Uploaded by:-

COMPUTER ENGINEERING (MU)

BY ONE CLICK TEAM

Telegram Channel

SECOND YEAR ENGINEERING

Computer Graphics HandWritten Notes

❖Join our **Computer Engineering Notes** Telegram Channel to get all Study Material For all Semester Engineering.
Click On This Link To Join.
👉https://t.me/compeng_notes

❖ Also Join our Official Telegram Group.
Click on this Link To Join.
👉https://t.me/compeng_notesgrp

MODULE 1

Introduction & Overview of computer Graphics System -

Def^n - computer graphics is an art of drawing pictures, lines, charts etc using computers with the help of programming

- A sub field of computer science which studies method for digitally synthesizing & manipulating visual content.
- Representation & manipulation of pictorial data by computer
- various technologies used to create & manipulate such pictorial data
- Anything which is prepared using computer program & is not text & audio is known as computer graphics

Important terms used in CG -

1) Pixel -
Smallest element of any digital image.

2) Resolution -
Maximum number of pixels displayed on computer screen without overlapping

3) Rendering -
Generation of 2D/3D image from object using computer programs.

4) 3D projection -
Mapping 3D points on 2D plane

Date
Page

5) Ray Tracing -
Generating Image by tracing path of light through pixels in an image plan.

* Advantages of CG -
- Provides tools for producing pictures/ films.
- It has ability to show moving pictures hence it is possible to produce animations with computer graphics
- With CG, user can control the animations speed, position of the view, the geometric relationship of object in a scene to one another, the amount of detail shown & so on.
- CG provides tool called motion dynamics with this user can move & tunible objects with respect to stationary observer or be can make objects stationary & the viewer moving around them
- GG provides update dynamics with update dynamics it is possible to change shape color or other properties of the objects' being viewed.

* Applications of Computer Graphics -
The following applications of computer graphics attempts to solve many of our day to day problems.

Date
Page

① Digital art -

A digital art, most commonly refers to an art created on a computer in a digital form.
A digital art can be purely computer generated, such as fractals & an algorithmic art taken from another source like a scanned photograph or image drawn using vector graphics soft ware.

② Special effects -

Illusions used in films, television & entertainment industries to simulate the imagined events in a story are traditionally called special effects (also known as SFX or SPFX)
In modern films, special effects are usually used to alter previously filmed elements by adding, removing or enhancing objects within the scene.

③ Visual effects -

Visual effects (or VFX) are the term given in which images or film frames are created & manipulated for film & video.
Visual effects usually involves the integration of live action footage with computer generated imagery or other elements (such as pyrotechnics

Date
Page

or model work) in order to create environments or scenarios which look realistic.

(4) Video Games -
Video games find a good place in the gaming & animation industry.

(5) Computer aided design -
Computer aided design (CAD) can be used by civil, mechanical & electronic engineers to build model of buildings, bridges, printed circuit boards (PCB) etc.

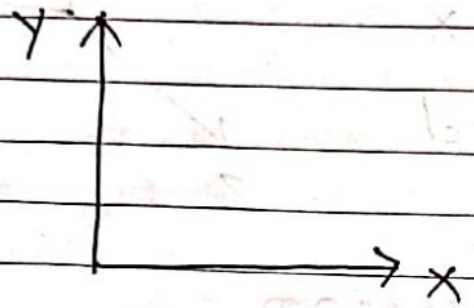
(6) Medical Imaging -
It can be used to train new doctors on understanding & experimenting surgery.
It is used in molecular biology to study the biological model in genes.

* Overview of co-ordinate system
- Coordinate system is a way of assigning number to points
- In 2D, we need a pair of numbers to specify a point & the coordinates are refersed to as x & y
- In 3D, we need 3 numbers to specify a point & the third coordinate is often called z

Date
Page

2-D coordinate system

- In 2D coordinate system, an actor's position can determine based on the x and y axes, left right & up down respectively.



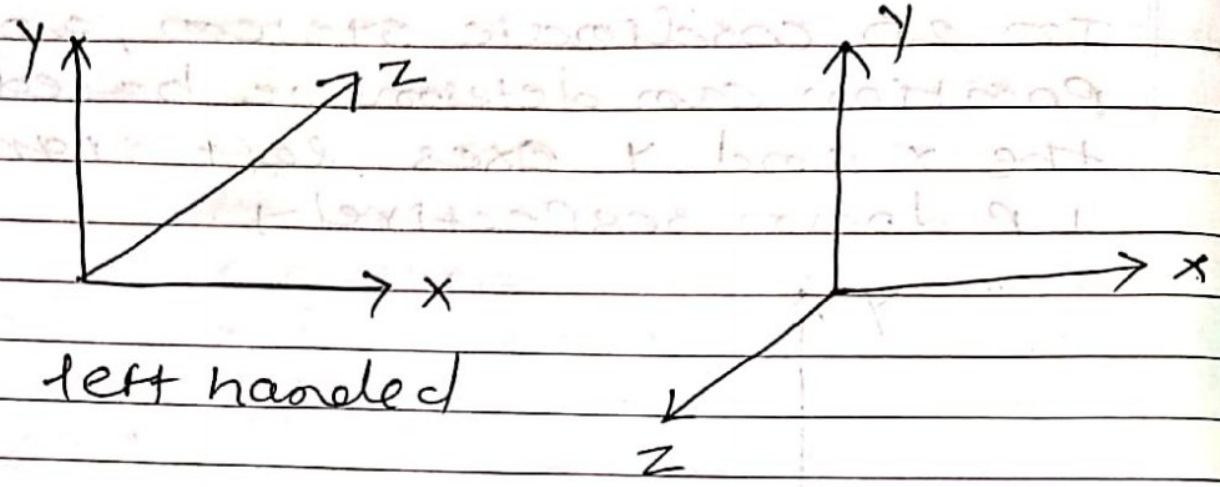
- Two types
1) Left handed
2) Right handed

3-D coordinate system

- In 3D coordinate system, along with the x & y dimensions we can determine the actor's position based on the z axis : the inclusion of depth.
- 3D graphics applications use 2 types of cartesian coordinate systems
1) Left handed
2) Right handed
- In both coordinate systems, the positive x-axis points to the right & the positive y-axis points up
- The fingers represent x & y direction
- The directions of thumb points, either toward or away, is the direction that the positive z-axis points for that

Date
Page

coordinate system



left handed

* Scan conversion -

- The process of representing continou graphics object as a collection of discrete pixels is known as scan con
- It is a process of representing graph objects by a collection of pixels.
- The graphics objects are continuous
- The pixels used are discrete.
- Each pixel can have either
- The circuitry of the video display device of the computer is capable of converting binary values (0, 1) into a pixel on & pixel off information.
- 0 is represented by pixel off
- 1 is represented using pixel on
- Using this ability computer graphics represent picture having discrete dots.
- Any model of graphics can be reproduced with a dense matrix of dots or points

Date
Page

Most human beings think graphics objects as points, lines, circles ellipses. For generating graphical object many algorithms have been developed

This can be demonstrated by a line that is defined by its 2 endpoints & the equation of that line.

Also by a circle that is represented by its mid point & the radius

The job of conversion of primitive object represented on the graphics screen into a set of pixels that is its basic form is termed as scan conversion or rasterization

Advantages of scan conversion

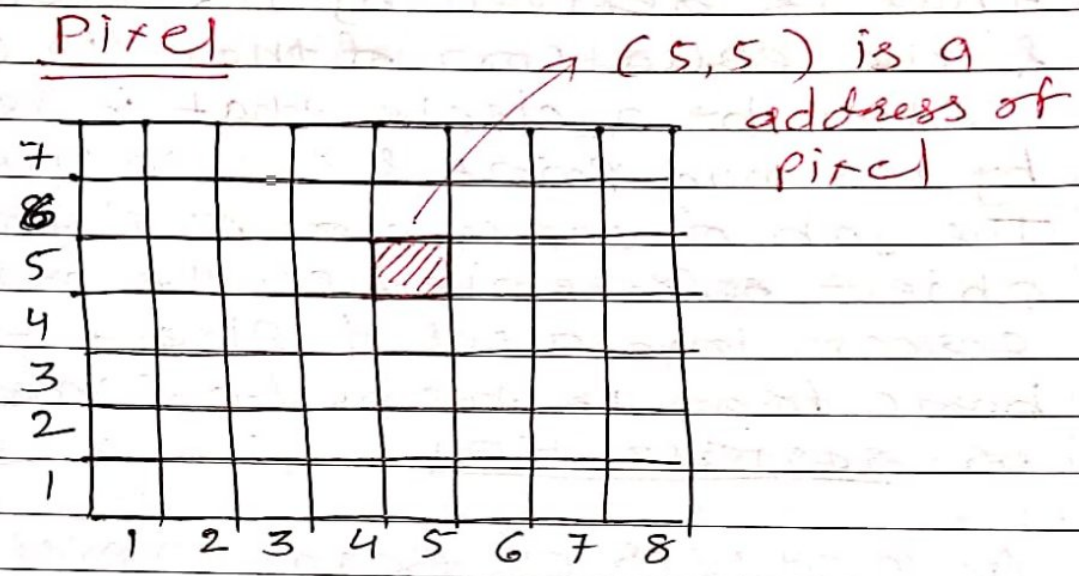
1) Algorithms can generate graphics objects at faster rate.
2) Memory can be used efficiently
3) Algorithms can develop a higher level of graphical objects.
4) Scan conversion methodology is used for many purposes including TV, the card used to capture a video, LCD monitor projects used for video display any more.

Objects that can be scan converted

1) Point
2) Line
3) Sector

4) Arc
5) Ellipse
6) Rectangle
7) Polygon
8) Characters
9) Filled Regions

Pixel



☐ each box represented are pixel

- A pixel is the smallest unit of a graphics image or referred to as a physical point on a raster
- The intensity of each pixel may vary as it depends on the image represented in the graphics image & the color combination is represented by the 3 basic colors i.e. red, green & blue.
- We can define a pixel as a rectangular dot centered at an integral position on an integral grid.
  It is a real point that is addressable
- The pixel is also called 'picture

element (Pel)

A pixel includes its own

Intensity -

The intensity of a pixel defines the brightness of the image that appears on the screen

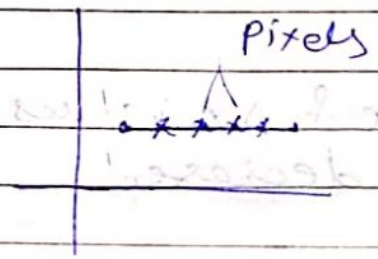
Name or address -

The name or address of a pixel defines the position of the pixel.

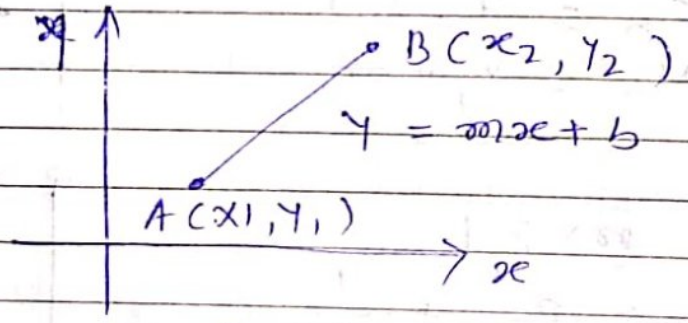
MODULE 2 (Output primitives)

DDA (Digital Differential Analyser)

- It is a scan conversion method for drawing lines.
- Incremental approach (To find out (.....) next point we realise previous point)
- Difference in pixel point is to be analysed



- What exactly is line



Straight line eqn is

$$y = mx + b$$

slope

if the line has 2 end point →
A (x_1, y_1) & B (x_2, y_2)

Slope of the line is

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

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

DDA algorithm based on calculation of values Δx, Δy.

$\Delta y = m * \Delta x \rightarrow (1)$

$\Delta x = \Delta y / m \rightarrow (2)$

Given line have the slope +ve or -ve

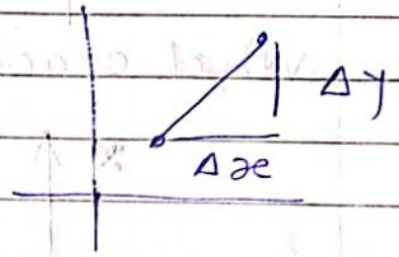
[$m > 0$ +ve
$m \leq 0$ -ve]

If +ve, the Δx & Δy values are increased else decresed

Case 1 → if $m < 1$

(new value) $x_n = x_1 + 1$

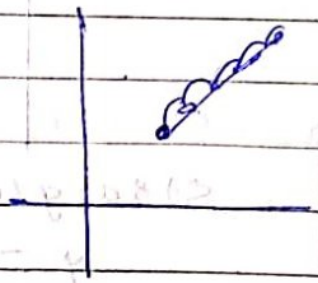
(new value) $y_n = y_1 + m$



Case 2 if $m > 1$

$x_n = x_1 + \frac{1}{m}$

$y_n = y_1 + 1$



Case 3 If $m = 1$

$x_n = x_1 + 1$

$y_n = y_1 + 1$

Algorithm

Step 1 - Accept start & end point coordinates (x_1, y_1) & (x_2, y_2)

Step 2 - calculate

$dx = x_2 - x_1$,

$dy = y_2 - y_1$,

Step 3 - If $abs(dx) > abs(dy)$ // $m < 1$

then $k = abs(dx)$

else $k = abs(dy)$

Step 4 - calculate $\Delta x = \frac{dx}{k}$

$\Delta y = \frac{dy}{k}$

Step 5 - Initialize $x = x_1$, $y = y_1$,

Step 6 - Display pixel (x, y)

Step 7 - $x = x + \Delta x$, $y = y + \Delta y$

Step 8 - Display pixel round (x), round (y)

Step 9 - repeat S_7 & S_8 'k' times

Draw the line betn

eg Start pts $(5, 5)$ ending $(10, 9)$

x_1, y_1 x_2, y_2

$dx = x_2 - x_1 = 10 - 5 = 5$ | $dx > dy$ | $dx < dy$

$dy = y_2 - y_1 = 9 - 5 = 4$ | $k = dx$ | $k = dy$

$\therefore dx > dy$ $5 > 4$

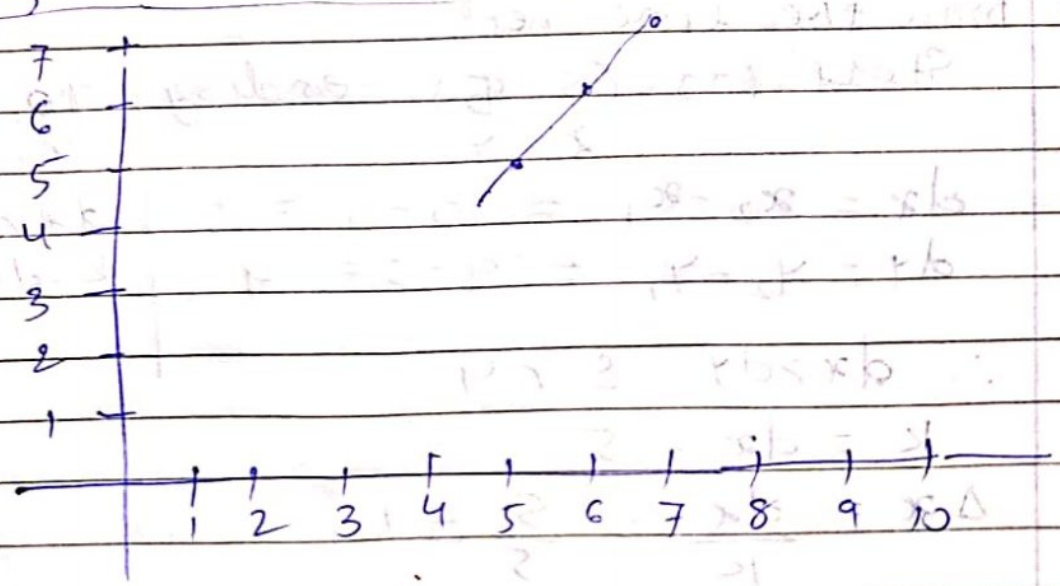
$k = dx = 5$

$\Delta x = \frac{dx}{k} = \frac{5}{5} = 1$

$$dy = \frac{dy}{k} = \frac{4}{5} = 0.8$$

$\boxed{\text{upto } k = 5}$

iteration (i)	plot (roundx, roundy)	$x + \Delta x$	$y + \Delta y$
0	$(5,5) \Rightarrow (x_1, y_2)$	$5+1=6$	$5+0.8 = 5.8$
1	$(6,6)$	$6+1=7$	$5.8+0.8 = 6.6$
2	$(7,7)$	$7+1=8$	$6.6+0.8 = 7.4$
3	$(8,7)$	$8+1=9$	$7.4+0.8 = 8.2$
4	$(9,8)$	$9+1=10$	$8.2+0.8 = 9$
5	$(10,9) \Rightarrow (x_2, y_2)$.	$10+1=11$	$9+0.8 = 9.8$



Date
Page

Bresenham's Line Generation (Drawing) Algorithm -

- It is an accurate & efficient raster line generation algorithm developed by Bresenham
- In this vertical axes shows position of pixel & horizontal axes identifies pixel columns.
- In this to draw a line from adjacent pixels which pixel will be nearer to line is determined.

Derivation -

- Assume that pixel at (x_k, y_k) is determined to display
- Next pixel needed to be decided to be placed in x_{k+1} column. So the choices are (x_{k+1}, y_k) & (x_{k+1}, y_{k+1})
- So the differences betn above 2 pixels can be d_1 & d_2
- Consider, y co-ordinate for pixel column position x_{k+1} is calculated as

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

&

$$d_1 = y - y_k$$
$$= m(x_{k+1}) + b - y_k$$

$$d_2 = (y_{k+1}) - y$$
$$= y_{k+1} - m(x_{k+1}) - b$$

So the difference between these 2 separations is

$$d_1 - d_2 = 2m(x_{k+1}) - 2y_k + 2b - 1$$

The eqn for decision parameter P_k is

$$P_k = dx(d_1 - d_2)$$

$$= dx[2m(x_{k+1}) - 2y_k + 2b - 1]$$

Now replacing m by dy/dx

$$= dx\left[2 \cdot \frac{dy}{dx}(x_{k+1}) - 2y_k + 2b - 1\right]$$

$$= dx\left[2 \cdot \frac{dy}{dx} x_k + \frac{2dy}{dx} - 2y_k + 2b - 1\right]$$

$$= dx\left[2 \cdot \frac{dy}{dx} x_k - 2y_k + \frac{2dy}{dx} + 2b - 1\right]$$

$$= dx\left[\frac{2dy \cdot x_k - 2dx y_k + 2dy + dx[2b-1]}{dx}\right]$$

$$P_k = 2dy\, x_k - 2dx\, y_k + c \rightarrow (2)$$

(here $c = 2dy + dx[2b-1]$)

At step $k+1$,

$$P_{k+1} = 2dy \cdot x_{k+1} - 2dx\, y_{k+1} + c$$

Hence

$$P_{k+1} - P_k = 2dy(x_{k+1} - x_k) - 2dx(y_{k+1} - y_k)$$

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

$y_k = y_{k+1}$

$P_{k+1} - P_k = 2dy(x_{k+1} - x_{k+1}) -$

$2dx(y_{k+1} - y_{k+1})$

$P_{k+1} - P_k = 2dy - 2dx$

$P_{k+1} = P_k + 2dy - 2dx \rightarrow (3)$

Now for original pixels (x_0, y_0) considering eqn (2)

$P_0 = 2dy - dx$

- Bresenham algorithm determines the pts of an n-dimensional raster that should be selected in order to form a close approximation to a straight line betn 2 pts.
- Only perform only integer addition subtraction, multiplication
- The operations performed rapidly so line can be generate quickly

Date
Page

Algorithm -

Start coordinate (x_0, y_0)
End coordinate (x_n, y_n)

Step 1 calculate Δx & Δy

$$\Delta x = x_n - x_0$$
$$\Delta y = y_n - y_0$$

Step 2 calculate decision parameter
(It is used to find exact point to draw line)

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

Step 3 suppose current pt (x_K, y_K), next pt (x_{K+1}, y_{K+1})

find next pt. depending on value of decision parameter P_K.

Case 1

If $P_K < 0$

$$P_{K+1} = P_K + 2\Delta y$$
$$x_{K+1} = x_K + 1$$
$$y_{K+1} = y_K$$

Case 2 If $P_K \geq 0$

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

Date
Page

Step 4 Repeat step 3 until end point is reached.

Advantages -

1. It is simple because it involves integer arithmetic
2. It avoids the generation of duplicate pt's
3. It can be implemented using H/w
4. It is faster.

Disadvantages

1. Means for basic line drawing only

eg. Starting coordinates $(x_0, y_0) = (9, 18)$
Ending coordinates $(x_n, y_n) = (14, 22)$

Step 1 -

$\Delta x = x_n - x_0$
$= 14 - 9$
$\Delta x = 5$

$\Delta y = y_n - y_0$
$= 22 - 14$
$\Delta y = 4$

Step 2 Calculate decision parameter

$P_k = 2\Delta y - \Delta x$
$= 2 \times 4 - 5$
$= 8 - 5$
$P_k = 3$

Date
Page

Step 3

As $P_K > 0$

here $P_K > 0$ case 2 is satisfied

$3 > 0$

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

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

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

$P_{K+1} = 3 + 2 \times 4 - 2 \times 5$

$= 3 + 8 - 10$

$P_{K+1} = 1$

$x_{K+1} = 9 + 1 = 10$

$y_{K+1} = 18 + 1 = 19$

P_K	P_{K+1}	x_{K+1}	y_{K+1}
		9	18
3	1	10	19
1	-1	11	20
-1	7	12	20
7	5	13	21
5	3	14	22

stop here

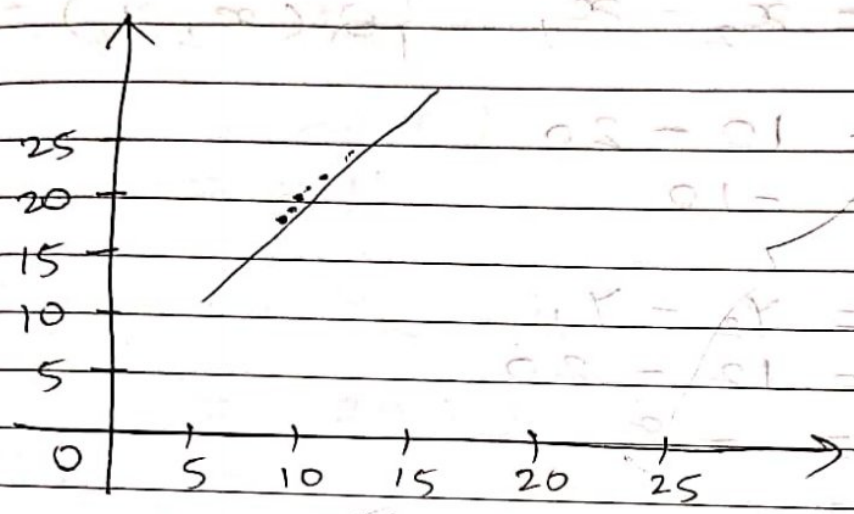
reached end points

$P_k < 0$

$P_{k+1} = P_k + 2\Delta y$

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

$y_{k+1} = y_k$



Date
Page

(2) Write Bresenham's line drawing algorithm. calculate the pixel co-ordin of line PQ using Bresenham's Algorithm. Were P, (20,20) & Q (10,12)

⇒ **Step 1**

$\Delta x = x_2 - x_1$ $\quad \begin{cases} P(x_1, y_1) = (20,20) \\ Q(x_2, y_2) = (10,12) \end{cases}$

$= 10 - 20$

$\Delta x = -10$

$\Delta y = y_2 - y_1$

$= 12 - 20$

$\Delta y = -8$

Step 2 Since Δx & Δy are negative we will take their absolute values

$\Delta x = -10 = 10$

$\Delta y = -8 = 8$

Now let's find decision factor P_K

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

$= 2(8) - 10$

$= 16 - 10$

$P_K = 6$

Step 3 As $P_K > 0$

here

$6 > 0$ case 2 is satisfied

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

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

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

Date
Page

$$P_{k+1} = P_k + 2\Delta y - 2\Delta x$$

$$= 6 + 2(8) - 2(10)$$

$$= 6 + 16 - 20$$

$$P_{k+1} = 2$$

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

$$= 20 + 1$$

$$x_{k+1} = 21$$

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

$$= 20 + 1$$

$$y_{k+1} = 21$$

P_k	P_{k+1}	x_{k+1}	y_{k+1}
		20	20
6	2	21	21
+2	-2	22	22
-2	14	23	22
14	10	24	23
10	6	25	24
6	2	26	25
2	-2	27	26
-2	14	28	26

Plot the line using Brashenham's line drawing algorithm
Starting co-ordinates $(x_0, y_0) = (5, 5)$
Ending co-ordinates $(x_n, y_n) = (13, 9)$

Step 1

$$\Delta x = x_n - x_0$$
$$= 13 - 5$$
$$\Delta x = 8$$

$$\Delta y = y_n - y_0$$
$$= 9 - 5$$
$$\Delta y = 4$$

Step 2 calculate decision Parameter

$$P_K = 2\Delta y - \Delta x$$
$$= 2(4) - 8$$
$$P_K = 0$$

Step 3 $P_K \geq 0$

$0 \geq 0$ case 2 is satisfied

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

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

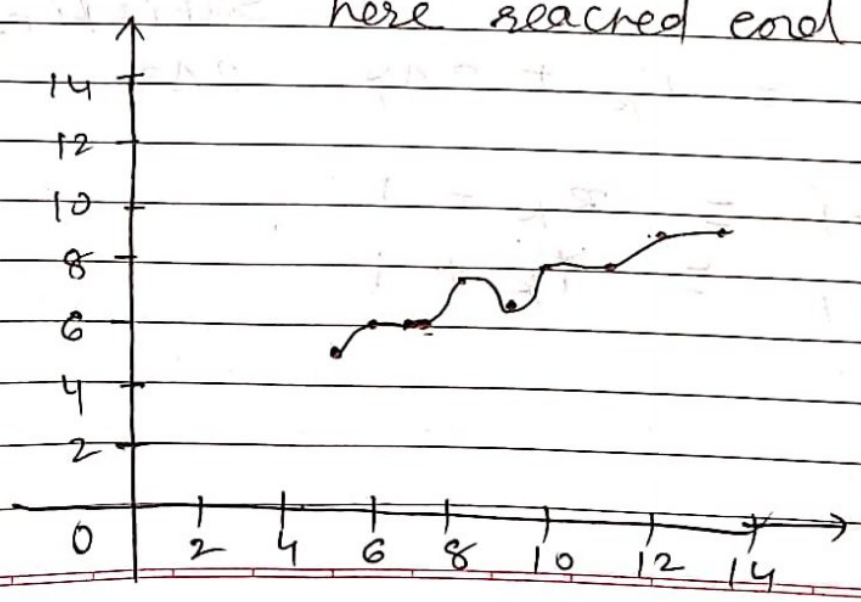
$$P_{k+1} = 0 + 2(4) - 2(8)$$
$$= 0 + 8 - 16$$
$$P_{k+1} = -8$$

$$x_{k+1} = 5 + 1$$
$$= 6$$

$$y_{k+1} = 5 + 1$$
$$= 6$$

P_k	P_{k+1}	x_{k+1}	y_{k+1}	
		5	5	
0	-8	6	6	
-8	0	7	6	
0	-8	8	7	
-8	0	9	7	
0	-8	10	8	
-8	0	11	8	
0	-8	12	9	
-8	0	13	9	stop

here reached end point



Date
Page

(4) $A(x_1, y_1) = (20, 10)$
$B(x_2, y_2) = (30, 18)$

⇒ **Step 1**

$$\Delta x = x_2 - x_1$$
$$= 30 - 20$$
$$\Delta x = 10$$

$$\Delta y = y_2 - y_1$$
$$= 18 - 10$$
$$\Delta y = 8$$

Step 2 calculate decision parameter

$$P_k = 2\Delta y - \Delta x$$
$$= 2(8) - 10$$
$$= 16 - 10$$
$$P_k = 6$$

Step 3 $P_k \geq 0$

$6 \geq 0$ case 2

$$P_{k+1} = P_k + 2\Delta y - 2\Delta x$$
$$= 0 + 2(8) - 2(10)$$
$$= 0 + 16 - 20$$
$$P_{k+1} = 4$$

$$x_{k+1} = x_k + 1$$
$$= 20 + 1$$
$$x_{k+1} = 21$$
$$y_{k+1} = y_k + 1$$
$$= 10 + 1$$
$$y_{k+1} = 11$$

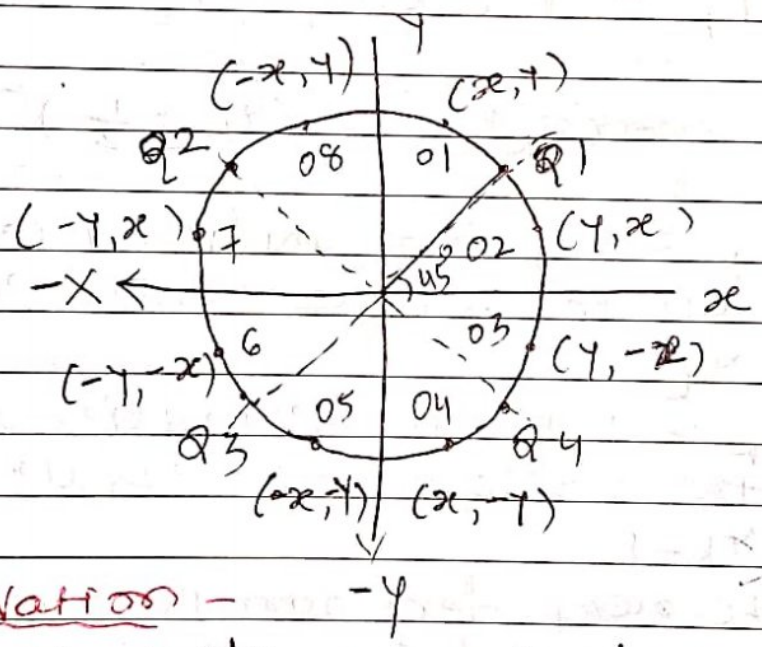
Date
Page

Midpoint

Circle Drawing Algorithm

Symmetry of Circle

- circle is symmetric in nature. There is symmetry between octants
- circle is collection of adjacent octants line with 45^0 divides 2 octants
- fig. above shows point (x, y) is mapped in seven circle points in different octants of xy plane.
- By taking circle geometry into consideration, circle can be drawn within the section from $x = 0$ to $x = y = x$



Circle is a set of all points that lie at an equal distance (radius) from a fixed point called centre.

Derivation -

To apply midpoint method circle function can be ~~determined~~ defined as

$$f(x, y) = x^2 + y^2 - r^2 \rightarrow (1)$$

The position of any point (x, y) can be determined by checking the sign of the circle function

Date
Page

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

Assume pixel (x_k, y_k) has been plotted, to determine next pixel position two choices are there (x_{k+1}, y_k) & (x_{k+1}, y_{k-1})

So decision parameter can be evaluated at the midpoint between these 2 pixels

$$P_k = f(x_{k+1}, y_k - 1/2)$$

$$= (x_k+1)^2 + (y_k - \tfrac{1}{2})^2 - r^2 \rightarrow (2)$$

If $P_k < 0$, then midpoint is inside the circle, so the result pixel will be y_k

If $P_k \geq 0$, the midpoint is outside or on the circle so the result pixel will by y_{k-1}

So the next decision parameter will be obtained by values

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

$$P_0 = f(1, r - 1/2)$$

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

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

Date
Page

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

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

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

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

[If r is specified as integer then replace $5/4$ by 1

$$P_0 = 1 - r$$

Algorithm –

If r is the radius of the circle to be drawn & ORIGIN is its centre then to plot first octant of the circle, do the following

Step 1

Plot the initial point (x_i, y_i) such that $x_i = 0$ & $y_i = r$

Step 2

Find initial decision parameter

$$P_i = \frac{5}{4} - r$$

Step 3

If $P_i < 0$ then

$$x_{i+1} = x_i + 1$$

$$y_{i+1} = y_i$$

$$P_{i+1} = P_i + 2x_i + 3$$

4 **Step 4**

If $P_i \geq 0$ then

$$x_{i+1} = x_i + 1$$

$$y_{i+1} = y_i - 1$$

$$P_{i+1} = P_i + 2(x_i - y_i) + 5$$

Step 5

Repeat steps 3, 4 until x becomes greater than or equal to y

To plot the complete circle, reflect each point of the first octant, into 7 other octants making use of 8-way symmetry.

Date
Page

① Plot the I^{st} octant of a circle centered at origin, having radius 10 units.

$\Rightarrow$ Plot $(0, 8) \rightarrow (0, 10) \rightarrow$

$P_0 = 1 - 8 = 1 - 10 = -9$

1 $P_0 < 0 \rightarrow P_1 = P_0 + 2x_0 + 3$ $\quad (x_1, y_1) = (1, 10)$

$P_1 = -9 + 2(0) + 3 = -6$

2. $P_1 < 0 \rightarrow P_2 = P_1 + 2x_1 + 3$ $\quad (x_2, y_2) = (2, 10)$

$P_2 = -6 + 2(1) + 3$

$P_2 = -6 + 2 + 3$

$P_2 = -1$

3. $P_2 < 0 \rightarrow P_3 = P_2 + 2x_2 + 3$ $\quad (x_3, y_3) = (3, 10)$

$P_3 = -1 + 2(2) + 3$

$P_3 = -1 + 4 + 3$

$P_3 = 6$

4 $P_3 > 0 \rightarrow P_4 = P_3 + 2(x_3 - y_3) + 5$ $\quad (x_4, y_4) = (4, 9)$

$P_4 = 6 + 2(3 - 10) + 5$

$P_4 = 6 + 2(-7) + 5$

$P_4 = 6 - 14 + 5$

$P_4 = -3$

5) $P_4 < 0 \rightarrow P_5 = P_4 + 2x_4 + 3$ $\quad (x_5, y_5) = (5, 9)$

$= -3 + 2(4) + 3$

$= -3 + 8 + 3$

$P_5 = 8$

Date
Page

6 $P_5 > 0$ $P_6 = P_5 + 2(x_5 - y_5) + 5$ $(x_6, y_6) = (6, 8)$

$= 8 + 2(5-9) + 5$

$= 8 + 2(-4) + 5$

$= 8 - 8 + 5$

$P_6 = 5$

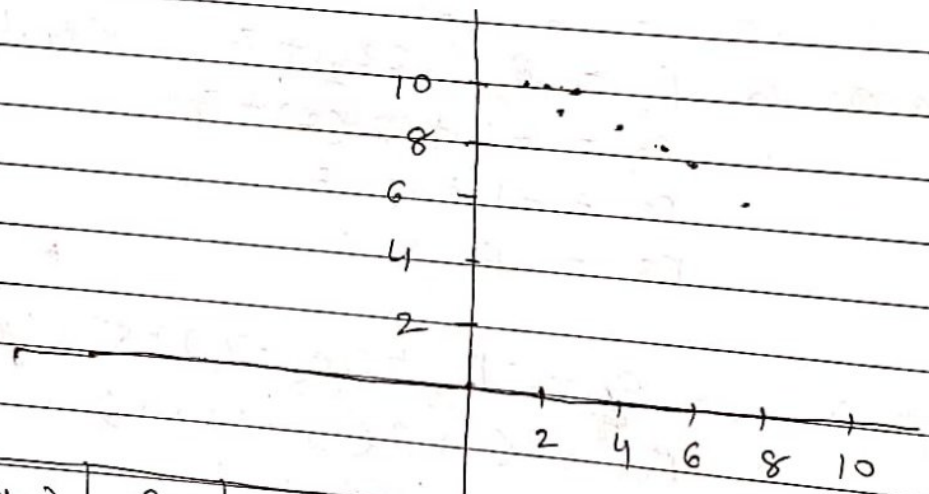
7 $P_6 > 0$ $P_7 = P_6 + 2(x_6 - y_6) + 5$ $(x_7, y_7) = (7, 7)$

$= 5 + 2(6-8) + 5$

$= 5 + 2(-2) + 5$

$= 5 - 4 + 5$

$P_7 = 6$



I	(x_k, y_k)	P_k	x_{k+1}, y_{k+1}
1	(0, 10)	-9	(1, 10)
2	(1, 10)	-6	(2, 10)
3	(2, 10)	-1	(3, 10)
4	(3, 10)	6	(4, 9)
5	(4, 9)	-3	(5, 9)
6	(5, 9)	8	(6, 8)
7	(6, 8)	5	(7, 7)

Date
Page

Plot a circle whose radius is 3 & whose centre co-ordinates are (0, 0)

$\Rightarrow$ Plot $(\overset{x_0}{0}, \overset{r_0}{3}) \rightarrow (0, 3)$

$P_0 = 1 - r = 1 - 3 = -2$

1) $P_0 < 0$ $\quad P_1 = P_0 + 2x_0 + 3 \quad (x_1, y_1) = (1, 3)$

$P_1 = -2 + 2(0) + 3$

$= -2 + 0 + 3$

$P_1 = 1$

2 $P_1 > 0$ $\quad P_2 = P_1 + 2(x_1 - y_1) + 5 \quad (x_2, y_2) = (2, 2)$

$= 1 + 2(1 - 3) + 5$

$= 1 + 2(-2) + 5$

$= 1 - 4 + 5$

$P_2 = 2$

3 $P_2 \geq 0$ $\quad P_3 = P_2 + 2(x_2 - y_2) + 5 \quad (x_3, y_3) = (3, 1)$

$= 2 + 2(2 - 2) + 5$

$= 2 + 2(0) + 5$

$= 2 + 0 + 5$

$P_3 = 7$

4 $P_3 > 0$ $\quad P_4 = P_3 + 2(x_3 - y_3) + 5 \quad (x_4, y_4) = (4, 1)$

$= 3 + 2(3 - 2) + 5$

$= 3 + 2(1) + 5$

$= 3 + 2 + 5$

$P_4 = 10$

I	(x_k, y_k)	P_k	x_{k+1}, y_{k+1}
1	(0, 3)	-2	(1, 3)
2	(1, 3)	1	(2, 2)
3	(2, 2)	2	(3, 1)

Date
Page

Draw a circle with radius 8 unit

$\Rightarrow$

Plot $(0, 8) \rightarrow (0, 8)$

$P_0 = 1 - 8 = 1 - 8 = -7$

1 $P_0 < 0$ $\quad P_1 = P_0 + 2x_0 + 3 \qquad (x_1, y_1) = (1, 8)$

$= -7 + 2(0) + 3$

$= -7 + 0 + 3$

$P_1 = -4$

2 $P_1 < 0$ $\quad P_2 = P_1 + 2(x_1) + 3 \qquad (x_2, y_2) = (2, 8)$

$= -4 + 2(1) + 3$

$= -4 + 2(-7) + 5$

$= -4 + 12 + 3$

$= -18 + 5$

$P_2 = +1$

3 $P_2 > 0$ $\quad P_3 = P_2 + 2(x_2 - y_2) + 5 \qquad (x_3, y_3) = (3, 7)$

$= 1 + 2(2 - 8) + 5$

$= 1 + 2(-6) + 5$

$= 1 - 12 + 5$

$P_3 = -6$

4) $P_3 < 0$ $\quad P_4 = P_3 + 2(x_3) + 3 \qquad (x_4, y_4) = (4, 7)$

$= -6 + 2(3) + 3$

$= -6 + 6 + 3$

$P_4 = 3$

Date
Page

5 $P_4 > 0$ $P_5 = P_4 + 2(x_4 - y_4) + 5$ $(x_5, y_6) = (5, 6)$

$= 3 + 2(4 - 7) + 5$

$= 3 + 2(-3) + 5$

$= 3 - 6 + 5$

$P_5 = 2$

6 $P_5 > 0$ $P_6 = P_5 + 2(x_5 - x_6) + 5$ $(x_6, y_7) = \underline{\underline{(6,5)}}$

$= 2 + 2(5 - 6) + 5$

$= 2 + 2(-1) + 5$

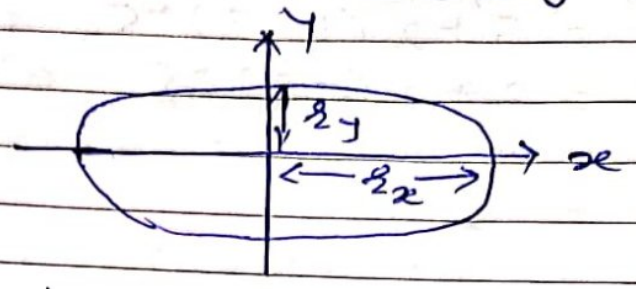
$= 2 - 2 + 5$

$P_6 = 5$

Iteration	(x_k, y_k)	P_k	x_{k+1}, y_{k+1}
1	(0, 8)	−7 ✓	(1, 8)
2	(1, 8)	−4 ✓	(2, 8)
3	(2, 8)	1 ✓	(3, 7)
4	(3, 7)	−6 ✓	(4, 7)
5	(4, 7)	3 ✓	(5, 6)
6	(5, 6)	~~2~~ 5 ✓	$\underline{\underline{(6, 5)}}$

Date
Page

** Midpoint Ellipse Drawing algorithm



- Ellipse is an elongated circle
- Equation of ellipse centered at (0,0) is

$$\frac{x^2}{r_x\, a^2} + \frac{y^2}{b^2\, r_y} = 1 \rightarrow (1)$$

- Major Axis $= 2a = 2r_x$
- Minor Axis $= 2b = 2r_y$

Eqn (1) can be rewritten as

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

$$x^2 b^2 + y^2 a^2 = a^2 b^2$$

$$b^2 x^2 + a^2 y^2 = a^2 b^2$$

$$b^2 x^2 + a^2 y^2 - a^2 b^2 = 0$$

$$r_y^2 x^2 + r_x^2 y^2 - a^2 b^2 = 0 \rightarrow (2)$$

Is the final eqn of ellipse.

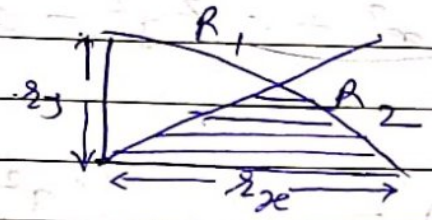
- If we put any point in eqn (2)

$= 0$	$< = 0$	> 0
Point lies ON ellipse	INSIDE the ellipse	OUTSIDE the ellipse

Date
Page

Difference between circle & ellipse

- circle has 8 way symmetry
- Ellipse has 4 way symmetry
- In circle we need to plot only one octant of any quadrant, but in ellipse we need to plot 2 octants i.e. 1 complete quadrant to plot entire ellipse.



Derivation - $\boxed{\text{Region 1 - Quadrant 1}}$

- Start point $(0, r_y)$
- Slope of curve ≤ -1
- Take unit steps in positive x direction direction till boundary between the 2 2 regions is reached.
- Where we do $x = x + 1$

$\boxed{\text{Region 2 - Quadrant 1}}$

- Start point $(r_x, 0)$
- Slope of curve ≥ -1
- Take unit steps in positive y direction till the end of quadrant
- Where we do $y \rightarrow y - 1$

on the boundary between 2 regions / octants, the slope of the curve is -1

Date
Page

Slope of the curve

Eq^n of Ellipse is as follows

$$r_y^2 x^2 + r_x^2 y^2 - a^2 b^2 = 0$$

Here consider the start position $(0, r_y)$

$$\frac{dy}{dx}\{r_y^2 x^2 + r_x^2 y^2 - a^2 b^2\} = 0$$

$$\frac{dy}{dx}(y^2) = \frac{dy}{dx}\left(\frac{a^2b^2 - r_y^2 x^2}{r_x^2}\right)$$

$$= \frac{dy}{dx}\left(\frac{r_x^2 r_y^2}{r_x^2}\right) - \frac{dy}{dx}\left(\frac{r_y^2 x^2}{r_x^2}\right)$$

$$\frac{dy}{dx} \cdot 2y = \frac{-2r_y^2 x}{r_x^2}$$

$$\frac{dy}{dx} = \frac{-2r_y^2 x}{2r_x^2 y}$$

$$\frac{-2r_y^2 x}{2r_x^2 y} = -1$$

$$\boxed{-2r_y^2 x = -2r_x^2 y}$$

Note

If $\frac{dy}{dx}$
- $< 0 \rightarrow (x, y)$ inside the ellipse boundary
- $= 0 \rightarrow (x, y)$ on ellipse boundary
- $> 0 \rightarrow (x, y)$ outside the ellipse boundary

Date
Page

Region 1

$(x_k+1, y_k), (x_k+1, y_k-1) \Rightarrow (\underbrace{x_k+1, y_k-\frac{1}{2}}_{\text{mid point}})$

Eqn $r_y^2 x^2 + r_x^2 y^2 - r_x^2 r_y^2 = 0$

$r_y^2 (x_k+1)^2 + r_x^2 (y_k - \frac{1}{2})^2 - r_x^2 r_y^2 = pl_k$ $\rightarrow$ (1)

mid Point

$r_y^2 (x_{k+1}+1)^2 + r_x^2 (y_{k+1} - \frac{1}{2})^2 - r_x^2 r_y^2 = pl_{k+1}$

$r_y^2 ((x_k+1)+1)^2 + r_x^2 (y_{k+1} - \frac{1}{2})^2 - r_x^2 r_y^2 = pl_{k+1}$ $\rightarrow$ (2)

eqn (2) − (1) $pl_{k+1} - pl_k$

$\rightarrow r_y^2 \{(x_k+1)+1\}^2 - r_y^2 (x_k+1)^2 + r_x^2 (y_{k+1} - \frac{1}{2})^2 - r_x^2 (y_k - \frac{1}{2})^2 - \cancel{r_x^2 r_y^2} + \cancel{r_x^2 r_y^2}$

$\rightarrow r_y^2 \{(x_k+1)^2 + 1 + 2(x_k+1)\} - r_y^2 (x_k^2 + 1 + 2x_k) + r_x^2 (y_{k+1}^2 + \frac{1}{4} - y_{k+1}) - r_x^2 (y_k^2 + \frac{1}{4} - y_k)$

$\rightarrow r_y^2 \{\cancel{x_k^2} + 1 + 2x_k + \cancel{1} + \cancel{2x_k} + 2 - \cancel{x_k^2} - \cancel{1} - \cancel{2x_k}\} + r_x^2 \{y_{k+1}^2 + \cancel{\frac{1}{4}} - y_{k+1} - y_k^2 - \cancel{\frac{1}{4}} + y_k\}$

$\rightarrow r_y^2 \{2(x_k+1)+1\} + r_x^2 \{y_{k+1}^2 - y_k^2 - y_{k+1} + y_k\}$

x_{k+1} $pl_k < 0$ y_k y_{k-1} ; $pl_k \geq 0$

$r_y^2 [(2x_k + 2) + 1]$

$2x_k + 2 + 1$

$2x_{k+1} r_y^2 + 2r_y^2 + r_y^2$

Date
Page

$$P1_{k+1} - P1_k = r_y^2 [2(x_k + 1) + 1] + r_x^2 [\underline{y_{k+1}^2} - y_k^2 - \underline{y_{k+1}} + y_k]$$

$\Rightarrow$

* When $P1_k < 0 \rightarrow y_{k+1} = \underline{\underline{y_k}}$

$$P1_{k+1} = P1_k + r_y^2 (2x_{k+1} + 1) \longrightarrow$$

$$\boxed{P1_{k+1} = P1_k + r_y^2 2x_{k+1} + r_y^2}$$

* When $P1_k \geq 0 \rightarrow y_{k+1} = y_k \underline{\underline{-1}}$

$$P1_{k+1} - P1_k = 2x_{k+1} r_y^2 + r_y^2 + r_x^2 [(y_k - 1)^2 - y_x^2 - (y_k - 1) + y_k]$$

$$P1_{k+1} = P1_k + 2x_{k+1} r_y^2 + r_y^2 + r_x^2 [\cancel{y_k^2} + 1 - 2y_k - \cancel{y_x^2} - \cancel{y_k} + 1 + \cancel{y_k}]$$

$$P1_{k+1} = P1_k + 2x_{k+1} r_y^2 + r_y^2 - 2y_{k+1} r_x^2$$

\# Initial Decision Parameter

put $(0, r_y)$ in $P1_k$

$$P1_k = r_y^2 (x_k + 1)^2 + r_x^2 (y_x - \tfrac{1}{2})^2 - r_x^2 r_y^2$$

$$P1_0 = r_y^2 (0 + 1)^2 + r_x^2 (r_y - \tfrac{1}{2})^2 - r_x^2 r_y^2$$

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

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

Date
Page

$$\boxed{P_0' = 2y^2 + \frac{2x^2}{4} - 2y\,x_2^2}$$

Region 2

$(x_k, y_k - 1), (x_k + 1, y_k - 1) \rightarrow (x_k + \frac{1}{2}, y_k - 1)$

midpoint

$$\cancel{\frac{x_k + x_k + \frac{1}{2}}{2}} = \cancel{\frac{2x_k + 2x_k + 1}{2}}$$

$$= \cancel{\frac{(2x_k + 1)}{2}} \cancel{\times \frac{1}{2}}$$

$=$

$$\frac{x_k + x_k + 1}{2} = \frac{2x_k + 1}{2}$$

$$= x_k + \frac{1}{2}$$

midpoint

$$\frac{y_k - 1 + y_k - 1}{2} = \frac{2y_k - 2}{2}$$

$$= y_k - 1$$

Date
Page

Eq^n of Ellipse

$$r_y^2 x^2 + r_x^2 y^2 - r_x^2 r_y^2$$

$$r_y^2 (x_k + \tfrac{1}{2})^2 + r_x^2 (y_k - 1)^2 - r_x^2 r_y^2 = P2_k \rightarrow (1)$$

$$r_y^2 (x_{k+1} + \tfrac{1}{2})^2 + r_x^2 (y_{k+1} - 1)^2 - r_x^2 r_y^2 = P2_{k+1} \rightarrow (2)$$

$$(2) - (1) : P2_{k+1} - P2_k$$

$$\rightarrow r_y^2 (x_{k+1} + \tfrac{1}{2})^2 - r_y^2 (x_k + \tfrac{1}{2})^2 + r_x^2 \underbrace{((y_k - 1) - 1)^2}_{a \quad b}$$

$$- r_x^2 (y_k - 1)^2 - \cancel{r_x^2 r_y^2} + \cancel{r_x^2 r_y^2}$$

$$\rightarrow r_y^2 [x_{k+1}^2 + \tfrac{1}{4} + x_{k+1}] - r_y^2 [x_k^2 + \tfrac{1}{4} + x_k]$$

$$+ r_x^2 [(y_k - 1)^2 + 1 - 2(y_k - 1)] + r_x^2 [y_k^2 + 1 + 2y$$

$$\rightarrow r_y^2 \{x_{k+1}^2 + \tfrac{1}{4} + x_{k+1} - x_k^2 - \tfrac{1}{4} - x_k\} +$$

$$r_x^2 \{y_k^2 + 1 - 2y_k + 1 - 2y_k + 2 - y_k^2 - 1 + 2y_k$$

$$\rightarrow r_y^2 \{x_{k+1}^2 + x_{k+1} - x_k^2 - x_k\} + r_x^2 \{-2y_{k+1} + 1$$

$$\rightarrow r_y^2 \{x_{k+1}^2 + x_{k+1} - x_k^2 - x_k\} +$$

$$r_x^2 \{-2y_{k+1} + 1\}$$

$$\rightarrow P2_{k+1} - P2_k = r_y^2 \{x_{k+1}^2 + x_{k+1} - x_k^2 - x_k$$

$$+ r_x^2 \{1 - 2y_{k+1}\}$$

$$r_y^2\left(x_{k+1}+\frac{1}{2}\right)^2 + r_x^2\left((y_k-1)-1\right)^2 - r_x^2 y_2^2 = P2_{k+1} \rightarrow (2)$$

$$P^2_{k+1} = P_2 k + r_y^2\{x^2_{k+1} + x_{k+1} - x_k^2 - x_k\} - 2y_{k+1} r_x^2 + r_x^2$$

If $P2_k > 0 \rightarrow x_{k+1} = x_k$

$\hookrightarrow P2_{k+1} = P2_k - 2y_{k+1} r_x^2 + r_x^2$

If $P2_k \leq 0 \rightarrow x_{k+1} = x_k + 1$

$\hookrightarrow$

$$P2_{k+1} = P2_k + r_y^2\{(x_k+1)^2 + x_k + 1 - x_k^2 - x_k\} - 2y_{k+1} r_x^2 + r_x^2$$

$$\rightarrow P2_{k+1} = P2_k + r_y^2\{\cancel{x_k^2} + 2x_k + 1 + \cancel{x_k} + 1 - \cancel{x_k^2} - \cancel{x_k}\} - 2y_{k+1} r_x^2 + r_x^2$$

$$\rightarrow \boxed{-2y_k + 2 \Rightarrow -2(y_k - 1) \Rightarrow -2y_{k+1}}$$

$$P2_{k+1} = P2_k + r_y^2(2x_{k+1}) - 2y_{k+1} r_x^2 + r_x^2$$

Initial Decision Parameter -

Obtained by putting the last point of Region 1 in the eqn

$$P2_k = r_y^2\left(x_k + \frac{1}{2}\right)^2 + r_x^2(y_k - 1)^2 - r_x^2 r_y^2$$

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

Midpoint ELLIPSE Drawing Algorithm -

- Read radius r_x & r_y
- Initialise starting point of Region 1 as $x = 0$, $y = r_y$ $(0, r_y)$
- Calculate

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

- calculate $dx = 2r_y^2 x$, $dy = 2r_x^2 y$

* Repeat while $(dx < dy)$ (Region 1)

Plot (x, y)

if $(P1 < 0)$

{

$x = x + 1$

update dx → $2r_y^2 x = \text{old } dx + 2r_y^2$

$P1 = P1 + 2r_y^2 x + r_y^2$

}

else → $P1 \geq 0$

{

$x = x + 1$, $y = y - 1$

update dx → $2r_y^2 x = \{\text{old } dx + 2r_y^2\}$

update dy → $2r_x^2 y = \{\text{old } dy - 2r_x^2\}$

$P1 = P1 + dx - dy + r_y^2$

}

* When $(dx \geq dy)$ Plot region 2 as

- Find $P2_0 = r_y^2 (x + \frac{1}{2})^2 + r_x^2 (y - 1)^2 - r_x^2 r_y^2$
- Repeat till $(y > 0)$
- Plot (x, y)
- if $(P_2 > 0)$

{

Date
Page

$x = x$

$y = y - 1$

update dy : $2r_x^2 y^2$

$P2 = P2 - dy + r_x^2$ → $2r_x^2 y$

}

else { $(P2 \leq 0)$

$x = x + 1$

$y = y - 1$

$dy = dy - 2r^2_x$

$dx = dx + 2r^2_y$

$P2 = P2 + dx - dy + r_x^2$

}

Date
Page

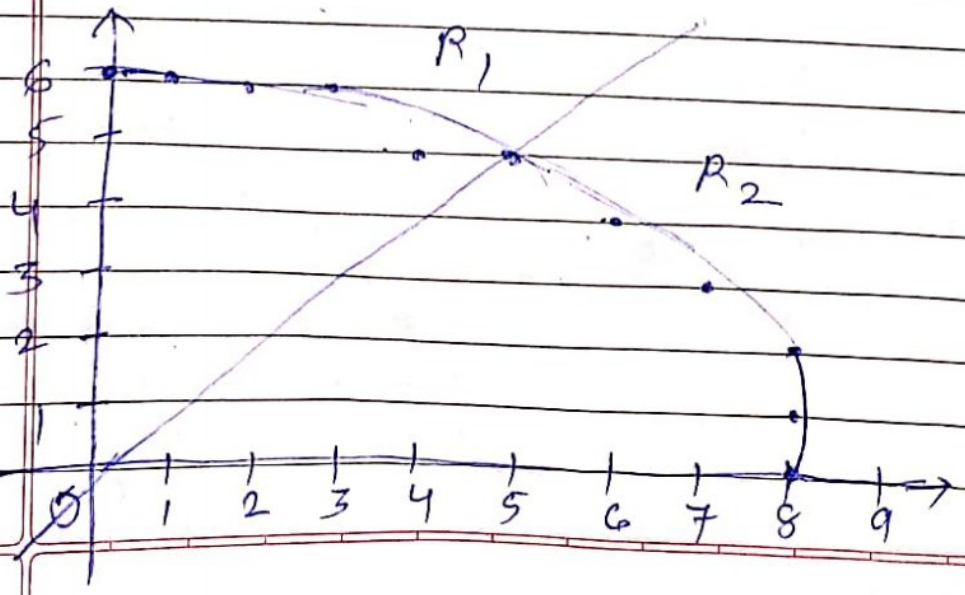
Draw ellipse using midpoint ellipse drawing algorithm where $r_x = 8$, $r_y = 6$

(x_k, y_k)	Decision Parameters P1 or P2	(x_{k+1}, y_{k+1})	$2x_{k+1} r_y^2$ (dx)	$2y_{k+1} r_x^2$ (dy)
R1				
0. (0,6)	$P1_0 = -332$	(1,6)	2(1)(36)=72	2(6)(64)=768
1. (1,6)	$P1_1 = -224$	(2,6)	2(2)(36)=144	2(6)(64)=768
2. (2,6)	$P1_2 = -44$	(3,6)	2(3)(36)=216	2(6)(64)=768
3. (3,6)	$P1_3 = 208$	(4,5)	2(4)(36)=288	2(5)(64)=640
4 (4,5)	$P1_4 = -108$	(5,5)	2(5)(36)=360	2(5)(64)=640
5 (5,5)	$P1_5 = 288$	(6,4)	2(6)(36)=432	2(4)(64)=512
6 (6,4)	$P1_6 = 244$	(7,3)	2(7)(36)=504	2(3)(64)=384

Here dx > dy so switch to Region 2

R2				
0 (7,3)	$P2_0 = -23$	(8,2)	2(8)(36)=576	2(2)(64)=256
1 (8,2)	$P2_1 = 361$	(8,1)	2(8)(36)=576	2(1)(64)=128
2 (8,1)	$P2_3 = 297$	(8,0)	—	—
3 (8,0)				

Points - (0,6), (1,6), (2,6), (3,6), (4,5), (5,5), (6,4), (7,3), (8,2), (8,1), (8,0)



R)

Date
Page

(0) $P_{10} = r_y^2 - r_x^2 r_y + \frac{1}{4}(r_x^2)$

$= (6)^2 - (8)^2(6)^2 + \frac{1}{4}(8)^2$

$= 36 - (64)(36) + \frac{1}{4}(64)$

$= 36 - 384 + 16$

$P_{10} = -332$

(1) $P_{11} = P_{10} + 2r_y^2 x + r_y^2$

$= -332 + 2(36)(1) + 36$

$= -332 + 72 + 36$

$= -224$

(2) $P_{12} = P_{11} + 2r_y^2 x + r_y^2$

$= -224 + 2(36)(2) + 36$

$= -224 + 144 + 36$

$= -44$

(3) $P_{13} = P_{12} + 2r_y^2 x + r_y^2$

$= -44 + 2(36)(3) + 36$

$= -44 + 216 + 36$

$P_{13} = 208$

(4) $\cancel{= r_y^2(x + \frac{1}{2})^2 + r_x^2(y-1)^2 - r_x^2 r_y^2}$

(4) $P_{14} = P_{13} + dx - dy + r_x^2$

$= 208 + 288 - 640 + 36$

$= -108$

(5) $P_{15} = P_{13} + 2r_y^2 x + r_y^2$

$= -108 + 2(36)(5) + 36$

$P_{15} = 288$

Date
Page

(6) $P1_6 = P1_5 + dx - dy + 2x^2$

$= 288 + 432 - 512 + 36$

$P1_6 = 244$

$\boxed{P2}$

(0) $P2_0 = 2y^2(x+\frac{1}{2})^2 + 2x^2(y-1)^2 - 2x^2 2y^2$

$= (36)(7+\frac{1}{2})^2 + (64)(3-1)^2 - (64)(36)$

$P2_0 = -23$

(1) $P2_1 = P2_0 + dx - dy + 2x^2$

$= -23 + 576 - 256 + 64$

$P2_1 = 361$

(2) $P2_3 = P2_1 - dy + 2x^2$

$= 361 - 128 + 64$

$P_{23} = 297$

Date
Page

(2) Plot the points for midpoint ellipse with $r_x = 3$ & $r_y = 4$

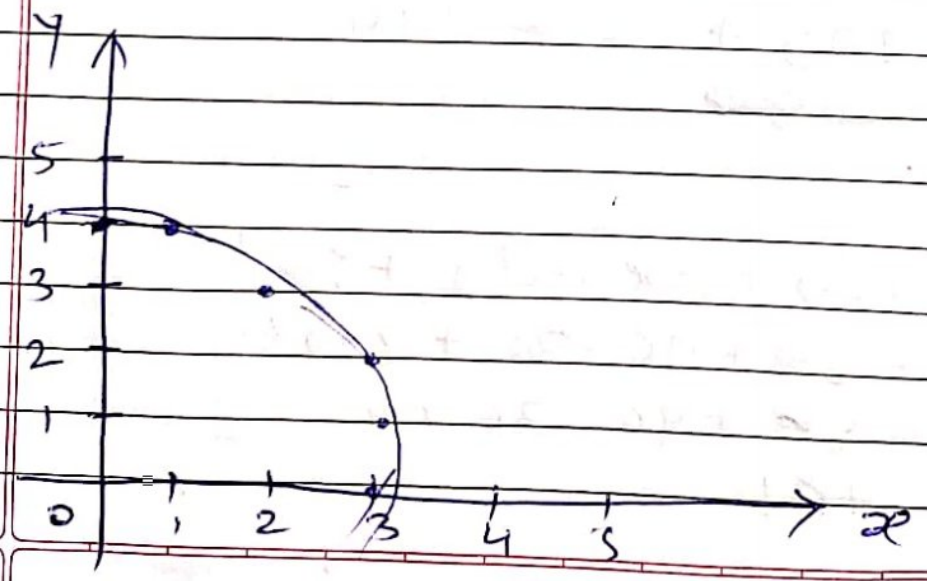
$\Rightarrow$

(x_k, y_k)	Decision parameters P1 or P2	(x_{k+1}, y_{k+1})	$2x_{k+1} r_y^2$	$2y_{k+1} r_x^2$
1. (0,4)	$P1_0 = -17.75$	(1, 4)	$2(1)(4)^2 = 32$	$2(4)(3)^2 = 72$
2. (1,4)	$P1_1 = 94.75$	(2, 3)	$2(2)(4)^2 = 64$	$2(3)(3)^2 = 54$

Here $dx > dy$ switch to Region 2

3 (2,3)	$P2_0 = -8$	(3, 2)	$2(3)(4)^2 = 96$	$2(2)(3)^2 = 36$
4 (3,2)	$P2_1 = +61$	(3, 1)	$2(3)(4)^2 = 96$	$2(1)(3)^2 = 18$
5 (3,1)	$P2_2 = 52$	(3, 0)	–	–

Points → (0, 4), (1, 4), (2, 3), (3, 2), (3, 1), (3, 0)



Date
Page

(1) $P1_0 = \delta y^2 - \delta x^2 \delta y + \frac{1}{4}(\delta x^2)$

$= (4)^2 - (3)^2(4) + \frac{1}{4}(3)^2$

$= 16 - 36 + 2.25$

$P1_0 = \cancel{-22.25} \quad -17.75$

(2) $P1_2 = P1_0 + 2\delta y^2 x + \delta y^2$

$= -17.75 + 2(4)^2(3) + (4)^2$

$= -17.75 + 32 + 16$

$P1_2 = \cancel{30.25} \quad 30.25$

(3) $\cancel{P1_3 = P1_2 + dx - dy + \delta y^2}$

$\cancel{= 30.75 + 64 - 54 + (4)^2}$

$\cancel{= 30.75 + 64 - 54 + 16}$

$\cancel{P1_3 = 115.75}$

(4) $P2_0 = \delta y^2 (x + \frac{1}{2})^2 + \delta x^2 (y-1)^2 - \delta x^2 \delta y^2$

$= (4)^2(3 + \frac{1}{2})^2 + (3)^2(3-1)^2 - (3)^2(4)^2$

$= 16(\frac{4+1}{2})^2 + (9)(2)^2 - (9)(16)$

$= 100 + 36 - 144$

$P2_0 = -8$

(5) $P2_1 = P2_0 + dx - dy + \delta x^2$

$= -8 + 96 - 36 + (3)^2$

$= -8 + 96 - 36 + 9$

$P2_1 = +61$

Date
Page

(6) $P_{22} = P_{21} - dy + 8x^2$

$= 61 - 18 + (3)^2$

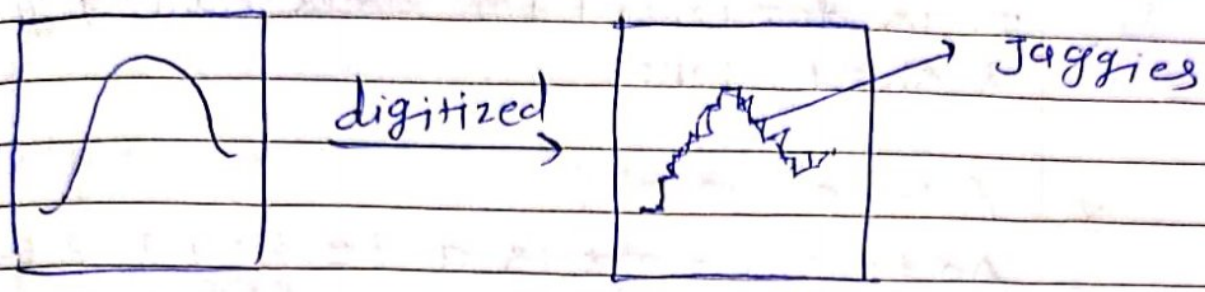
$= 61 - 18 + 9$

$P_{22} = 52$

Date
Page

Aliasing & Anti Aliasing -

Aliasing -



Digital line - aliasing occurs

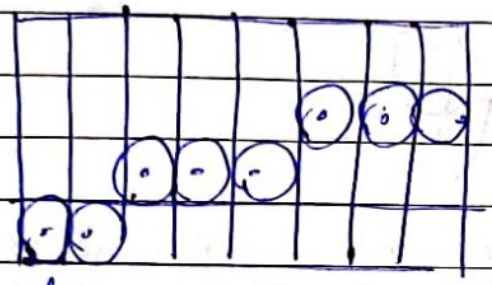


fig. Aliasing effect

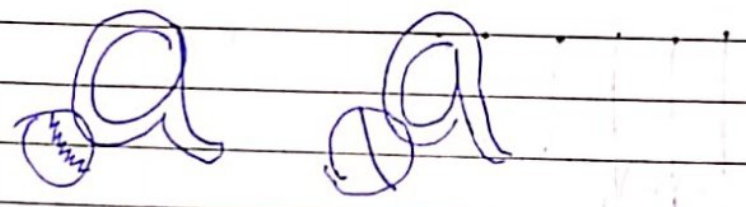
- In computer graphics the process by which smooth curves & other lines become jagged because the resolution of the graphics device or file is not high enough to represent a smooth curve
- In the line drawing algorithm, we have seen that all rasterized locations do not match with the true line & we have to select the optimum raster locations to represent a straight line.
- This problem is severe in low resolution screens.

Date
Page

- In such screens line appears like a stair-step as shown in the figure
- This effect is known as 'aliasing'
- It is dominant for lines having gentle & short slopes

* Anti aliasing

- Anti aliasing is a technique used in CG to remove the aliasing effect



- The aliasing effect can be reduced by adjusting the intensities of the pixels along the line.
- The process of adjusting intensities of the pixels along the line to minimize the effect of aliasing is called as antialiasing.
- The aliasing effect can be minimized by increasing resolution of the raster display. But it comes at the price of cost of memory, band width of memory & scan conversion line.

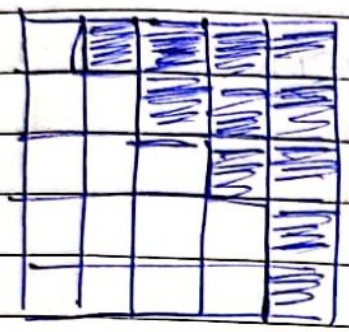
Anti aliasing Methods -

1. High resolution display
2. Post filtering (Super sampling)
3. Pre-filtering (Area sampling)

Date
Page

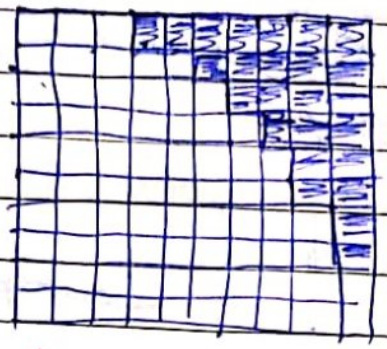
① High Resolution Display -

- Doubling resolution in x & y coordinates
- This method only lessens the problem
- costs 4 times memory, memory band width & scan conversion time.



Low resolution image

Staircase effect is more



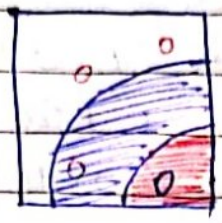
High resolution image

Staircase effect is less

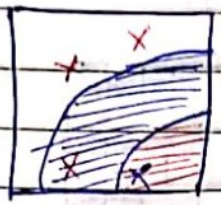
- cost conversion time increases as memory ~~increases~~ increases

② Post Filtering (Super Sampling)

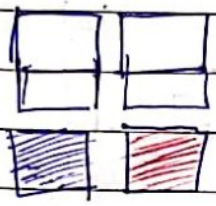
- For each displayed pixel
- It takes several samples from the scene & computes an average of the samples to determine the pixel's color.
- Eg - Suppose the display resolution is 512 x 512. Sampling at 3 times the width & 3 times the height of the display resolution would yield 1536 x 1536 samples
- The color of each pixel in the rendered image will be an average of several samples



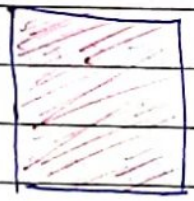
pixel with sampling positions



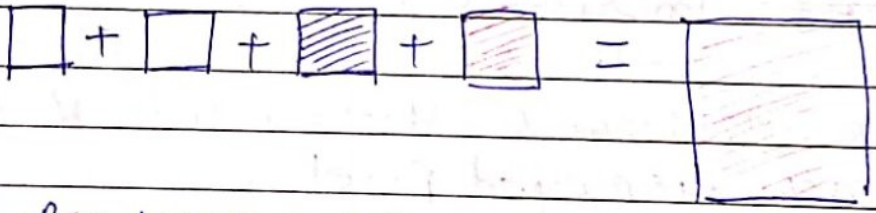
Pixel with sample position



Sampled colours



Average = displayed colour

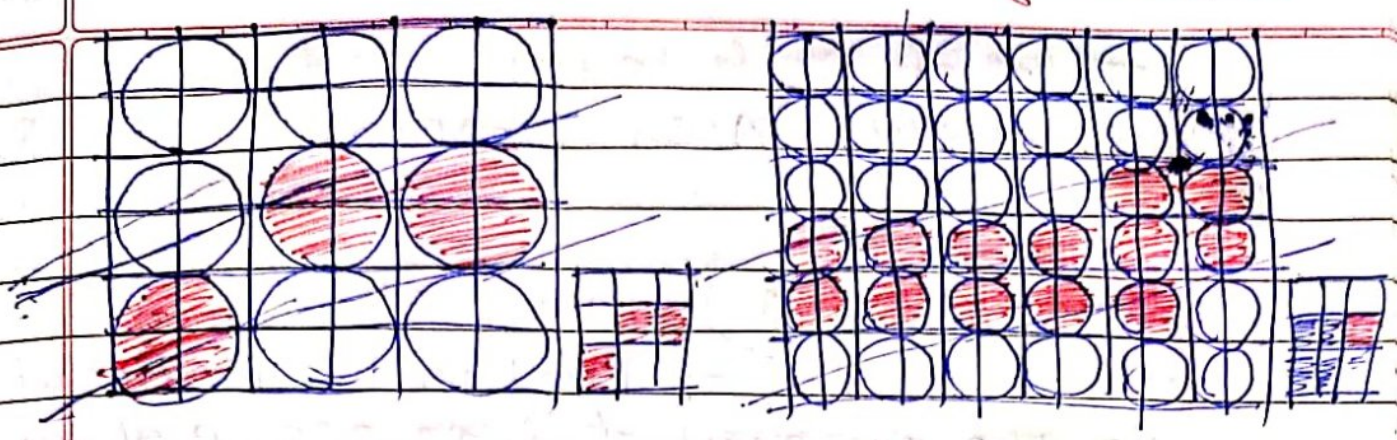


(Brighter Pitch)

Resulting color

* calculating the end color value

eg ② - Divide pixel up into sub-pixels : 2×2, 3×3 4×4 etc
- sub pixel is colored if inside line
- pixel color is the average of its sub pixel colors
- Easy to implement (in S/w & H/w)



No antialiasing

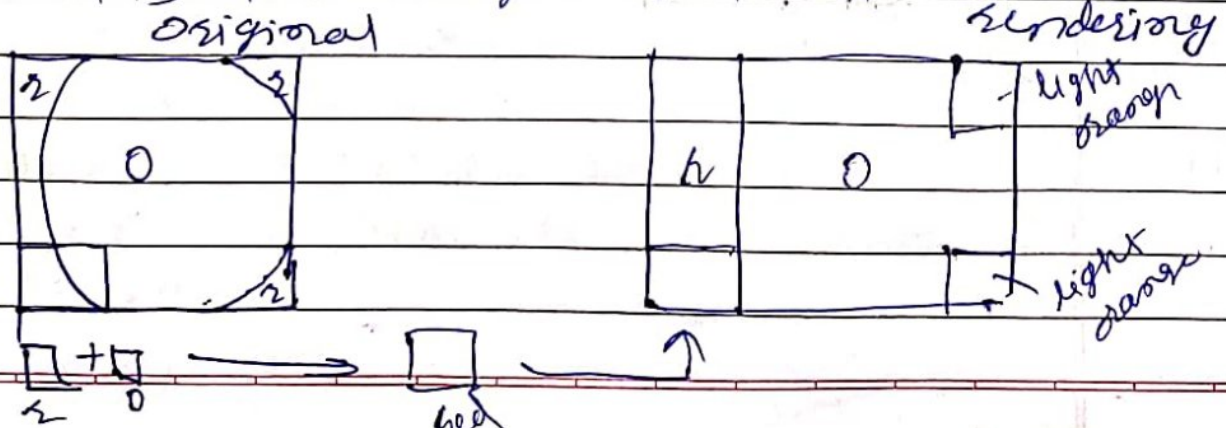
Antialiasing (2x2 Super Sampling)

③ Pre Filtering (Area Sampling)

- Pre Fitering methods treat a pixel as an area & compute pixel color based on the overlap of the scene's objects with a pixel's area.
- The original scene is a filled orange circle on a red background
- All of the pixels inside the circle are 100 % orange.
- All of the pixels on the boundary of the circle have some area that is red & some area that is orange.

40% of Highlighted pixel is orange & 60% of its area is red.

computed color for the highlighted pixel is 40% orange & 60% red



Date
Page

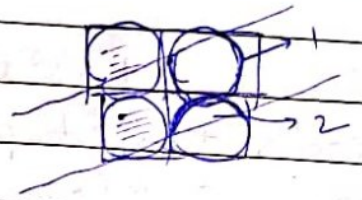
Types of Pre filtering

(1) Unweighted Area Sampling

(2)

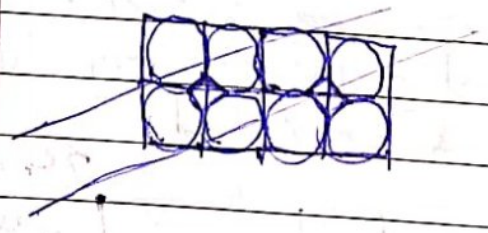
Unweighted Area Sampling

The intensity of pixel is proportional to the amount of line occupied by the pixel area.



Weighted Area Sampling -

- Equal areas contribute unequally i.e. a small area closer to the pixel center has greater intensity than does one at a greater distance.
- The intensity of the pixel is dependent on the line area occupied & the distance of area from the pixel's center



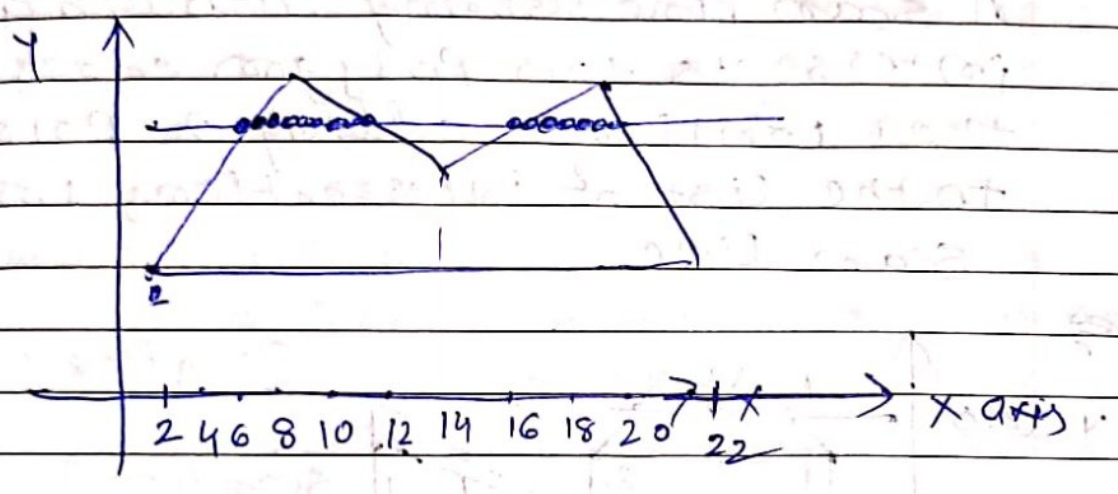
Date
Page

* Filled Area Primitive

1. Scan line polygon Fill algorithm
2. Inside Outside tests
3. Boundary Fill algorithm
4. Flood Fill algorithm

1. Scan line Polygon Fill algorithm

Scan line polygon filling algorithm is used for solid color filling in polygons



* Steps to perform

1. Find the intersections of the scan line with all edges of the polygon
2. Sort the intersections by increasing X co-ordinate i.e. from left to right

eg. intersection points are
6, 12, 16, 20
Sort the intersection points
6, 12, 16, 20

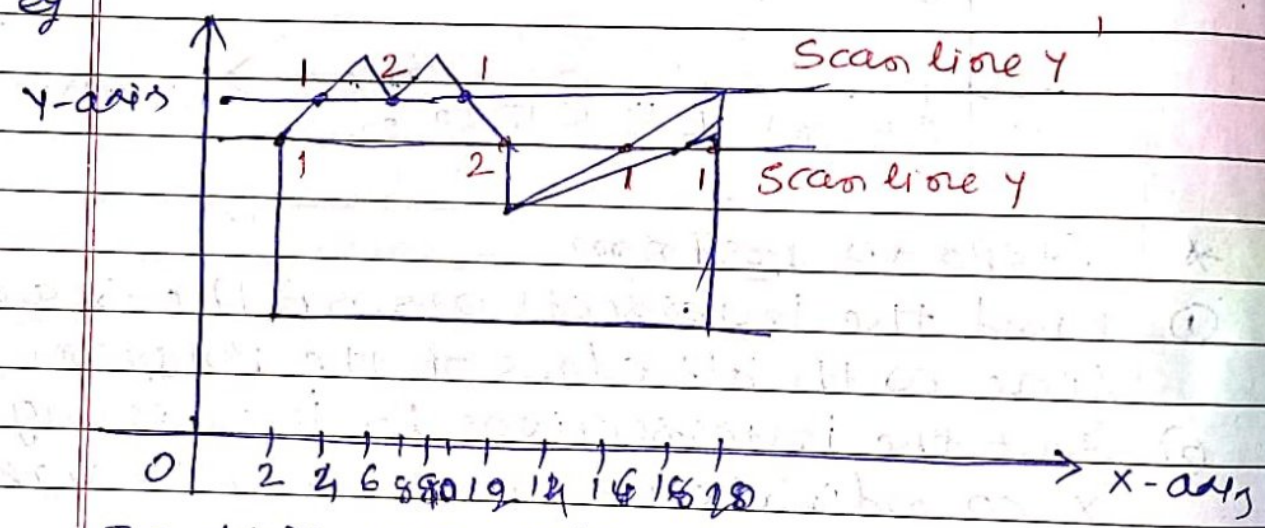
3. Make pairs of the intersections & fill in color within all the pixels inside the pair

eg. Make the pairs as follows
(6,12) & (16,20)
Now fill all pixels with color inside the pair

Special cases

- Some scan line intersections at polygon vertices require special handling
- A scan line passing through a vertex intersects two polygon edges at that positions, adding 2 points to the list of intersections for the scan line.

eg



- In this example scan line y & scan line y' both pass through an vertex or an edge endpoint. Now in case of scan line y' the scan line is intersecting 4 edges i.e. even no. of edges & also passing through a vertex / endpoint
- Also both the edges that are connected to the vertex are on SAME SIDE of the

Date
Page

scan line.
So we need to count the vertex / endpoint Twice so that we can make pairs of intersection points as.
(4, 8) & (8, 12)
In scan line y the scan line is intersecting with 5 different edges i.e. ODD no. & also passing through a vertex / endpoint.
So in this case we need to do some extra processing i.e. we need to check if the 2 edges at the endpoint through which the scan line is passing are they on opposite sides or on same sides?
In case of scan line y' they are on same sides & in case of scan line y both edges at the vertex are on opposite sides.
So Now we count the vertex as a single intersection point
Now we just need to sort the intersection points & make pairs of them & fill all pixels which lie inside the pair
eg (2, 12) & (16, 20)

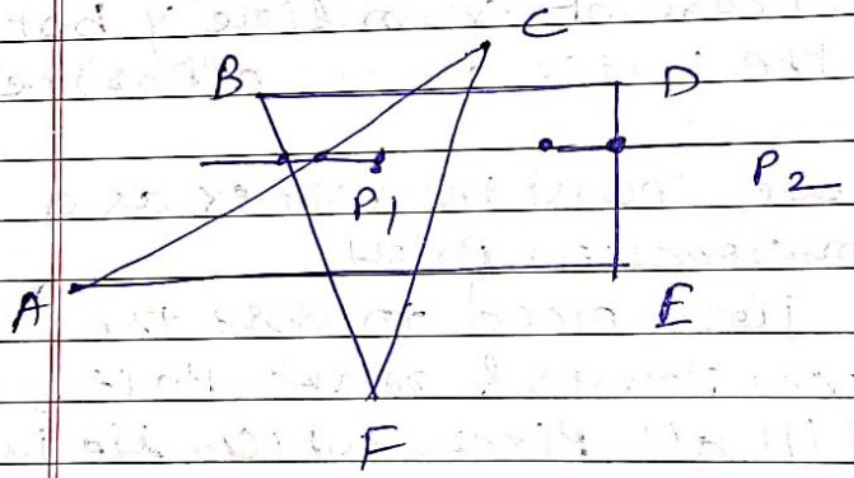
Inside outside Tests –
(counting number method)

- A test used in CG to determine if a point is inside or outside of a polygon.
- While filling an object we often need to identify whether particular point is inside the object or outside it

Methods

1. Even-odd Rule / odd parity Rule
2. Non zero winding rule

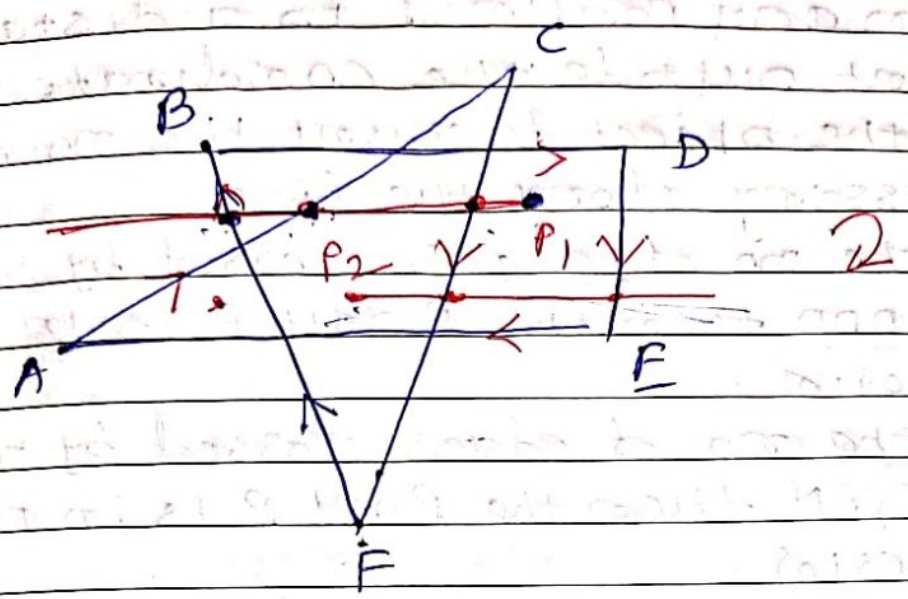
1. Even-odd Rule



No. of intersect
odd → inside the polygon
even → outside the polygon

in fig P_1 is having 2 intersect po so that is outside the polygon
P_2 is having 1 intersect point so that is inside the polygon

② Non zero winding Rule



wind
left to Right → −1
Right to left → +1

P_1 intersect at 3 edges
$P_1 = 0 + 1 - 1 - 1 = -1$

$\boxed{\text{Initial value of } P_1 \text{ is } = 0}$

if sum = 0 (outside)
sum = non zero (inside) the polygon

∴ P_1 is inside the polygon

$P_2 = 0 + 1 + 1 = 2$

∴ P_2 is inside the polygon

Date
Page

Even odd rule :-

- In even odd rule we draw a line from any position P to a distant point outside the coordinate extent of the object & count the no. of edges crossing along the line.
- If the no. of edges crossed by the line is ODD then the point P is in the interior.
- If the no. of edges crossed by the line is EVEN then the point P is in the interior.

condition -

You need to make sure that the line drawn from the point does not intersect a vertex or endpoint.

Non zero winding Number

- In this method we need to know the direction of edge in the polygon i.e. whether the edge is clockwise or counter-clockwise.
- The winding number is number of times the polygon edges wind around a particular point in the counterclockwise direction.

How to apply

(1) To ~~keep~~ apply non zero winding rule, first we keep the initial value of winding number = 0

Date
Page

(2) Then we imagine drawing a line from point P to outside the polygon which does not pass through any vertex.

(3) Now we add 1 to the winding no. every time we intersect a polygon edge that crosses the line from right to left & we subtract 1 every time we intersect an edge that crosses from left to right.

Note

left to Right (clockwise) → −1

Right to left (counterclockwise) → +1

Date
Page

③ Boundary Fill algorithm

- In boundary fill algorithm the basic concept is filling the color in closed area by starting at a point inside a region & paint the interior outward towards the boundary
- one requirement for boundary fill algorithm is that the boundary has to have a single color.

Working

- you start from a point inside the region & fill the color interior outwards towards the boundary pixel by pixel.
- The Process is that you check the default color of the pixel before filling, if the color is not boundary color, then you fill it with the fill color & move to the next pixel & check for the same criteria till you encounter the boundary colored pixel or the boundary

Methods

① 4 - connected pixel
② 8 - connected pixel

Date
Page

① 4- connected Pixel

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

Step 1 First initialize the 4 values namely X, Y, Fill color & default color
Where X & Y are coordinate positions of the initial interior pixel we start the boundary fill algorithm & Fill color is the color we want to fill & default color is the default color of the interior pixel.

Step 2 Define the value of the boundary pixel color or Boundary color.

Step 3 Now check if the current pixel is of default color & if yes then repeat step 4 & 5 till the boundary pixels are reached

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

Step 5 Repeat step 3 & 4 for the neighbouring 4 pixels

Step 6 Exit

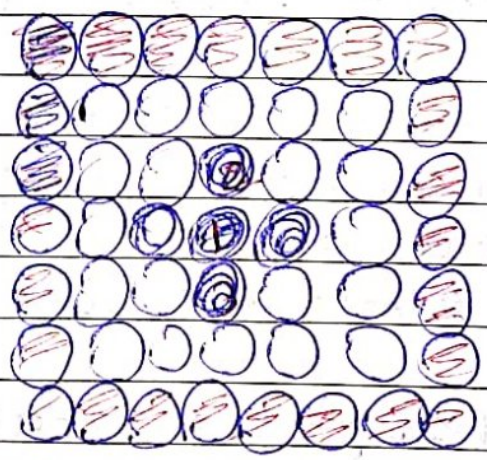
Date
Page

Code for 4 connected

```
void boundaryFill4 (int x, int y, int fill,
                    int boundary, int default)
{
int current;
current = getPixel(x, y);
if ((current != boundary) &&
    (current != default))
{
  setPixel (x, y, fill);
(right)  boundaryFill4 (x+1, y, fill, default);
(left)   boundaryFill4 (x-1, y, fill, default);
(above)  boundaryFill4 (x, y+1, fill, default);
(below)  boundaryFill4 (x, y-1, fill, default)
```

Problem in 4-connected:

There is problem with this technique that, it cannot fill all the pixels



Date
Page

8-connected pixel

To solve the problems that can occur in 4 connected pixel method, a new method was introduced called 8-connected pixel so in this method we instead of filling just 4 adjacent pixels we fill also the adjacent diagonal pixels positions such as
(x+1, (y+1)

code for 8-connected -

```
void boundaryFill8 (int x, int y, int fill,
                    int boundary, int default)

  {
   int current;
   current = getPixel(x, y);
   if ((current != boundary) && (current =
                                  default))
   {
    setPixel(x, y, fill);
    boundaryFill8 (x+1, y, fill, default);
    boundaryFill8 (x-1, y, fill, default);
                  (x, y+1, fill, default);
                  (x, y-1, fill, default);
(top right)       (x+1, y+1, fill, default);
(bottom right     (x+1, y-1, fill, default);
(left top)        (x-1, y+1, fill, default);
(left bottom)     (x-1, y-1, fill, default);
   }
  }
```

diagonal Pixel

(4) Flood Fill Algorithm

- Flood Fill algorithm is useful in cases where there is no single color boundary for polygon i.e. the boundary has multiple colors.
- In this algorithm instead of Filling color till you encounter a specific boundary color you just fill the pixels with default color.
- It is used in the "bucket" fill tool of Paint programs to fill connected pixels, similarly-colored areas with a different color.

Algorithm -

Flood-fill (node, target-color, replacement-color);

1. If target-color = replacement-color, return
2. If the color of node is not equal to target-color, return
3. Set the color of node to replacement-color.

Date
Page

4. Perform Flood fill (one step to the south of node, target-color, replacement-color)
Perform Flood Fill (one step to the north of node, target-color, replacement-color)
Perform flood fill (one step to the west of node, target color, replacement color)
Perform Flood fill (one step to the east of node, target-color, replacement-color)
5. return

Methods of Implementation

There are 2 methods
1. 4-connected pixel
2. 8-connected Pixel

(1) 4-connected Pixel

- In 4-connected Pixel method you check 4 Pixels adjacent to the current Pixel, namely towards the North (top) South (bottom) west (left) East (right)
- So you fill the area with 4-connected pixel method by following these steps

Step 1.

First initialize the 4 values namely x, y, fill color & Default color. Where x and y are co-ordinate Positions of the initial interior pixel we start the flood fill algorithm & fill color is the color we want to

Date
Page

fill & Default color is the default color of the interior pixel.

Step 2

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

Step 3

Set the color of node to replacement color

Step 4

set the color of node to the south of node to replacement - color

Step 5

Set the color of node to the north of node to replacement - color

Step 6

set the color of node to the east of node to replacement - color

Step 7

Set the color of node to the west of node to replacement - color

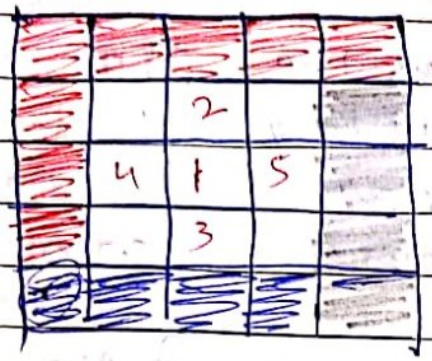
Step 8

Exit

Date
Page

problem in 4-connected

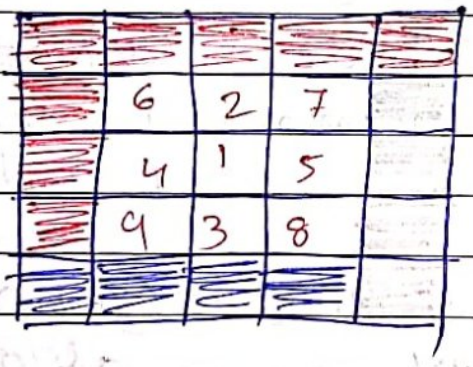
There is problem with this techniqe that 4 connected pixels techniqe



old color - white
new color - green
current = old color
pixel (1)
white = white.

(2) 8-connected Pixel -

- To solve the problems that can occure in 4-connected pixel method, a new method was introduced called 8-connected pixel
- So in this method we instead of filling just 4 adjacent pixels we fill also the adjacent diagonal pixels positions, such as (x+1, y+1)



Date
Page

Difference between Flood fill Algorithm and Boundary fill Algorithm

Flood Fill Algorithm	Boundary Fill Algorithm
① It can process the image containing more than one boundary colours	① It can process the image containing single boundary colour.
② Algorithm is comparatively slower than the boundary fill algorithm	② Algorithm is faster than the flood fill algorithm
③ In this a random colour can be used to paint the interior portion then the old one is replaced with a new one	③ In boundary fill algorithm interior points are continuously searching for the boundary color.
④ It requires huge amount of memory	④ Memory consumption is relatively low in Boundary fill algo.
⑤ Algorithm is simple & efficient	⑤ complexity of algorithm is high

Algorithm 8-connected

```
void flood-fill (int x, int y, int old color,
                 int new color)
{
  if (getPixel(x,y) == old color)
  {
    PutPixel (x, y, newcolor);
    floodfill (x+1, y, oldcolor, newcolor);
    flood-fill (x-1, y, old color, newcolor);
    flood-fill (x, y-1, old color, newcolor);
    flood-fill (x, y+1, old color, newcolor);
    flood-fill (x+1, y+1, old color, newcolor);
    flood-fill (x+1, y-1, old color, newcolor);
    flood-fill (x-1, y+1, old color, newcolor);
    flood-fill (x-1, y-1, old color, newcolor);
  }
}
```

Algorithm 4-connected

```
void flood-fill (int x, int y, int old color,
                 int new color)

{
  if (getPixel (x,y) == old color)
  {
    PutPixel (x, y, newcolor)
    flood-fill (x+1, y, oldcolor, newcolor);
    flood-fill (x-1, y, old color, newcolor);
    flood-fill (x, y-1, old color, newcolor);
    flood-fill (x, y+1, old color, newcolor);
  }
}
```

Date
Page

MODULE 3

2 Dimentional Geometric Transformations

* Basic Transformations

Transformation

- Transformation means changes in orientation, size & shape of the image or object
- They are used to change position the object, to change the shape of the object & even to change how something is viewed
- So we can modify the image by performing some basic transformation as

Basic transformations

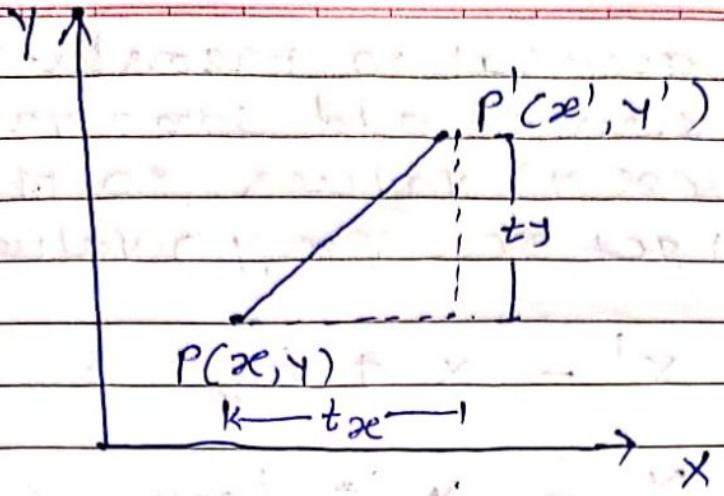
1. Scaling Transformation
2. Rotation Transformation
3. Translation Transformation.

& other desired transformations are

1. Reflection
2. Shearing

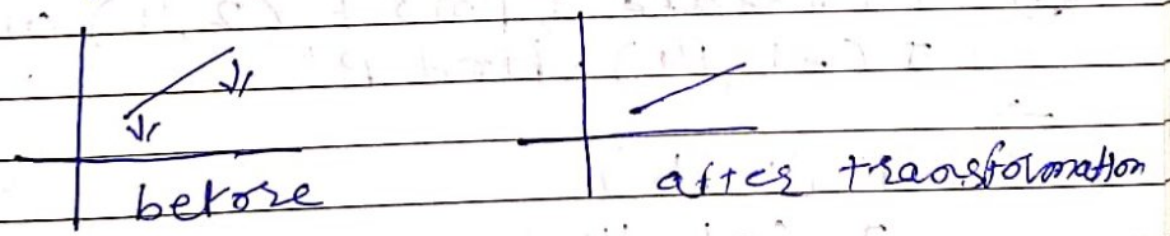
1. Translation Transformation

- Change in position of the image
- It is repositioning an object along straight line path from one co-ordinate location to another
- Translation is rigid body transformation that moves an object without deformation

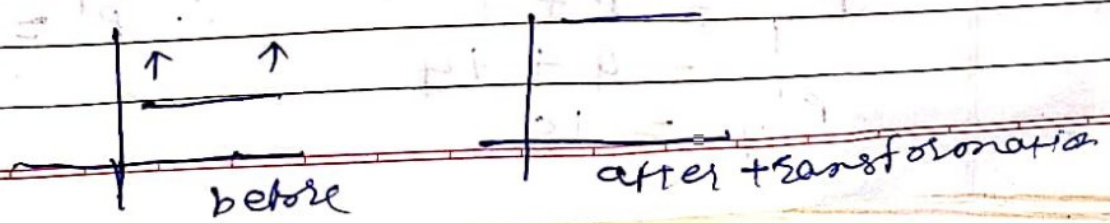


$(t_x, t_y) \rightarrow$ Translation vector or shift vector

- Moving the whole image is called translation. We can easily shift the image by adding some value to all points of image by which we want to shift the image.
- If we want to shift the image down by 5 units then we have to add -5 to all y values of that image. Same thing for x values



- Similarly if we want to shift image up by 5 units, then add 5 to all y values of image. Same thing for x values



Date
Page

- In general to translate the image we have to add some value to old (x, y) values, so that we will get new (x, y) values

$$\therefore x' = x + t_x$$

$$y' = y + t_y$$

Where $t_x \rightarrow$ Translation factor for x-axis

$t_y \rightarrow$ Translation factor for y-axis

The matrix representation will be

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

Example -

① Translate a point $(2, 4)$ where $T(-1, 14)$. Find P'

$\Rightarrow$

$P\ (2, 4)$

$T\ (-1, 14)$

$P'_x = P + T$

$= 2 + (-1)$

$P'_x = +1$

$P'_y = P + T$

$= 4 + 14$

$P'_y = 18$

$\therefore P' = (1, 18)$

Date
Page

(2) Translate a polygon with coordinates A (2,7), B (7,10), C (10,2) by 3 units in x-direction & 4 units in y-direction

$\Rightarrow$

$t_x = 3$, $t_y = 4$

1) $A' = A + T$

$= \begin{bmatrix} 2 \\ 7 \end{bmatrix} + \begin{bmatrix} 3 \\ 4 \end{bmatrix}$

$A' = \begin{bmatrix} 5 \\ 11 \end{bmatrix}$

$A' = (5, 11)$

2) $B' = B + T$

$= \begin{bmatrix} 7 \\ 10 \end{bmatrix} \begin{bmatrix} 3 \\ 4 \end{bmatrix}$

$= \begin{bmatrix} 10 \\ 14 \end{bmatrix}$

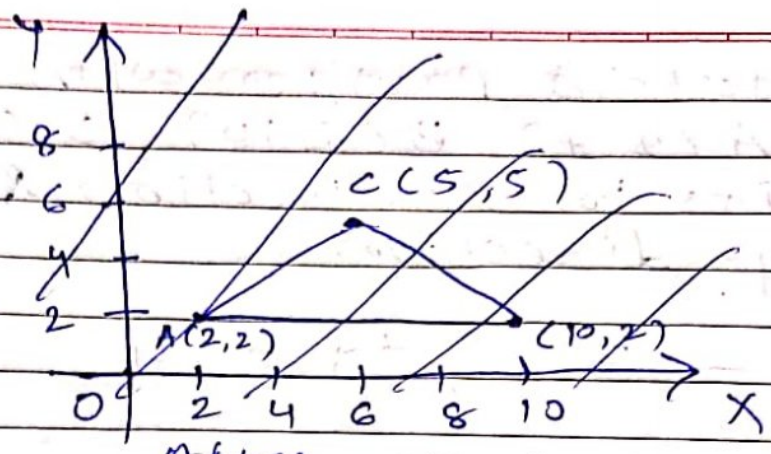
$B' = (10, 14)$

3) $C' = C + T$

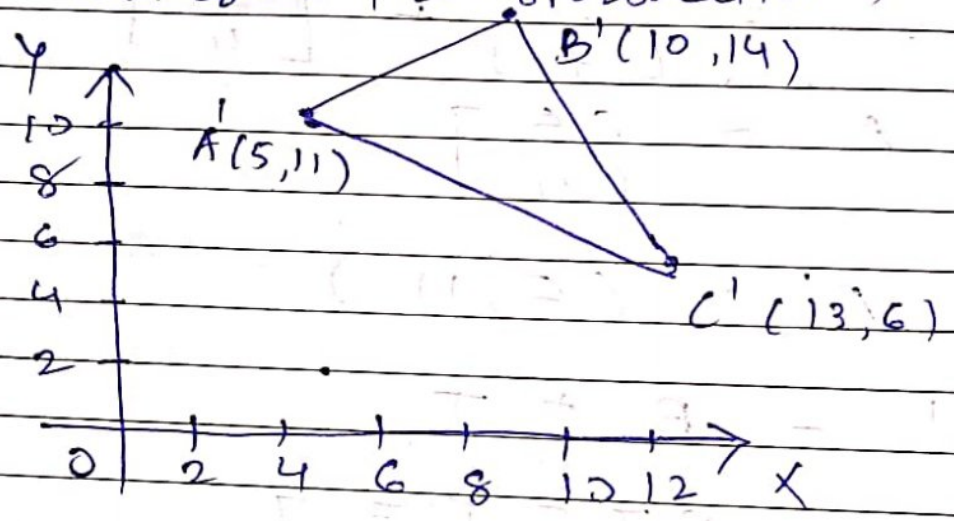
$= \begin{bmatrix} 10 \\ 2 \end{bmatrix} \begin{bmatrix} 3 \\ 4 \end{bmatrix}$

$= \begin{bmatrix} 13 \\ 6 \end{bmatrix}$

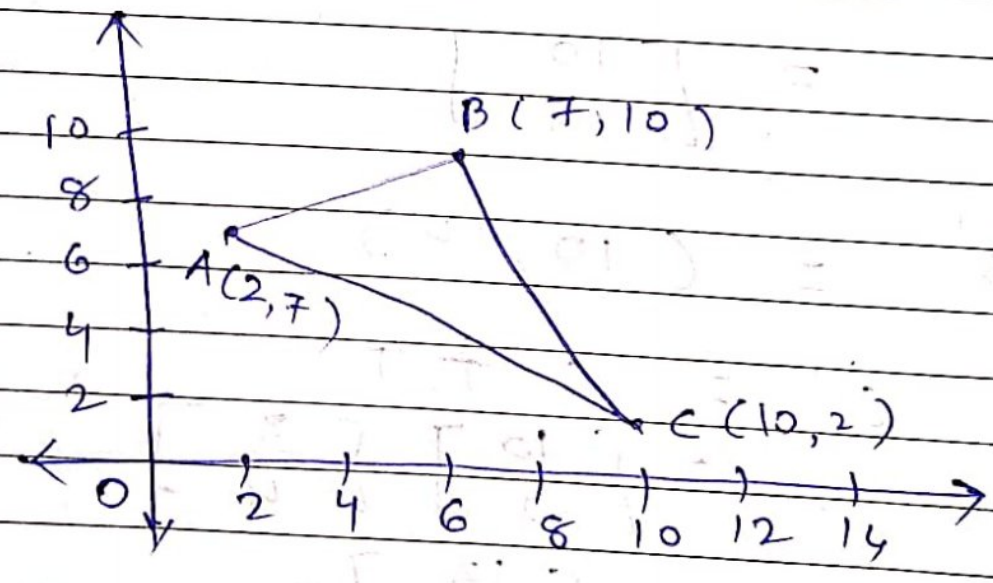
$C' = (13, 6)$



After
~~Before~~ Transformation)



Before Transformation



Date
Page

(2) A(2, 2), B(10, 2), C(5, 5) Translate the triangle with $dx = 5$, $dy = 6$

$\Rightarrow$ $dx = 5$, $dy = 6$

(1) $A' = A + T$

$= \begin{bmatrix} 2 \\ 2 \end{bmatrix} + \begin{bmatrix} 5 \\ 6 \end{bmatrix}$

$A' = \begin{bmatrix} 7 \\ 8 \end{bmatrix}$

$A' = (7, 8)$

(2) $B' = B + T$

$= \begin{bmatrix} 10 \\ 2 \end{bmatrix} + \begin{bmatrix} 5 \\ 6 \end{bmatrix}$

$= \begin{bmatrix} 15 \\ 8 \end{bmatrix}$

$B' = (15, 8)$

(3) $C' = C + T$

$= \begin{bmatrix} 05 \\ 5 \end{bmatrix} + \begin{bmatrix} 5 \\ 6 \end{bmatrix}$

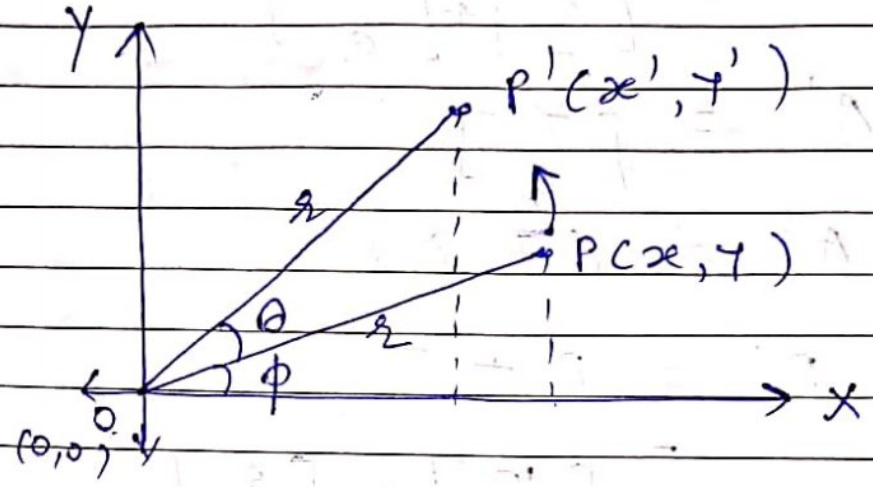
$= \begin{bmatrix} 10 \\ 11 \end{bmatrix}$

$C' = (10, 11)$

Date
Page

(2) Rotation Transformation

- It is a process of changing the angle of the object.
- It can be done in clockwise or anticlockwise.
- Let coordinate of P in the polar form



ϕ - Is the angle between x-axis & point P

r - Is the distance from origin to P

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

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

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

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

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

Matrix Representation

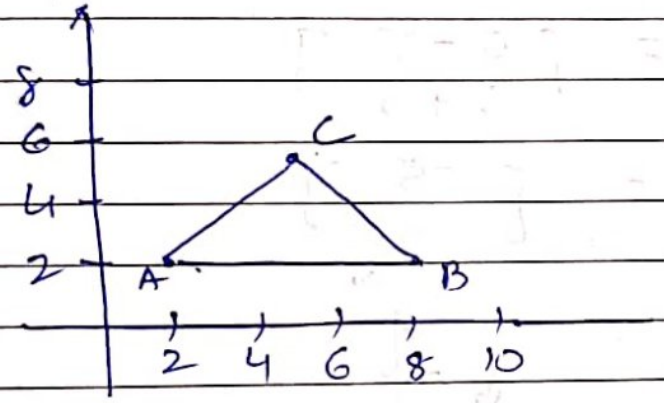
$P' = RP$

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

Example

① One triangle is given $A(2,2)$, $B(8,2)$, $C(5,5)$. Rotate the triangle by $90°$.

⇒



Here $\theta = 90°$, $P' = RP$

$$R = \begin{bmatrix} \cos 90 & -\sin 90 \\ \sin 90 & \cos 90 \end{bmatrix}$$

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

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

Date
Page

$$B' = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 8 \\ 2 \end{bmatrix}$$

$$= \begin{bmatrix} 0\times 8 + (-1\times 2) \\ 1\times 8 + (0\times 2) \end{bmatrix}$$

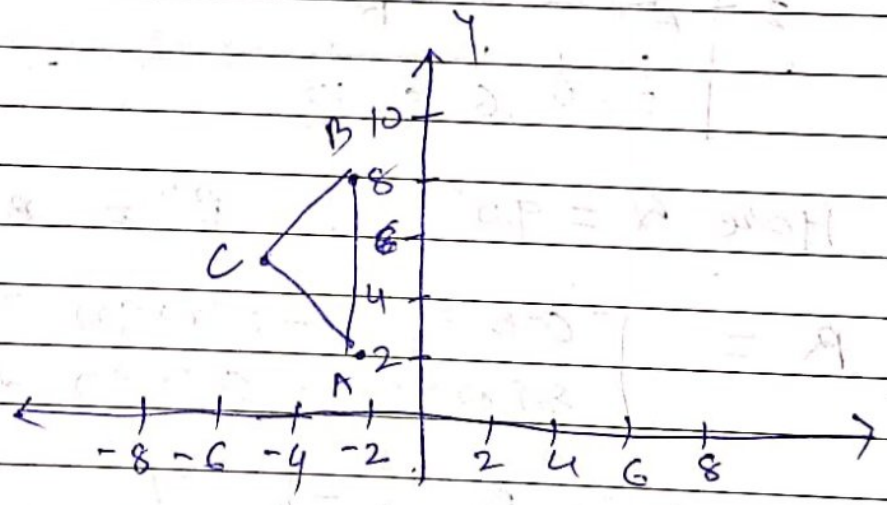
$$B' = \begin{bmatrix} -2 \\ 8 \end{bmatrix}$$

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

$$= \begin{bmatrix} 0\times 5 + (-1\times 5) \\ 1\times 5 + (0\times 5) \end{bmatrix}$$

$$= \begin{bmatrix} 0-5 \\ 5+0 \end{bmatrix}$$

$$C' = \begin{bmatrix} -5 \\ 5 \end{bmatrix}$$



After Rotation

(2) A Point (4, 3) is rotated in counter clockwise direction by the angle of 45°. Find the rotation matrix R & the resultant points.

$\Rightarrow$

$$P' = RP$$

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

$$= \begin{bmatrix} \cos 45 & -\sin 45 \\ \sin 45 & \cos 45 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$

$$= \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{4}{\sqrt{2}} - \frac{3}{\sqrt{2}} \\ \frac{4}{\sqrt{2}} + \frac{3}{\sqrt{2}} \end{bmatrix}$$

Date
Page

(3) Scaling Transformation :-

- Scaling transformation alters the size of an object
- Scaling of a polygon requires multiplying the coordinate value of each vertex by the scaling factor to get new coordinate value
- In the scaling process object is either expanded or compressed.

$$x' = x \cdot \delta_x$$
$$y' = y \cdot \delta_y$$

Where δ_x & δ_y are scaling factors.

Scaling factor :

δ_x & $\delta_y > 1$ object enlarged

δ_x & $\delta_y < 1$ object compressed

$\delta_x = \delta_y$ Uniform Scaling

In matrix form it can be represented by

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

$$P' = SP$$

Date
Page

① Scale a Polygon with coordinates A(2,5), B(7,10); C(10,2) by 2 units in x-direction & 3 units in y direction

⇒ Here $\delta x = 2$ & $\delta y = 3$

$$P' = SP$$

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

① A (2, 5)

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 2 \\ 5 \end{bmatrix}$$

$$A' = \begin{bmatrix} 4 \\ 15 \end{bmatrix} \quad \therefore A' = (4, 15)$$

② B (7, 10)

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 7 \\ 10 \end{bmatrix}$$

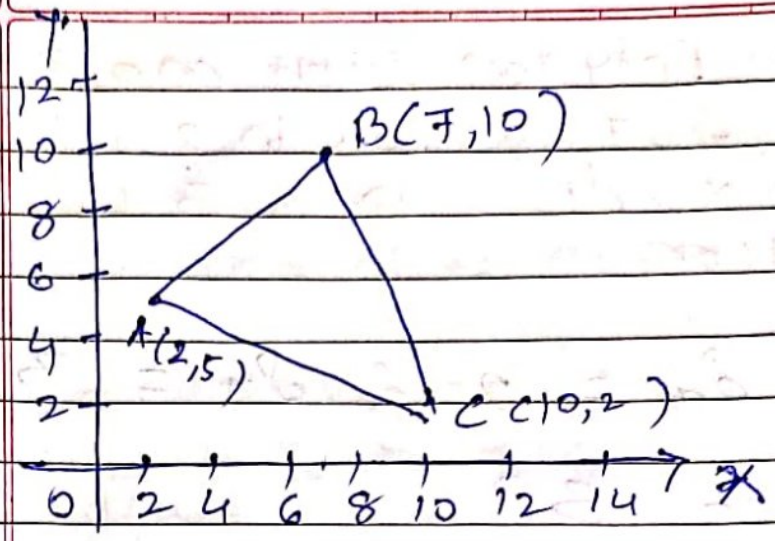
$$B' = \begin{bmatrix} 14 \\ 30 \end{bmatrix} \quad \therefore B' = (14, 30)$$

③ C (10, 2)

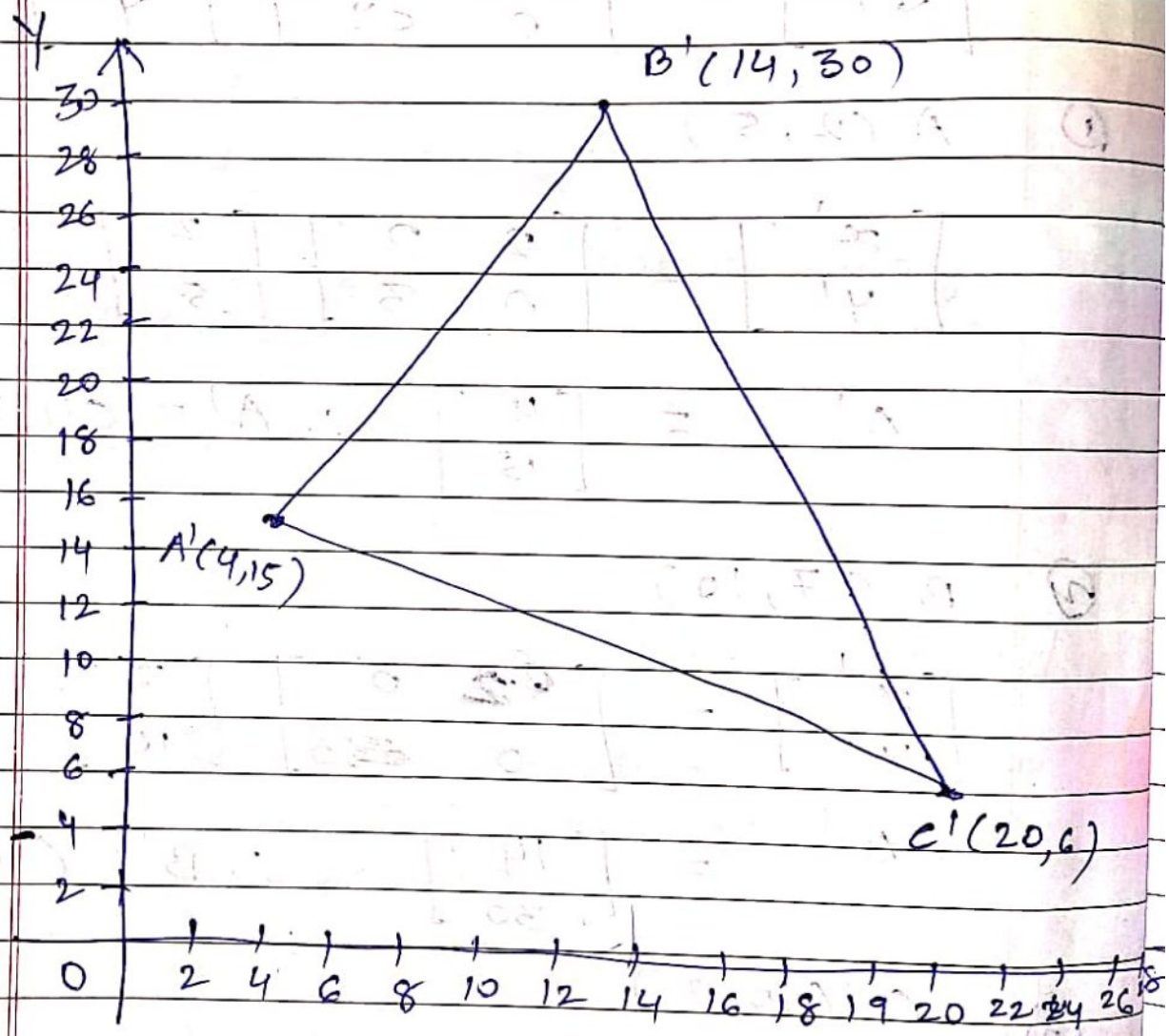
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 10 \\ 2 \end{bmatrix}$$

$$C' = \begin{bmatrix} 20 \\ 6 \end{bmatrix} \quad \therefore C' = (20, 6)$$

Date
Page



Before scaling



Date
Page

(2) Translate the square ABCD A (0,0), B (3,0), C (3,3) & D (0,3) by 2 units in both direction & then scale it by 1.5 units in x direction & 0.5 units in y direction. Determine the resultant coordinates of polygon

⇒ (1) Translation

$t_x = 2$ & $t_y = 2$

$$A' = A + T$$
$$= \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$
$$A' = (2, 2)$$

$$B' = B + T$$
$$= \begin{bmatrix} 3 \\ 0 \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$
$$B' = (5, 2)$$

$$C' = C + T$$
$$= \begin{bmatrix} 3 \\ 3 \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$
$$C' = (5, 5)$$

$$D' = D + T$$
$$= \begin{bmatrix} 0 \\ 3 \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$
$$D' = (2, 5)$$

② Scaling

$d_x = 1.5$ and $d_y = 0.5$

$P' = SP$

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

① $A'(2,2)$

$$A'' = \begin{bmatrix} 1.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

$$= \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

$A'' = (3,1)$

② $B'(5,2)$

$$B'' = \begin{bmatrix} 1.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} 5 \\ 2 \end{bmatrix}$$

$$= \begin{bmatrix} 7.5 \\ 1 \end{bmatrix}$$

$B'' = (7.5, 1)$

③ $C' = (5,5)$

$$= \begin{bmatrix} 1.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} 5 \\ 5 \end{bmatrix}$$

$$= \begin{bmatrix} 7.5 \\ 2.5 \end{bmatrix}$$

$C' = (7.5, 2.5)$

Date
Page

(4) $D'' = (2, 5)$

$$D'' = \begin{bmatrix} 1.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} 2 \\ 5 \end{bmatrix}$$

$$= \begin{bmatrix} 3 \\ 2.5 \end{bmatrix}$$

$D'' = (3, 2.5)$

(3) Magnify the triangle with vertices A(0,0), B(1,1), C(5,2) to twice its size while keeping (5,2) fixed

$\Rightarrow$ $\Delta^{le} \xrightarrow{\text{scaling}} \Delta^{le} \xrightarrow{\text{translate}} \Delta^{le}$

(1) <u>Scaling</u>

$d_x = 2, \; d_y = 2$

$P' = SP$

(1) $A(0, 0)$

$$A' = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

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

$A' = (0, 0)$

(2) $B = (1, 1)$

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

$$= \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

$B' = (2, 2)$

Date
Page

$C = (5, 2)$

(3) $C' = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 5 \\ 2 \end{bmatrix}$

$= \begin{bmatrix} 10 \\ 4 \end{bmatrix}$

$C' = (10, 4)$

(2) <u>Translation</u>

Since the point $C(5, 2)$ should remain fixed we have to apply translation with

$t_x = -5$ & $t_y = -2$

$A'(0, 0)$

(1) $A'' = A' + T$

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

$= \begin{bmatrix} -5 \\ -2 \end{bmatrix}$

$A'' = (-5, -2)$

(2) $B'(2, 2)$

$B'' = B' + T$

$= \begin{bmatrix} 2 \\ 2 \end{bmatrix} + \begin{bmatrix} -5 \\ -2 \end{bmatrix}$

$= \begin{bmatrix} -3 \\ 0 \end{bmatrix}$

$B'' = (-3, 0)$

Date
Page

(3) $C' = (10, 4)$

$C'' = C' + T$

$= \begin{bmatrix} 10 \\ 4 \end{bmatrix} + \begin{bmatrix} -5 \\ -2 \end{bmatrix}$

$= \begin{bmatrix} 5 \\ 2 \end{bmatrix}$

$C'' = (5, 2)$

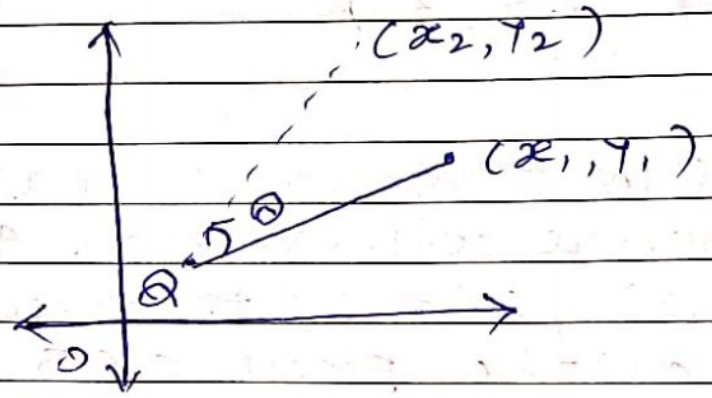
(4) Desire the transformation matrix to magnify the triangle with vertices A (0,0), B (1,2), C (3,2) to twice its size so that the point C (3,2) remain fixed.

⇒ Answer $A''(-3, -2)$
$B''(-1, 2)$
$C''(3, 2)$

Date
Page

* Homogeneous co-ordinate System

- We have seen matrices for rotation about origin in clockwise & anticlockwise direction
- But suppose we want to rotate a point (x, y) with respect to a point other than the origin i.e. Q as shown in fig



- Then in that case we can not use our rotation matrices, because they are considering reference point as origin
- So to handle such situations we have to use series of transformation i.e. we have to first translate that image until the centre of rotation becomes origin.
- Now we can perform rotation with respect to origin by angle θ in clockwise or in anticlockwise direction. But it doesn't finish our job, we have to again translate this new image back to its position by shifting the image with

same amount which we have used when we shifted image to origin
So here we have to use 3 transformations
① Translation
② Rotation
③ Translation
So instead of 3 transformation we can combine different transformation by using <u>Homogenous co-ordinates</u>
- In homogenous co-ordinate we use 3×3 matrix instead of 2×2 matrix & introduce additional dummy variable w.
- Generally, Points are represented as (x, y) but here every point is specified by a set of 3 values (xw, yw, w)
- For simplicity we are keeping the the value of w as 1
- In normal co-ordinates the scaling matrix is

$$S = \begin{vmatrix} S_x & 0 \\ 0 & S_y \end{vmatrix}$$

<u>Translation</u>

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

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

$$P' = T(t_x, t_y)\, P$$

Rotation

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

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

In homogenous co-ordinates.

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

Scaling

$$x' = x\delta_x$$
$$y' = y\delta_y$$

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

In homogenous co-ordinates

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

Date
Page

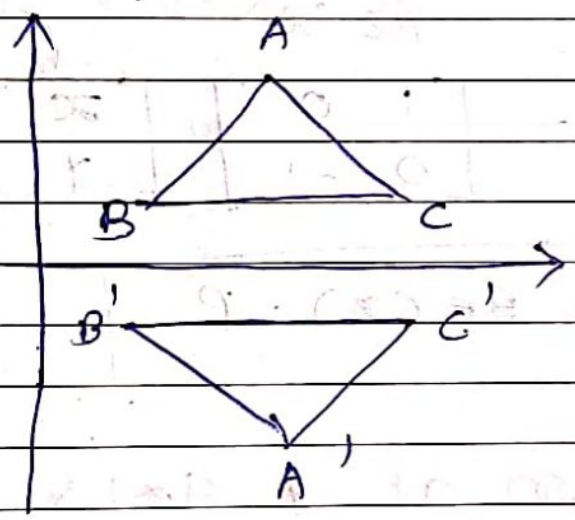
Other types of Transformation -

1. Reflection Transformation
2. Shear Transformation

1. Reflection Transformation -

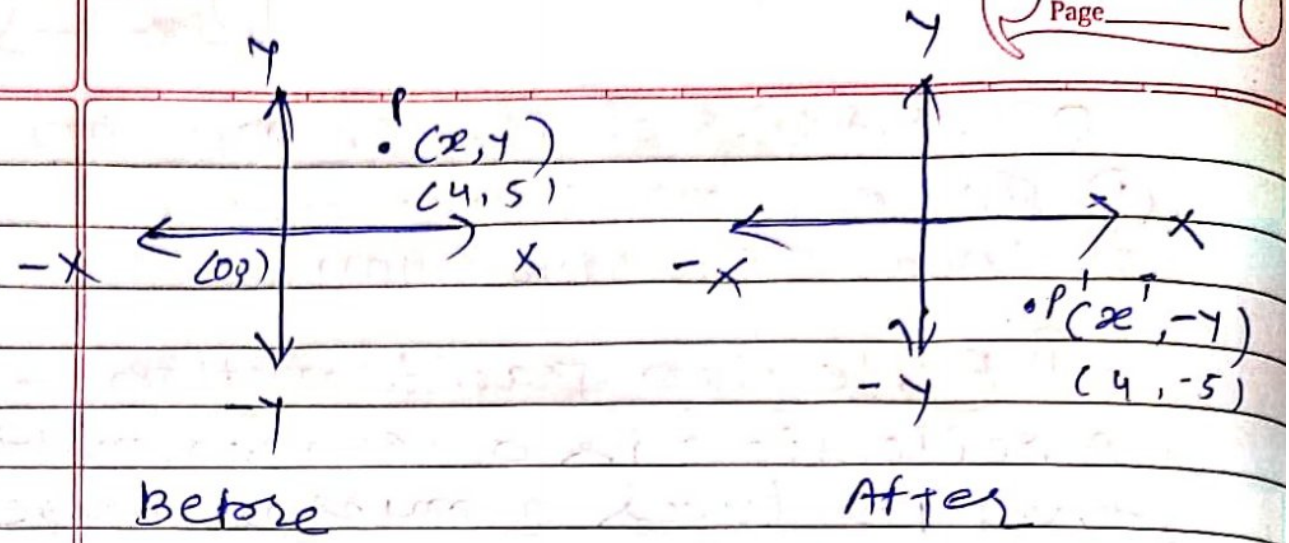
- A reflection is a transformation that produces a mirror image of an object
- The mirror image of 2D reflection is generated relative to an axis of reflection by rotating the object 180° to the reflection axis

Reflection at x - axis



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

- Reflection at x-axis will be similar to reflection at y-axis
- It means now we are keeping mirror at x-axis
- So if the point is (x, y) then its reflection at x-axis will become as $(x, -y)$



Before After

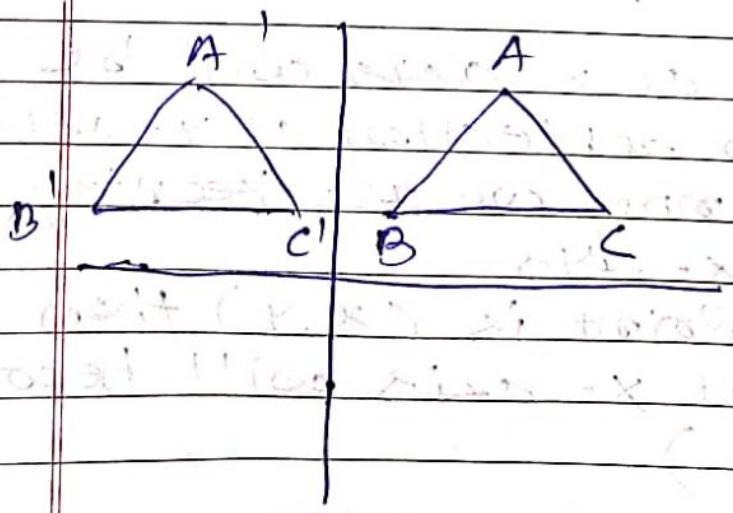
- Here after reflection the values of x-coordinates remains same but y coordinates values gets changed by sign only.
- So the transformation matrix

$Re(x)$

$$\underset{P'}{\begin{bmatrix} x' \\ y' \end{bmatrix}} = \underset{Re(x)}{\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}} \underset{P}{\begin{bmatrix} x \\ y \end{bmatrix}}$$

$$\therefore \boxed{P' = Re(x) \cdot P}$$

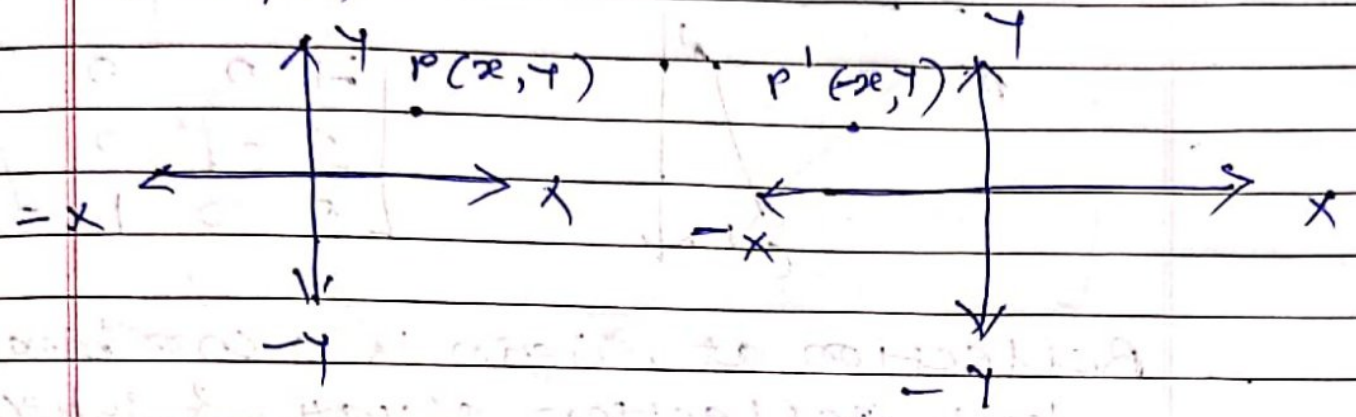
Reflection at y axis



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

Date
Page

- Reflection at Y-axis is similar to placing a mirror image at Y-axis & taking the mirror image of an object
- If we have point (x, y), then the reflection at Y-axis will become $(-x, y)$.



- Here magnitude of both X & Y remains same only the sign of X-coordinate gets changed.

$\boxed{Re(Y)}$

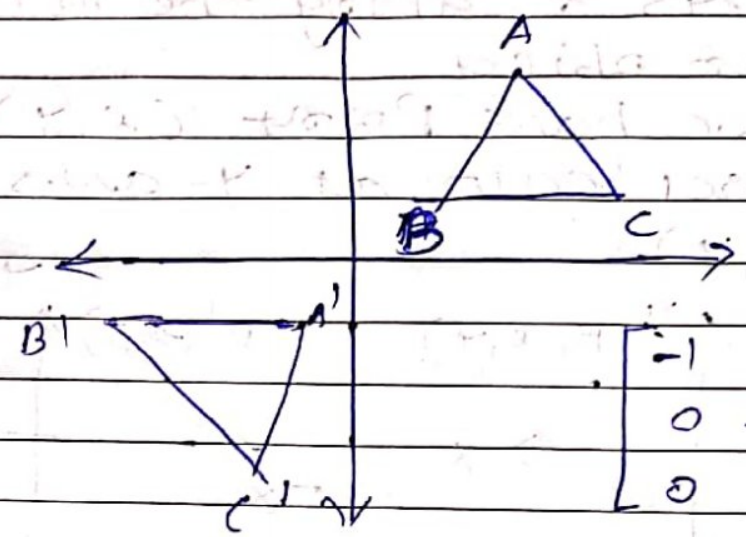
$$\underset{P'}{\begin{bmatrix} x' \\ y' \end{bmatrix}} = \underset{Re(Y)}{\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}} \underset{P}{\begin{bmatrix} x \\ y \end{bmatrix}}$$

$$\therefore \boxed{P' = Re(Y) \cdot P}$$

Reflection at Origin

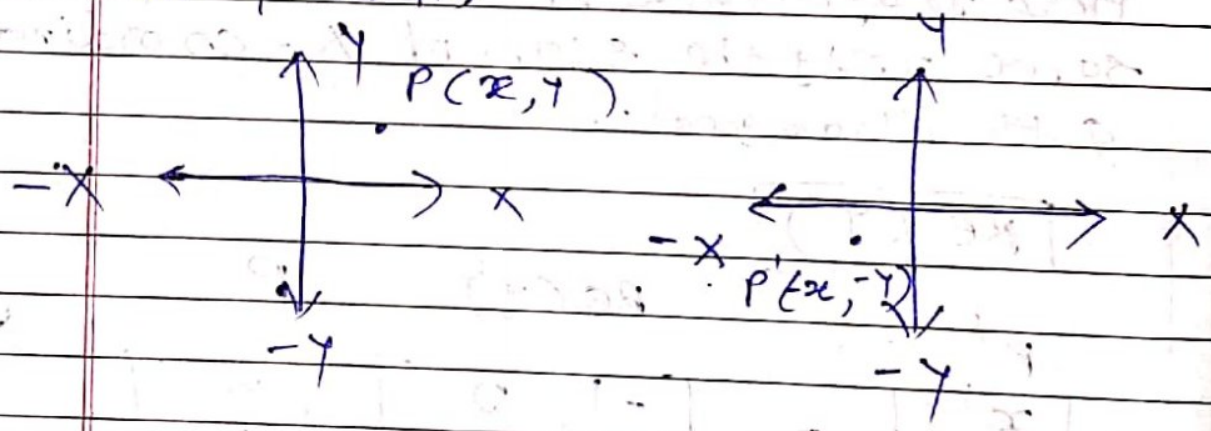
- We have seen reflection at X-axis & reflection at Y-axis. But we have seen them seperately
- suppose if we want both these transformations simultaneously then

we have to apply reflection in the origin



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

- Reflection at origin is a combination of both reflection at x axis & reflection at y-axis



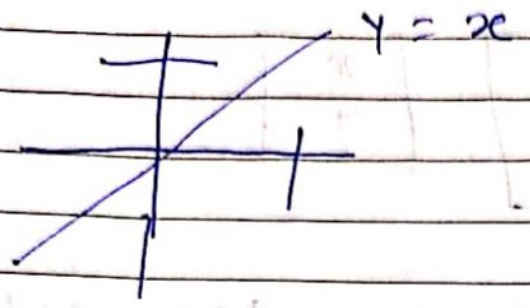
$\therefore$ P' Ref(ori) P

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

$$\therefore \boxed{P' = Re(ori) \cdot P}$$

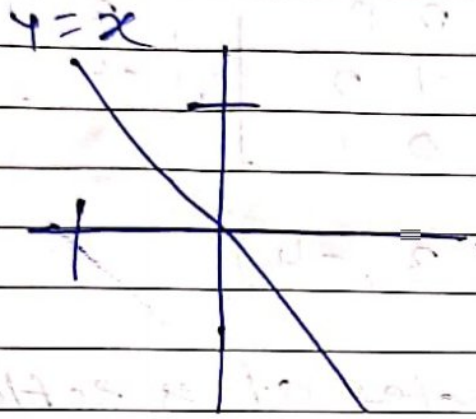
Date
Page

Reflection in the line $y = x$



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

Reflection in the line $x = y$

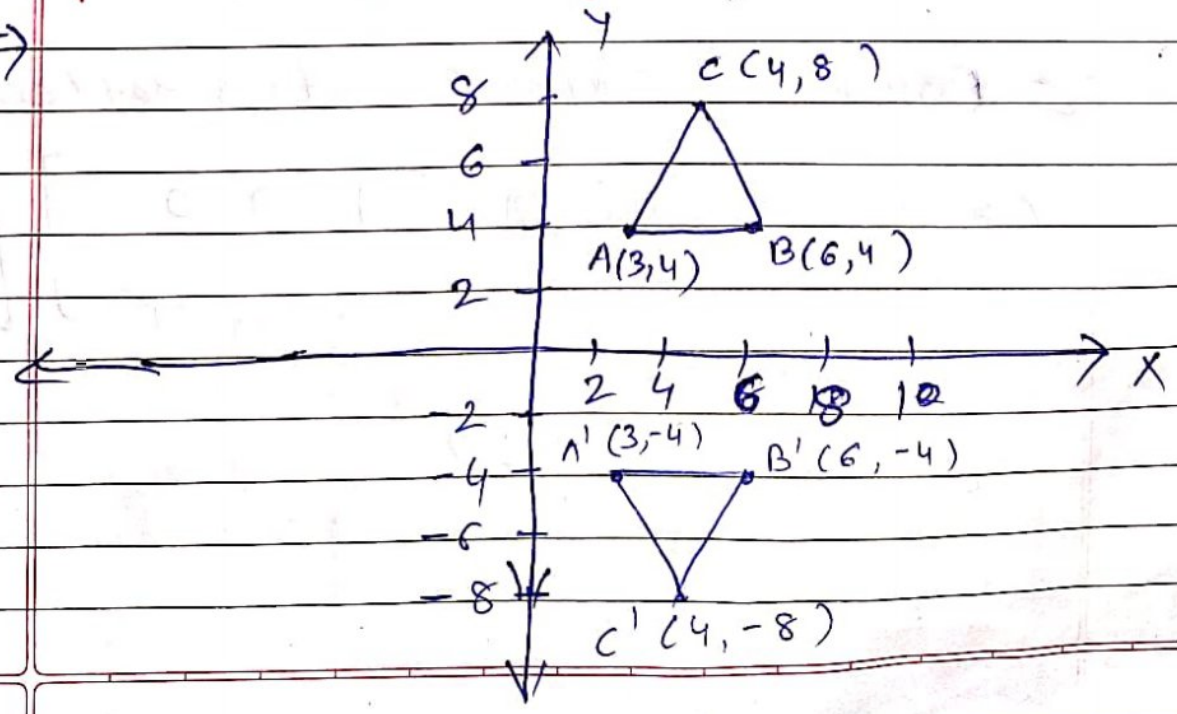


$$\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

Example

(1) A triangle ABC is given, the coordinates of ABC are given as A (3, 4) B (6, 4), C (4, 8). Find the reflected position of triangle to the X-axis

→



Date
Page

= The matrix for Re (x) is

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

A point coordinates after Reflection

$$(x^*, y^*) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ -4 \end{bmatrix}$$

$$A' = (3, -4)$$

B point coordinates after reflection

$$(x, y) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 6 \\ 4 \\ 1 \end{bmatrix}$$

$$B' = (6, -4)$$

C Point co-ordinates after reflection

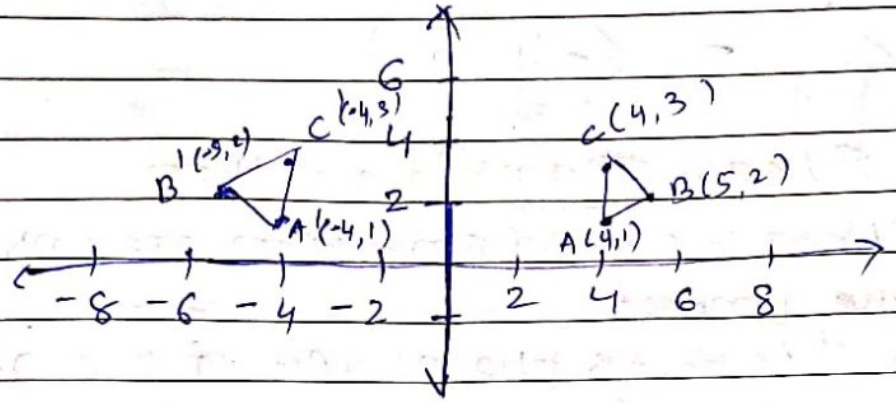
$$(x, y) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 8 \\ 1 \end{bmatrix}$$

$$C' = (4, -8)$$

Date
Page

(2) Determine the transformation matrix of triangle A (4,1), B (5,2), C (4,3) about the line $x = 0$ & determine the resultant coordinates.

⇒ About y - axis we have to find.



The matrix for Reflection y is

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

A Point coordinates after reflection

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

$A' = (-4, 1)$

B Point coordinates after reflection

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

$B' = (-5, 2)$

C Point coordinates after reflection

$$C' = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \\ 1 \end{bmatrix}$$

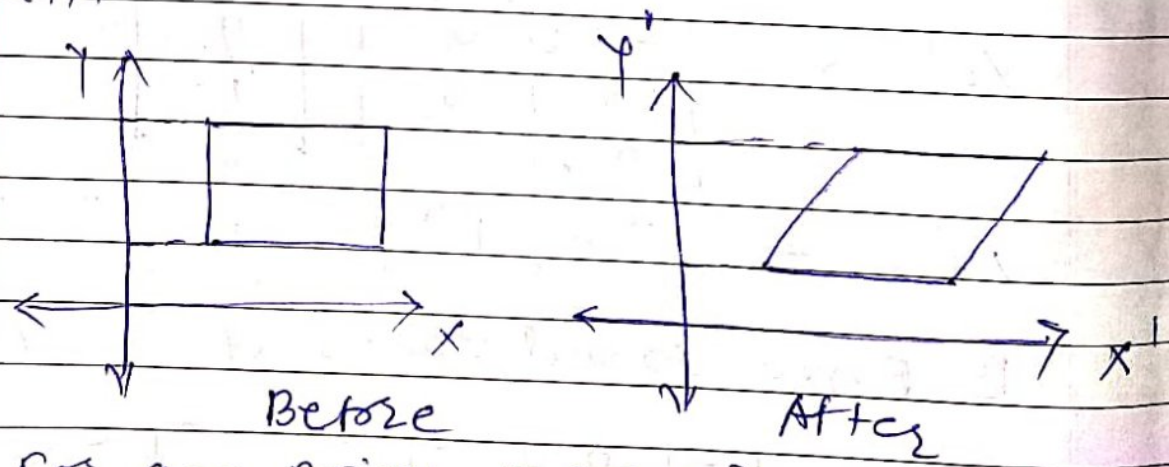
$$C' = (-4, 3)$$

② Shear Transformation

- Shear transformation means slanting the image
- It distorts the shape of the object
- 2 types of shear transformation
  1. X - shear
  2. Y - Shear
- It change the shape & size along x & y axis

① X - Shear

- It preserves the y-coordinates values but changes the x-coordinate values causing vertical lines to tilt.



- For any point P (x, y), new point P (x', y')

Date
Page

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

Matrix Form

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

Homogeneous coordinates

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

where sh_x is the shearing vector in x direction

② Y-Shear

- The X coordinates remains same while there is change in Y coordinate values.

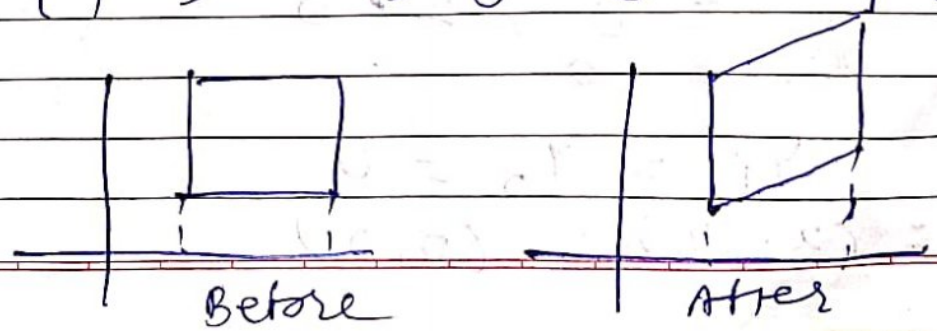
$$P(x, y) = P'(x', y')$$

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

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

Homogenous coordinates

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

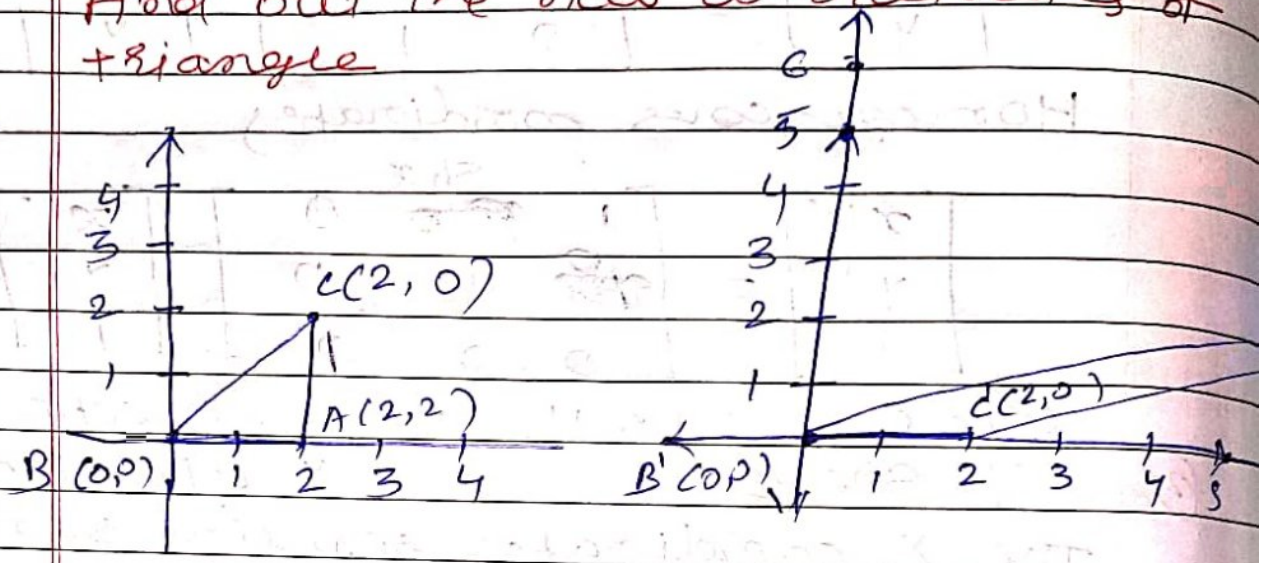


Date
Page

Example

① A triangle with A(2,2), B(0,0), C(2,0). Apply shearing factor 2 on x-axis & 2 on y-axis. Find out the new co-ordinates of triangle

⇒



A (2,2), B(0,0), C(2,0) $Sh_x = 2$
$Sh_y = 2$

X - Shear

$A' = (x_1, y_1)$ $A(x_0, y_0) = (2,2)$

$x_1 = x_0 + Sh_x \cdot y_0$
$= 2 + 2*2$
$x_1 = 6$
$y_1 = y_0 = 2$ $A' = (x_1, y_1) = (6,2)$

$B' = (x_1, y_1)$ $B(x_0, y_0) = (0,0)$
$x_1 = x_0 + Sh_x \cdot y_0$
$= 0 + 2 \cdot 0$
$x_1 = 0$

$y_1 = y_0 = 0$
$B' = (x_1, y_1) = (0,0)$

Date
Page

$C' = (x_1, y_1)$ $C(x_0, y_0) = (2, 0)$

$x_1 = x_0 + sh_x \cdot y_0$

$= 2 + 2 \cdot 0$

$x_1 = 2$

$y_1 = y_0 = 0$

$C' = (x_1, y_1) = (2, 0)$

$A(2,6)$ $\boxed{Y\text{-Shear}}$

$A' = (x_1, y_1)$, $A(x_0, y_0) = (2, 2)$

$x_1 = x_0 = 2$

$y_1 = y_0 + sh_y \cdot x_0$

$= 2 + 2 \cdot 2$

$y_1 = 6$

$A' = (x_1, y_1) = (2, 6)$

$B' = (x_1, y_1)$, $B(x_0, y_0) = (0, 0)$

$x_1 = x_0 = 0$

$y_1 = y_0 + sh_y \cdot x_0$

$= 0 + 2 \cdot 0$

$y_1 = 0$

$B' = (x_1, y_1) = (0, 0)$

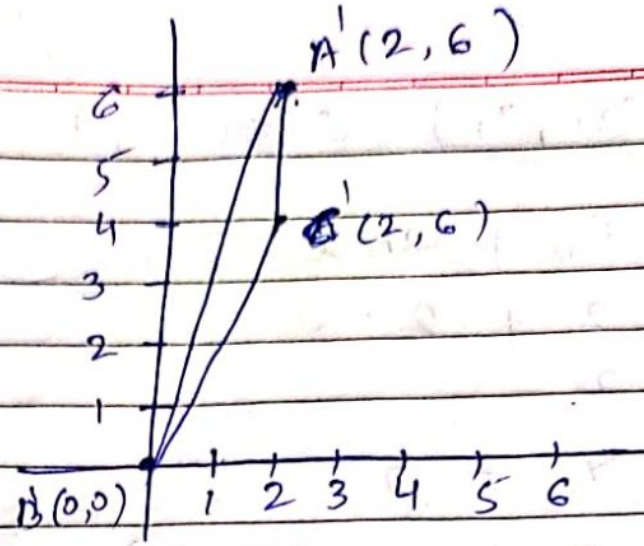
$C' = (x_1, y_1) = B(x_0, y_0) = (2, 0)$

$x_1 = x_0 = 2$

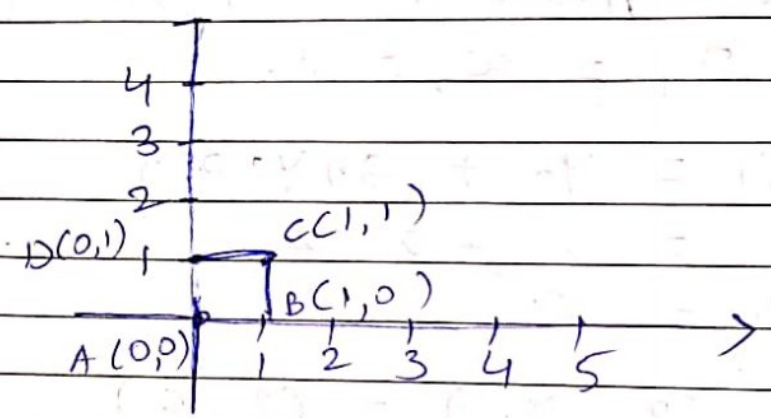
$y_1 = y_0 + sh_y \cdot x_0$

$= 0 + 2 \cdot 2$ $C' = (x_1, y_1) = (2, 4)$

$y_1 = 4$



② Shear a Polygon A(0,0), B(1,0), C(1,1) D(0,1) by shearing vector $Sh_x = 2$. Determine the new coordinates



A(0,0), B(1,0), C(1,1), D(0,1)

$Sh_x = 2$

X - Shear

1) $A' = (x_1, y_1)$ $\quad A(x_0, y_0) = (0,0)$

$x_1 = x_0 + Sh_x \cdot y_0$

$= 0 + 2 \times 0$

$x_1 = 0$

$y_1 = y_0 = 0$

$\therefore A' = (x_1, y_1) = (0,0)$

Date
Page

(2) $B' = (x_1, y_1)$ $\quad B(x_0, y_0) = (1, 0)$

$x_1 = x_0 + sh_x \cdot y_0$

$= 1 + 2 \cdot 0$

$x_1 = 1$

$y_1 = y_0 = 0$

$\therefore B' = (x_1, y_1) = (1, 0)$

(3) $C' = (x_1, y_1)$ $\quad C(x_0, y_0) = (1, 1)$

$x_1 = x_0 + sh_x \cdot y_0$

$= 1 + 2 \cdot 1$

$x_1 = 3$

$y_1 = y_0 = 1$

$\therefore C' = (x_1, y_1) = (3, 1)$

(4) $D' = (x_1, y_1)$ $\quad D(x_0, y_0) = (0, 1)$

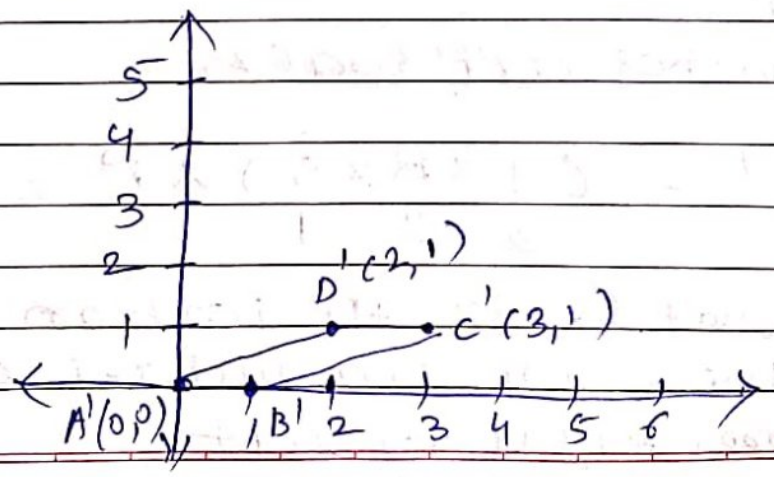
$x_1 = x_0 + sh_x \cdot y_0$

$= 0 + 2 \cdot 1$

$x_1 = 2$

$y_1 = y_0 = 1$

$\therefore D' = (x_1, y_1) = (2, 1)$



Date
Page

Composite Transformation -

- Sequence of different transfor techniques on one object is kno as composite transformations
- We can perform multiple tran-mations on a single object.
- Suppose we want to
  1. Scale a point
  2. Then rotate it
  3. Then translate it
- We have to do transformations as follows

$$P' = (T \times (R \times (S \times P)))$$

- We will apply scaling on the poin we will get some intermediate co-ordinates on which we apply rotation, again we will get some intermediate points on which we will apply translation
- This approach is correct but sometimes not that much appropri-ate.
- Another approach

$$P' = (\underset{3}{T} \times \underset{2}{R} \times \underset{1}{S}) \times P$$

- As matrix multiplication is associative & not commutative
- from Right to left

Date
Page

→ Transformations can be combined by matrix multiplication

$$\underset{P'}{\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}} = \left(\underset{T(t_x, t_y)}{\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}} \underset{R(\theta)}{\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}} \underset{S(S_x, S_y)}{\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}} \right) \underset{P}{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}$$

① There is a triangle ABC A(0,0), B(1,1), C(5,2): Scale the image twice as large & then translate it one unit to the left.

⇒ we have to
1 scale the triangle
2 Translate the triangle

$$P' = (T \times S) \times P$$

More to right → t_x is Positive
More to left → t_x is negative
More upwards → t_y is Positive
More downwards → t_y is negative

$S_x = 2 \quad S_y = 2$
$t_x = -1 \quad t_y = 0$

Date
Page

* Object Matrix

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

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

* Scaling Matrix

$$S(s_x, s_y) = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

* Translation Matrix

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

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

Date
Page

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

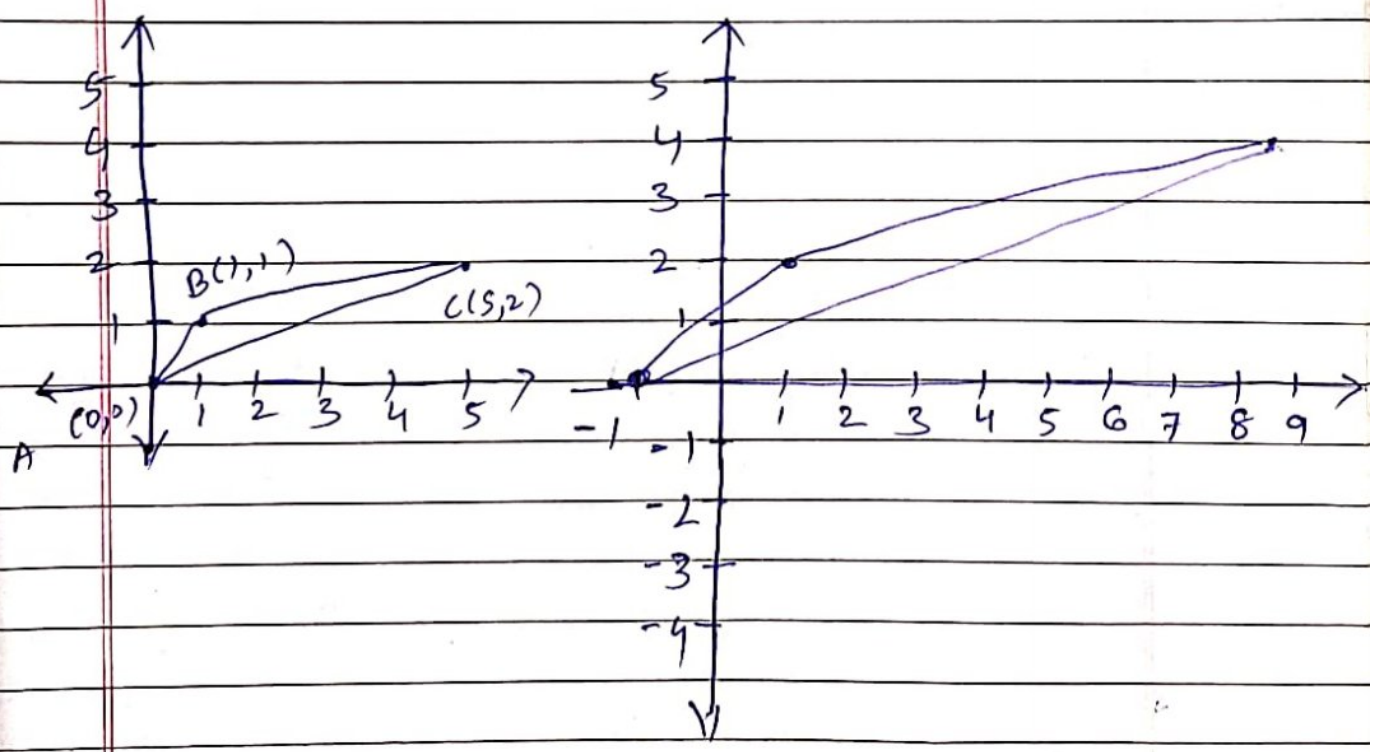
$$O' = \underbrace{\begin{bmatrix} 2 & 0 & -1 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}} \begin{bmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} -1 & 1 & 9 \\ 0 & 2 & 4 \\ 1 & 1 & 1 \end{bmatrix}$$

$A' = (-1, 0)$

$B' = (1, 2)$

$C' = (9, 4)$



PAGE No.
DATE / /

MODULE 4

2 Dimentional Viewing & Clipping

* 2 Dimentional Viewing & Clipping -

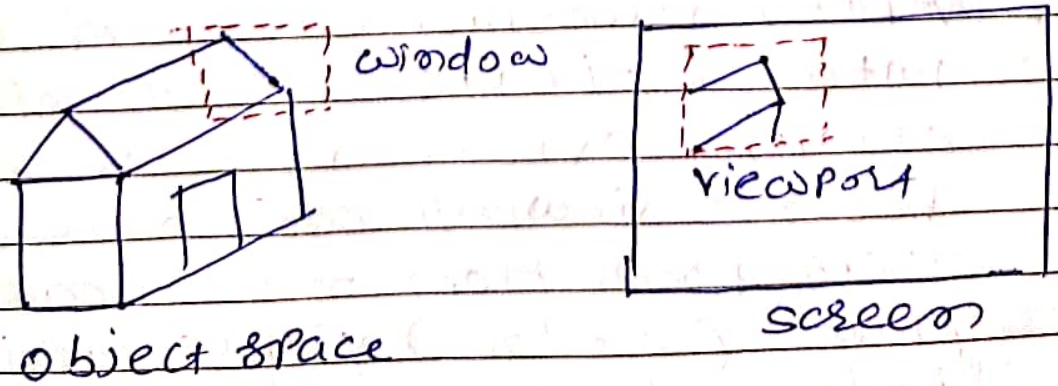
Window

- Window is a world co-ordinate system.
- World co ordinate area selected for display is called window.
- It defines what is to be viewed
- What to be displayed -

Viewport

- It's an area on a display device to which a window is mapped is called viewport.
- Where to be displayed.

So the window defines what is to be viewed, clipping means what to omit & viewport defines where it is to be displayed on display device.



PAGE No.
DATE / /

* The viewing pipeline

The 2D viewing transformation pipeline is as follows -

MC →	construct World co-ordinate scene using Modelling co-ordinat transformation	WC →	convert world coordinate to viewing co-ordinate	VC →	Map viewing coordinates to normalized viewing co-ordinates using window viewport specification	NVC →	Map normalized viewport to Device coordinate	DC

- Viewing pipeline describes a series of transformation which are passed by geometry data to display on device in a form of image
- The 2D pipeline describes this process to display 2D images
- The coordinates in which individual objects (actual models) are created are called as Model coordinates.
- Combination of such models, assembled into a scene are described by world coordinates
- After transformation of scene into coordinate system of a camera (viewer), those coordinates become viewing coordinates
- Their projection onto a common plane (window) yields device independent normalized coordinates

PAGE No.
DATE / /

- Now mapping of those normalized co-ordinates to any suitable specific device, we get device coordinates.

* Window to viewport co-ordinate transformation

- In general the mapping of world co-ordinate scene of device coordinates is referred as viewing transformation.
- Sometimes it is also called as window to viewport transformation or windowing transformation.
- This transformation is a process of three steps.

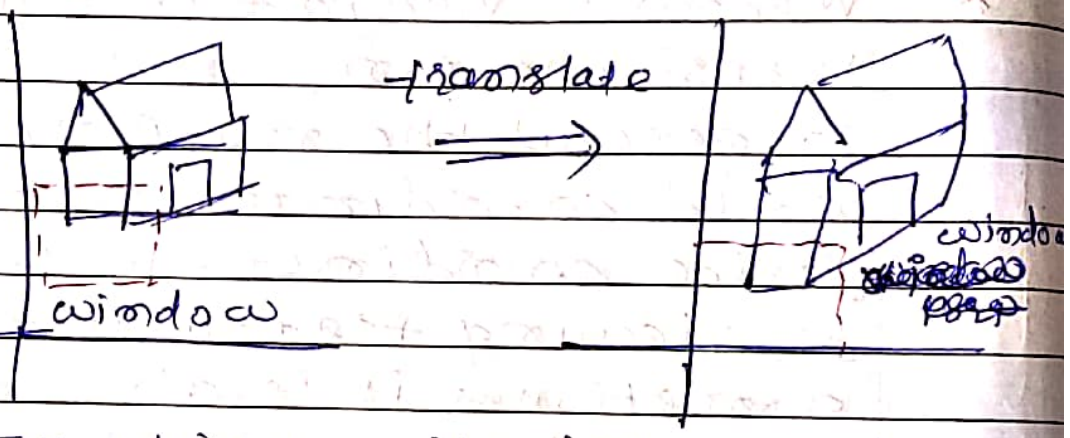
Step 1 Translate window

- Translate window towards origin
- To shift window towards origin lower left or upper left corner of window, we will have to subtract.
- Hence translation factor will become negative
  $(-t_x, -t_y)$
- So $(-wx_{min}, -wy_{min})$ when origin is lower left corner of

screen

- or

$(-wx_{min} - wy_{max})$ where origin is upper left corner of screen



The object with its window is shifted or translated until the lower left corner of the window matches to the origin.

Step 2 Resize Window

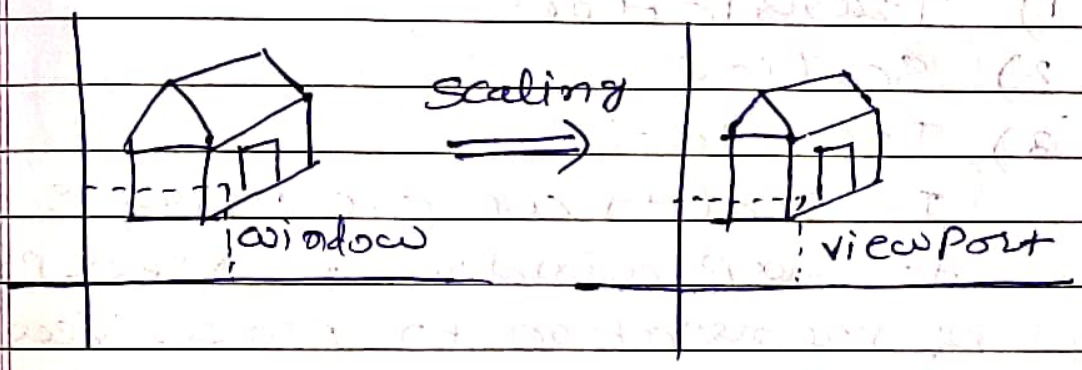
- Resize window to the size of viewport
- Following computation is required to scale window as per viewport

$$\delta x = \frac{XV_{max} - XV_{min}}{XW_{max} - XW_{min}}$$

$$\delta y = \frac{YV_{max} - YV_{min}}{YW_{max} - YW_{min}}$$

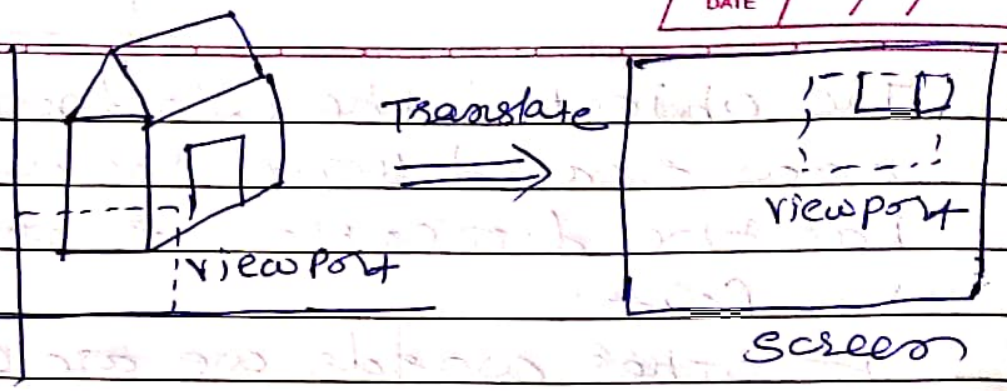
PAGE No.
DATE / /

- The object & the window are now scaled until the window has the dimension same as viewport.
- In other words we are converting the object into image & window in viewport.
- For this we have to use scaling



Step 3 Translate window

- Translate window (keep the same position as viewport)
- If lower corner of viewport is at origin (0,0) the no need of transformation of window or else we have to add translation factor
- Here the translation factor will be (tx, ty) which is (XVmin, YVmin)
- This translation is performed to move the viewport to its correct position on the screen



So the viewing transformation performs 3 steps

1) Translation
2) Scaling
3) Translation

In reality we are using scaling to map window to viewport & translation to place viewport properly on screen

Let us generalize the matrix

① The matrix for translation with respect to window is

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -xw_{min} & -yw_{min} & 1 \end{bmatrix}$$

② The matrix for scaling will be

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

PAGE No.
DATE / /

where

$$\delta x = \frac{\text{Width of Viewport}}{\text{Width of Window}}$$

$$\delta x = \frac{XV_{max} - XV_{min}}{XW_{max} - XW_{min}}$$

and

$$\delta y = \frac{\text{height of Viewport}}{\text{height of Window}}$$

$$\delta y = \frac{YV_{max} - YV_{min}}{YW_{max} - YW_{min}}$$

∴ Scaling Matrix will be

$$\begin{vmatrix} \frac{XV_{max} - XV_{min}}{XW_{max} - XW_{min}} & 0 & 0 \\ 0 & \frac{YV_{max} - YV_{min}}{YW_{max} - YW_{min}} & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

③ Translation Matrix with respect to viewport will be

$$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ XV_{min} & YV_{min} & 1 \end{vmatrix}$$

PAGE No.
DATE / /

Collective generalized viewing transformation matrix will be

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -XW_{min} & -YW_{min} & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{XV_{max} - XV_{min}}{XW_{max} - XW_{min}} & 0 & 0 \\ 0 & \frac{YV_{max} - YV_{min}}{YW_{max} - YW_{min}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ XV_{min} & YV_{min} & 1 \end{bmatrix}$$

If we will solve this we will get

$$\begin{bmatrix} \frac{XV_{max} - XV_{min}}{XW_{max} - XW_{min}} & 0 & 0 \\ 0 & \frac{YV_{max} - YV_{min}}{YW_{max} - YW_{min}} & 0 \\ (XV_{min} - XW_{min}) \cdot \frac{XV_{max} - XV_{min}}{XW_{max} - XW_{min}} & (YV_{min} - YW_{min}) \cdot \frac{YV_{max} - YV_{min}}{YW_{max} - YW_{min}} & 1 \end{bmatrix}$$

But if height & width of window is not in proportion with viewport then distortion occurs

PAGE No.
DATE / /

and image may get elongated or minimized.

PAGE No.
DATE / /

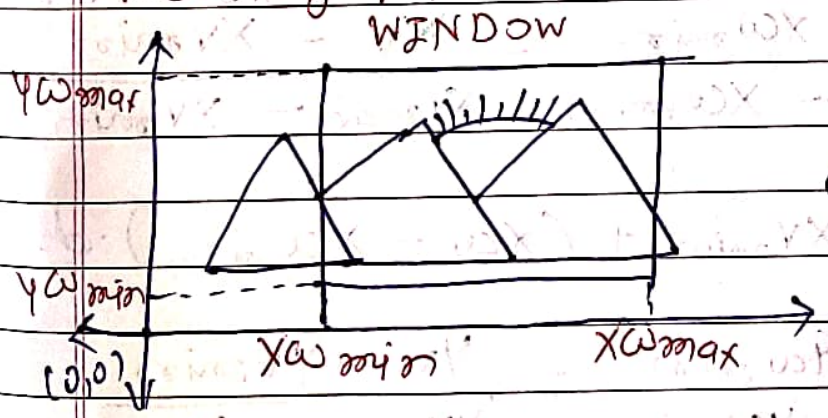
2D Viewing

A graphics package that allows a user to specify which part of a defined picture is to be displayed & where that part is to be displayed on the display device using a concept known as clipping

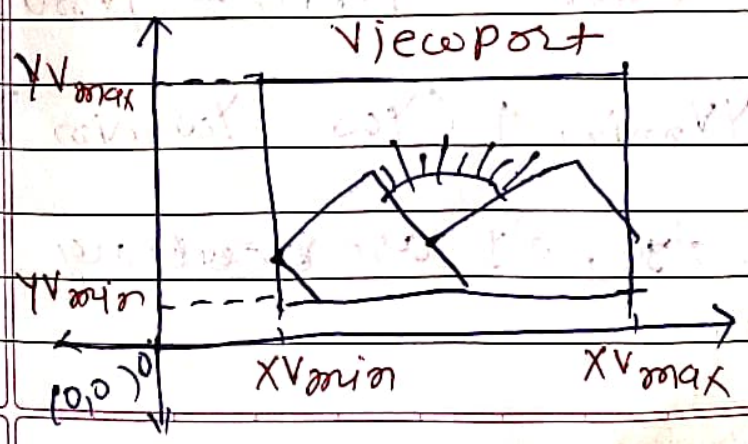
World coordinate system -
This is the object space or the space in which application model is defined.

Screen coordinate system -
The space in which the image is displayed is called as screen coordinate system

Viewing Pipeline

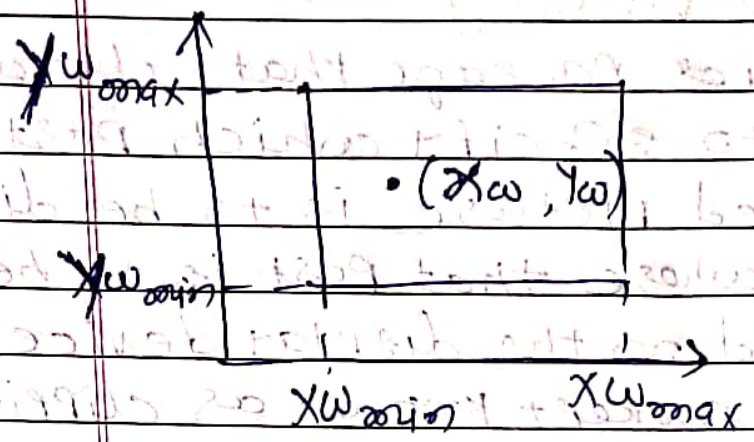


World coordinate

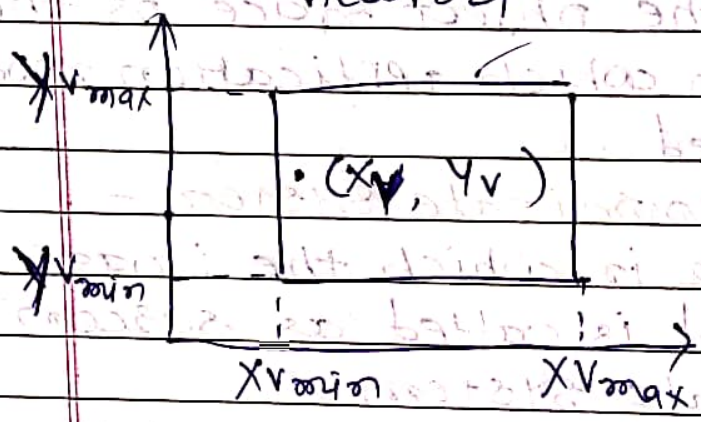


Device coordinate

Window



viewport



Relative position will not change only size will change a/c to w & v

$$\frac{x_w - X_{w_{min}}}{X_{w_{max}} - X_{w_{min}}} = \frac{x_v - X_{v_{min}}}{X_{v_{max}} - X_{v_{min}}}$$

$$X_v = X_{v_{min}} + (X_w - X_{w_{min}}) \cdot d_x$$

$$\frac{y_w - y_{w_{min}}}{y_{w_{max}} - y_{w_{min}}} = \frac{y_v - y_{v_{min}}}{y_{v_{max}} - y_{v_{min}}}$$

$$y_v = y_{v_{min}} + (y_w - y_{w_{min}}) d_y$$

where d_x, d_y are scaling factor

PAGE No.
DATE / /

$$dx = \frac{XV_{max} - XV_{min}}{XW_{max} - XW_{min}}$$

$$dy = \frac{YV_{max} - YV_{min}}{YW_{max} - YW_{min}}$$

eg. $XV_{min} = 30$ $\quad$ $XW_{min} = 20$
$XV_{max} = 60$ $\quad$ $XW_{max} = 80$
$YV_{min} = 40$ $\quad$ $YW_{min} = 40$
$YV_{max} = 60$ $\quad$ $YW_{max} = 80$
$(XV, YV = ?)$ $\quad$ $XW, YW = (30, 60)$
Find out

$\Rightarrow$

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

$$\frac{XV - 30}{60 - 30} = \frac{30 - 20}{80 - 20}$$

$$\frac{XV - 30}{30} = \frac{10}{60}$$

$$XV = \left(\frac{10}{60}\right) \times 30$$

$$XV - 30 = 5$$

$$XV = 5 + 30$$

$$XV = 35$$

PAGE No.
DATE / /

$$\frac{Y_v - Y_{v_{min}}}{Y_{v_{max}} - Y_{v_{min}}} = \frac{Y_\omega - Y_{\omega_{min}}}{Y_{\omega_{max}} - Y_{\omega_{min}}}$$

$$\frac{Y_v - 40}{60 - 40} = \frac{60 - 40}{80 - 40}$$

$$\frac{Y_v - 40}{20} = \frac{20}{40}$$

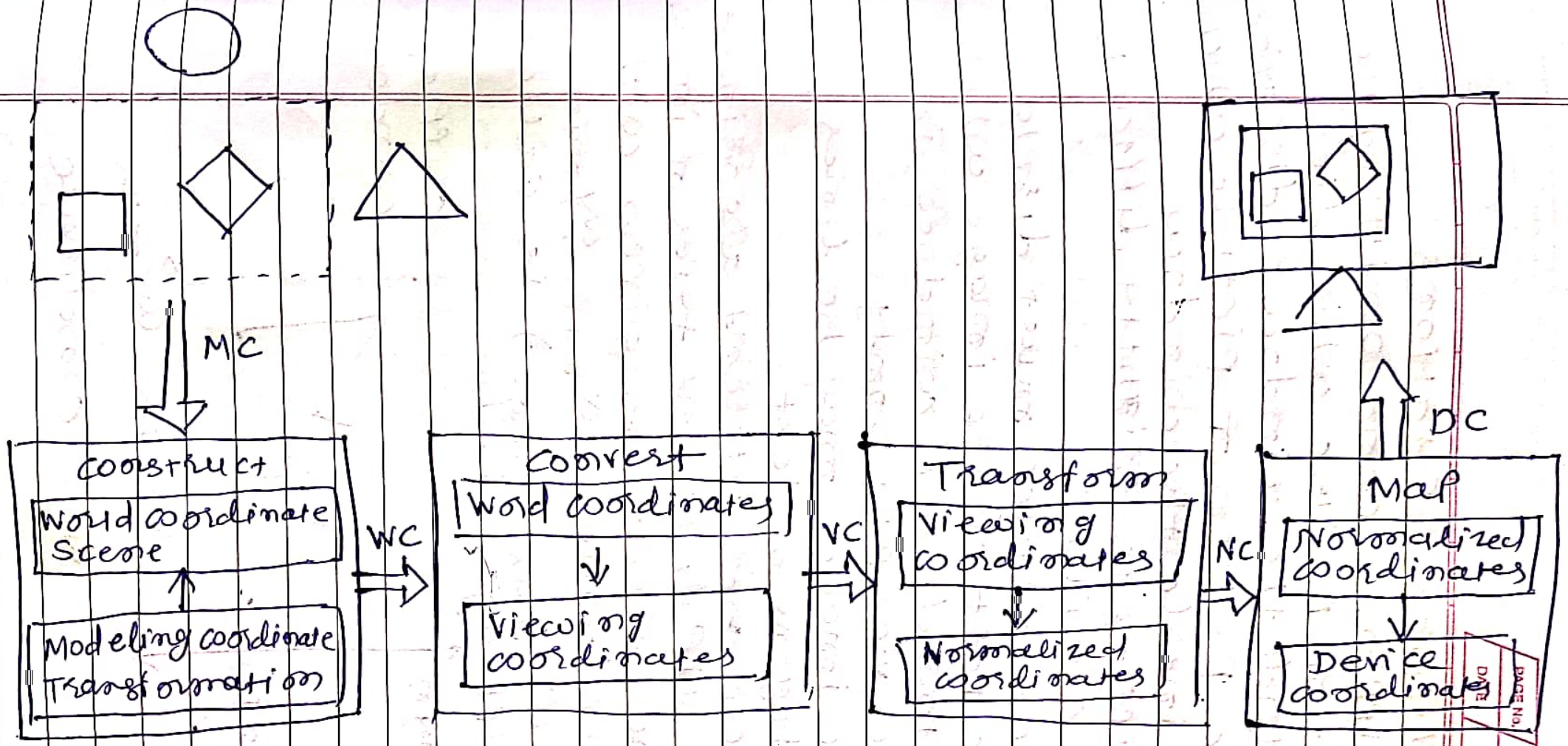
$$Y_v - 40 = \left(\frac{20}{40}\right) \times 20$$

$$Y_v - 40 = 10$$

$$Y_v = 10 + 40$$

$$Y_v = 50$$

$$\therefore (X_v, Y_v) = (35, 50)$$



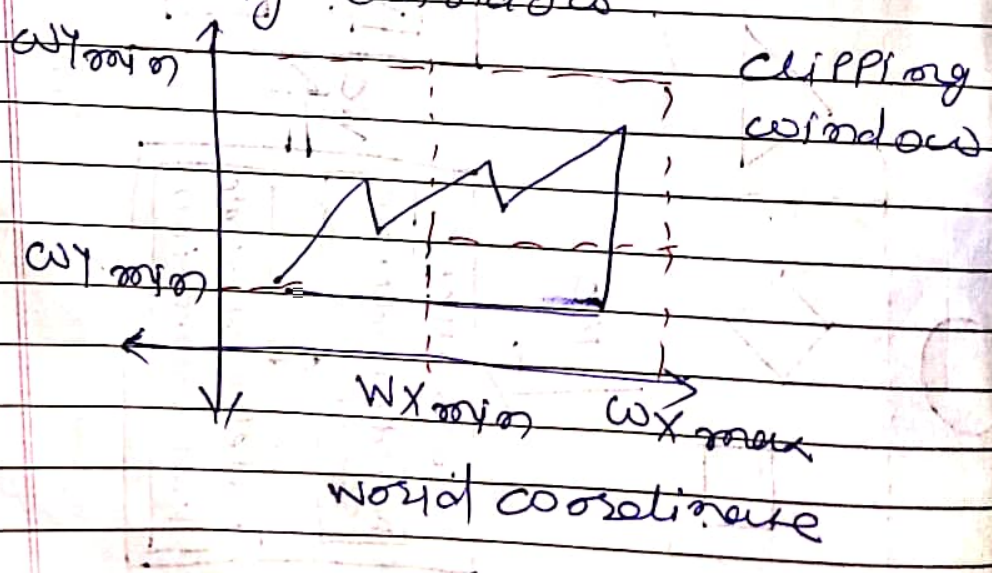
The Viewing Pipeline

PAGE No.
DATE / /

Clipping operations

Clipping means omitting the part of picture which lies outside the window & displaying the portion of picture which is inside the window

- Clipping is slightly difficult to computes.
- The computer must display the part of the image which is inside the window, in the viewport only.
- The images may be lines, polygons or text.
- There are different algorithms to clip different types of images
- one window is given with coordinates (wx_{min}, wy_{min}) & (wx_{max}, wy_{max}) is called clipping window.

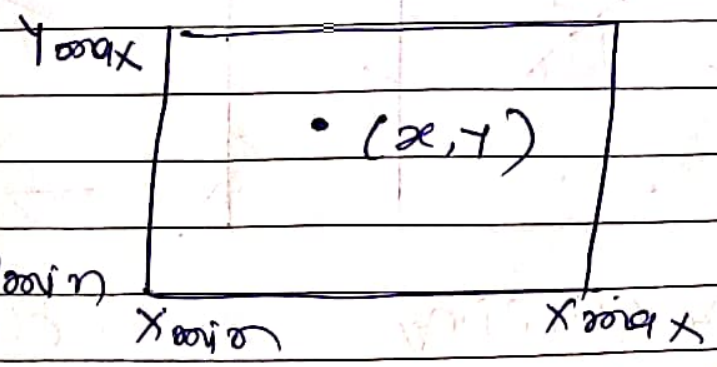


PAGE No.
DATE / /

* Application of clipping
- It will extract part we desire
- To identify the visible & invisible area in 3D objects
- For creating object using solid modeling.
- For drawing operations
- For deleting, copying, moving part of objects.

* Types of clipping
① Point clipping
② Line clipping

① Point clipping
- It is used to determining whether the point is inside the window or not
- If the point is coming inside the clipping window we have to display it, otherwise no need to display it.



* conditions to check whether point is inside clipping window

PAGE No.
DATE / /

or not

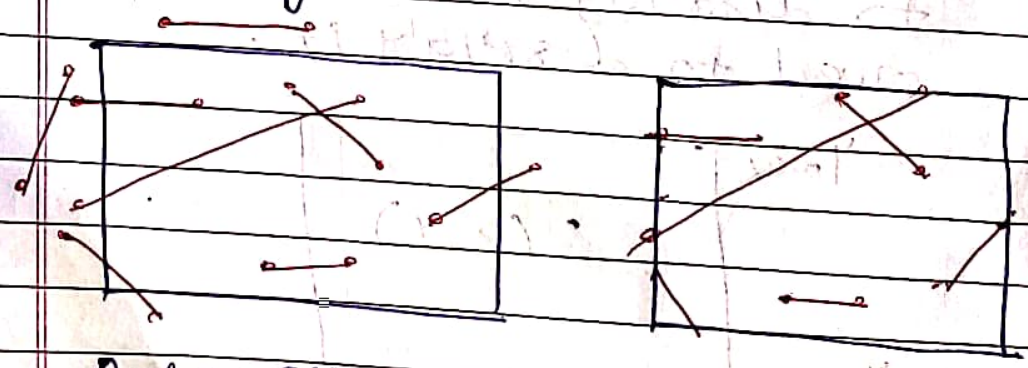
1) $x \leq X_{max}$
2) $x \geq X_{min}$
3) $Y \leq Y_{max}$
4) $Y \geq Y_{min}$

If any of the above condition is <u>TRUE</u> the point lies inside the clipping window.
If any of the above condition is <u>FALSE</u> the point lies outside the clipping window

② Line clipping

- ~~The part of the line inside the window~~
- It is simplest to clip a straight line.

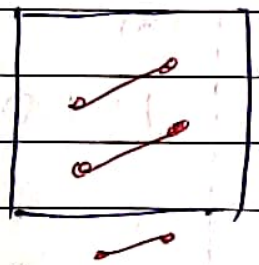
Fig shows lines of various types before clipping & after clipping



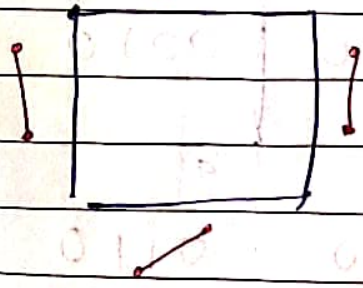
Before Clipping After Clipping

PAGE No.
DATE / /

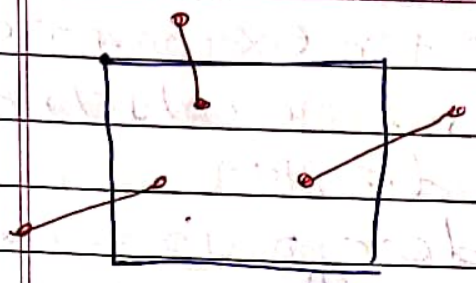
- Here we have to examine each & every line which we are going to display.
- We have to determine whether or not a line is ① completely inside the window or
② lies completely outside the window or
③ crosses the boundary of window
- If the line is completely inside then display it fully.
- If the line is totally outside then discard that line
- But if the line is crossing to the window boundary, then we have to find the intersection point of the line with the edge of window & draw the portion which lies inside



Accept the lines don't clip } Visible lines



Reject the lines } Not visible lines

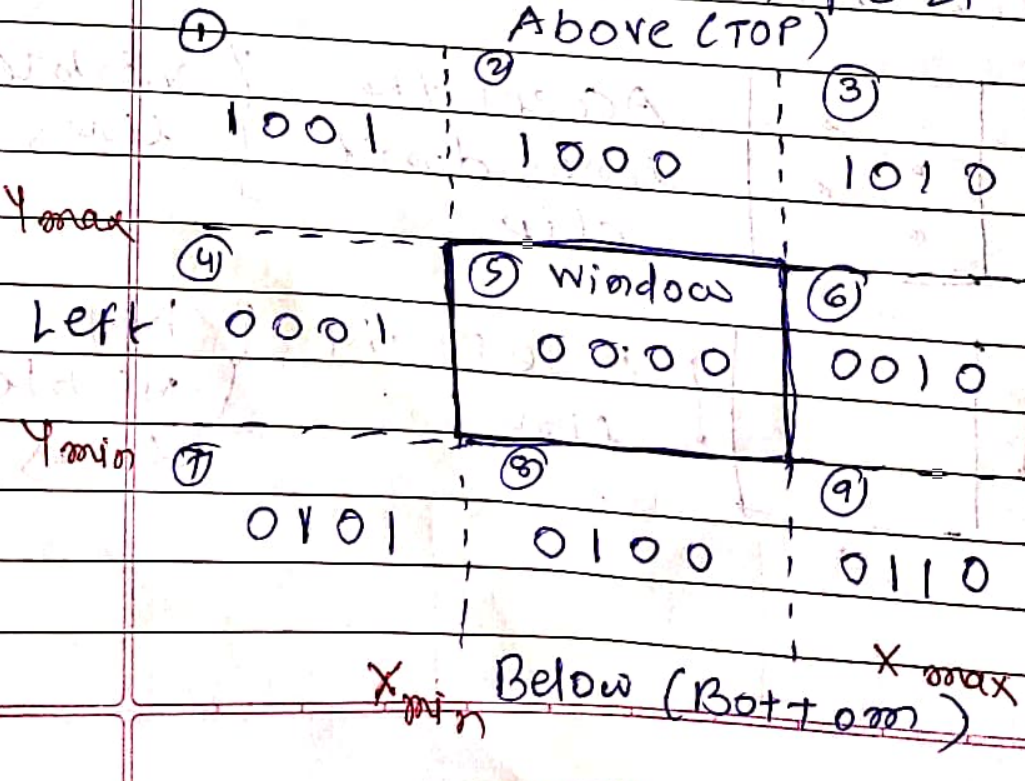


Clipping is required } Partially visible lines

Line Clipping Algorithms
1. Cohen Sutherland
2. Liang Barsky

1. Cohen Sutherland Line Clipping Algorithm -

- This algorithm immediately removes the lines which are lying totally outside the window
- This algorithm divides the plane in nine (9) parts & assigns outcode or binary number to each part



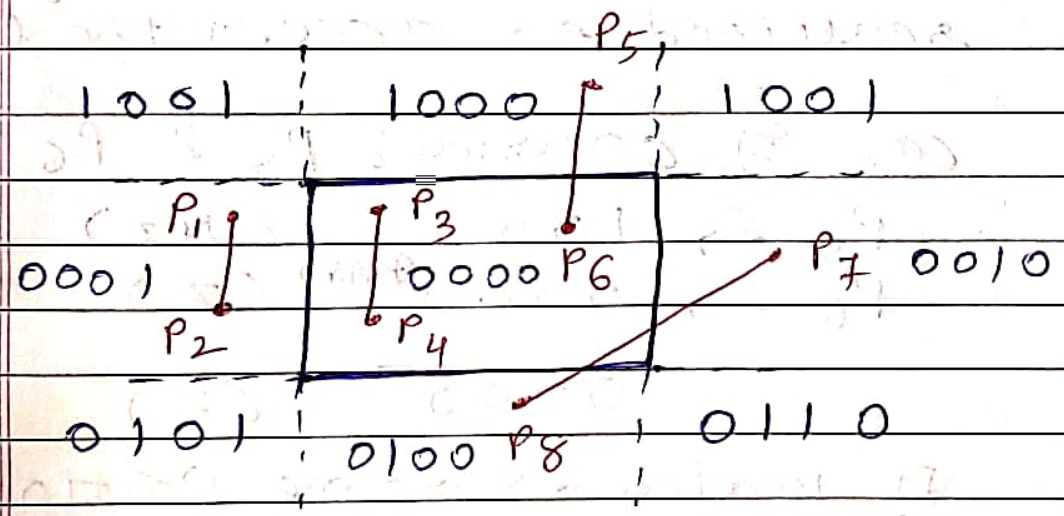
PAGE No.
DATE / /

4 bit code called as TBRL

T - TOP
B - Bottom
R - Right
L - Left

T : $\Rightarrow y > y_{max}$
B : $\Rightarrow y < y_{min}$
R : $\Rightarrow x > x_{max}$
L : $\Rightarrow y < x_{min}$

Example



case ①

consider line P_1 & P_2

$P_1 \rightarrow$ 0001 (Non zero value)
$P_2 \rightarrow$ 0001 (Non zero value)

P_1 & P_2 are non zero values so perform AND operation

$$\begin{array}{r} 0001 \\ 0001 \\ \hline 0001 \end{array} \quad \oplus \text{ AND}$$

(Non zero value

(If Result of AND operation is NZ)

If all non zero values are there it indicates that P_1 & P_2 are completely outside the window so Reject line

Case (2) consider P_3 & P_4

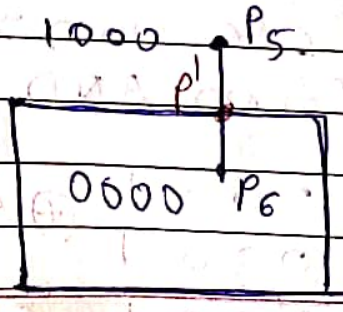
$P_3 \rightarrow$ 0000 (zero value)
$P_4 \rightarrow$ 0000 (zero value)
0000 (zero value)

It indicates that P_3 & P_4 are completely inside the window so no clipping is required so accept the line.

Case (3) consider P_5 & P_6

$P_5 \rightarrow$ 1000 (NZ)
$P_6 \rightarrow$ 0000 ⊕AND (Z)
0000 (Z)

It indicates some portion of of line lies inside the window & some portion of line lies outside the window so clipping is required.
Find the intersection point.



PAGE No.
DATE / /

consider P' and P_6

$P' \rightarrow$ 0000 (zero)

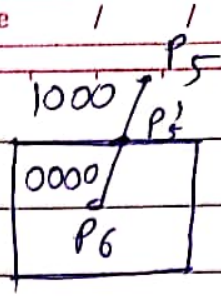
$P_6 \rightarrow$ 0000 (zero)

0000 (zero)

Accept P' and P_6

clip P' and P_5

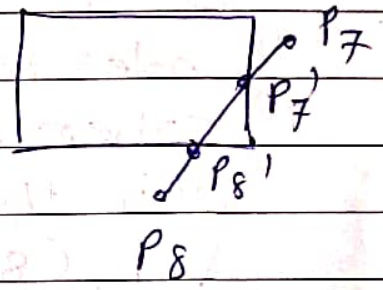
Consider P_5' & P_6



P_5'	0000	Z
P_6	0000	Z
	0000	

Accept P_5' & P_6
clip P_5 & P_5'

Consider P_7 & P_8



P_7	0100	NZ
P_8	0010	NZ
AND	0000	Z

you need to clip

P_7 & P_7' - clipped

P_7' & P_8

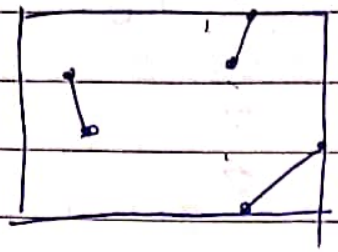
P_7'	0000	Z
	0010	NZ
	0000	Z

Again clip

P_8 & P_8' - clipped

P_8'	0000	Z
P_7'	0000	Z
	0000	Z

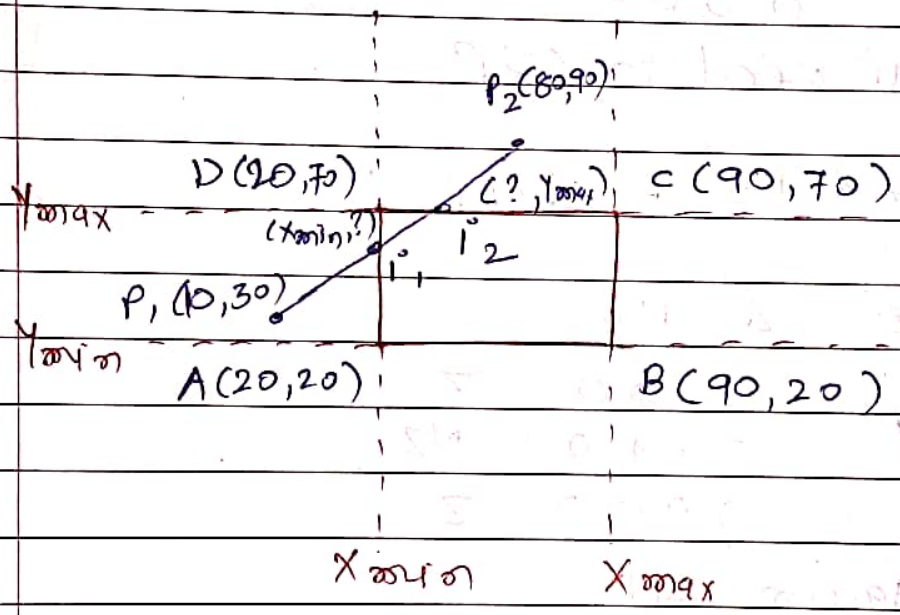
Accept the line



Example

① Let ABCD be the rectangular window with A (20,20), B (90,20), C (90,70) & D (20,70). Find region codes for the end points & use Cohen Sutherland algorithm to clip the lines P_1P_2 with P_1(10,30) & P_2(80,90)

⇒



Finding Region code

P_1	0001	NZ
P_2	1000	NZ
	0000	Z

Since it is 0000, line is partially visible so needs clipping

$$\text{Slope, } m = \frac{Y_2 - Y_1}{X_2 - X_1}$$

$$= \frac{90-30}{80-10}$$

$$= \frac{60}{70}$$

$$m = 0.857$$

$$Y = m(X_{min} - X_1) + Y_1$$

$$= 0.857(20-10) + 30$$

$$Y = 38.57$$

$$i_1(X_{min}, Y) = (20, 38.57)$$

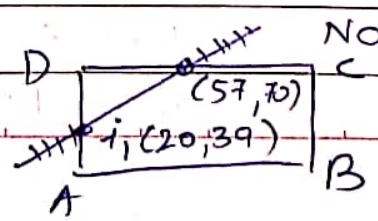
1st Intersection point

$$X = \frac{1}{m}(Y_{max} - Y_1) + x_1$$

$$= \frac{1}{0.857}(70-30) + 10$$

$$X = 56.67$$

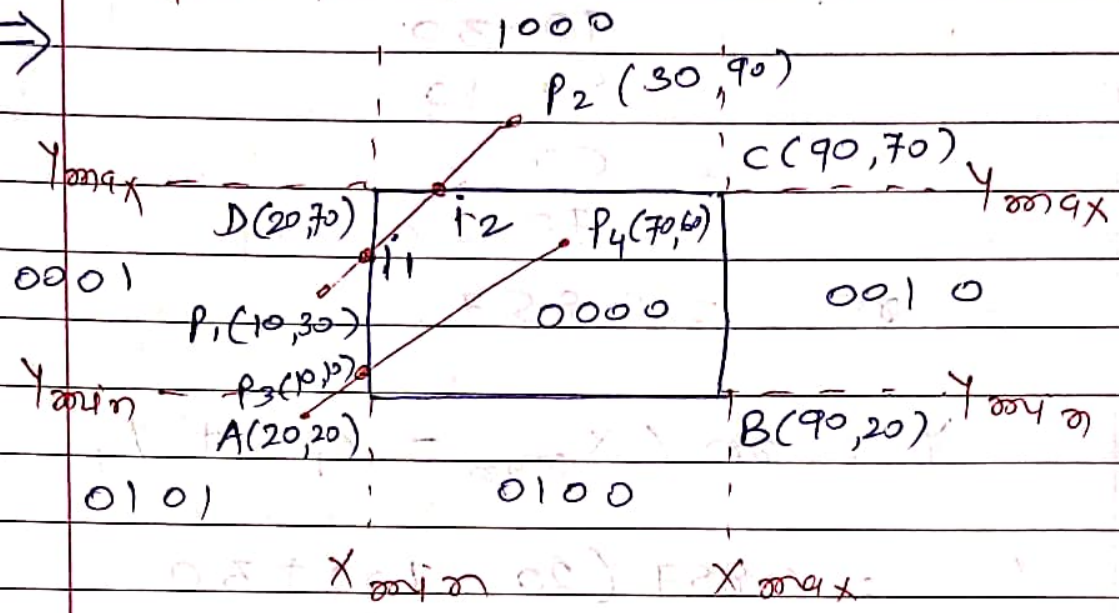
$$\therefore i_2(56.67, 70) \cong (57, 70)$$



Not visible clipped

(2) Given coordinates of window
A (20,20), B (90,20), C (90,70), D (20,70)
Clip the lines P_1P_2 & P_3P_4.
P_1 (10,30), P_2 (80,90), P_3 (10,10),
P_4 (70,60)

⇒



Region codes

P_1	0001	NZ	P_3	0101	NZ
P_2	1000	NZ	P_4	0000	Z
AND	0000	Z		0000	Z

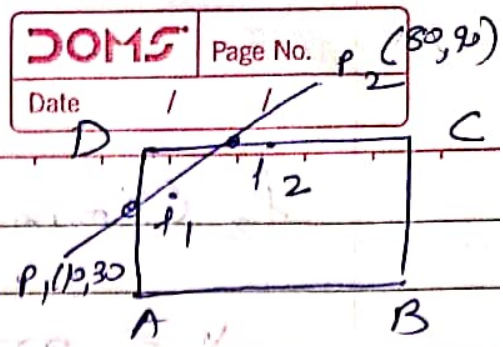
visible partially
So trivially accept it

Find the intersection point

(1) For P_1P_2 P_1 (10,30), P_2 (80,90)

$$\text{Slope } m = \frac{y_2 - y_1}{x_2 - x_1}$$

$$= \frac{90-30}{80-10}$$



$$m = \frac{60}{70}$$

$$m = 0.857$$

$$i_1(x_{min}, y) = (20, ?)$$

$$y = m(x_{min} - x_1) + y_1$$

$$y = 0.857(20 - 10) + 30$$

$$y = 0.857(10) + 30$$

$$y = 38.57$$

$$\therefore i_1(20, 38.57)$$

Now finding i_2

$$i_2(x, y_{max}) = (?, 70)$$

$$\therefore x = \frac{1}{m}(y_{max} - y_1) + x_1$$

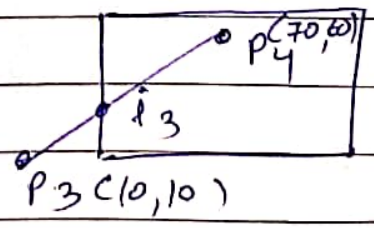
$$= \frac{1}{0.857}(70 - 30) + 10$$

$$= \frac{1}{0.857}(40) + 10$$

$$x = 56.67$$

$$\therefore i_2(56.67, 70)$$

(2) for $P_3 P_4$



$$\text{slope } m = \frac{y_2 - y_1}{x_2 - x_1}$$

$$= \frac{60 - 10}{70 - 10}$$

$$m = \frac{50}{60} = 0.832$$

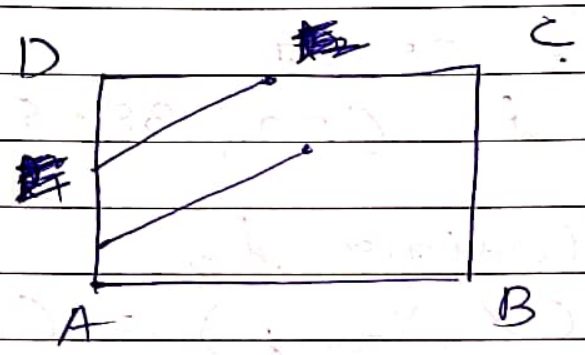
$i_3 (X_{min}, Y) = (20, ?)$

$Y = m(X_{min} - x_1) + Y_1$

$= 0.832 (20 - 10) + 10$

$Y = 18.33$

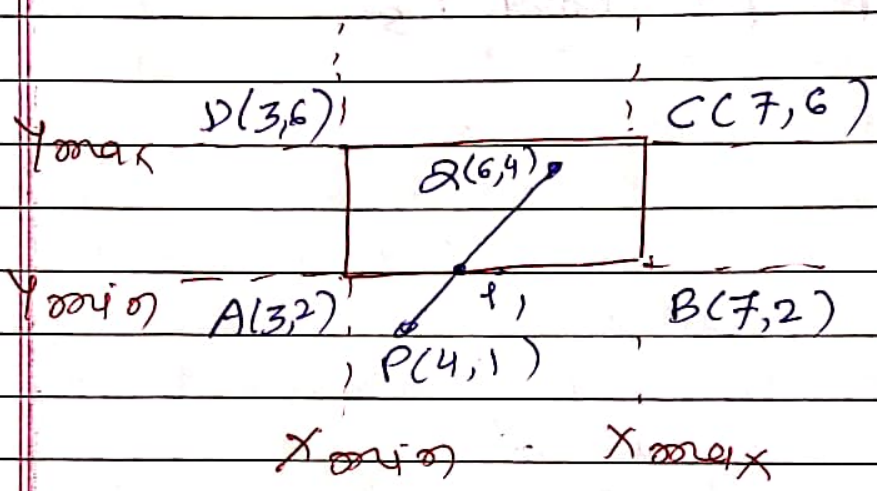
$i_3 (X_{min}, Y) = (20, 18.33)$



PAGE No.
DATE / /

(3) Clip the line PQ having coordinates A(4,1) and B(6,4) against the clipwindow havig vertices A(3,2), B(7,2), C(7,6) & D(3,6) Using cohen sutherland line clipping algorithm.

⇒



Finding the Region code

P	0100	N2	
Q	0000	+	AND
		2	
	0000	2	

since it is 0000, line is Partially visible so needs clipping.

$$\text{slope } m = \frac{y_2 - y_1}{x_2 - x_1}$$

$$= \frac{4-1}{6-4}$$

$$= \frac{3}{2} = 1.5$$

PAGE No.
DATE / /

~~Now dog [illegible]~~

Find the intersection point.

$$x = \frac{1}{m}(y_{min} - y_1) + x_1$$

$$= \frac{1}{1.5}(2 - 1) + 4$$

$$= \frac{1}{1.5} + 4$$

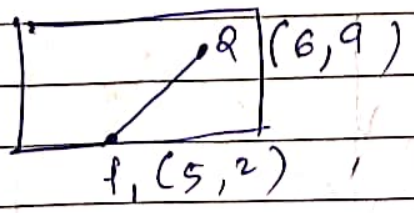
$$x = 4.66 \cong 5$$

$$\therefore i_1 (x, y_{min}) = (5, 2)$$

So the co-ordinates of intersection point i_1 will be (5, 2)

By finding out the intersection point we have divided a line PQ into Pi_1 & i_1Q.

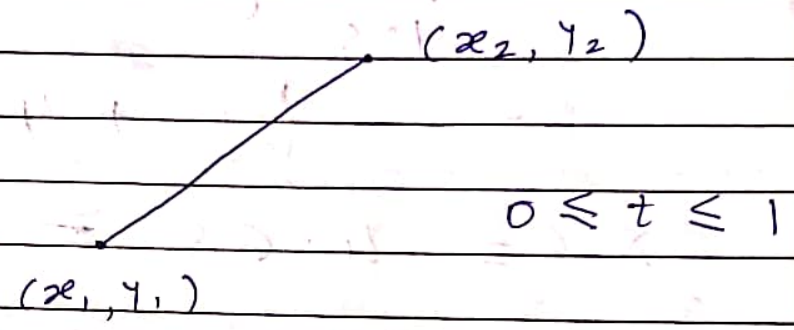
Since Pi_1 subline is outside the window means we have to discard this line.

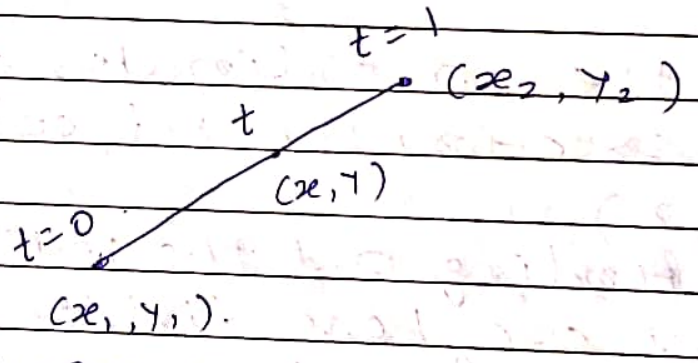


PAGE No.
DATE / /

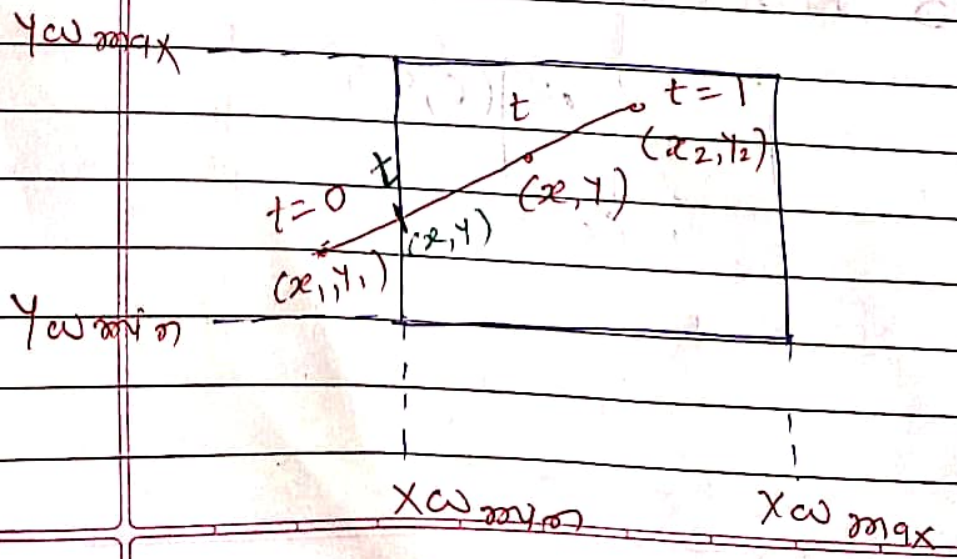
Liang Barsky Line clipping Algorithm

- This algorithm uses Parametric equation of line.
- We have to draw a line in 1 sec





suppose we want to find the x & y values at particular time like shown in fig.



PAGE No.
DATE / /

Line Parametric equation

$$x = (1-t)x_1 + t x_2$$

$$y = (1-t)y_1 + t y_2$$

$$x = (1-t)x_1 + t x_2$$
$$= x_1 - t x_1 + t x_2$$
$$x = x_1 + t(x_2 - x_1)$$
$$\boxed{x = x_1 + t\Delta x}$$
$$y = (1-t)y_1 + t y_2$$
$$= y_1 - t y_1 + t y_2$$
$$= y_1 + t(y_2 - y_1)$$
$$\boxed{y = y_1 + t\Delta y}$$

As per point clipping algorithm

$$x_{wmin} \leq x \leq x_{wmax}$$

$$y_{wmin} \leq y \leq y_{wmax}$$

$$x_{wmin} \leq x_1 + t\Delta x \leq x_{wmax}$$

$$y_{wmin} \leq y_1 + t\Delta y \leq y_{wmax}$$

$$x_1 + t\Delta x \geq x_{wmin}$$

$$x_1 + t\Delta x \leq x_{wmax}$$

$$y_1 + t\Delta y \geq y_{wmin}$$

$$y_1 + t\Delta y \leq y_{wmax}$$

PAGE No.
DATE / /

$t\Delta x \geq x_{w_{min}} - x_1 \rightarrow (1)$

$t\Delta x \leq x_{w_{max}} - x_1$

$t\Delta y \geq y_{w_{min}} - y_1 \rightarrow (2)$

$t\Delta y \leq y_{w_{max}} - y_1$

Multiply eqn (1) & (2) by -ve sign

$-t\Delta x \leq x_1 - x_{w_{min}}$

$t\Delta x \leq x_{w_{max}} - x_1$

$-t\Delta y \leq y_1 - y_{w_{min}}$

$t\Delta y \leq y_{w_{max}} - y_1$

General Equation form

$tp_k \leq q_k \quad \{k = 1, 2, 3, 4\}$

$p_1 = -\Delta x$	$q_1 = x_1 - x_{w_{min}}$
$p_2 = \Delta x$	$q_2 = x_{w_{max}} - x_1$
$p_3 = -\Delta y$	$q_3 = y_1 - y_{w_{min}}$
$p_4 = \Delta y$	$q_4 = y_{w_{max}} - y_1$

If $p_k = 0$ Line is parallel

PAGE No.
DATE / /

If $q_k < 0$ line is outside

If P_k other than nonzero
& If $P_k < 0$ then t_1 value find

the new value of t

$$t_1 = \max\left(0, \frac{q_k}{P_k}\right)$$

else mean $P_k > 0$
then t_2 find

$$t_2 = \min\left(1, \frac{q_k}{P_k}\right)$$

Now check $t_1 > t_2$
line is completely outside &
reject it

If $t_1 < t_2$ then we will use
it

$$x = x_1 + t\Delta x$$
$$y = y_1 + t\Delta y$$

Steps (Algorithm)

1. Take the endpoints of line (x_1, y_1) to (x_2, y_2)
2. calculate $\Delta x, \Delta y, P_1, P_2, P_3, P_4, q_1, q_2, q_3, q_4$
3. Assign $t_1 = 0$, $t_2 = 1$

PAGE No.
DATE / /

i) If $P_k = 0$ $(k = 1, 2, 3, 4)$
line is parallel to window

ii) If $q_k < 0$ $(k = 1, 2, 3, 4)$
line is outside

iii) for nonzero value of P_k
if $P_k < 0$ (t_1 find)

$$t_1 = \max\left(0, \frac{q_k}{P_k}\right)$$

else $P_k > 0$ (t_2 find)

$$t_2 = \min\left(1, \frac{q_k}{P_k}\right)$$

If $t_1 > t_2$ line is completely outside & reject it

If $t_1 < t_2$ find (x, y) by using

$$x = x_1 + t\Delta x$$
$$y = y_1 + t\Delta y$$

PAGE No.
DATE / /

Example

① Window A (20,20), B (90,20), C (90,70) D (20,70). Line P_1 (10,30), P_2 (80,90)

⇒

$x_{wmin} = 20$
$y_{wmin} = 20$
$x_{wmax} = 90$
$y_{wmax} = 70$

$x_1 = 10$
$y_1 = 30$
$x_2 = 80$
$y_2 = 90$

$\therefore \Delta x = x_2 - x_1$
$= 80 - 10$
$\Delta x = 70$

$\Delta y = y_2 - y_1$
$= 90 - 30$
$\Delta y = 60$

$p_1 = -\Delta x = -70$
$p_2 = \Delta x = 70$
$p_3 = -\Delta y = -60$
$p_4 = \Delta y = 60$

$q_1 = x_1 - x_{wmin}$
$= 10 - 20$
$q_1 = -10$

PAGE No.
DATE / /

$q_2 = x_{w\,max} - x_1$
$= 90 - 10$
$q_2 = 80$

$q_3 = y_1 - y_{w\,min}$
$= 30 - 20$
$q_3 = 10$

$q_4 = y_{w\,max} - y_1$
$= 70 - 30$
$q_4 = 40$

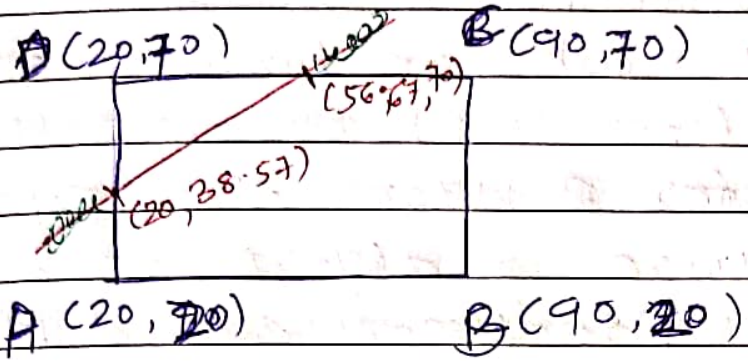
$p_k < 0$	$p_k > 0$
p_1, p_3	p_2, p_4
$t_1 = \max\left(0, \frac{-10}{-70}, \frac{10}{-60}\right)$	$t_2 = \min\left(1, \frac{80}{70}, \frac{40}{60}\right)$
$t_1 = \frac{1}{7}$	$t_2 = \frac{2}{3}$

$x = x_1 + t\Delta x$
$y = y_1 + t\Delta y$

$x = 10 + \frac{1}{\cancel{7}} \times \cancel{70}^{10}$	$x = 10 + \frac{2}{3} \times 70$
$x = 20$	$x = 56.66$
$y = 30 + \frac{1}{7} \times 60$	$y = 30 + \frac{2}{\cancel{3}_1} \times \cancel{60}^{20}$
$y = 38.57$	$y = 70$

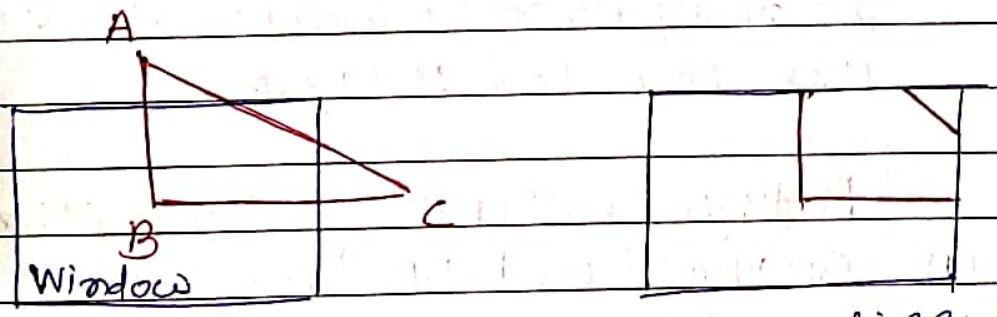
PAGE No.
DATE / /

(20, 38.57) | (56.67, 70)



* Polygon clipping

- In the last section we have seen how to clip line
- Now we will see how to clip a polygon.
- Basically polygon is a set of lines only. But it is a closed figure.
- So we can use line clipping algorithm to clip a polygon

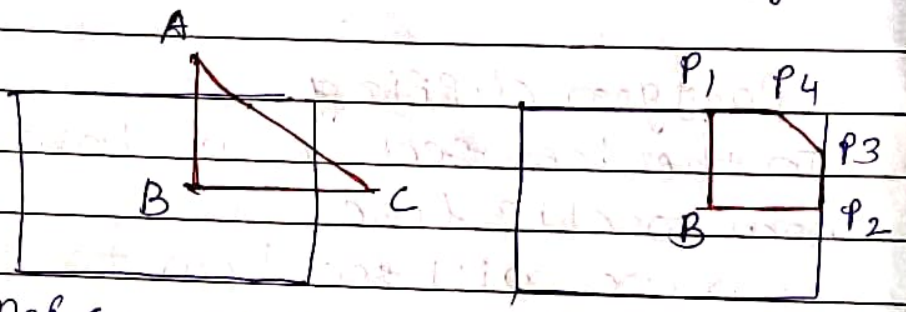


Before clipping After clipping

- If we are using line clipping algorithm, then each line of polygon is separately considered & clipped
- After clipping we will get again a set of lines.
- But can we call that set of lines as a polygon? surely not, because

PAGE No.
DATE / /

these lines are not forming a close figure.
That's why we have to modify the line clipping algorithm so for polygon clipping, we require an algorithm which will produce a closed figure



Before clipping After clipping

- clipping a polygon may produce a polygon with fewer vertices or one with more vertices than the original one.
It may even produce several unconnected polygon.

Polygon clipping Algorithm
1. Sutherland Hodgeman
2. Weiler - Atherton

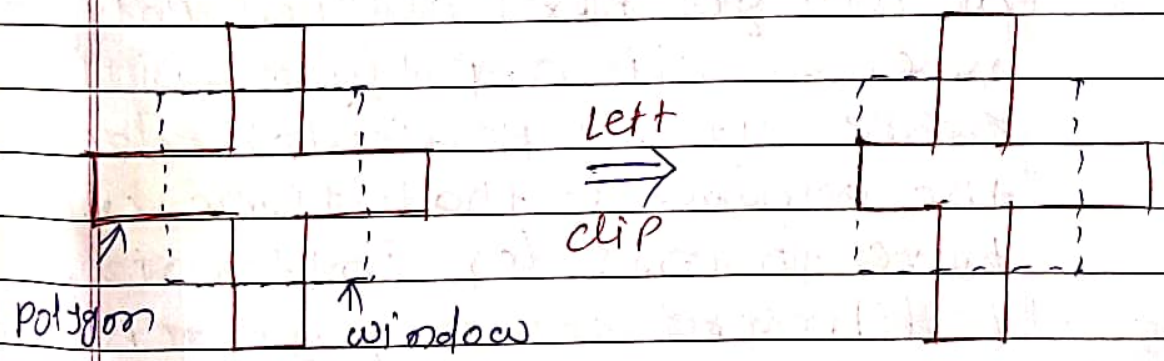
1. Sutherland Hodgeman
- We can clip a polygon by considering whole polygon against each boundary edge of the window.
- To represent a polygon we need a set of vertices.

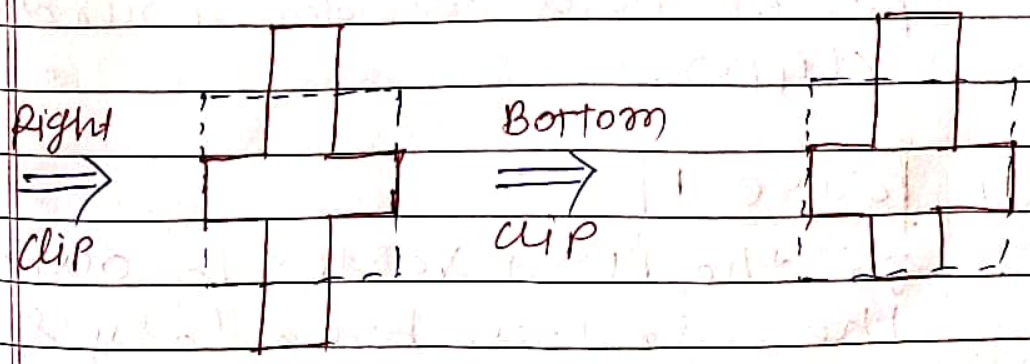
PAGE No.
DATE / /

- We will pass pass this set of vertices of a polygon to a procedure which will clip the polygon against left edge of the window.
- This left clip procedure generates new set of vertices which indicates left clipped polygon.

Again this new set of vertices is passed to the right boundary clipper procedure.

Again we will get new set of vertices. Then we will pass this new set of vertices to bottom boundary clipper & lastly to top boundary clipper procedure.



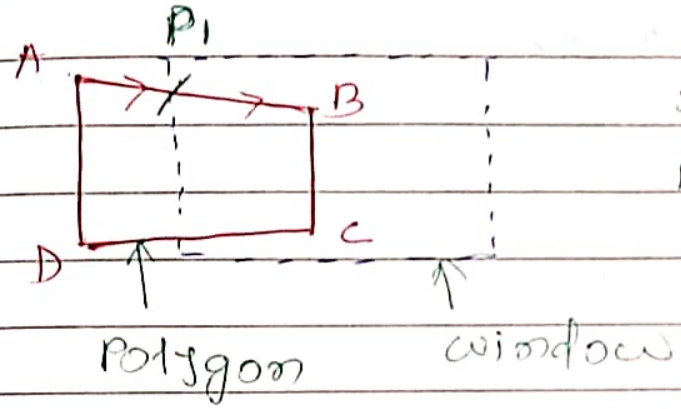


- Above fig shows 4 different stages which are required to clip a polygon.
- At the end of every clipping stage a new set of vertices is generated & this new set of modified polygon is passed to the next clipping stage.
- After clipping a polygon with respect to all the 4 boundaries we will get final clipped polygon
- When we clip a polygon with respect to any particular edge of the window at that time we have to consider following 4 different cases, as each pair of adjacent polygon vertices is passed to a window boundary clipper.

① **Case 1**

If the first vertex is outside the window boundary & the second vertex is inside the window, then the intersection

point of polygon with boundary edge of window & the vertex which is inside the window is stored in a output vertex list (i.e. it will act as new vertex points)

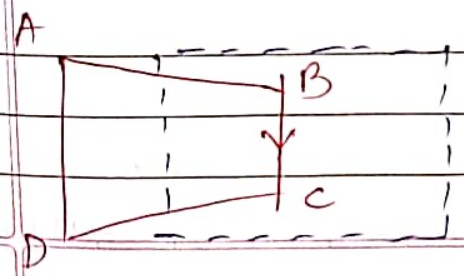


store P_1 & B in output vertex list.

So for edge AB instead of storing vertex A & B we are storing P_1 & B in output vertex list.

case 2

If both i.e. first & second vertices of a polygon are lying inside the window, then we have to store the second vertex only in output vertex list

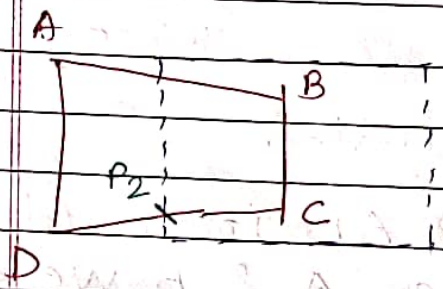


store only C in output vertex list

PAGE No.
DATE / /

Case 3

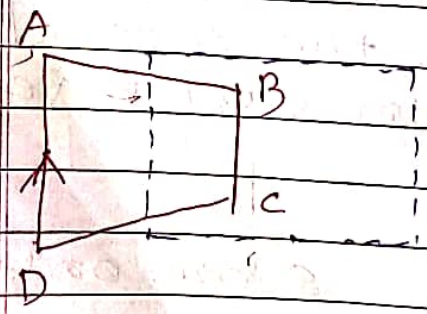
- If the first vertex is inside the window & second vertex is outside the window i.e. opposite to case 1, then we have to store only intersection point of that edge of polygon with window in output vertex list



Store P_2 only in output vertex list

Case 4

- If both first & second vertices of a polygon are lying outside the window then no vertex is stored in output vertex list

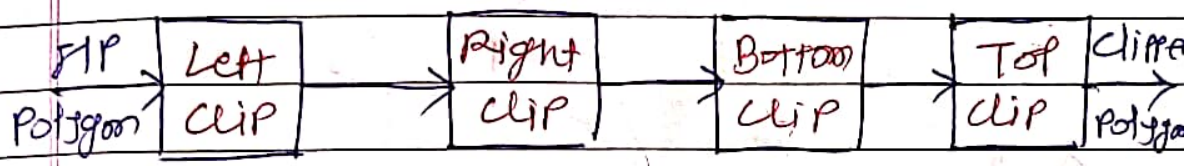


Nothing is stored in output vertex list

Once all vertices have been considered for one clip window

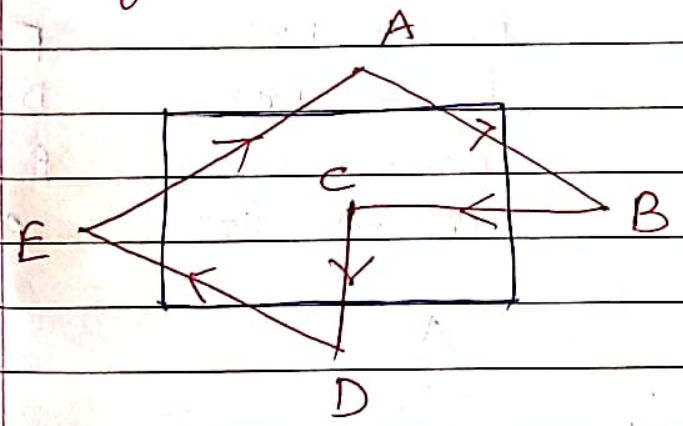
PAGE No.
DATE / /

boundary, the output list of vertices is clipped against the next window boundary.
So the block diagram for this algorithm will be as shown in fig.



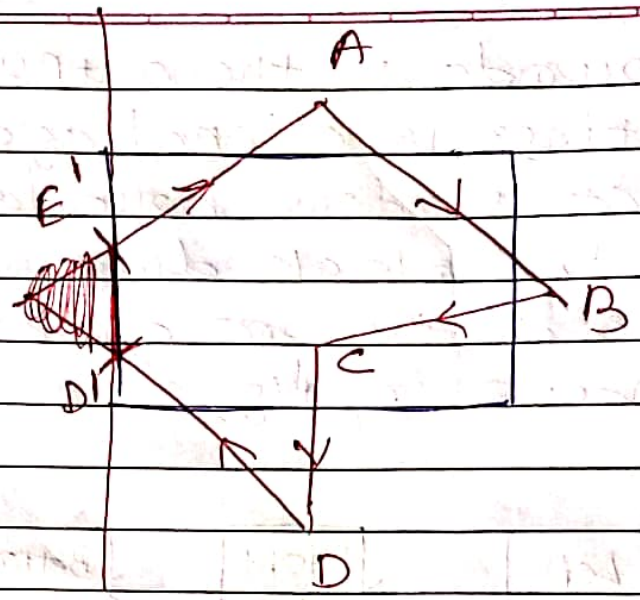
Example

① Clip the polygon using Sutherland Hodgeman polygon clipping algorithm



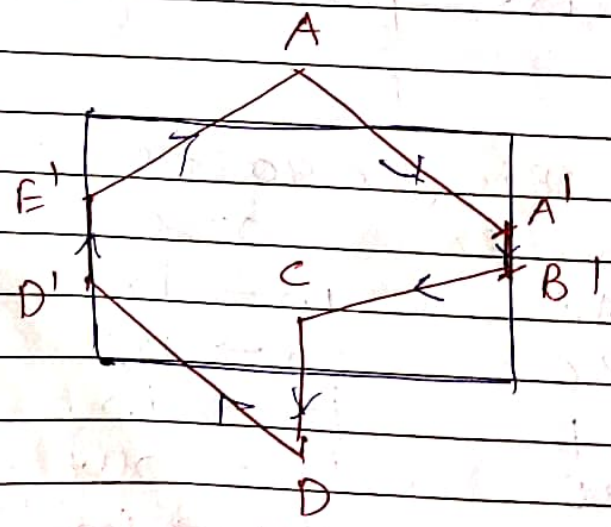
⇒ Clipping against left edge

	Vertex	case	output
①	AB	in → in	B
②	BC	in → in	C
③	CD	in → in	D
④	DE	in → out	D'
⑤	EA	out → in	E' A



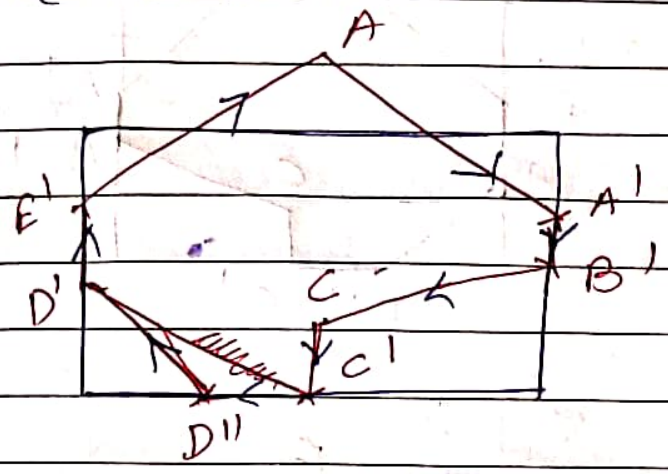
(2) clipping against right edge of window

	vertex	case	output
(1)	AB	in → out	A'
(2)	BC	out → in	B' C
(3)	CD	in → in	D
(4)	DD'	in → in	D'
(5)	D'E'	in → in	E'
(6)	E' A	in → in	A



PAGE No.
DATE / /

(3) Clipping against Bottom edge of window

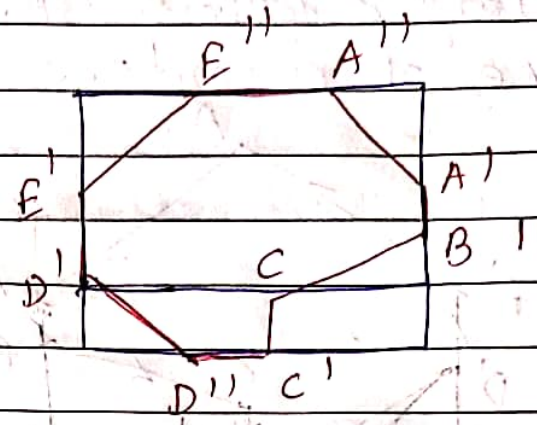


	Vertex	Case	Output
(1)	AA'	in → in	A'
(2)	A'B'	in → in	B'
(3)	B'C	in → in	C
(4)	CD	in → out	C'
(5)	DD'	out → in	D'' D'
(6)	D'E'	in → in	E'
(7)	E'A	in → in	A

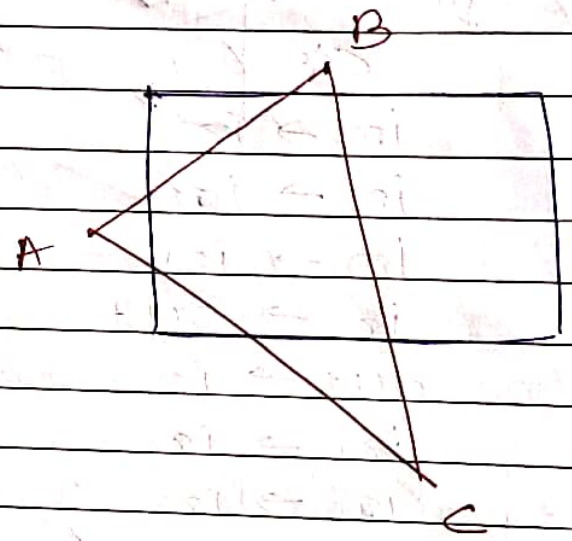
(4) Clipping against top edge of window

	Vertex	Case	Output
(1)	AA'	out → in	A'' A'
(2)	A'B'	in → in	B'
(3)	B'C	in → in	C
(4)	CC'	in → in	C'
(5)	C'D''	in → in	D''
(6)	D''D'	in → in	D'
(7)	D'E'	in → in	E'
(8)	E'A	in → out	E''

PAGE No.
DATE / /



(2)

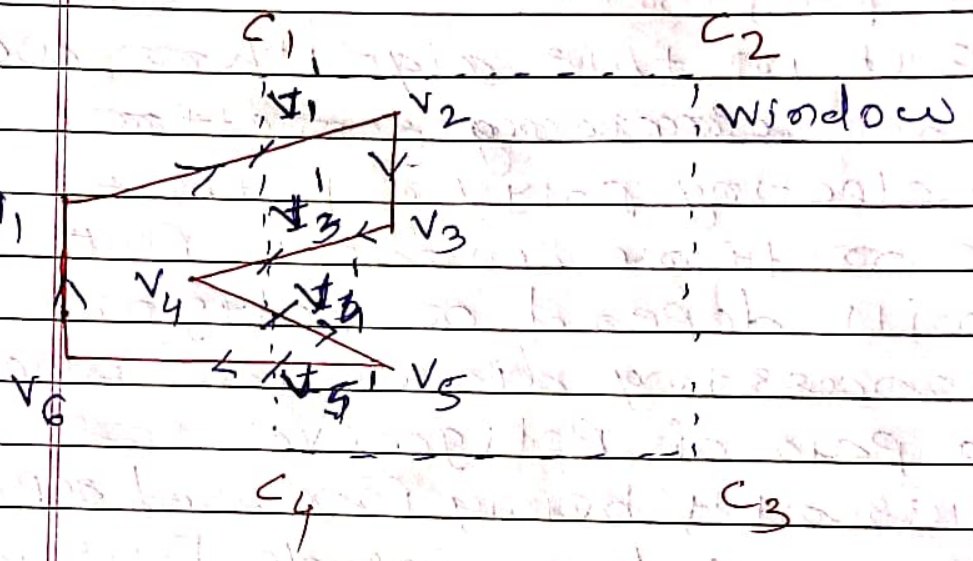


PAGE No.
DATE / /

② Weiler Atherton Polygon clipping

- This is one of the generalized polygon clipping algorithm
- In sutherland Hodgeman we always follow the path of polygon.
- But in this algorithm we are ~~[illegible]~~ sometime selecting polygon path & sometime window path, that will depend on polygon processing direction & whether a pair of polygon vertices currently being processed represent an outside to inside pair or an inside to outside pair
- We can follow polygon processing path either clockwise or anti-clockwise.
- When we are following a path of polygon in clockwise direction at that time we have to use following rules.
- We have to follow the polygon boundary, same as that of sutherland Hodgeman algorithm, if the vertex pair is outside to inside.

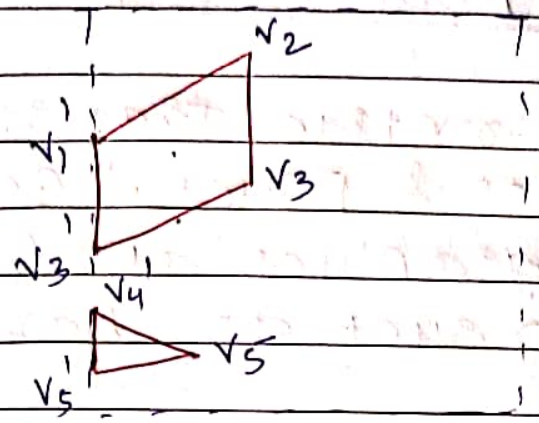
We have to follow the window boundary in clockwise direction, if the vertex pair is inside to outside.



Original Polygon	Clipped Window
V_1	C_1
V_1'	C_2
V_2	C_3
V_3	C_4
V_3'	V_5'
V_4	V_4'
V_4'	V_3'
V_5	V_1'
V_5'	
V_6	
V_1	

first polygon = $\{V_1' - V_2 - V_3 - V_3' - V_1'\}$

second polygon = $\{V_4' - V_5 - V_5' - V_4'\}$



After clipping.

PAGE No.
DATE / /

MODULE - 5

3D Geometric Transformations, Curves & Fractal Generation

- Some graphics applications are 2D such as charts, graphs, maps.
- But we live in a world of 3D. In many design applications we must deal with objects describing in 3D.

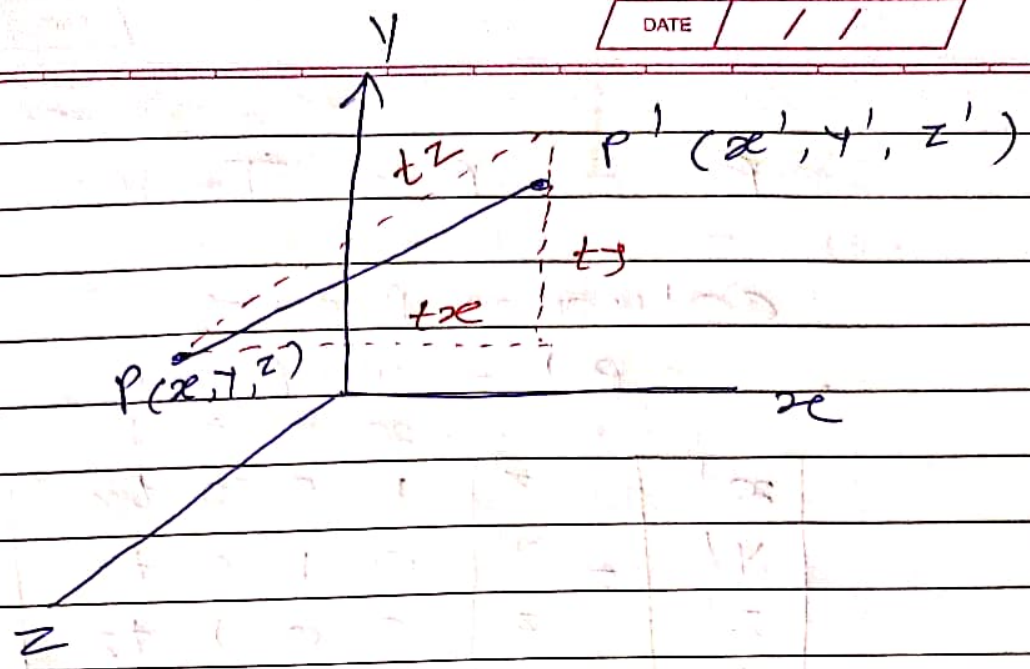
Basic 3D Transformations

- 3D transformations are similar to our normal 2D transformations
- Rather in 3D we are extending 2D transformations by considering one additional parameter i.e. z-co-ordinate
- Basic 3D transformations are as follows.

1. Translation
2. Scaling
3. Rotation.
4. Reflection

1. 3D Translation -

Used to ~~change~~ shift a point or moving whole object



- In 2D for transformation we have used a 3×3 matrix
- In 3D we need 3 translation factor. So the transformation matrix in 3D will be of 4×4

$$x' = x + tx$$
$$y' = y + ty$$
$$z' = z + tz$$
$$1 = 1$$

Row major

$$[x'\ y'\ z'] = [x\ y\ z] \cdot T$$
$$P' = P \cdot T$$

$$[x'\ y'\ z'] = [x\ y\ z] \cdot \begin{array}{c|cccc} & x' & y' & z' & 1 \\ \hline x & 1 & 0 & 0 & 0 \\ y & 0 & 1 & 0 & 0 \\ z & 0 & 0 & 1 & 0 \\ 1 & tx & ty & tz & 1 \end{array}$$

PAGE No.
DATE / /

$$p' = p \cdot T$$

Column Major

$$p' = T \cdot p$$

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

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

(1) A point has a co-ordinates (x, y, z) in direction $(5, 6, 7)$

Translation is done in x direction by 3 coordinates, y direction in 3 coordinates & z direction in 2 coordinates of point are (5, 6, 7)

Translation vector

$t_x = 3, \; t_y = 3, \; t_z = 2$

Translation matrix $= \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$

$$p' = T \cdot p$$

PAGE No.
DATE / /

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

$$= \begin{bmatrix} 8 \\ 9 \\ 9 \\ 1 \end{bmatrix}$$

$\therefore x' = 8, \; y' = 9, \; z' = 9$

(2) 3D Scaling -

Scaling is the translation which will change the object size based on scaling factors S_x, S_y, S_z along the x-axis, y-axis & z-axis.

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

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

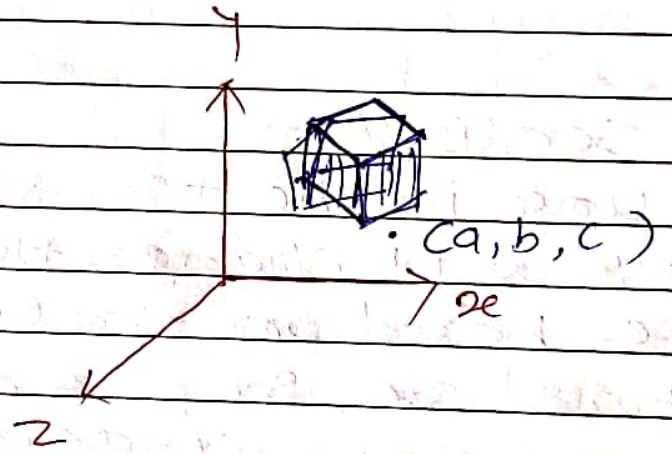
S_x, S_y & S_z are scaling factors

PAGE No.
DATE / /

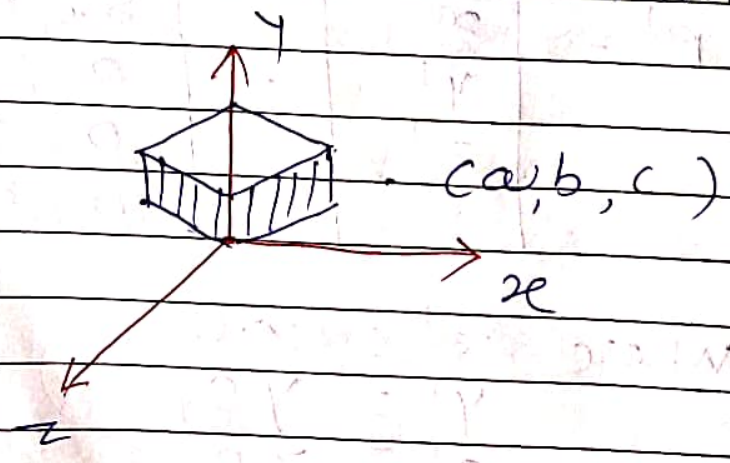
Scaling of object relative to fixed point (a, b, c)

- Translate fixed point to the origin
- Scale the object relative to the origin
- Translate a object back to its original position

① object & point for scaling

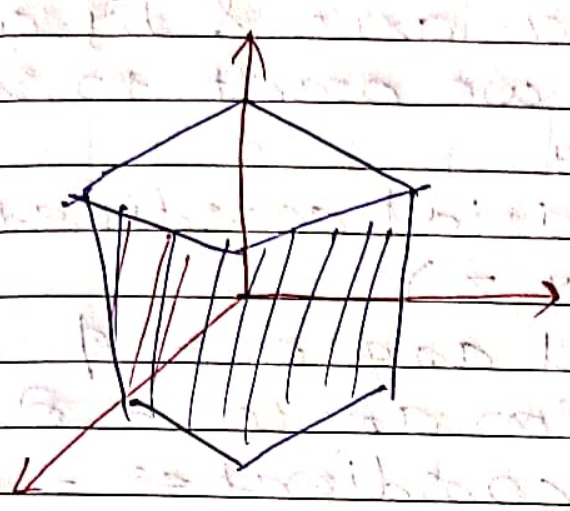


② Object translation at origin

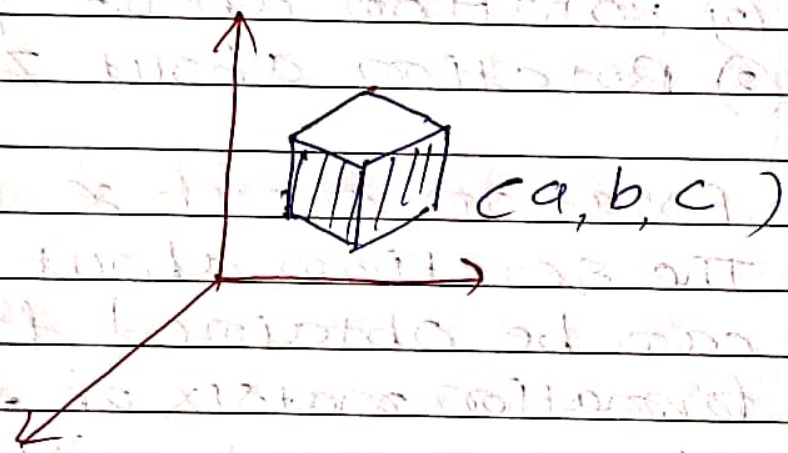


PAGE No.
DATE / /

(3) Scaling of object



(4) object shifted to its original position after scaling.



(3) 3D Rotation -

- 3D rotation is rotating an object with respect to an angle
- In this we are performing rotation in 3 different planes i.e. XY, XZ & YZ
- Rotation can be performed in clockwise or in anticlockwise direction.

PAGE No.
DATE / /

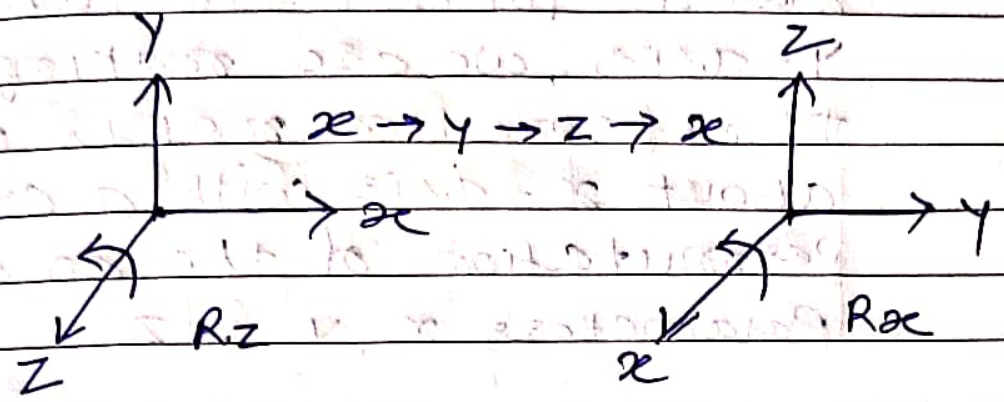
- For rotation we must specify to which axis & by which angle we want to rotate the object.
- Let initial coordinates are $P(x, y, z)$.
  initial angle $= \phi$
  Rotation angle $= \theta$
  New coordinates $= (x', y', z')$
- 3 possible types of rotation are
  1. Rotation about x-axis
  2. Rotation about y-axis
  3. Rotation about z-axis

1) Rotation about x-axis
- The rotation about x-axis can be obtained from transformation matrix of rotation about z-axis with a cyclic permutation of the co-ordinate parameters x, y and z.
- It means we have to replace z co ordinate by x co-ordinate y by z & x by y co-ordinates

$$\rightarrow x \rightarrow y \rightarrow z$$

PAGE No.
DATE / /

We can pictorially represent the equation:



The equation of rotation about x-axis is

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

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

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

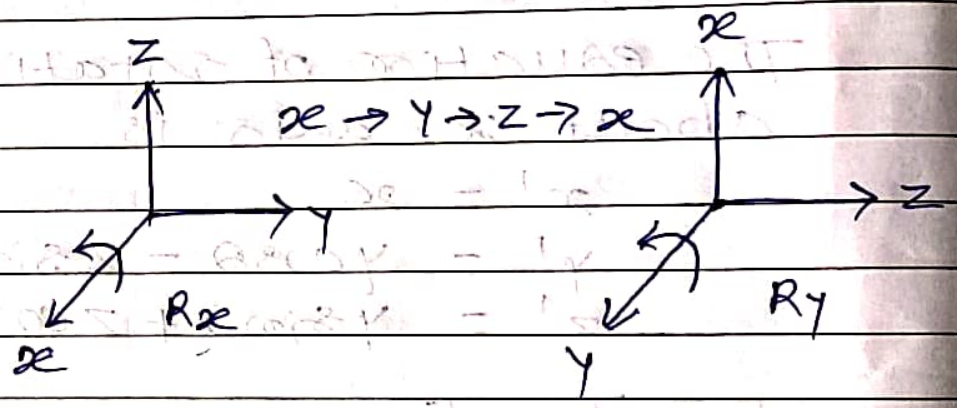
$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ for clock wise

$R_x \rightarrow$ Rotation about a axis

PAGE No.
DATE / /

② Rotation about Y-axis

To form a transformation matrix for rotation about Y-axis, we are making use of transformation matrix of rotation about x-axis with a cyclic permutation of the co-ordinate parameters x, y & z



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

Here y co ordinate remains unchanged & x & z values gets changed

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

PAGE No.
DATE / /

Therefore the transformation matrix will be

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

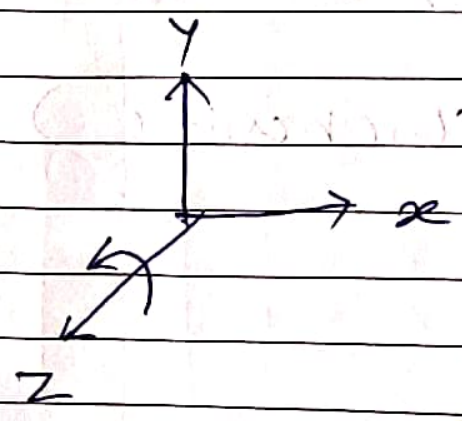
(for anti clockwise)

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

(for clockwise)

3) Rotation about z axis

In rotation about z axis, z co ordinates remain unchanged and x & y co ordinates changes



PAGE No.
DATE / /

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

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

$z' = z$

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

The transformation matrix

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

(for anticlockwise)

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

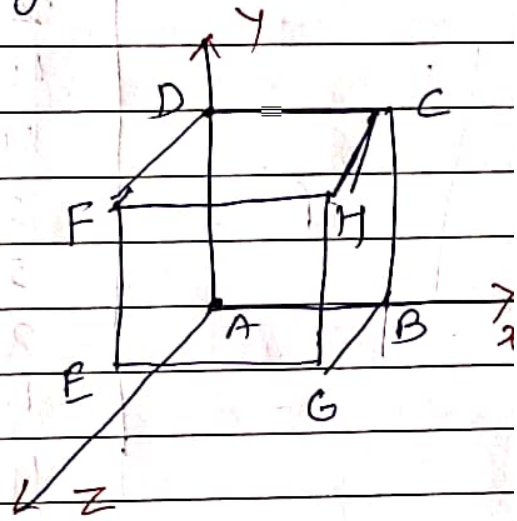
(for clockwise)

PAGE No.
DATE / /

① A cube is defined by 8 vertices A(0,0,0), B(2,0,0), C(2,2,0), D(0,2,0), E(0,0,2), F(0,2,2), G(2,0,2), H(2,2,2)
perform the following transformations on the above cube
i) Translation ($t_x = 2$, $t_y = 4$, $t_z = 0$)
ii) Scaling ($S_x = 0.5$, $S_y = 1$, $S_z = 1$)

⇒ A cube is having 8 vertices

0	0	0
2	0	0
2	2	0
0	2	0
0	0	2
0	2	2
2	0	2
2	2	2



① Translation

Here we have to represent a cube matrix in homogeneous co-ordinate form

$t_x = 2$
$t_y = 4$
$t_z = 0$

$P' = P \cdot T$

PAGE No.
DATE / /

(1)

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

$$P' = \begin{bmatrix} 2 & 4 & 0 & 1 \\ 4 & 4 & 0 & 1 \\ 4 & 6 & 0 & 1 \\ 2 & 6 & 0 & 1 \\ 2 & 4 & 2 & 1 \\ 2 & 6 & 2 & 1 \\ 4 & 4 & 2 & 1 \\ 4 & 6 & 2 & 1 \end{bmatrix}$$

(2) <u>Scaling</u>

$S_x = 0.5, \; S_y = 1, \; S_z = 1$

$P' = T \cdot P$

$$P' = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 1 \\ 2 & 2 & 0 & 1 \\ 0 & 2 & 0 & 1 \\ 0 & 0 & 2 & 1 \\ 0 & 2 & 2 & 1 \\ 2 & 0 & 2 & 1 \\ 2 & 2 & 2 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

PAGE No.
DATE / /

$$P' = \begin{vmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 2 & 0 & 1 \\ 0 & 2 & 0 & 1 \\ 0 & 0 & 2 & 1 \\ 0 & 2 & 2 & 1 \\ 1 & 0 & 2 & 1 \\ 1 & 2 & 2 & 1 \end{vmatrix}$$

(2) A homogenous co-ordinate point P [3, 2, 1] is translated in x, y & z direction by -2, -2 & -2 unit respty followed by successive rotation 60° about x-axis. Find the final position of homogenous co-ordinate.

⇒ (1) Translation -

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

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

PAGE No.
DATE / /

(2) Rotation 60° about x-axis

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

$$\theta = 60^\circ$$

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.5 & 0.866 & 0 \\ 0 & -0.866 & 0.5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Resultant transformation

$$T_R = T \times R$$

$$T_R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -2 & -2 & -2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.5 & 0.866 & \\ 0 & -0.866 & 0.5 & \\ 0 & 0 & 0 & \end{bmatrix}$$

$$T_R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.5 & 0.866 & 0 \\ 0 & -0.866 & 0.5 & 0 \\ -2 & 0.732 & -2.732 & 1 \end{bmatrix}$$

Final Position = Initial Position $\times T_R$

PAGE No.
DATE / /

$P' = P[3\ 2\ 1\ 1] \times T_R$

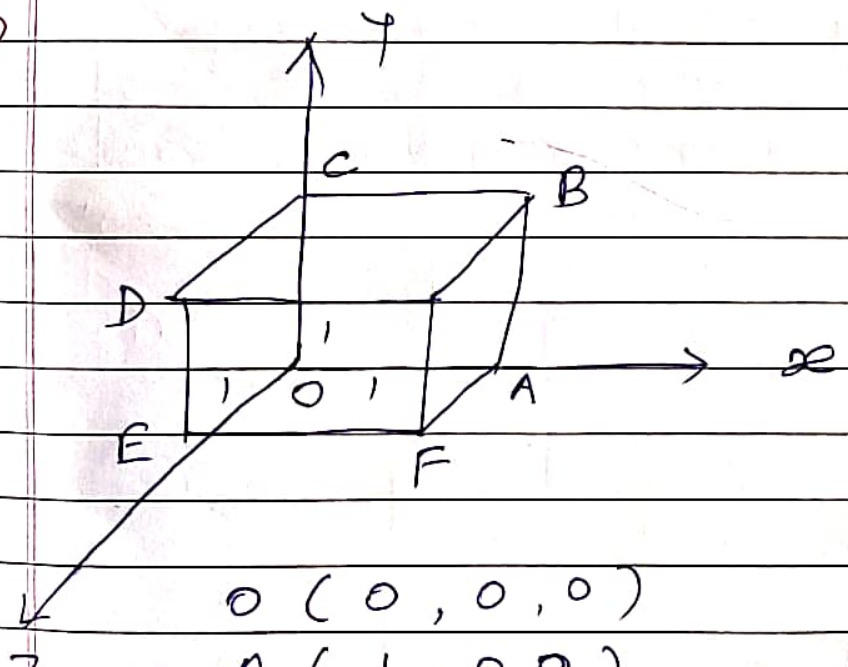
$P' = [1\ \ 0.864\ \ -0.5\ \ 1]$

Final Position

$P' = \{1, 0.864, -0.5, 1\}$

(3) There is a unit cube which is translated with 2 units in x-axis, 3 units in y axis & 4 units in z direction

⇒



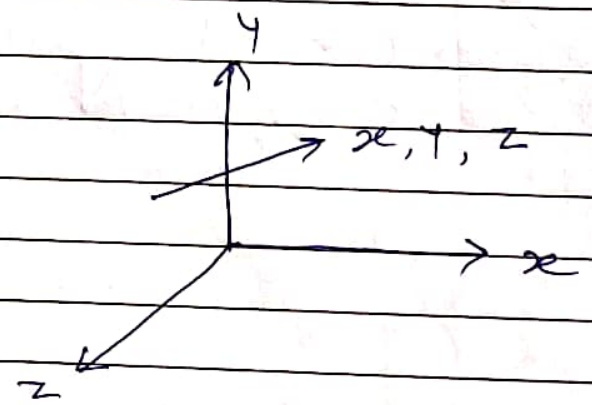
O (0, 0, 0)
A (1, 0, 0)
B (1, 1, 0)
C (0, 1, 0)
D (

Arbitary → undetermined, not assigned a specific value.

Page No.	
DATE	/ /

* Rotation about an Arbitary axis

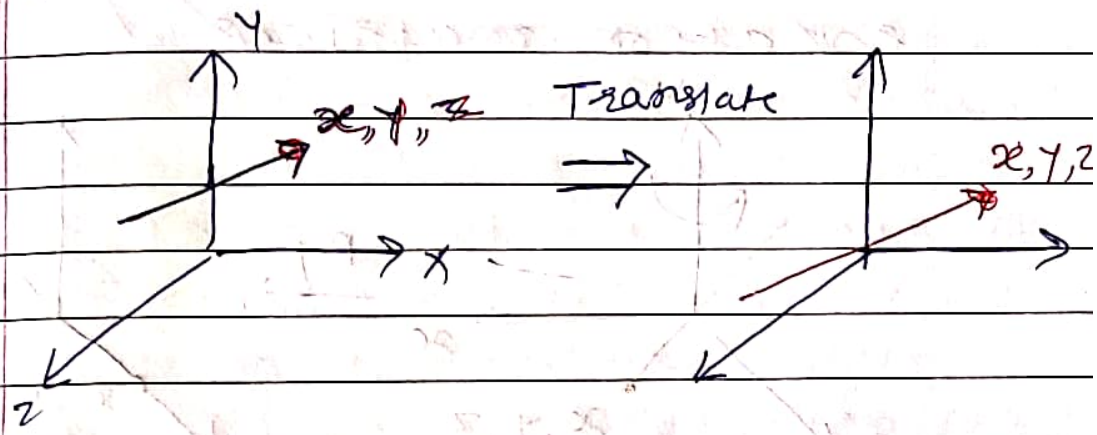
Until now we have seen how to rotate a point or object with respect to co-ordinate axes such as x, y & z.
But suppose we want to rotate a point or object at angle θ about an arbitary line which does not coinside with any of the co-ordinate axes, then we have to set up a composite transformations involving combination of translations & the co-ordinate axes rotations.



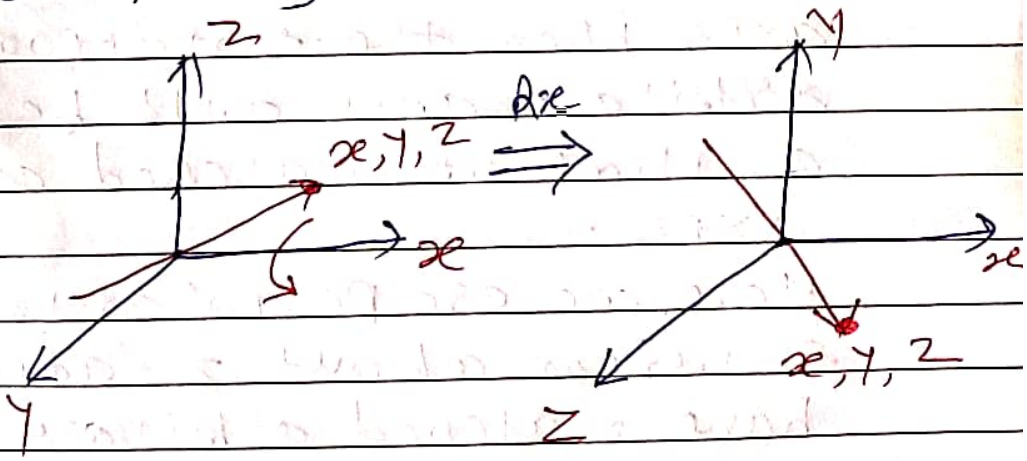
If we we want to rotate a point (x, y, z) by an angle θ, which is in space & lying on an arbitary axis as shown in fig. Then we have to follow the steps as

below in sequence

① Translate the arbitary axis so that it will pass through the origin.
When we are translating axis a point also gets translated.



② perform rotation about x-axis until the arbitary axis completely lies in xz plane

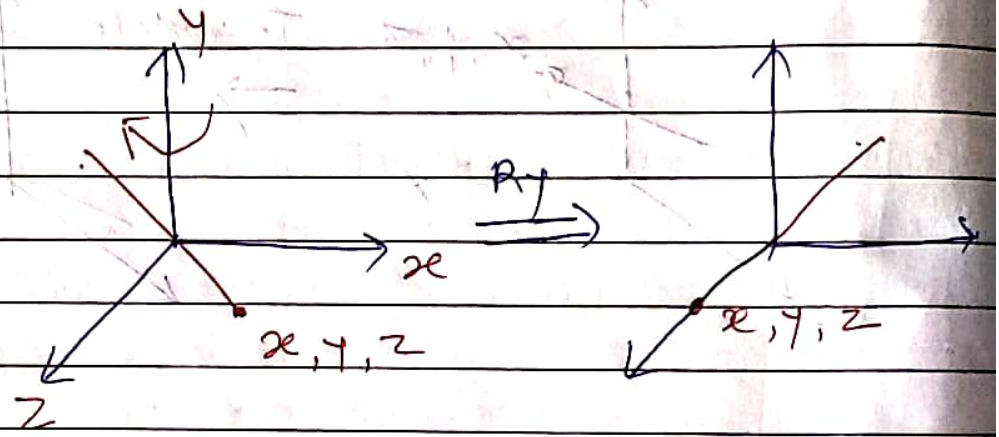


③ Now perform rotation about y-axis until the arbitary axis coincides with

PAGE No.
DATE / /

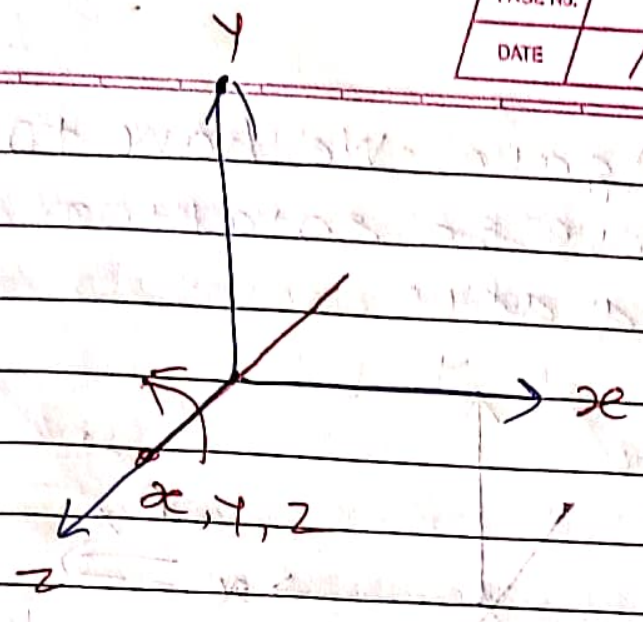
Z - axis.

We can also perform rotation about y - axis until arbitary axis coincides with x - axis. Here we are matching the arbitary axis to any one co-ordinate axis. So that we can make use of standard rotation matrices.



(4) one the arbitary axis coincides with any one co-ordinate axis then the rotation of arbitary axis will be same as that of matched co-ordinate axis.

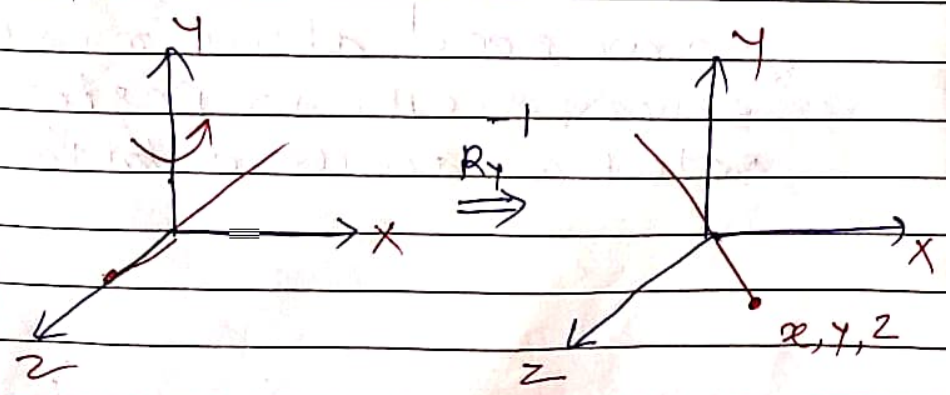
Here we are performing rotation about z - axis as we have matched arbitary axis to z - axis



5) After performing rotation about z-axis (i.e. rotation about arbitrary axis). We have to perform reverse transformation. Here we should not perform reverse rotation about z-axis because it is rotation about arbitrary axis.

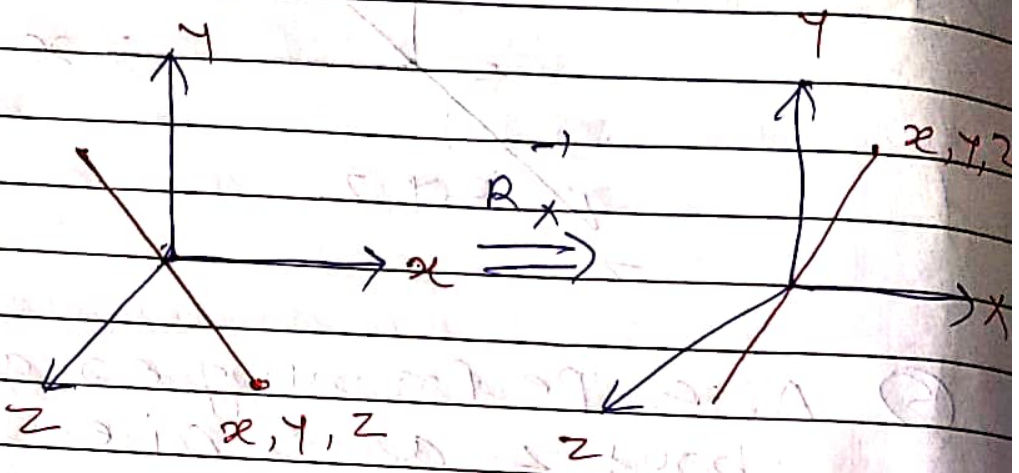
So we have to perform <u>reverse rotation about y-axis</u>.

i.e. if earlier we rotated y-axis in anticlockwise direction, now we have to rotate it in clockwise direction

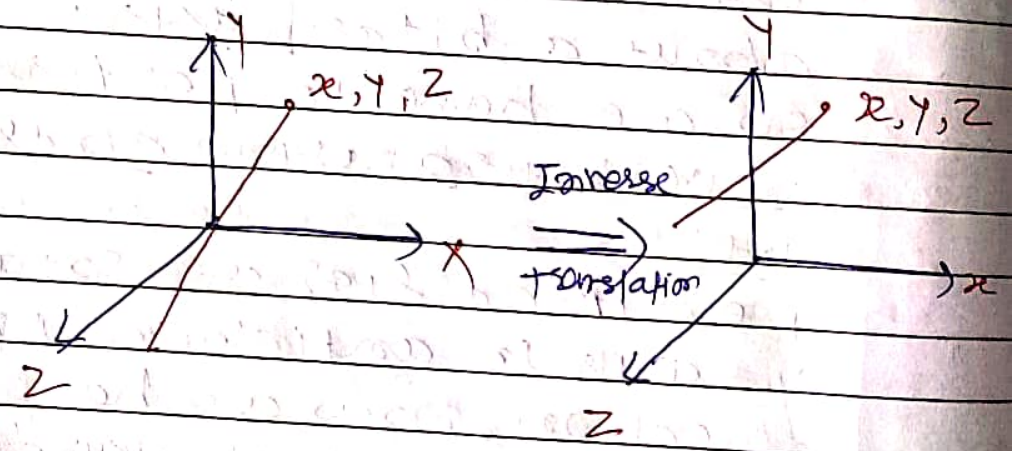


PAGE No.
DATE / /

(6) Again we have to perform reverse rotation about a axis.



(7) Lastly we have to apply inverse translation to bring the arbitary axis back to its original position.

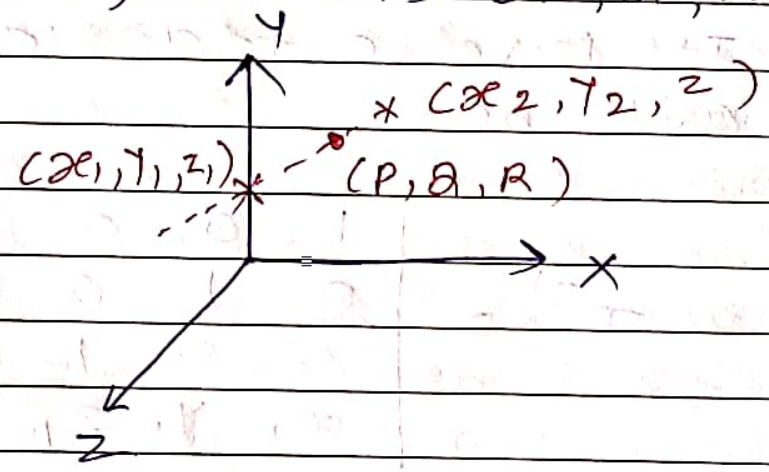


So we need above mentioned seven steps to perform rotation about arbitary axis

PAGE No.
DATE / /

* Now we can determine the transformation matrix for rotation about arbitary axis.

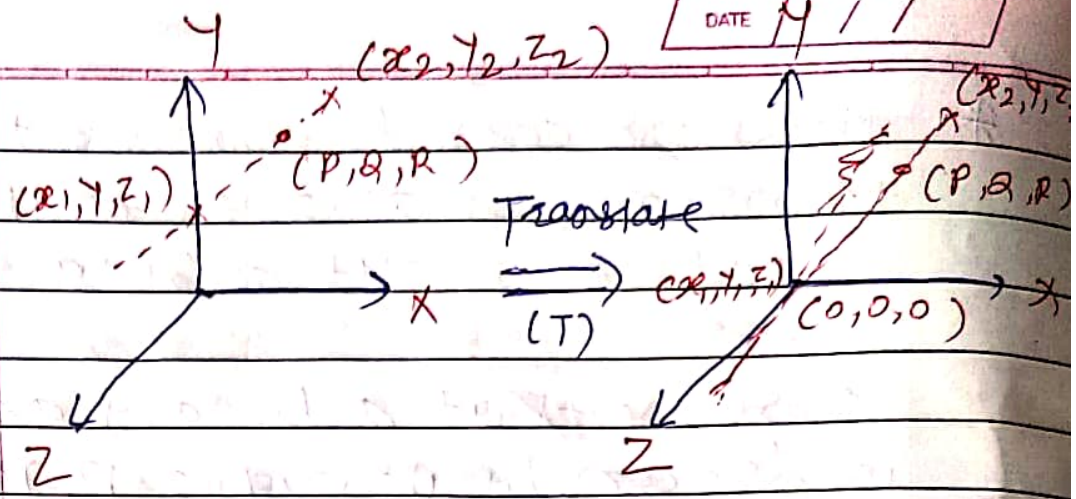
- Assume that arbitary axis is drawn by using 2 points (x_1, y_1, z_1) and (x_2, y_2, z_2).
- Let us say point on this arbitary axis is (P, Q, R)



① Translation Matrix will be

$$T = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_1 & -y_1 & -z_1 & 1 \end{vmatrix}$$

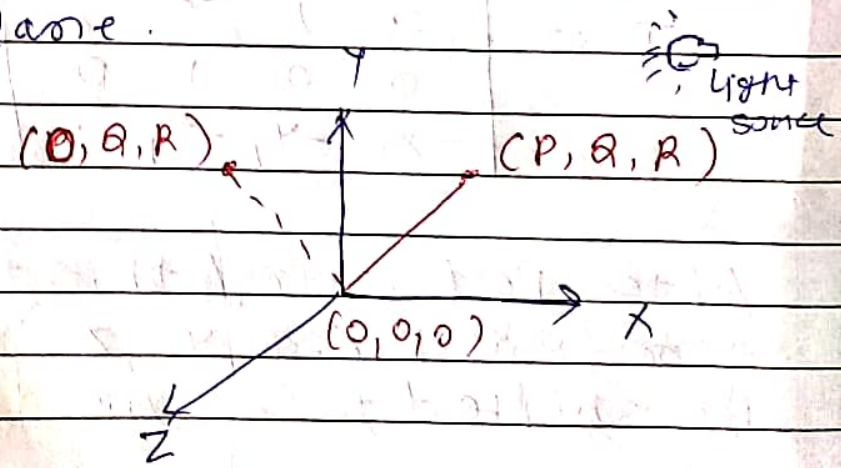
After performing this translation point (x_1, y_1, z_1) will be shifted to origin.



- Immediately we will find Inverse translation also
- Its inverse translation matrix will be

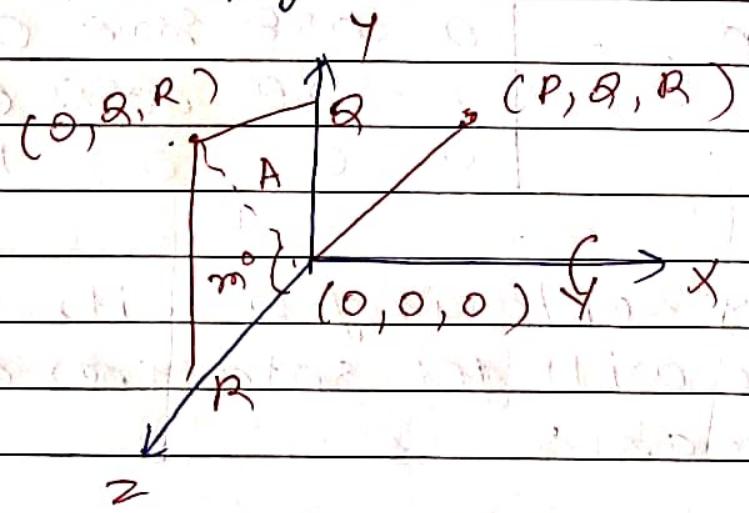
$$T^{-1} = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x_1 & y_1 & z_1 & 1 \end{vmatrix}$$

* Now the next step is rotation about X-axis, so that the arbitary axis will lie in XZ plane.



PAGE No.
DATE / /

- Here we are considering a line segment (0,0,0) to (P,Q,R) which is a part of rotation axis.
- When we are focusing some ray's in x-direction this line segments shadow will lie on yz plane & the length of shadow will be (0,0,0) to (0,Q,R)
- Now if we perform rotation about x-axis until arbitary axis is in xz plane, the shadow of line (0,0,0) & (0,Q,R) will coincide with z-axis as shown in fig



- Now we have to decide by how much angle we have to rotate the shadow, so that it will match to the z-axis let's call that angle as 'M' & length of shadow as 'A'

PAGE No.
DATE / /

$\therefore$ Length of shadow (A) $= \sqrt{(Q)^2 + (R)^2}$

$$\sin M = \frac{\text{opposite side}}{\text{hypotenuse}}$$

$$\sin M = \frac{Q}{A}$$

$$\cos M = \frac{\text{Adjacent side}}{\text{hypotenuse}}$$

$$\cos M = \frac{R}{A}$$

The matrix for rotation about x-axis, in anticlockwise direction is

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

By replacing θ with angle M we will get rotation about x-axis

$$R_x = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos M & \sin M & 0 \\ 0 & -\sin M & \cos M & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

PAGE No.
DATE / /

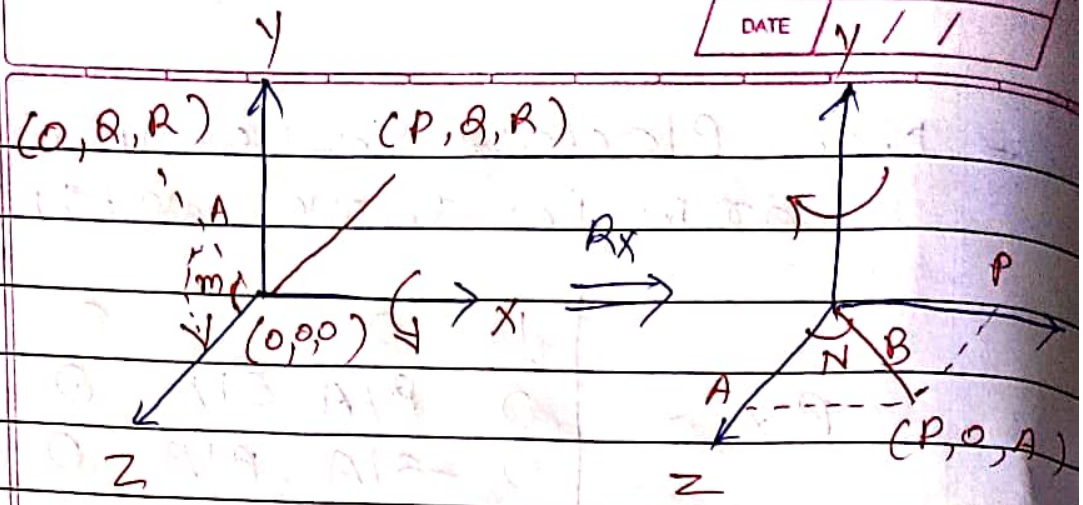
- Now Placing values of cosM & sinM in above matrix

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & R/A & Q/A & 0 \\ 0 & -Q/A & R/A & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- immediately we will find reverse rotation of this
- If earlier we have performed anticlockwise rotation, now we have to perform clockwise rotation.
- Magnitude remains unchanged
- The matrix for reverse rotation about x-axis will be

$$R_x^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & R/A & -Q/A & 0 \\ 0 & Q/A & R/A & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- After performing rotation about x-axis, the shadow will coincide with z-axis & the actual line will lie in xz plane.



When we are performing rotation about x-axis the values of x-coordinates remains unchanged. But value of y co-ordinates becomes zero because our actual point lies on XZ plane & the value of z-coordinate will be replaced by A instead of R.

So Now the actual points coordinates will be (P, 0, A)

Now let's find the length of actual point from origin

$$B = \sqrt{(P)^2 + (A)^2}$$

But $A^2 = Q^2 + R^2$

$$\therefore B = \sqrt{(P)^2 + (Q)^2 + (R)^2}$$

PAGE No.
DATE / /

Now we have to perform rotation about Y-axis in clockwise direction by angle N. So that the line of arbitary axis aligns to Z-axis

$$\therefore \sin N = \frac{\text{Opposite side}}{\text{hypotenuse}}$$

$$\sin N = \frac{P}{B}$$

$$\cos N = \frac{\text{Adjacent side}}{\text{hypotenuse}}$$

$$\cos N = \frac{A}{B}$$

We know that matrix for rotation about Y-axis in clockwise direction is

$$R_Y = \begin{vmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Replacing θ by angle N, we will get

$$R_Y = \begin{vmatrix} \cos N & 0 & \sin N & 0 \\ 0 & 1 & 0 & 0 \\ -\sin N & 0 & \cos N & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

PAGE No.
DATE / /

Replacing the values of cos N & sin N in above matrix we will get

$$R_y = \begin{vmatrix} A/B & 0 & P/B & 0 \\ 0 & 1 & 0 & 0 \\ -P/B & 0 & A/B & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Immediately we will find reverse rotation of this it will be

$$R_y^{-1} = \begin{vmatrix} A/B & 0 & -P/B & 0 \\ 0 & 1 & 0 & 0 \\ +P/B & 0 & A/B & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Here at this stage we have matched an arbitary axis to z-axis.

Now we can use our normal rotation about z-axis which will be nothing but rotation about arbitary axis by some angle θ

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

PAGE No.
DATE / /

So the final transformation matrix which will be a 3D transformation pipeline for rotation about arbitrary axis will be the product of above transformation

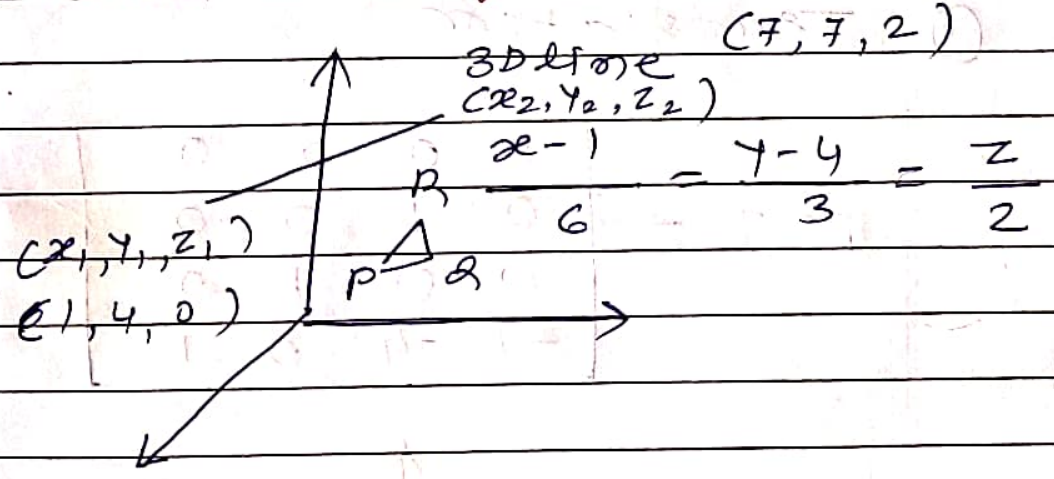
$$R_A = T \cdot R_x \cdot R_y \cdot R_z \cdot R_y^{-1} \cdot R_x^{-1} \cdot T$$

Problem

* A triangle PQR is defined by P(3, 3, 7), Q(3, 5, 7), R(5, 3, 7). Rotate this triangle by angle 55° about a line $\frac{x-1}{6} = \frac{y-4}{3} = \frac{z}{2}$

Find the co-ordinates of transformed triangle

⇒



Equation of a 3D line is

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{x_2 - x_1} = \frac{z - z_1}{z_2 - z_1}$$

$$\frac{x-1}{6} = \frac{y-4}{3} = \frac{z}{2}$$

PAGE No.
DATE / /

$x_1 = 1$
$y_1 = 4$
$z_1 = 0$

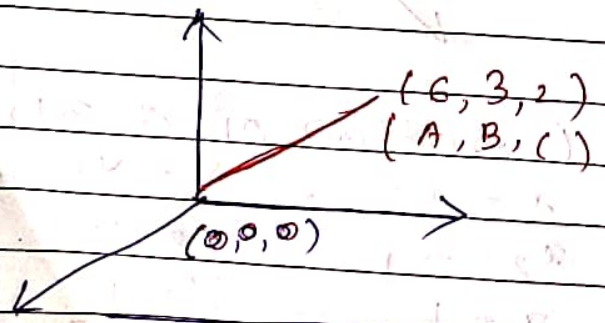
$x_2 - x_1 = 6$
$x_2 - 1 = 5 + 1$
$x_2 = 7$

$y_2 - y_1 = 3$
$y_2 - 4 = 3$
$y_2 = 3 + 4$
$y_2 = 7$

$z_2 - z_1 = 2$
$z_2 - 0 = 2$
$z_2 = 2$

(1) Translation

$$[T] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1 & -4 & 0 & 1 \end{bmatrix}$$



$A = \sqrt{Q^2 + R^2} = \sqrt{3^2 + 2^2}$
$A = \sqrt{9 + 4} = \sqrt{13}$

PAGE No.
DATE / /

Rotation about x-axis in clockwise direction.

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & R/A & Q/A & 0 \\ 0 & -Q/A & R/A & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2/\sqrt{13} & 3/\sqrt{13} & 0 \\ 0 & -3/\sqrt{13} & 2/\sqrt{13} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation about y axis in clockwise direction

$$R_y = \begin{bmatrix} A/B & 0 & -P/B & 0 \\ 0 & 1 & 0 & 0 \\ -P/B & 0 & A/B & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B = \sqrt{P^2 + Q^2 + R^2}$$

$$= \sqrt{6^3 + 3^2 + 2^2}$$

$$= \sqrt{36 + 9 + 4}$$

$$= \sqrt{49}$$

$$B = 7$$

PAGE No.
DATE / /

$$R_y = \begin{bmatrix} \sqrt{13}/7 & 0 & -6/7 & 0 \\ 0 & 1 & 0 & 0 \\ 6/7 & 0 & \sqrt{13}/7 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(4) Rotation about z axis

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

$$= \begin{bmatrix} \cos 55 & \sin 55 & 0 & 0 \\ -\sin 55 & \cos 55 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(5) $$[R_y^{-1}] = \begin{bmatrix} 0.573 & 0.819 & 0 & 0 \\ -0.819 & 0.573 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(6) $[R_z^{-1}]$

(7) $[T^{-1}]$

PAGE No.
DATE / /

Total Transformation Matrix

$$R_X = T \cdot R_X \cdot R_Y \cdot R_Z \cdot R_Y^{-1} R_X^{-1} T^{-1}$$

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

PAGE No.
DATE / /

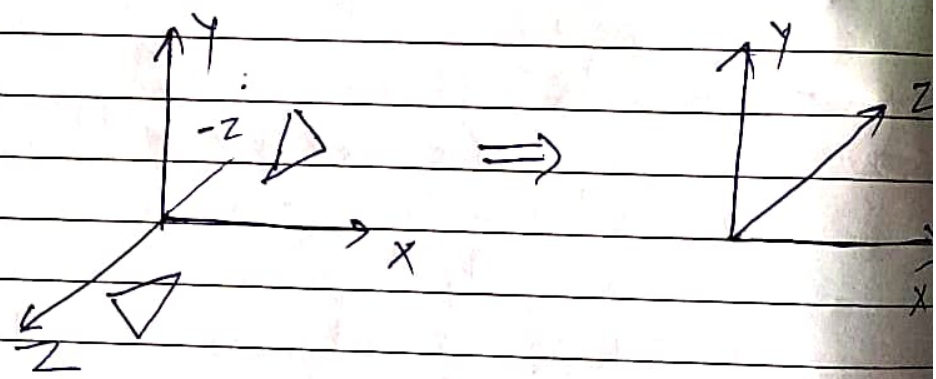
Other Transformations

① Reflection

- can be done with respect to different planes

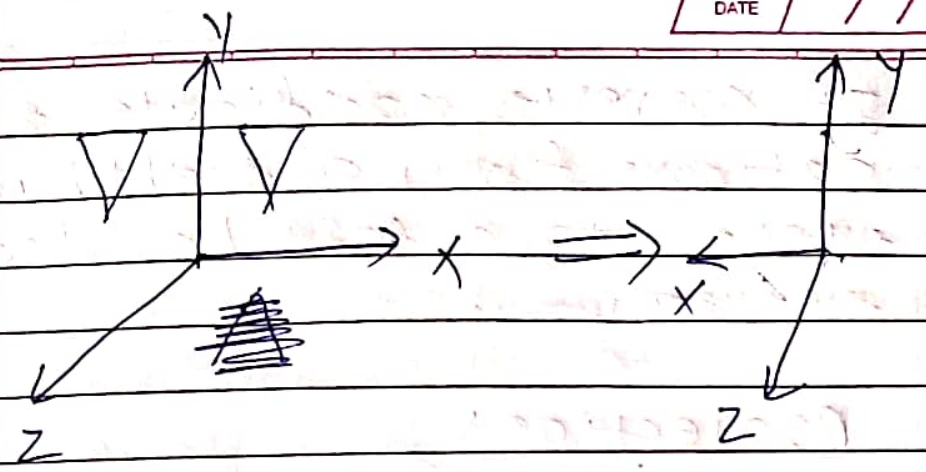
Reflection w.r.t XY plane (Re_{xy})

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



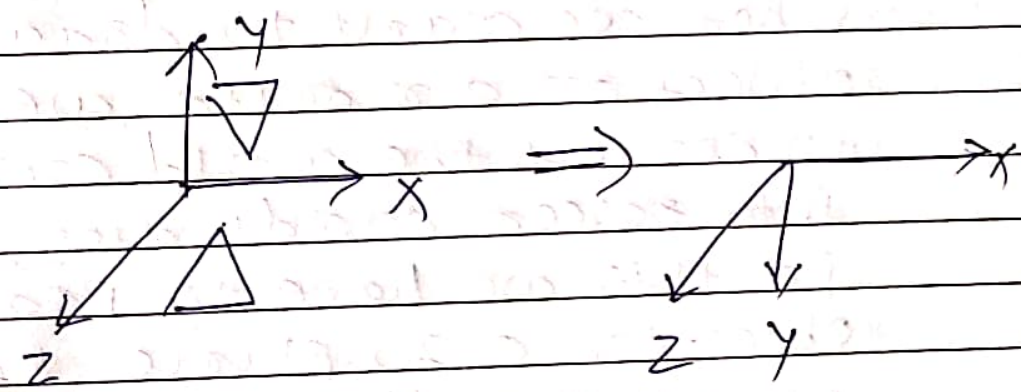
Reflection w.r.to YZ plane (Re_{yz})

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



Reflection w.r.t. to xz plane
(Rexz)

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



- Reflection is also called as mirror image of an object
- For this reflection axis & reflection of plane is selected
- Reflection is 180° about the given axis
- For reflection xy, xz or yz plane is selected

PAGE No.
DATE / /

It converts coordinate specifications from a right handed system to a left handed system

Projections

- We know that the objects available in nature are described in world co-ordinate system.
- But all graphics display media are 2D such as screen of the monitor, printout of printer etc.
- When we want to draw a 3D object on a monitor, we have to convert the world coordinates into screen coordinates.
- For this we have to project 3D object on a 2D plane, here 2D plane will be screen of monitor.
- projection is nothing but a shadow of the object.
- Ideally an object is projected by projecting each of its points.
- As there are infinite points in an objects, we cannot produce ~~Instead~~ projections of all these points, instead of that we will

PAGE No.
DATE / /

take projections of only corner points of an object on 2D plane & then we will join these projected points by straight line in 2D plane

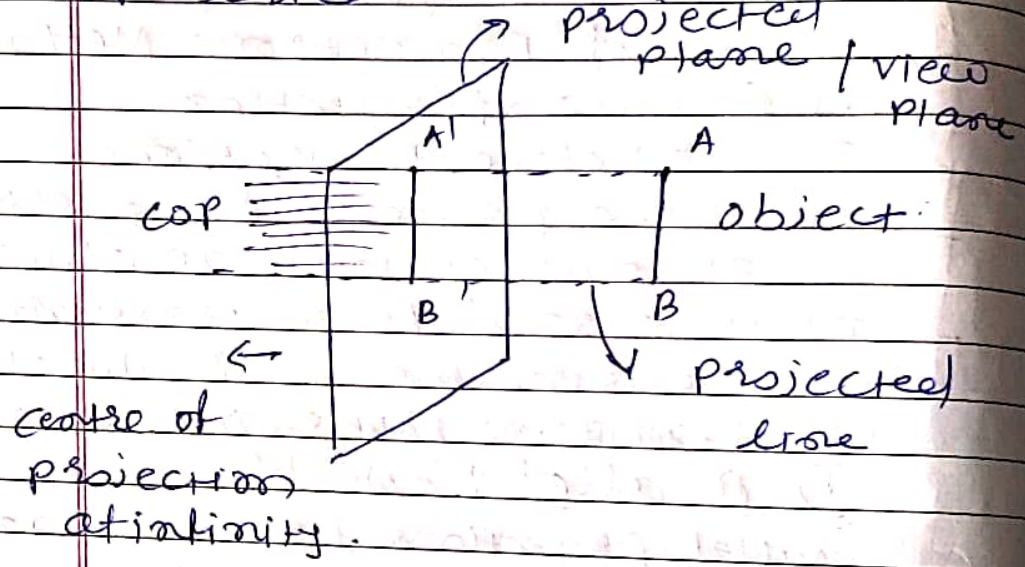
- Types of projections Methods
  1. Parallel projection
  2. perspective projection
- It is the process of converting a 3D object into 2D object
- It is also defined as mapping or transformation of object in projection plane or view plane

1. Parallel projection

- Parallel projections discards z-coordinates & parallel lines from each vertex on the object are extended until they intersect the view plane.
- Here we have to specify the direction of projection instead of centre of projection
- Here the distance from the centre of projection to project plane is infinite.
- In this type of projection we connect the projected vertices by line segments which correspond to connections on the original object

PAGE No.
DATE / /

- Projections are less realistic but good for exact measurements
- Here parallel lines remain parallel & angles are not preserved.



Types of Parallel Projection

- Orthographic
  1. Top
  2. Front
  3. Side
- Oblique
  1. Cavalier
  2. Cabinet

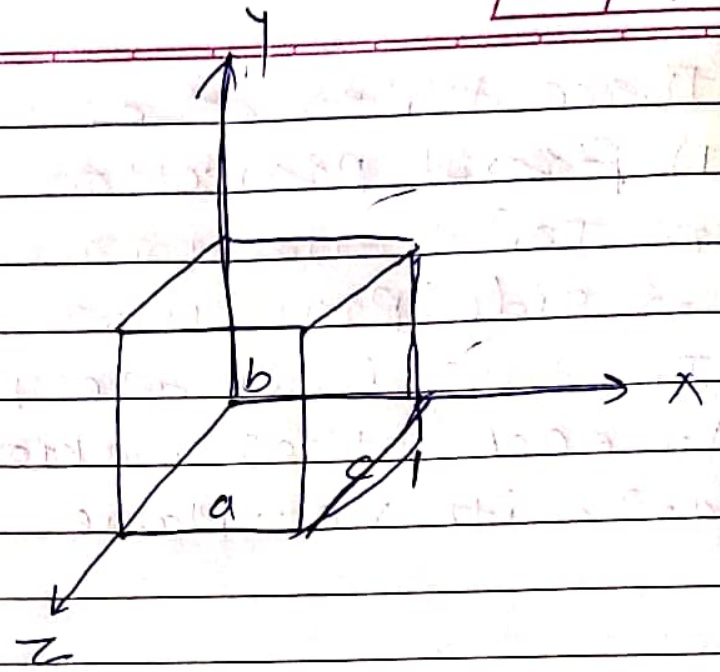
1. Orthographic projection -
- In this the direction of projection is normal to the projection of the plane.

PAGE No.
DATE / /

Three types are
1) Front projection
2) Top projection
3) Side projection
In this lines are parallel to each other making an angle 90° with view plane

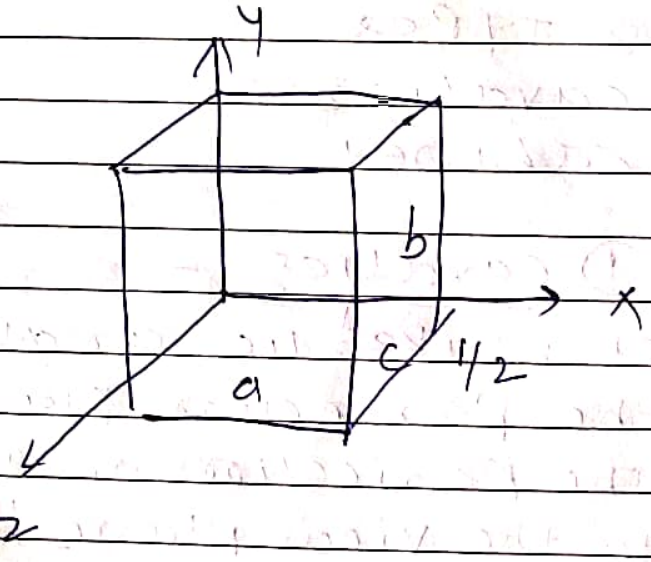
② Oblique projection
In this the direction of projection is not normal to the projection plane
We can view the object better than orthographic projection.
2 types
① cavalier
② cabinet

① Cavalier -
It makes 45° angle with the projection plane
The projection of a line ⊥ to the view plane has the same length as the line itself in cavalier projection



(2) cabinet

- It makes 63.4 angle with the projection plane
- Here lines ⊥er to the viewing surface are projected at 1/2 their actual length



PAGE No.
DATE / /

(2) Perspective projection

- In perspective projection, the distance from the center of projection to project plane is infinite & the size of the object varies inversely with distance which looks more realistic
- The distance & angles are not preserved & parallel lines do not remain parallel instead, they all converge at a single point called center of projection
- 3 types

(1) One point
(2) Two point
(3) Three point

PAGE No.
DATE / /

Curves

- In CG we often need to draw different types of objects onto the screen
- objects are not flat all the time & we need to draw curves many times to draw an object

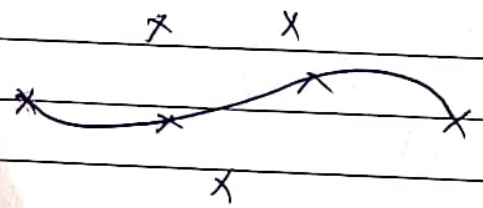
Types of curves

- curve is infinitely large set of points
- Each point has 2 neighbours except end points

1. Implicit curve
2. Explicit curve
3. Parametric curve

1. Implicit curve

- If the point is present on the curve



It is defined as $f(x, y) = 0$
f is a scalar function which returns a single real no.

PAGE No.
DATE / /

(2) Explicit curve

It is defined as

$y = f(x)$

For each values of x a single value y is normally computed by the function

(3) Parametric curves -

curves which have parametric values

$P(t) = f(t), g(t)$ or
$P(t) = x(t), y(t)$ } 2D

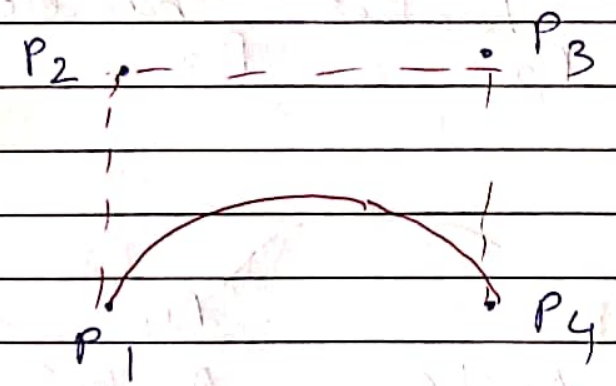
x, y are co-ordinates

t is the parametric values

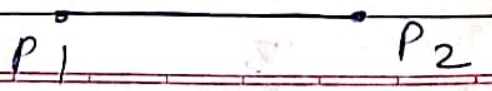
$0 < t < 1$

(1) Bezier curve

- It is defined by control point



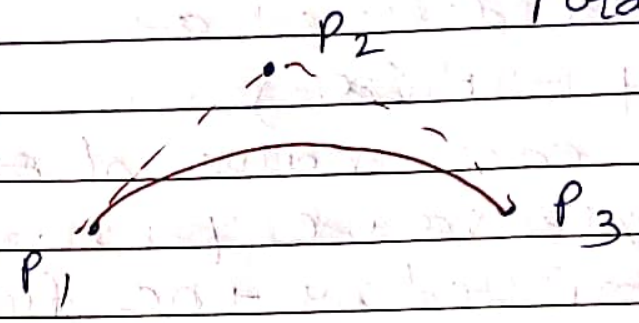
- control points are P_1, P_2, P_3, P_4
- 2 point curve



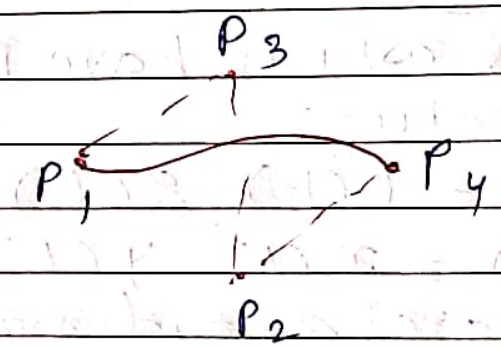
(linear curve)

PAGE No.
DATE / /

3 point curve (Quadratic Parabolic)

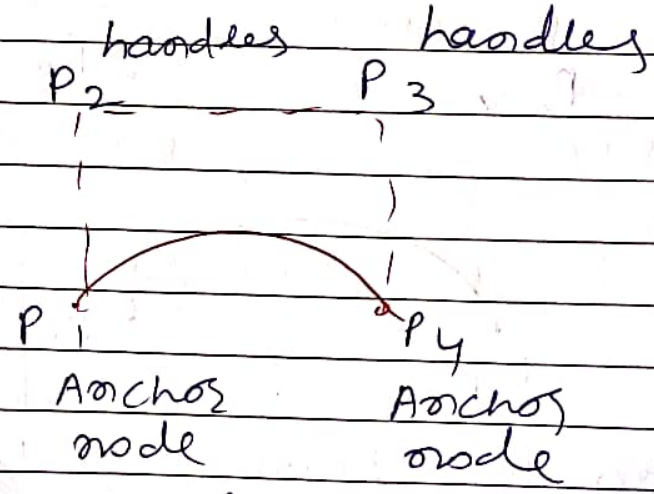


- 4 point curve (cubic)



- Points are not always on curve.
- The curve order

Bezier cubic



- curve is defined as 4 pts

degree = 4 - 1
= 3

$P(u)$ = Any point lying on the bezier curve

P_i = i^{th} control point lying on bezier curve

n = degree of curve

~~$B_{i,n}(u)$ = Blending function~~

DATE / /

Derivation of Bezier curve

Blending / basis function

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

where $0 \le u \le 1$ — u is parametric value

Assume $n+1 \Rightarrow$ control points

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

$$\Downarrow$$

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

$$B_{i,n}(u) = \frac{n!}{i!\,(n-i)!}\, u^i (1-u)^{n-i}$$

cubic Bezier $n = 3$

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

$$P(u) = P_0 B_{0,3}(u) + P_1 B_{1,3}(u) + P_2 B_{2,3}(u) + P_3 B_{3,3}(u)$$

$$B_{0,3}(u) = C(3,0) \cdot u^0 \cdot (1-u)^{3-0}$$

$$= \frac{3!}{0!\,3!} (1-u)^3$$

$$= (1-u)^3$$

PAGE No.
DATE / /

$$B_{1,3}(u) = C(3,1) \cdot u^1 (1-u)^{3-1}$$

$$= 3u \cdot (1-u)^2$$

$$B_{2,3}(u) = C(3,2) \cdot u^2 \cdot (1-u)^{3-2}$$

$$= 3u^2 \cdot (1-u)$$

$$B_{3,3}(u) = C(3,3) \cdot u^3 (1-u)^{3-3}$$

$$= u^3$$

So the final Blending function of cubic Bezier curve

$$P(u) = P_0(1-u)^3 + P_1 \cdot 3u(1-u)^2 + P_2 \cdot 3u^2 \cdot (1-u) + P_3 \cdot u^3 .$$

* $P(u_x) = P_0 x_0 (1-u)^3 + P_1 x_1 \cdot 3u \cdot (1-u)^2 + P_2 x_2 \cdot 3u^2 (1-u) + P_3 x_3 \cdot u^3$

* $P(u_y) = Py_0 (1-u)^3 + Py_1 \cdot 3u \cdot (1-u)^2 + Py_2 \cdot 3u^2 \cdot (1-u) + Py_3 \cdot u^3$

* $P(z_0) = Pz_0 (1-u)^3 + Pz_1 \cdot 3u(1-u)^2 + Pz_3 \cdot 3u^2 (1-u) + Pz_3 \cdot u^3$

PAGE No.
DATE / /

Example

Design a Bezier curve controlled by 4 points A(1,1), B(2,3), C(4,3), D(6,4)

$\Rightarrow$

control points = 4

$n = 4 - 1 = 3$

$A(1,1) = P_0$
$B(2,3) = P_1$
$C(4,3) = P_2$
$D(6,4) = P_3$

Eqn of bezier curve

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

$$B_{i,n}(u) = \frac{n!}{i!(n-i)!} \cdot u^i \cdot (1-u)^{n-i}$$

$n=3$

$$P(u) = P_0 B_{0,3}(u) + P_1 B_{1,3}(u) + P_2 B_{2,3}(u) + P_3 B_{3,3}(u)$$

$$P(u) = P_0 \cdot (1-u)^3 + P_1 \cdot 3u(1-u)^2 + P_2 \cdot 3u^2(1-u) + P_3 u^3$$

To find $x(u)$ & $y(u)$

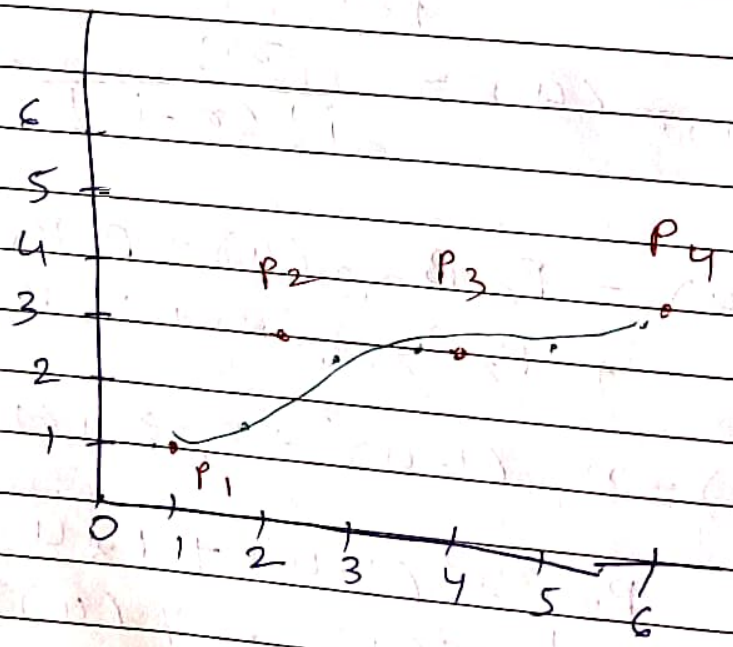
$P_0 = (\underset{x_0}{1}, \underset{y_0}{1})$
$P_1 = (\underset{x_1}{2}, \underset{y_1}{3})$
$P_2 = (\underset{x_2}{4}, \underset{y_2}{3})$, $P_4(\underset{x_3}{6}, \underset{y_3}{4})$

PAGE No.
DATE / /

$$x(u) = (1-u)^3 + 6u(1-u)^2 + 12u^2(1-u) + 6u^3$$

$$y(u) = (1-u)^3 + 9u(1-u)^2 + 9u^2(1-u) + 4u^3$$

u	$x(u)$	$y(u)$
0	1	1
0.2	1.712	0.984
0.4	2.616	2.632
0.6	3.664	3.088
0.8	4.808	3.496
1	6	4

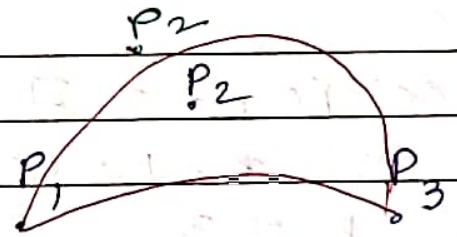


PAGE No.
DATE / /

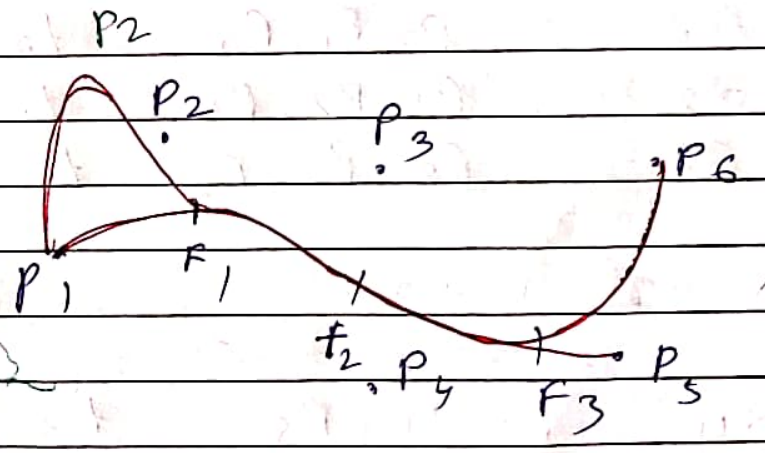
B - Spline curve

- Bezier curve is global control.
- B - spline curve is a local control.
- In this the degree of polynomial does not depends on the no. of control points
- It depends upon the order of polynomial.
- i.e. It is based on no. of control points if order varies then blending function degree will be different

Bezier (global control)



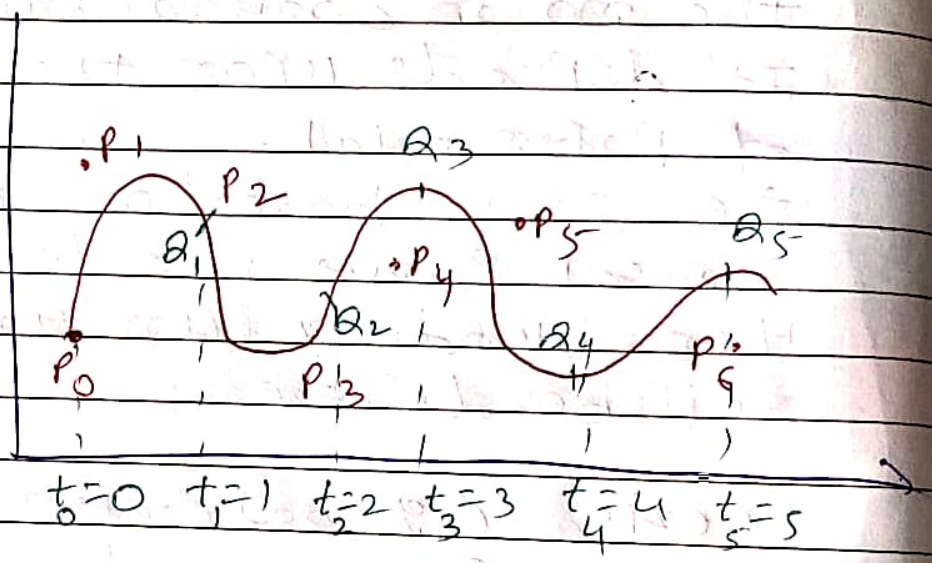
B - Spline (local control)



- It is made of n+1 control points & the order of curve (k)
- It is used to draw open & closed curve

PAGE No.
DATE

- Approximate spline curve
- Local control point
- Blending function necessary = non zero
- Here we can specify the degree of polygon specifying order of curve (k)



Control point = 7

$n = n - 1 = 6$

$k = 3$

Segment	Control points	Parameter
Q_1	P_0, P_1, P_2	$t_0=0, t_1=1$
Q_2	P_1, P_2, P_3	$t_1=1, t_2=2$
Q_3	P_2, P_3, P_4	$t_2=2, t_3=3$

$\cancel{n = k + 2}$

$\cancel{6 = 3 + 2 = 5}$

Q_4	P_3, P_4, P_5	$t_3=3, t_4=4$
Q_5	P_4, P_5, P_6	$t_4=4, t_5=5$

PAGE No.
DATE / /

$n - k + 2$

$6 - 3 + 2 = 5$

Blending function of B Spline curve

$$P(u) = \sum_{i=0}^{n} B_{i,k}(u) P_k$$

$P_k (x_k, y_k, z_k)$

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

$P_i \rightarrow$ control point $(P_0, P_1, \ldots$

u = Parameter, n = C.P. − 1

$0 \leq u \leq n - k + 2$ k = order of curve

$2 \leq k \leq n + 1$

k = responsible for segment

suppose $n = 6$

$k = 3$

control points $= n + 1$

$6 + 1$

7

$u = n - k + 2 = 6 - 3 + 2 = 5$

$0 \leq u \leq 1$

Generalization of expression

$$Y(u) = \sum_{i=0}^{n} B_{i,k}(u) Y_k$$

PAGE No.
DATE / /

$$B_{i,k}(u) = \frac{(u - t_i) B_{i,k-1}(u)}{t_{i+k-1} - t_i} + \frac{(t_{i+k} - u) B_{i+1,k-1}(u)}{t_{i+k} - t_{i+1}}$$

t = knot vector (joint point)

$t_i \Rightarrow (0 \leq i \leq n+k)$

$t_i = 0$ if $i < k$
$t_i = i - k + 1$ if $k < i < n$
$t_i = n - k + 2$ if $i > n$

Fractals

- complex picture created using iteration & single formula
- Ex. Mountains, clouds

Types of Fractals

1. Self similar
2. Self affine
3. Invarient

Self Similar -

Those fractals have parts as scaled down version of entire object

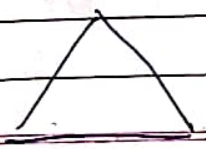
PAGE No.
DATE / /

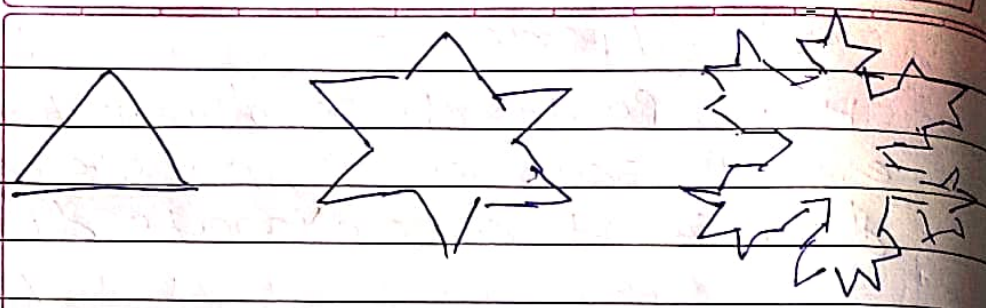
Self affine -
Have parts formed with different scaling parameters in different co-ordinate direction

Invarient -
formed with non linear transformations

Fractal Dimension
Fractal dimension is measure of roughness or fragmentation of object
- More fractal dimension in case of more jagged looking objects
- Using some iterative procedure we can calculate fractal dimension D

Koch curve
- Each straight line is replaced with 4 equal sized lines of scaling factor 1/3
- Also known as snowflake pattern
- Fractal dimension = 1.2619



PAGE No.

DATE / /