Chap 1.
Introduction and overview of Graphics System

* Use of Computer Graphics

① Major use of computer graphics is in design processes i.e. in engineering and architectural systems.
CAD used in the design of buildings, automobiles, aircraft, spacecraft, computers, textiles and many other products.

② CG used in fine art and commercial art applications.
- Artists use a variety of other computer technologies to produce images.
- They uses a combination of 3D modeling packages, texture mapping, drawing programs etc.

③ Entertainment
- CG methods are used in making motion pictures, music videos & TV shows.

④ Education and Training
- computer generated models of physical, financial and economic systems are often used in education.
- For some training applications, special systems are designed.
Ex:- simulators for practice sessions or training of ship captains, aircraft pilots, heavy equipment operators.

⑤ Visualization

- Scientists, engineers, medical personnel, business analysts & others often need to analyze large amounts of info or to study the behaviour of certain processes.
- Mathematicians, physical scientists and others use visual techniques to analyze mathematical functions and processes or produce interesting graphical representations.

⑥ Image processing

- To apply image processing method, we first digitize a photograph or other picture into an image file. Then digital methods can be applied to rearrange picture parts, to enhance color separations or to improve the quality of shading.
- Medical applications also use image-processing techniques for picture enhancements, in tomography and in simulations of operations.

⑦ GUI

- Interfaces also display menu & icons for fast selection of processing operations.
- The icons represent options for painting, drawing, zooming, typing text strings & other operations connected with picture construction.

Page No.:
Date:

* Coordinate System

- A coordinate system is a system consisting of a set of points, lines or surfaces with each point having a unique location or co-ordinate i.e. assigned to it.

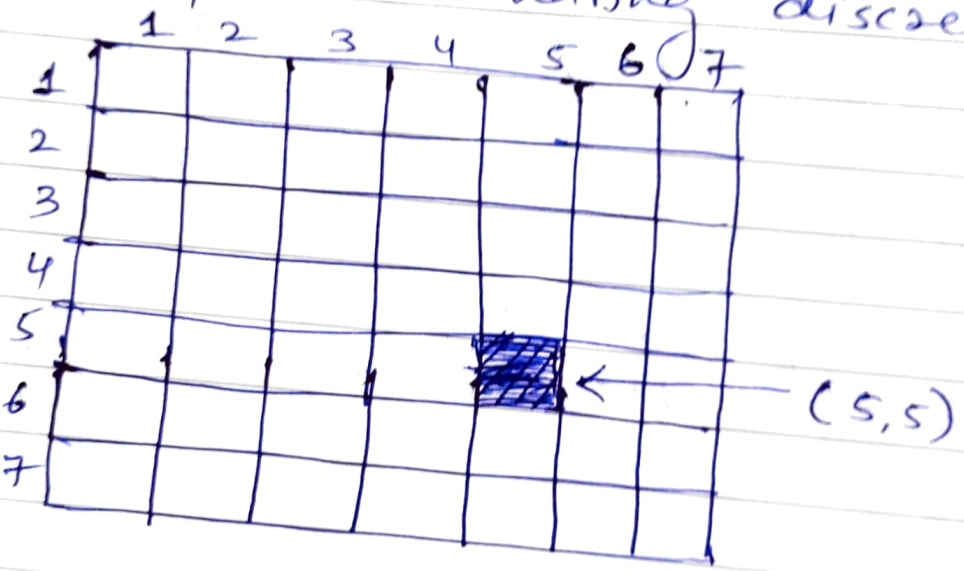
- The device coordinate system is the physical coordinate system of the selected plotting device.

- Device coordinates are integers, ranging from (0,0) at the bottom-left corner to (Vx-1, Vy-1) at the upper-right corner

- Vx and Vy are the number of columns and rows addressable by the device.

* Scan conversion

- It is process of representing graphics objects which is a collection of pixels.
- The pixels used are discrete. Each pixel can have either on or off state. 0 is represented by pixel off. 1 is represented on pixel.
- Using this ability graphics computer represent picture having discrete dots.



□ → Pixel

- Diff graphics objects can be generated by setting the diff intensity of pixels and diff colors of pixels.
- Each pixel has some co-ordinate value. The coordinate is represented using row and column.

* Rasterization and Rendering

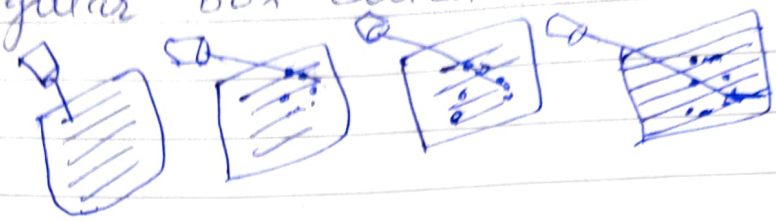
- Rasterization is the process by which most modern display systems turn electronic data or signals into projected images, such as video or graphics.

- All graphics are just a 2D array of pixe 3D graphics is thus a system of producing colors for pixels that convince you that the scene are at 3D world rather than 2D image.

The process of converting a 3D world into 2D image of that world is called rendering.

* Raster scan & random scan displays

- Raster scan display is based on intensity control of pixels in the form of a rectangular box called Raster. On the screen.



ex:- TV

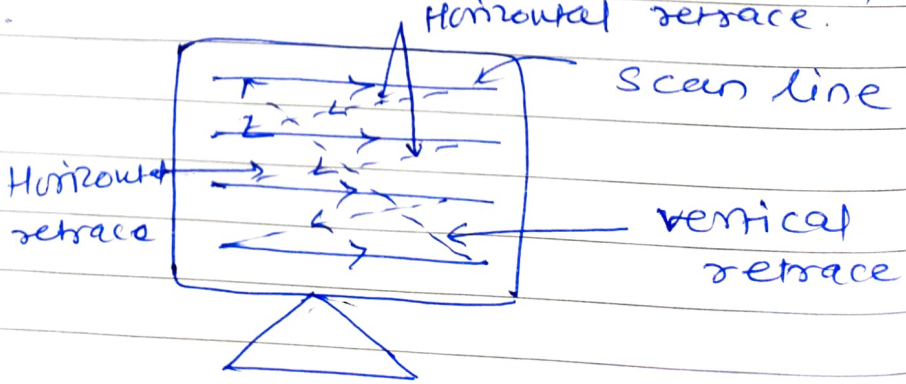
- The Raster scan system can store information of each pixel position, so it is suitable for realistic display of objects.

- Raster scan provides a refresh rate of 60 to 80 frames per second.

- Beam refreshing is of two types - First is horizontal retracing and second is vertical retracing.

- When the Beam start from the top left corner and reaches the bottom right scale, it will again return to the the top left side called a vertical retrace.

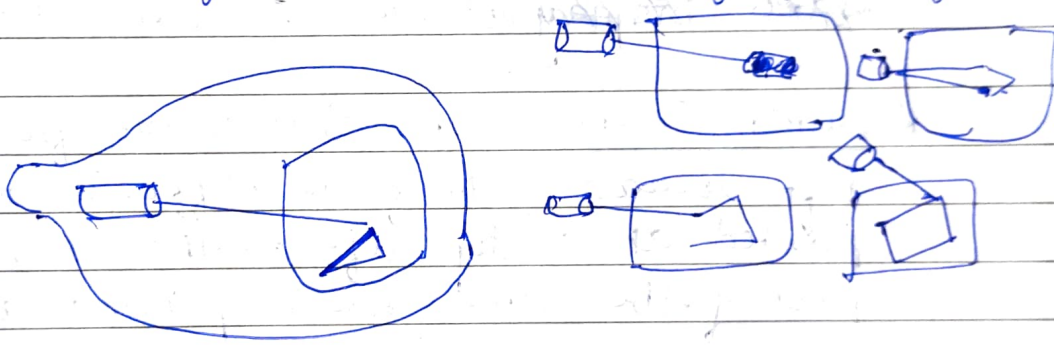
  Then it will come again more horizontal from top to bottom call as horizontal retracing.



Random Scan Display.

- Random scan system uses an electron beam which operates like a pencil to create a line image on the CRT screen.
- The picture is constructed out of a sequence of straight-line segments.
- Each line segment is drawn on the screen by directing the beam to move from one point on the screen to the next, where x & y co-ordinates define each point.
- After drawing the picture, the system cycles back to the first line and design all the lines of the image 30 to 60 time each second.

- Random scan monitors are also known as vector displays or 'stroke-writing display.



- This sys are designed for line drawing application & cannot display realistic shaded screens.

- Random displays produce smooth line drawing whereas raster produces jagged lines that are plotted as discrete point sets.

- Instead of pixel, the picture is stored in terms of line drawing commands, so it is called vector display.
- Electron beam only follow border of polygon instead of scanning each scan line.

Random scan	Raster scan
1. It has high Resolution	1. Low Resolution
2. More expensive	2. Less expensive
3. Solid pattern is tough to fill	3. Solid pattern is easy to fill
4. Refreshing rate depends on resolution	4. Refresh does not depend on pictu
5. It is restricted to line drawing applica	5. It is suitable for realistic displa
6. scan only part of the screen where picture info is presen	6. scan entire screen to draw a pictur
7. ~~video display~~ not requi	7. Scan conversion is requi

* Raster scan - Intensity value from the top left location of the frame buffer is retrived & painted at a top left on the screen. The second pixel of the same row is painted after that & this process continues.

After plotting all pixels in the first row the electron beam moves to the first pixel of the second row & paints all pixels in the second row on screen by retrieving intensity values in horizontal order.

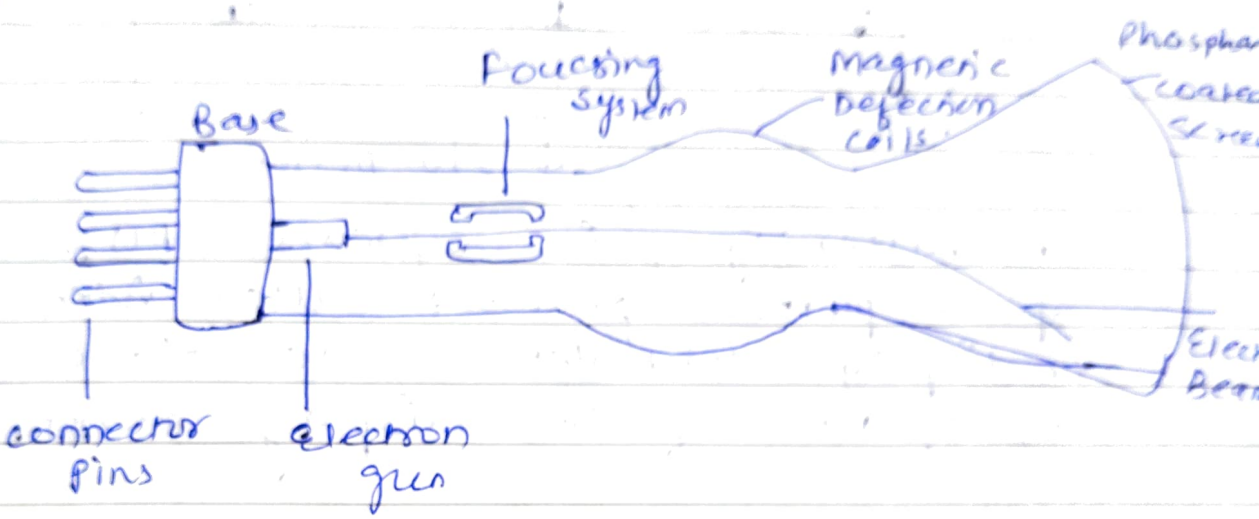
After finishing each scan line (row), electron guns are turned off & move back to the first pixel of the next scan line. That horizontal movement is called Horizontal retrace.

After Reaching to last pixel of the screen electron gun move to the first pixel of the first row i.e. called vertical retrace.

* Video Display Devices.

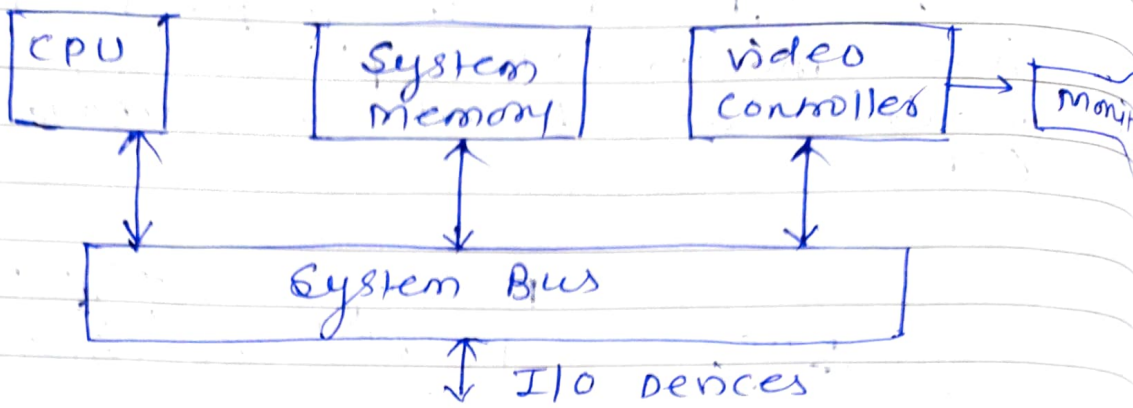
* Architecture of Raster Graphics system

- The o/p device in graphics system is video monitor. The operation of most video monitors are based on cathode ray Tube (CRT).



- A beam of electrons emitted by an electron gun, passes through focusing & deflection system that direct the beam toward specified positions on the phosphor coated system screen.
  The phosphor then emits small spot of ligur at the position where electron beam contacted the screen.
- Since amount of ligur emitted by phosphor coating depends on no of electrons striking the screen, we can control brightness of a display by varying the voltage on control grid.

* Architecture of Raster scan display.



- In raster graphics systems CPU, a special purpose processor called video control or display controller is used to control the operation of the display device.

- Here, the frame buffer can be anywhere in the system memory & the video controller accesses the frame buffer to refresh the screen.

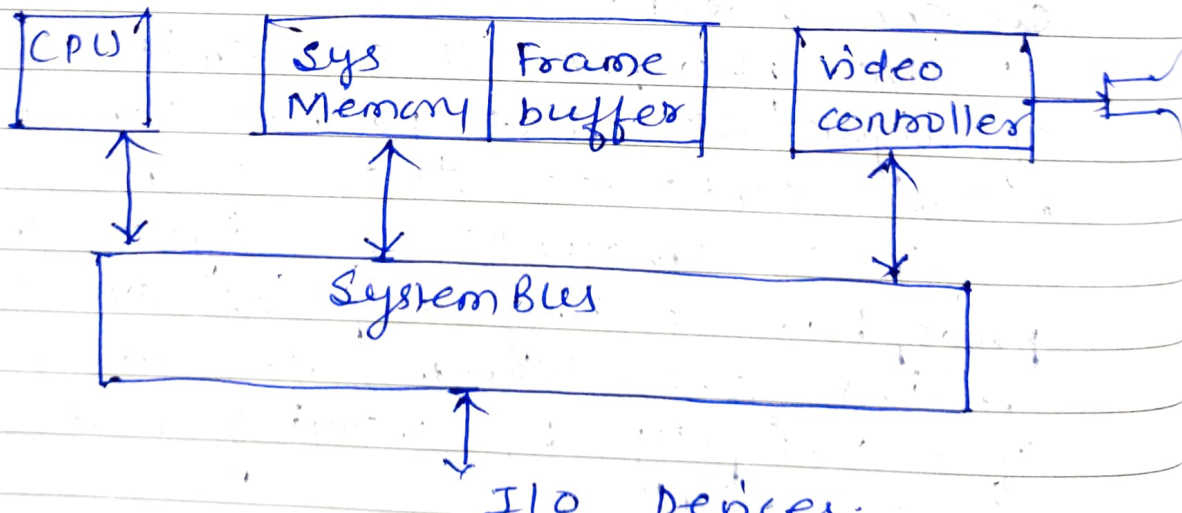


fig:- Architecture of simple Raster graphics

Raster Scan Display Processor

One way to set up raster system is having a separate display processor called as graphics controller / display processor.

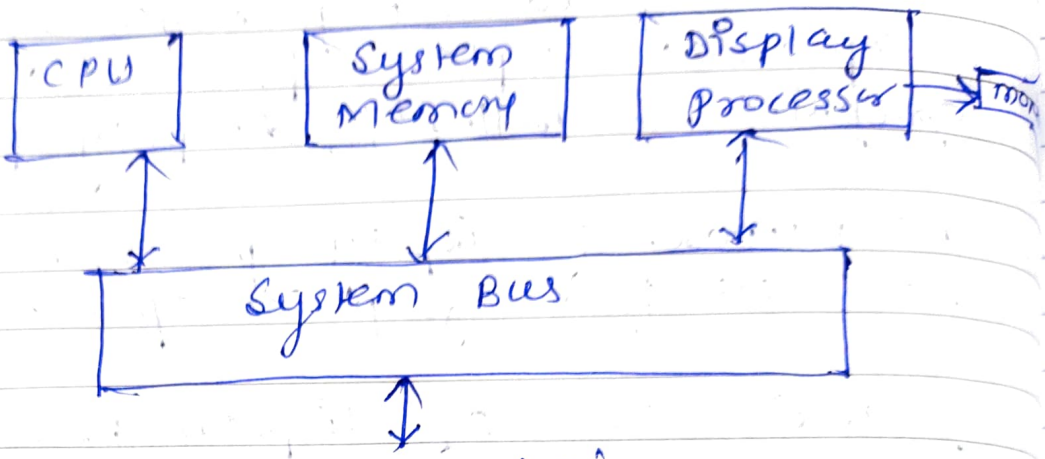
The purpose of display processor is to free the CPU from graphics operations.

A main task of display processor is digitizing the picture definition of application program into set of pixel intensity values for storage in frame buffer. This digitization process is called scan conversion.

Display processor perform other operations such as generating various line styles, (dashed, dotted or solid) displaying color areas & performing transformation & manipulations on display objects.

Display processors are designed to interface with iteractive input devices such as a mouse.

* Random scan Display



- An application program is input & st in system memory along with graphic package.
- Graphics commands are translated by graphics package into display file sto in system memory. This display file is then accessed by display processor h refresh the screen.
  This display processor is also call as a display processing unit or graph controller.

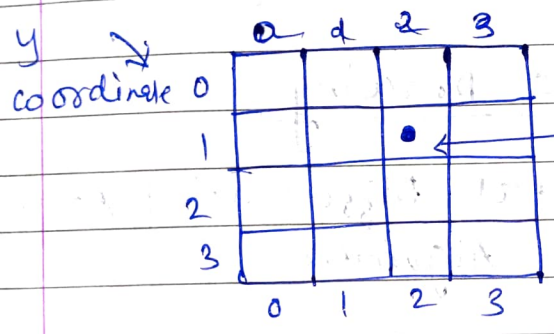
Page No.
Date:

Chap 2.
Output Primitives

* Scan conversion of point, line, circle & ellipse:

- Each pixel on the graphics display does not represent a mathematical point. Scan converting a point involes illuminating the pixel that contains the point.

Ex:- point p(x, y) represent by the integer part of x & the integer part of y.



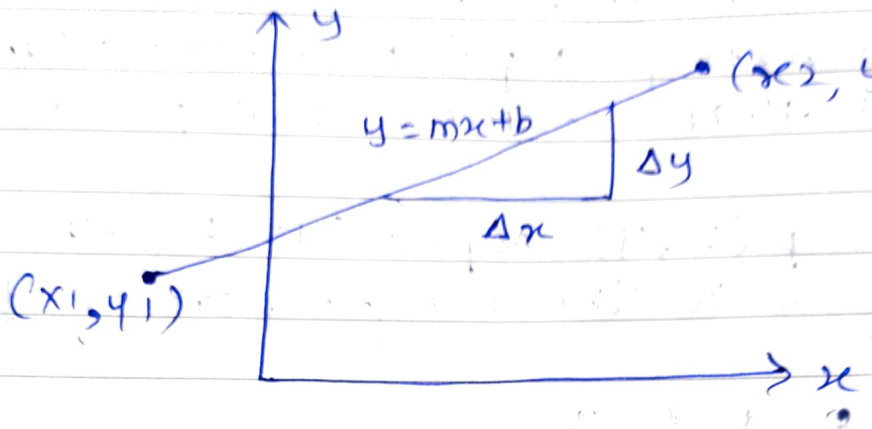
- Scan conversion is a process of representing graphics objects collection of pixels.

- The graphics objects are continuous. The pixel used are discrete. Each pixel can have either on or off state.

- The circuitry of the video display device of the computer is capable of converting binary values (0, 1) into a pixel on & off information.
Using this ability graphics computer represent Picture having discrete dots.

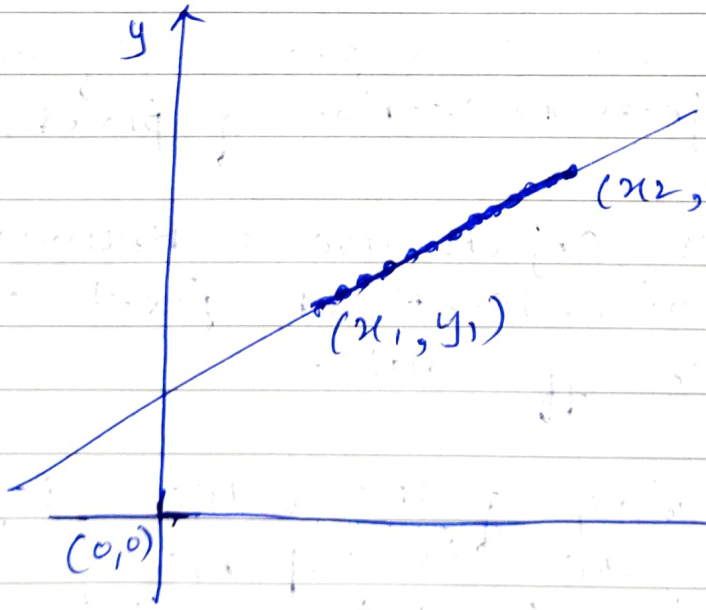
Scan Conversion of Line

- A straight line may be defined by 2 endpoints & an equation. The equation of the line is used to det the x, y coordinates of all the points that lie between these two endpoi



Algorithm for Line Drawing

1. DDA (Digital Differential Analy
2. Bresenham's Algorithm



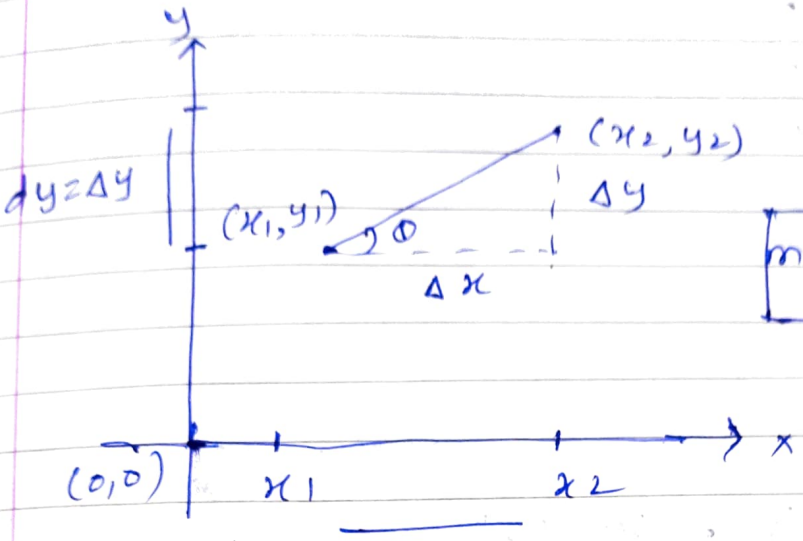
Line eq^n

① $y = mx + c$ ← slop

② $y - y_1 = \frac{y_2 - y_1}{x_2 - x_1} \cdot (x - x$

* DDA Algorithm

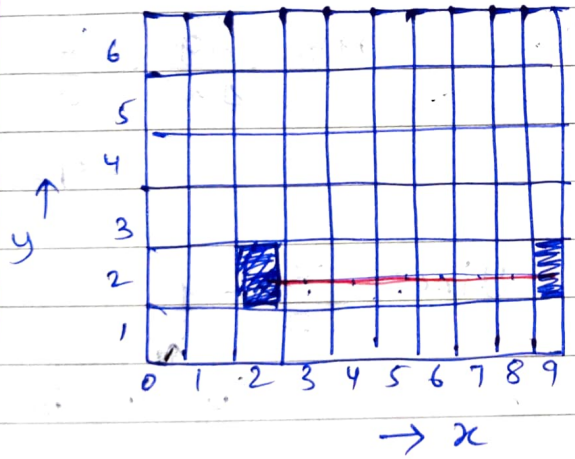
- It is an incremental method of scan conversion of line.



$$m = \frac{\Delta y}{\Delta x}$$ slope.

$dx = \Delta x$

(distance bet^n x_1 & x_2)



(x_1, y_1) (x_2, y_2)

$(2, 2)$ $(9, 2)$

$\Delta x = 9 - 2 = 7$

$\Delta y = 2 - 2 = 0$

$m = \frac{\Delta y}{\Delta x} = \frac{0}{7}$

Steps = 7

$x_{inc} = \frac{7}{7} = 1$ ↑ inc by 1

$y_{inc} = \frac{0}{7} = 0$ ↑ not inc

x	y
2	2
3	2
4	2
5	2
6	2
7	2
8	2
9	2

ex: $(x_1, y_1) \quad (x_2, y_2)$

$(2, 5) \quad (2, 12)$

$\Delta x = 2 - 2 = 0$

$\Delta y = 12 - 5 = 7$

$m = \frac{\Delta y}{\Delta x} = \frac{7}{0} = \infty$

steps = 7

(y va

$x_{inc} = \frac{0}{7} = 0$ ↑ x not inc.

$y_{inc} = \frac{7}{7} = 1$ ↑ (y inc by 1)

x	y
2	5
2	6
2	7
2	8
2	9
2	10
2	11
2	12

ex:- (x_1, y_1) (x_2, y_2)

$(5, 4)$ $(12, 7)$

$\Delta x = 12 - 5 = 7$

$\Delta y = 7 - 4 = 3$

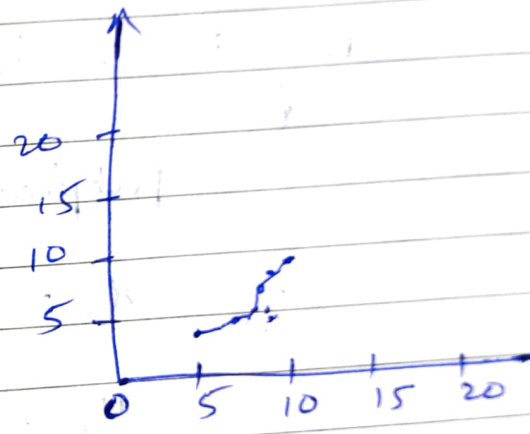
$m = \frac{\Delta y}{\Delta x} = \frac{3}{7} = 0.4$

steps 7

$x\,inc = \frac{7}{7} = 1$ $y\,inc = \frac{3}{7} = 0.4$

Here $m < 1$ (slope)

x	y	round (y)
5	4	4
6	4.4	4
7	4.8	5
8	5.2	5
9	5.6	6
10	6	6
11	6.4	6
12	6.8	7



* Algorithm DDA (x_1, y_1, x_2, y_2)

```
dx = x2 - x1
dy = y2 - y1

if ( abs (dx) > abs (dy))

    step = abs (dx)
else
    step = abs (dy)

xinc = dx / step

yinc = dy / step

for ( i = 1; i <= step; i++)
  {
      putpixel (x1, y1);

      x1 = x1 + xinc;

      y1 = y1 + yinc;
  }
```

- Algorithm takes xinc & yinc in floats calculation of floats take extra time than integer values. So Algo is slo
- We are getting points which are not accurate one so we have to round off
- Line will not smooth line.

- This is drawback of DDA Algo. So we go for Brekenbam's Algorithm.

* (DDA) Algo (Practical)

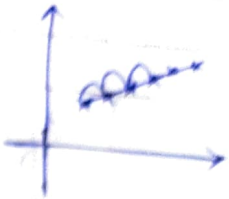
→ It is a scan conversion method for drawing line

→ It has incremental approach

→ It analysed difference in pixel points.

→ straight line equation is,

$$y = mx + b$$



if the line has two end points $A(x_1, y_1)$ & $B(x_2, y_2)$

slope of the line is $m = \frac{y_2 - y_1}{x_2 - x_1}$

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

→ DDA based on calculation of values Δx, Δy.

$$\therefore \Delta y = m * \Delta x \quad — (1)$$

$$\Delta x = \frac{\Delta y}{m} \quad — (2)$$

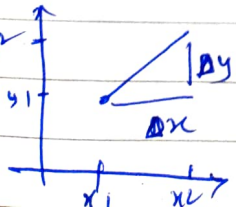
→ Given line have the slope +ve or -ve

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

case (1) if $m < 1$

$x_n = x_1 + 1$

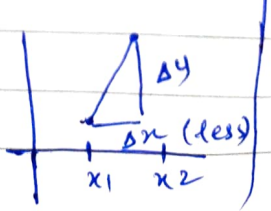
$y_n = y_1 + m$



(2) if $m > 1$

$x_n = x_1 + 1/m$

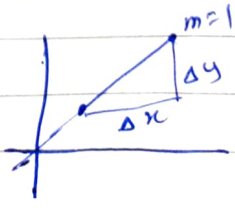
$y_n = y_1 + 1$



(3) if $m = 1$

$x_n = x_1 + 1$

$y_n = y_1 + 1$



DDA Algorithm

1) Steps : Accept start & end point coordinates (x_1, y_1) & (x_2, y_2)

2) calculate $dx = x_2 - x_1$
$dy = y_2 - y_1$

3) if $abs(dx) > abs(dy)$ // $m < 1$
then $k = abs(dx)$
else
$k = abs(dy)$

4) calculate $\Delta x = \frac{dx}{k}$, $\Delta y = \frac{dy}{k}$

5) Initialize $x = x_1$, $y = y_1$

6) Display pixel (x, y)

7) $x = x + \Delta x$ $\quad y = y + \Delta y$

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

9) Repeat step 7 & 8 'k' times.

Advantage

→ faster method : when compared to direct use of line eq^n.

Q. Draw the line between (5,5) & (10,9) using DDA Algo.

→ $(x_1, y_1) - (5,5)$ $(x_2, y_2) - (10,9)$

$dx = x_2 - x_1 = 10 - 5 = 5$

$dy = y_2 - y_1 = 9 - 5 = 4$

$(dx > dy) \rightarrow (5 > 4) \rightarrow K = dx.$

$K = dx = 5$

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

$\Delta y = \frac{dy}{K} = \frac{4}{5} = 0.8$

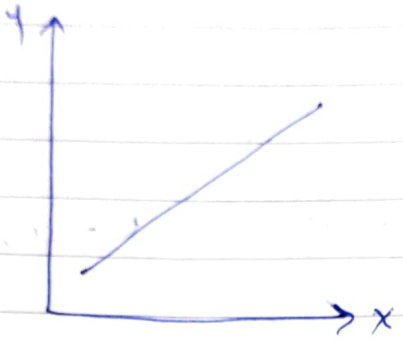
step. K=5 plot (round (x), round (y))

$x + \Delta x$ $y + \Delta y$.

iterat^n	plot (x,y)	$x + \Delta x$	$y + \Delta y$
0	(5,5)	5+1=6	5+0.8=5.8
1	(6,6)	6+1=7	5.8+0.8=6.8
2.	(7,7)	7+1=8	6.8+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)	10+1=11	9+0.8=9.8

* Bresenham's Line Drawing Algorithm

- Drawback of DDA algo i.e: floating point number (processing time is more)

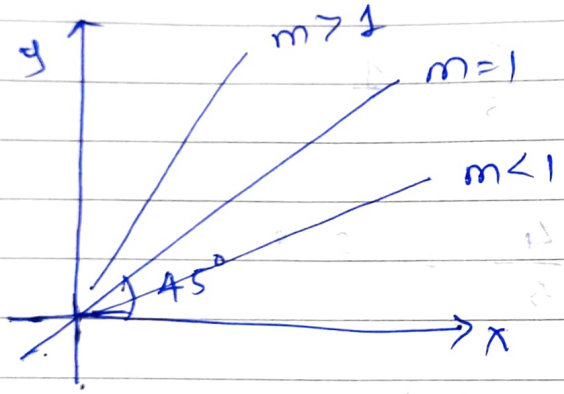


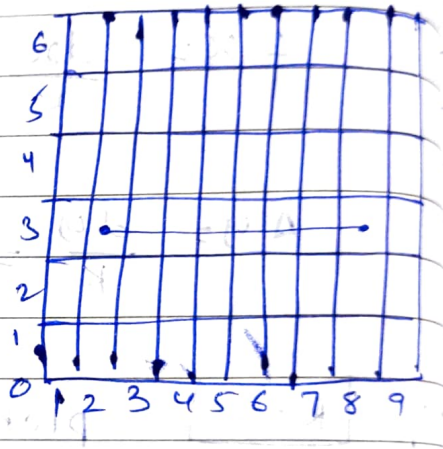
vector

$y = mx + c$

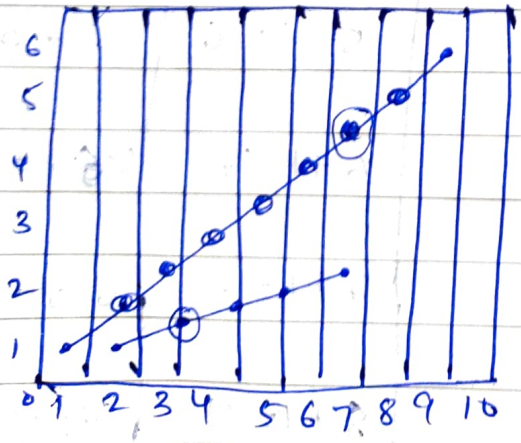
$\frac{\Delta y}{\Delta x}$

raster

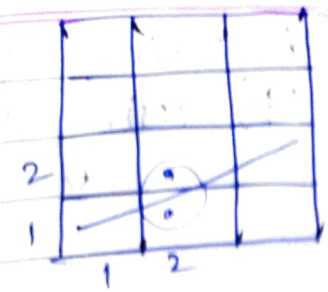


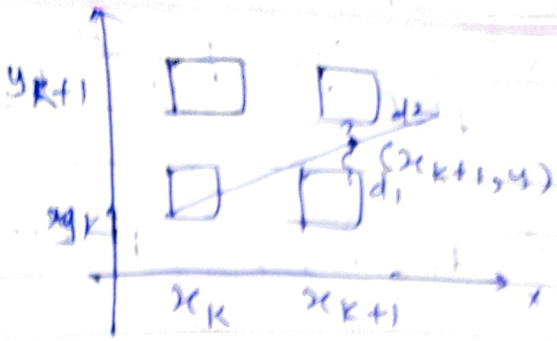


- Converting $y = mx + c$ into screen co-ord is called Rasterization.



Line is going partially to another pixel al



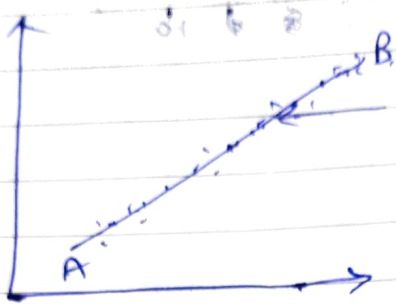


$x_{inc} = x_{k+1}$

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

* Bresenham's Line Drawing Algo

- It determines the point of an n-dimen raster that should be selected in ord to form a close approximation to a straight line between two points.



It will take close approximation of pixel.

→ A (x_1, y_1) B (x_2, y_2)

- It is efficient because it only take integer addⁿ, sub, & multiplication.
- Operations performed rapidly

- Algorithm

1) Cal start & end co-ordinates.
   (x_1, y_1) & (x_2, y_2)

2) Cal Δx & Δy
   $dx = x_2 - x_1$
   $dy = y_2 - y_1$

3) Cal decision parameter (It is used to find Extract point to draw line)

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

4) current point (x_k, y_k)
next point (x_{k+1}, y_{k+1})
find next point depending on value of decision parameter P_k.

case (1)

if $P_k < 0 \Rightarrow$ $P_{k+1} = P_k + 2\Delta y$
$x_{k+1} = x_{k+1}$
$y_{k+1} = y_k$

case (2)

if $P_k \geq 0 \Rightarrow$ $P_{k+1} = P_k + 2\Delta y - 2\Delta x$
$x_{k+1} = x_{k+1}$
$y_{k+1} = y_{k+1}$

5) ~~Repeat~~ Repeat step (4) until end point is reached.

Advantages

- It involve integer arithmetic, very simple.
- It avoids the generation of duplicate pts
- It can be implemented using H/w.
- It is faster than DDA.

Example

1) $(x_1, y_1) = (9, 18)$
$(x_2, y_2) = (14, 22)$

→ $\Delta x = x_2 - x_1 \rightarrow 14 - 9 = 5$
$\Delta y = y_2 - y_1 \rightarrow 22 - 18 = 4$

Cal decision parameter

$$P_k = 2\Delta y - \Delta x$$
$$= 2 \times 4 - 5$$
$$\boxed{P_k = 3}$$

$\therefore P_k > 0 \rightarrow$ case (2)

$$P_{k+1} = P_k + 2\Delta y - 2\Delta x$$
$$= P_k + 2(\Delta y - \Delta x)$$
$$= 3 + 2(4 - 5)$$
$$= 3 + 2(-1)$$
$$= 3 - 2$$
$$= 1$$

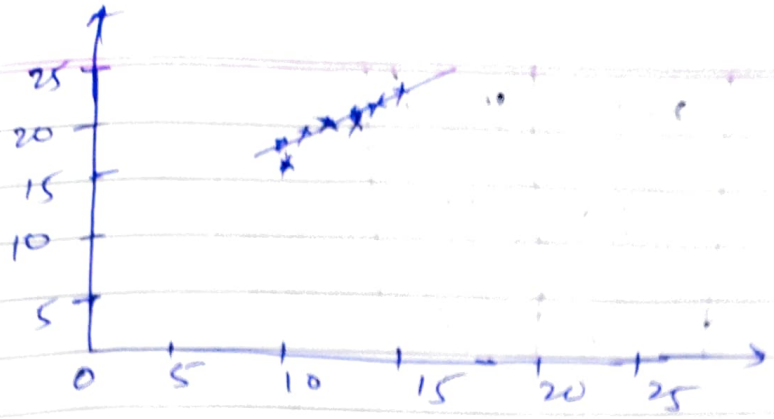
$x_{k+1} = x_{k+1} = 9 + 1 = 10$
$y_{k+1} = y_{k+1} = 18 + 1 = 19$

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
~~3~~	~~1~~	~~15~~	~~23~~

$P_k < 0$

$P_{k+1} = P_k + 2y_{k+1} - 2x_{k+1}$
$x_{k+1} = x_{k+1}$
$y_{k+1} = y_k$

$= -1 + 2(4)$
$= -1 + 8 = 7$



2) Explain Bresenham line drawing Algo. ~~with~~ and identify the pixel positions of A (10,10) & B (18,16).

→ A (10,10)
B (18,16)

$\Delta x = x_2 - x_1$ $\quad 18-10 = 8$
$\Delta y = y_2 - y_1$ $\quad 16-10 \quad 6$

Decision parameter

$$P_K = 2\Delta y - \Delta x = 2 \times 6 - 8 = 4$$

$P_K > 0$ → case (2)

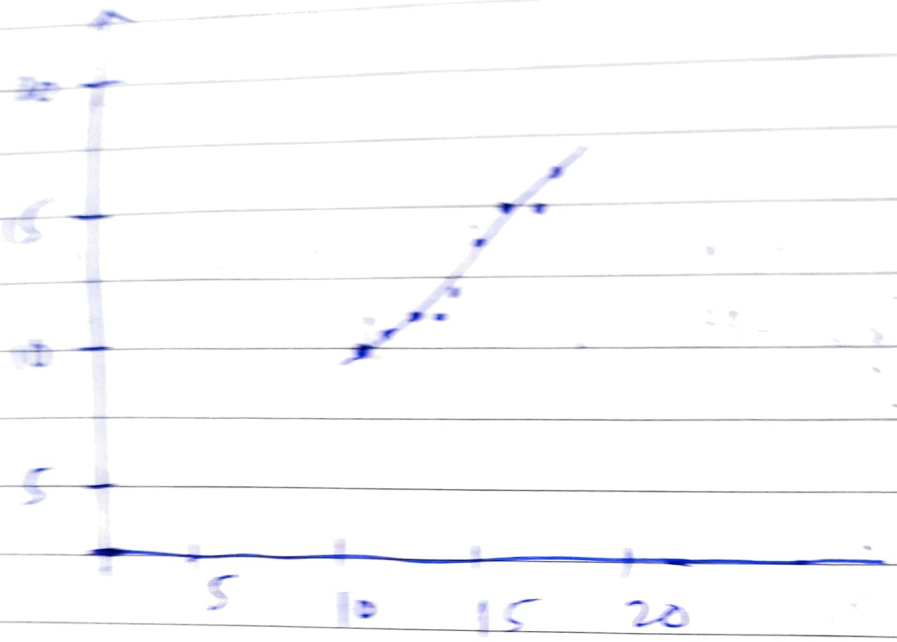
$P_{K+1} = P_K + 2(\Delta y - \Delta x)$
$= 4 + 2(6-8)$
$= 4 - 4 = 0$

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

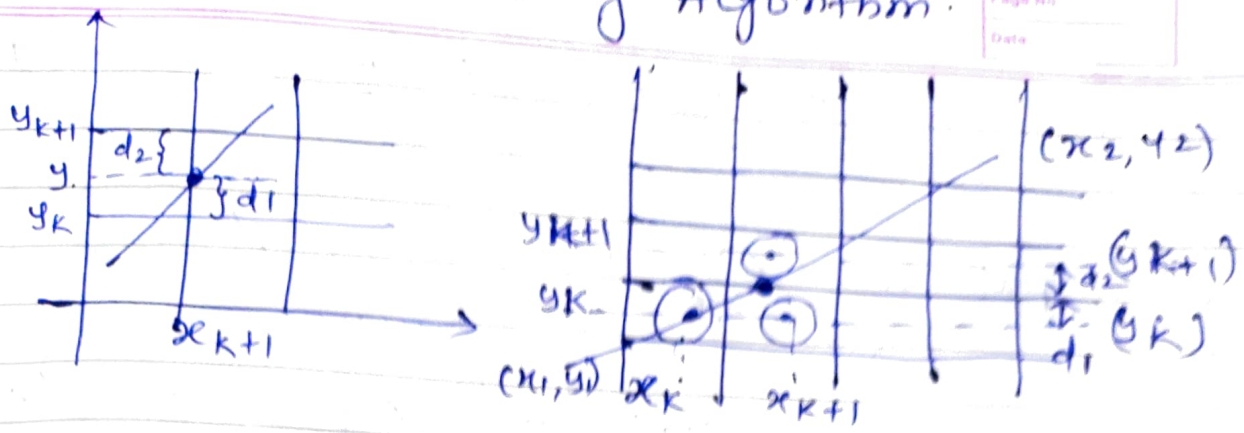
P_K	P_{K+1}	x_{K+1}	y_{K+1}
		10	10
4	0	11	11
0	-4	12	12
-4	8	13	12
8	4	14	13
4	0	15	14

$P_K \geq 0$
$P_K < 0$
$P_K > 0$

P_k	P_{k+1}	x_{k+1}	y_{k+1}
0	-4	16	15
-4	13	17	15
5	4	18	16



Derivation of Bresenham's Line Drawing Algorithm.



$y = mx + c$. At sampling position x_{k+1}, corresponding y is computed as,

$$y = m(x_{k+1}) + c$$
$$= m \cdot x_k + m + c$$

$$d_1 = y - y_k$$
$$= (m \cdot x_k + m + c) - y_k$$

$$d_2 = y_{k+1} - y$$
$$= y_{k+1} - (m \cdot x_k + m + c)$$

(for next scan line $y_{k+1} = y_{k+1}$)

$$d_1 - d_2 = (m \cdot x_k + m + c - y_k) - (y_{k+1} - m \cdot x_k - m - c)$$
$$= m \cdot x_k + m + c - y_k - y_{k+1} + m \cdot x_k + m + c$$
$$= \boxed{2m \cdot x_k + 2m - 2y_k + 2c - 1}$$

To simplify the computation, multiply it by Δx,

Decision parameter at step k, P_k

$$P_k = \Delta x (d_1 - d_2)$$
$$= \Delta x (2m \cdot x_k + 2m - 2y_k + 2c - 1)$$ const

Derivative

$$d_1 = y - y_k$$
$$= [m(x_k+1) + c] - y_k$$

$$d_2 = y_{k+1} - y$$
$$= y_{k+1} - m(x_k+1) - c$$

if $d_1 - d_2 < 0$ $\quad y_{next} = y_k$

$d_1 - d_2 > 0$ $\quad y_{k+1}$

$$d_1 - d_2 = [m(x_k+1) + c - y_k] - [y_{k+1} - m(x_k+1) - c]$$
$$= 2m(x_k+1) + c - y_k - y_k - 1 + c$$
$$= 2m(x_k+1) - 2y_k + 2c - 1$$

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

Again a float value so remove float by multiplying Δx on both side.

$$\Delta x(d_1 - d_2) = \Delta x\left[\frac{2\Delta y}{\Delta x}(x_k+1) - 2y_k + 2c - 1\right]$$

$$\Delta x(d_1 - d_2) = 2\Delta y(x_k+1) - 2\Delta x y_k + 2\Delta x c - \Delta x$$

$$= \underbrace{2\Delta y x_k - 2\Delta x y_k} + \underbrace{2\Delta y + 2\Delta x c - \Delta x}_{\text{constant}} \quad \text{---(A)}$$

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

$$\Delta x(d_1 - d_2) = p_k \text{ — decision p}$$

$$\therefore p_k = 2\Delta y x_k - 2\Delta x y_k$$

Calculate next decision parameter

$$P_{next} = 2\Delta y\, x_{next} - 2\Delta x\, y_{next}$$

$$P_{next} - P_k = [2\Delta y\, x_{next} - 2\Delta x\, y_{next}] - [2\Delta y\, x_k - 2\Delta x\, y_k]$$

$$= 2\Delta y\, x_{next} - 2\Delta x\, y_{next} - 2\Delta y\, x_k + 2\Delta x\, y_k$$

$$= 2\Delta y\,(x_{next} - x_k) - 2\Delta x\,(y_{next} - y_k)$$

Here $x_{next} = x_{k+1}$

But y_{next} has y_k or y_{k+1}

So take decision

$$P_{next} - P_k < 0$$

then $y_{next} = y_k$, $x_{next} = x_{k+1}$

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

$$\therefore \boxed{P_{next} = P_k + 2\Delta y}$$

if $P_{next} - P_k >= 0$ then

$x_{next} = x_{k+1}$ & $y_{next} = y_{k+1}$

$P_{next} - P_K >= 0$ then

$x_{next} = x_{k+1}$ & $y_{next} = y_{k+1}$

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

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

Now put $P_{next} = P_{k+1}$

when $P < 0$ → $\therefore$ $\boxed{P_{k+1} = P_K + 2\Delta y}$ $P_{k+1} - P_k <$

when $P >= 0$ → $\boxed{P_{k+1} = P_K + 2\Delta y - 2\Delta x}$ $P_{k+1} - P_k >$

Now cal initial value of decision parameter P_0 use eqn (A)

$$P_K = 2\Delta y x_K - 2\Delta x y_K + 2\Delta x c - \Delta x$$

Put $K = 1$

$$P_1 = 2\Delta y x_1 - 2\Delta x y_1 + 2\Delta x c - \Delta x$$

$$c = y_1 - \frac{\Delta y}{\Delta x} x_1$$

$$P_1 = 2\Delta y x_1 - 2\Delta x y_1 + 2\Delta y + 2\Delta x \left[y_1 - \frac{\Delta y}{\Delta x} x_1\right]$$

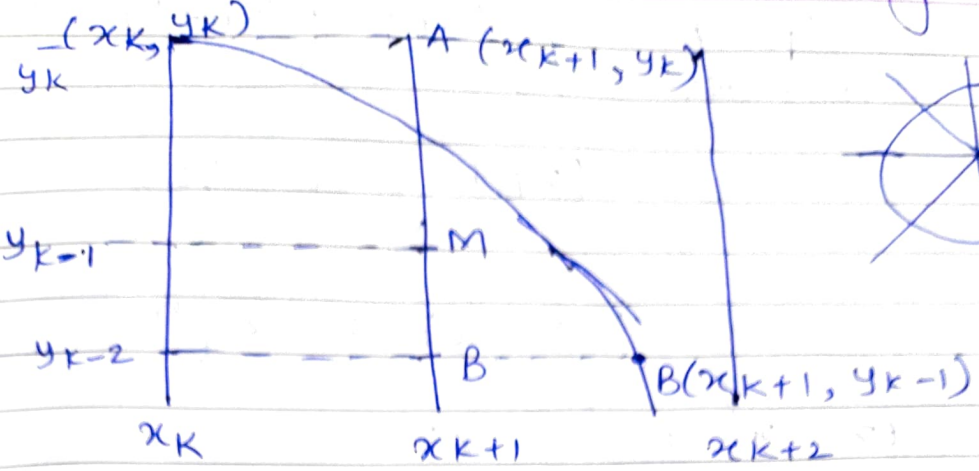
$$= 2\Delta y x_1 - 2\Delta x y_1 + 2\Delta y + 2\Delta x y_1 - 2\Delta y x_1 - \Delta x$$

$$P_1 = 2\Delta y - \Delta x$$ ← Initial value of decision parameter.

Q1 A (1, 1)
B (8, 5)

Q2 (4, 4)
(12, 9)

Mid-Point Circle Drawing Algorithm

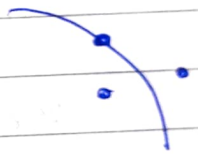


→ $x^2 + y^2 = r^2$

→ $x^2 + y^2 - r^2 = 0$

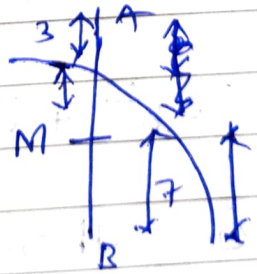
(x_m, y_m) → are midpoint co-ordinates

$(x_m)^2 + (y_m)^2 - r^2$

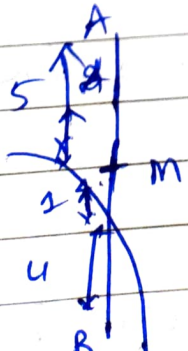


0 : point lies on circle
A → < 0 : point lies inside
B → > 0 : point lies outside

Find minimum distance from M of point A & B from midpoint to select for drawing circle.



select shortest distance.
M inside.



of m outside

Midpoint coordinates $= \left[\frac{(x_{k+1} + x_{k+1})}{2}, \frac{(y_{k+1} + y_{k+1})}{2} \right]$

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

Put above into circle eqn.

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

$$d_k = (x_{k+1})^2 + (y_k - 1/2)^2 - r^2 \quad \text{---(1)}$$

d_k is decision parameter.

$$\frac{d_{k+1}}{dinc} = (x_{\underbrace{k+1}_{next}} + 1)^2 + (y_{\underbrace{k+1}_{next}} - 1/2)^2 - r^2 \quad \text{---(2)}$$

$$d_{k+1} - d_k$$

x_{k+1} will same in next step

Substitute $x_k + 1$ in eqn (2) in place of x_{k+1}

$$d_{k+1} = d_k + 2x_k + 3 + (y_k + 1)^2 - y_{k+1} - y_k^2 + y_k$$

If $d_k < 0$ then y coordinate only update

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

Substitute y_k in place of y_{k+1}

$$d_{k+1} = d_k + 2x_k + 3$$

If $d_k > 0$ then $y_{k+1} = y_k - 1$

$$d_{k+1} = d_k + 2x_k + 2y_k + 5$$

Now find initial decision parameter

initial co-ordinates (x_k, y_k) substitute this with $(0, r)$ i.e. $x_k = 0$ & $y_k = r$

Now substitute $(0, r)$ in eqn (1)

$$d_0 = (0+1)^2 + (r - 1/2)^2 - r^2$$

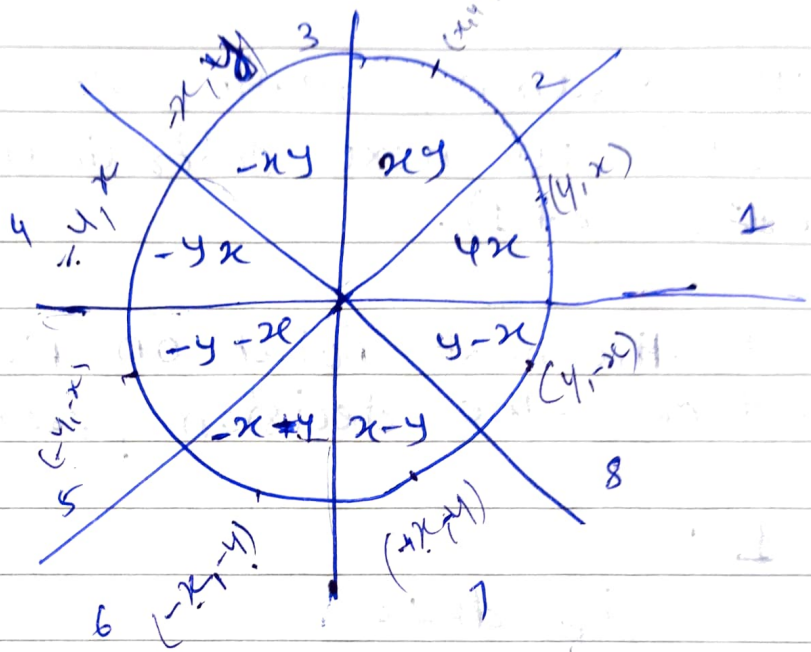
$$d_0 = \frac{5}{4} - r \Rightarrow 1 - r$$

$\frac{5}{4} \cong 1$ $\therefore r$ is an integer

so $d = 1 - r$

initial value (0, r)

circle divide into 8 octants on 2D plan



Page No.:
Date:

Algorithm

1. Consider a center coordinates (x_1, y_1) as

$$x_1 = 0;$$
$$y_1 = r;$$

2. Cal the starting decision parameter d_1

$$d_1 = 1 - r$$

3. Let us assume, starting co-ordinates as (x_k, y_k) so next co-ordinates are (x_{k+1}, y_{k+1}).

Find the next point on first octant based on the decision parameter (d_k)

4. Case 1 :

if $d_k < 0$

then

$$x_{k+1} = x_{k+1}$$
$$y_{k+1} = y_k$$
$$d_{k+1} = d_k + 2(x_{k+1}) + 1$$

case 2 :

if $d_k \geq 0$

then

$$x_{k+1} = x_{k+1}$$
$$y_{k+1} = y_k - 1$$
$$d_{k+1} = d_k - 2(x_{k+1} + 2(x_{k+1})) + 1$$
$$= d_k - 2y_{k+1} + 2$$

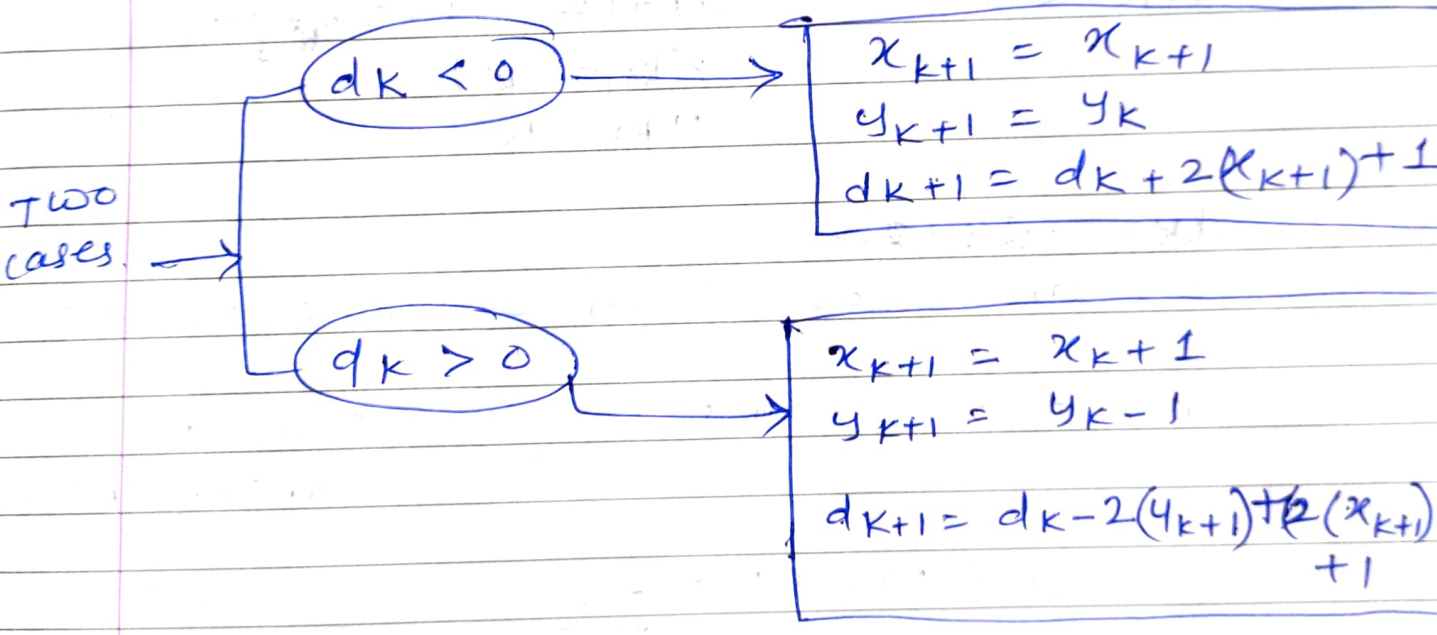
5. If the center co-ordinate pt (x_1, y_1) is not at the origin $(0, 0)$ then finding the point as -

for x co-ordinate = $x_c + x_1$

for y co-ordinate = $y_c + y_1$

{ x_c & y_c contains current value of x & y }

6. Repeat step 4 & 5 till we get $x >= y$



+1

Example

① Draw a circle using mid-pt circle at origin with radius = 15.

→ $(x, y) = (0, 0)$ & radius $= 15$

$x = 0$
$y = 0$
$r = 15$

$d_k = 1 - r$
$= 1 - 15$
$= -14$

$d_k < 0$ $\quad d_{k+1} = d_k + 2x_{k+1} + 1$
$x_{k+1} - x_k$
$y_{k+1} - y_k$

As d_k is less than 0, $x_n = 1$, $y_n = 15$

$d_k = -14 + 2 \times 1 + 1 = -11$ again less than

$\therefore x_n = 2$, $y_n = 15$

$d_k = -11 + 2 \times 2 + 1 = -6$ again < 0

$x_n = 3$, $y_n = 15$

$d_k = -6 + 2 \times 3 + 1 = 1$ $d_k > 0$, case

$x_n = 4$, $y_n = 14$

$d_k = 1 + 2 \times 4 + 2 \times 14$
$= 1 + 8 + 28$
$= 37$

$d_k > 0$

$x_{k+1} = 4 \quad y_{k+1} = 14$

$d_{k+1} = 1 - 2 \times 14 + 2 \times 4 + 1$

$= 1 - 28 + 9$

$= -18$

$x_{k+1} = (x_{k+1}) + 1$

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

$d_{k+1} = d_k - 2x_{k+1} + 2x_{k+1} + 1$

$d_{k+1} = -7$

$d_{k+1} = -7 + 2 \times 6 + 1$

$= -7 + 13$

$= 6$

$d_{k+1} = 6 - 2 \times 13 + 2 \times 7 + 1$

$= 6 - 26 + 15$

$= -5$

$d_{k+1} = -5 + 2 \times 8 + 1$

$= -5 + 17$

$= 12$

$d_{k+1} = 12 - 2 \times 12 + 2 \times 9 + 1$

$= 12 - 24 + 19$

$= 12 - 5$

$= 7$

$= 2 \times 11 + 2 \times 10 + 1$

$d_k < 0$

$x_{k+1} = 5, \quad y_{k+1} = 14$

$d_{k+1} = d_k + 2x_{k+1} + 1$

$= -18 + 2 \times 5 + 1$

$= -18 + 10 = -7$

$d_{k+1} < 0$

$x_{k+1} = 6 \quad y_{k+1} = 14$

$d_{k+1} > 0$

$x_{k+1} = 7 \quad y_{k+1} = 13$

$d_{k+1} < 0$

$x_{k+1} = 8 \quad y_{k+1} = 13$

$d_k > 0$

$x_{k+1} = 9 \quad y_{k+1} = 12$

$d_k > 0$

$x_{k+1} = 10 \quad y_{k+1} = 11$

$= 7 - 22 + 24$

$= 7 - 1$

$= 6$

$\Rightarrow$

$d_k > 0$

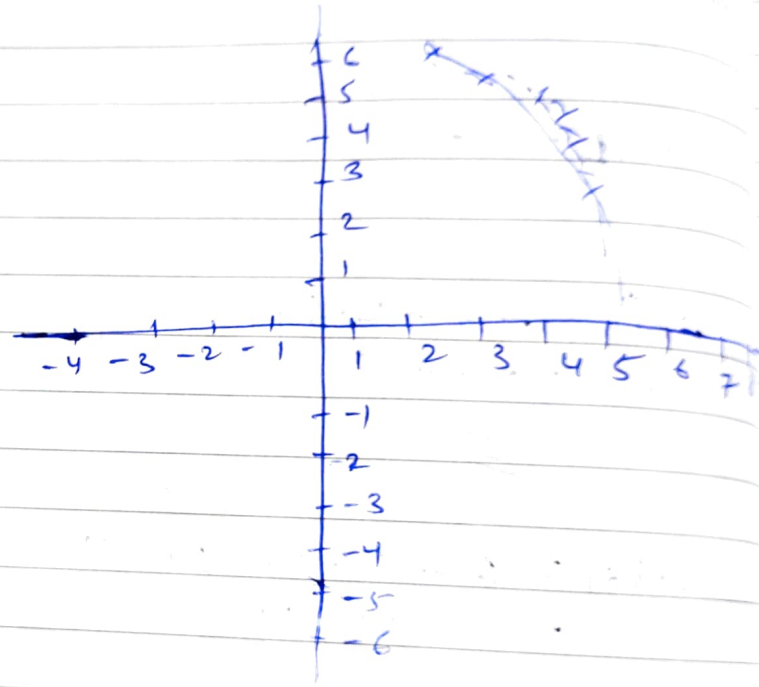
$x_{k+1} = 11 \quad y_{k+1} = 10$

Here we stop because

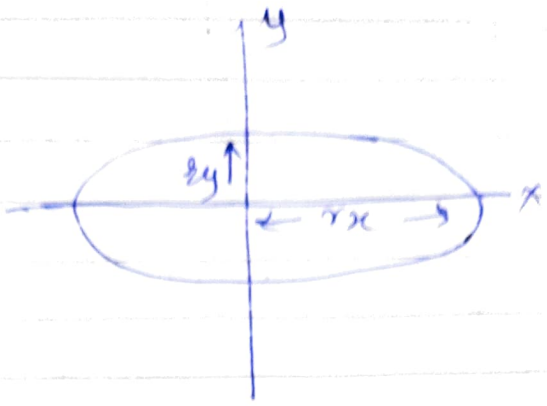
$x > y$

points are -

(1, 15)
(2, 15)
(3, 15)
(4, 14)
(5, 14)
(6, 14)
(7, 13)
(8, 13)
(9, 12)
(10, 11)
(11, 10)



Midpoint Ellipse Algorithm



ellipse is an elongated circle

eq^n of ellipse is centered at (0,0)

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

were $a = r_x$ & $b = r_y$

Major Axis $= 2a = 2r_x$ (longer axis)
Minor Axis $= 2b = 2r_y$ (short axis)

Semi major Axis $= a = r_x$
semi minor Axis $= b = r_y$

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

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

put $a = r_x$ & $b = r_y$

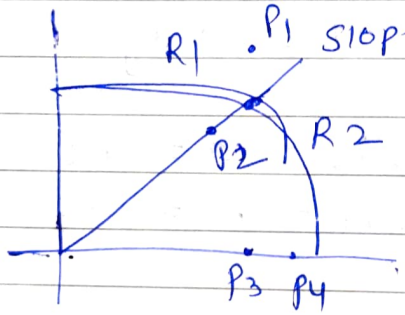
$$\therefore x^2r_y^2 + y^2r_x^2 - r_x^2r_y^2 = 0 \quad -(1)$$

If we put any point in eq (1)

case 1	case 2	case 3
$= 0$	< 0	> 0
on boundary	inside boundary	outside boundary

In Circle, 8 way symmetry
In ellips 4 way symmetry

In circle, we need to plot one octant & any quadrant but in ellips we need to plot 2 octants i.e. one complete quadrant.



In Region R_1 we can have next point at x increment, y to take decision.

(x_{k+1}, y_k) or (x_{k+1}, y_{k-1})

But in Region R_2 we are considering 2 points on major axis P_3 & P_4.

So here it depends on ~~this~~ whether to increment x or decrement x

In this Algo, we plot 1st quadrant which has 2 regions & have different slope in region R_1 & R_2 so we have to plot both regions by using different formulas.

Quadrant 1 → Region 1

i) start point : $(0, ry)$
ii) slope of ellips < -1
iii) Take unit steps in positive x direction till boundary between 2 regions is reached. → $x = x+1$

Quadrant 1 → Region 2

slope of ellips > -1

Take unit steps in negative y direction till the end of quadrant.

on the boundary between 2 regions, slope of the ellips is -1.

Here y is decremented by 1 so $y-1$ & will decide whether to increment x or not

Example - Midpoint Ellipse Algo

- Read radius r_x & r_y
- Initialize starting point & region 1 as
  $x = 0$, $y = r_y$ → $(0, r_y)$
- calculate $\boxed{P1_0 = r_y^2 - r_x^2 \cdot r_y + 1/4\, r_x^2}$
- calculate $dx = 2 r_y^2 x$, $dy = 2 r_x^2 y$
- Repeat while $(dx < dy)$ - Region 1

* plot (x, y)
* if $(P1 < 0)$
{
$x = x + 1$
update $dx = 2 r_y^2 \cdot x \approx$ old $(dx + 2 r_y^2)$
$P_1 = P_1 + 2 r_y^2 x + r_y^2$
}

* else
{
$x = x + 1$
$y = y - 1$
update $dx \rightarrow 2 r_y^2 x$ {old $dx + 2 r_y^2$}
update $dy \rightarrow 2 r_x^2 y$ {old $dy + 2 r_x^2$}
$P_1 = P_1 + dx - dy + r_y^2$
}

when $(dx \geq dy)$ plot Region 2 as :-

- Find $\boxed{P2_0 = r_y^2 (x + 1/2)^2 + r_x^2 (y-1)^2 - r_x^2 r_y^2}$
- Repeat till $(y > 0)$

* plot (x, y)
* if $(p_2 > 0)$
{
$x = x$
$y = y - 1$
update $dy = 2 r_x^2 y$
$p_2 = p_2 - dy + r_x^2$
}
else
{
$x = x + 1$
$y = y - 1$
$dy = 2 r_x^2$
$dx = 2 r_y^2$
$p_2 = p_2 + dx - dy + r_x^2$
}

Q. 1. $r_x = 8$, $r_y = 6$ given.

Iterat^n	(x_k, y_k)	p_1 or p_2	(x_{k+1}, y_{k+1})	dx $2x_{k+1} r_y^2$	dy $2y_{k+1} r_x^2$
0.	$(0, r_y)$ $(0, 6)$	$p_{10} = r_y^2 - r_x^2 . r_y + 1/4 . r_x^2$ $= 36 - 64 \times 6 + 1/4 . 64$ $= 36 - 384 + 16$ $= -332$ $\therefore p_1 < 0$	$(1, 6)$ x inc, y as it is	$2 \times 1 + 36 = 72$	$2 \times 6 + 64 = 768$
1.	$(1, 6)$	$p_1 = p_1 + 2 r_y^2 x + r_y^2$ $= -332 + 2 \times 36 + 36$	Now $dx < dy$		

					dx	dy
	$= -224$		(2, 6)	$= 72 \times 2$ $= 144$		768
2. (2, 6)	$P_1 = -224 + 144 + 36$ $= -44$		(3, 6)	$= 6(36)$ $= 216$		768
3. (3, 6)	$P_1 = -44 + 216 + 36$ $= 208$		(4, 5)	$8(36)$ $= 288$		10×64 $= 640$
					dx < dy	
4. (4, 5)	$P_1 = 208 + 288 + 36 - 640$ $= -108$		(5, 5) (6, 4)	$10(36)$ $= 360$		$= 640$
					dx < dy	
5. (5, 5)	$P_1 = -108 + 360 + 36$ $= 288$		(6, 4)	$= 12(36)$ $= 432$		$= 8(64)$ $= 512$
					dx < dy	
6. (6, 4)	$P_1 = 288 + 432 - 512 + 36$ $= 244$		(7, 3)	$= 14(36)$ $= 504$		$= 6(64)$ $= 384$

NOW $dx > dy$

P2

Plot Region 2

1. (7, 3)	$P_2 = ry^2(x + 1/2)^2 + rx^2(y-1)^2 - rx^2 \cdot ry^2$ $= 36(7 + 1/2)^2 + 64(3-1)^2 - 36 \times 64$ $= 1078 + 256 - 2304$ $= -23$	(8, 2)	$16(36)$ $= 576$	4×64 $= 256$
2. (8, 2)	$P_2 = -23 + 576 - 256 + 64$			

	$p_2 = 361$	(8,1)	16(36) = 576	2 (64) = 128
2. (8,1)	$p_2 = 361 - 128 + 64 = 297$	(8,0) Now stop when y=0	~~16(36)~~ —	—
3. (8,0)	—	—	—	—

Page No.:
Date:

chap 3.
2D Geometric Transformations.

* Transformation

- The process of changing the scale, shape or position of the object is called Transform

  - Translation
  - scaling
  - Rotation.

* Matrix Representation & Homogeneous Coordina

① Translation Homogeneous Matrix

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

$x' = x + tx$
$y' = y + ty$
$1 = 1$

② Rotation –

x-axis

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

y-axis

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

2-axis

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

③ scaling

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

$x' = x \cdot Sx \quad y' = y \cdot Sy$

Translation

- Shifting of an object along a straight path
- It does not alter the shape or size of the object. It just moves the entire object from one location to another location along a straight path.

$$T = [t_x, t_y]$$

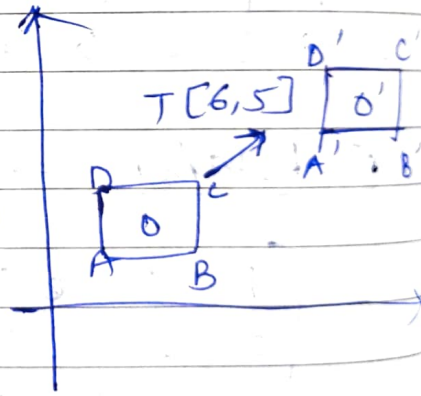
Translation operation -

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

where (x, y) is the original point, $[t_x, t_y]$ is shift vector & (x', y') is translated point.

$$P' = T + P$$

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



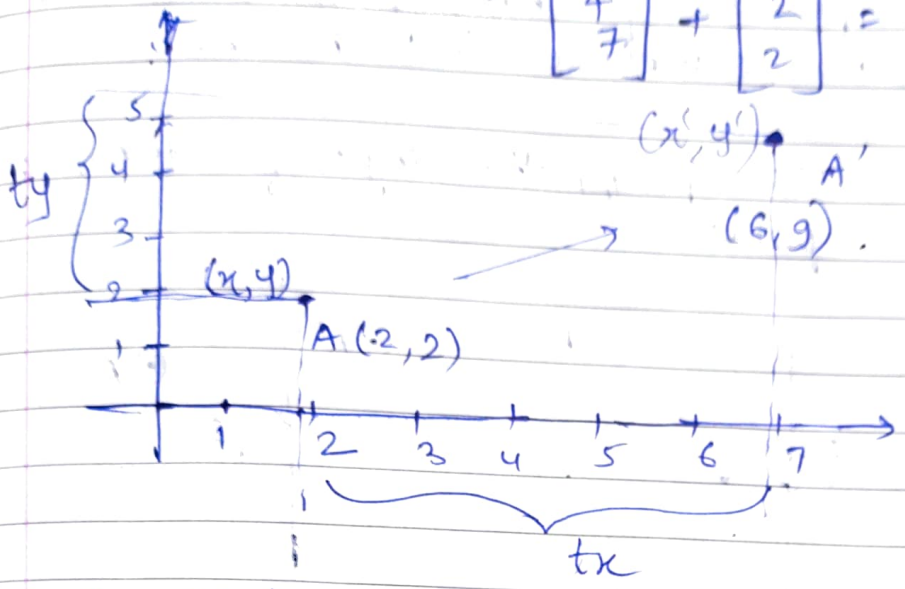
Ex:- Translate a pixel with co-ordinates A(2,2) with translation vector [4 7].

$$T = \begin{bmatrix} 4 \\ 7 \end{bmatrix}$$

$$\boxed{P' = T + P}$$

$\therefore A' = T + A$

$= \begin{bmatrix} 4 \\ 7 \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 6 \\ 9 \end{bmatrix}$



ex:- consider the line with endpoints $A(x_1, y_1)$ & $B(x_2, y_2)$

$\therefore x_1' = tx + x_1 \qquad x_2' = tx + x_2$

$y_1' = ty + y_1 \qquad y_2' = ty + y_2$

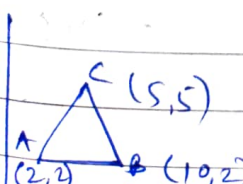
$\boxed{P' = T \cdot P}$

Homogeneous Representation of transformation would be

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

(1) A(2,2), B(10,2) & C(5,5) Translate the triangle with $tx = 5$ & $ty = 6$

→



$\boxed{P' = P + T}$

$A' = A + T$

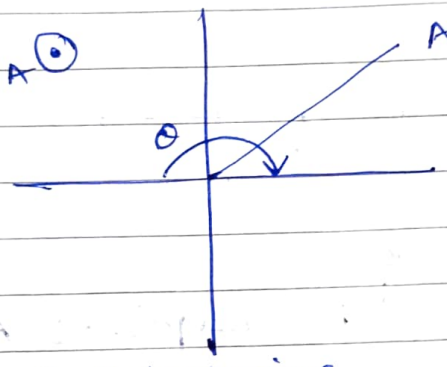
$A' = \begin{bmatrix} 2 \\ 2 \end{bmatrix} + \begin{bmatrix} 5 \\ 6 \end{bmatrix} = \begin{bmatrix} 7 \\ 8 \end{bmatrix}$

$B' = \begin{bmatrix} 10 \\ 2 \end{bmatrix} + \begin{bmatrix} 5 \\ 6 \end{bmatrix} = \begin{bmatrix} 15 \\ 8 \end{bmatrix}$

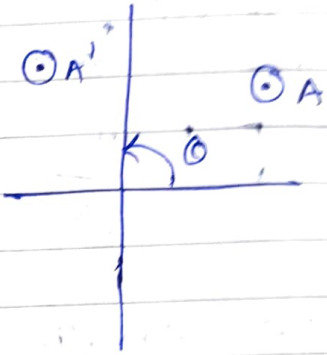
$C' = \begin{bmatrix} 5 \\ 5 \end{bmatrix} \begin{bmatrix} 5 \\ 6 \end{bmatrix} = \begin{bmatrix} 10 \\ 11 \end{bmatrix}$

* Rotation

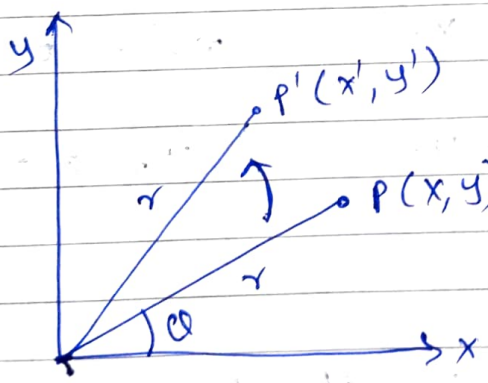
- It is a process of changing the object it can be clockwise or anticlockwise.
- We have to specify the angle of rotation & rotation point.



Anticlockwise



clockwise



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

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

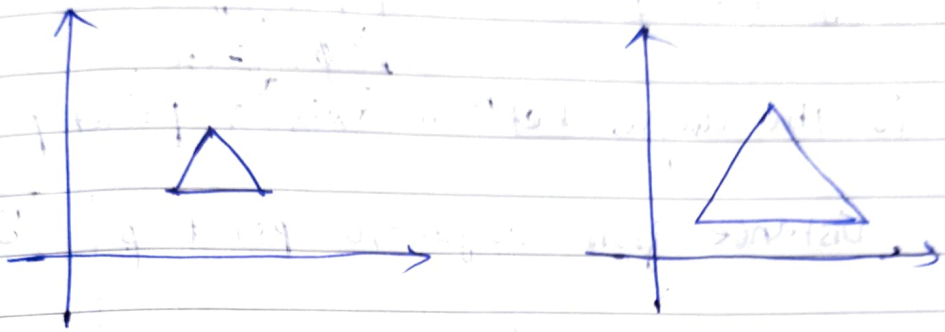
$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

$$P' = R \cdot P$$

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

* Scaling

- It is used to alter or change the size of objects. Change is done using scaling factors.



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

multiplying the vertex coordinates by scaling parameters S_x & S_y.

$$\boxed{P' = S \cdot P}$$

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

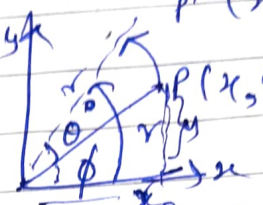
$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Anticlockwise direction (+ve)
clockwise direction (-ve)

(2) Rotation

Let consider a p in the polar. (Anticlockwise)

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



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

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

ϕ is the angle betn x axis & point p

r = Distance from origin to point p

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

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

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

$\therefore \boxed{x' = x\cos\theta - y\sin\theta}$

$y' = r[\sin\phi\cos\theta + \cos\phi\sin\theta]$
$= r\sin\phi\cos\theta + r\cos\phi\sin\theta$

$\therefore \boxed{y' = x\sin\theta + y\cos\theta}$

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

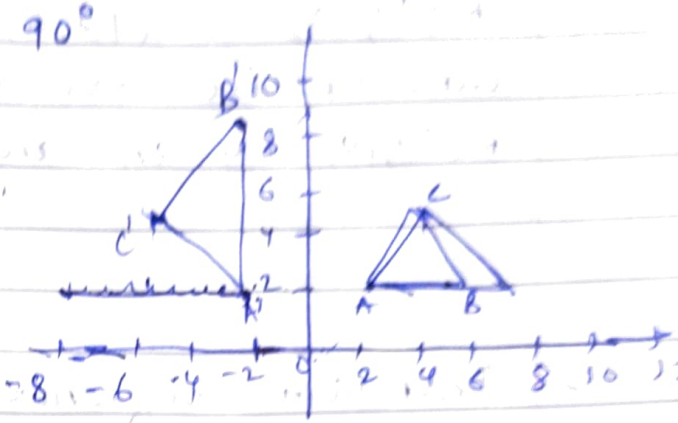
$$\boxed{P = R \cdot P}$$

Ex:- One triangle is given (2,2) (8,2) (5,5)
Rotate the triangle 90°

→ $\theta = 90°$

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

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



$A' = R \cdot A$

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

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

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

* Clockwise (-ve direction)

$x' = x\cos(-\theta) - y\sin(-\theta) \Rightarrow \boxed{x\cos(\theta) + y\sin\theta}$

$y' = x\sin(-\theta) + y\cos(-\theta) \Rightarrow \boxed{-x\sin\theta + y\cos\theta}$

Cos θ Always +ve

Q1 Consider a square P(0,0), Q(0,10) R(10,10) S(10,0). Rotate the square anticlockwise about fixed point R(10,10) by an angle 45°.

→

1. Translate reference point R to the origin
2. Perform a rotation by 45° in anticlockwise direction.
3. Inverse translate of point R.

$$P' = R \cdot P$$

Reference point (10,10)

$$P' = \begin{bmatrix} 1 & 0 & 10(tx) \\ 0 & 1 & 10(ty) \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos 45 & -\sin 45 & 0 \\ \sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & -10 \\ 0 & 1 & -10 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 10 & 10 \\ 0 & 10 & 10 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Multiply first two & last two matrixes

$$= \begin{bmatrix} 1 & 0 & 10 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.70 & -0.70 & 0 \\ 0.70 & 0.70 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

$$= \begin{bmatrix} 0.70 & -0.70 & 10 \\ 0.70 & 0.70 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -10 & -10 & 0 & 0 \\ -10 & 0 & 0 & -10 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$P' = \begin{bmatrix} 10 & 2.91 & 10 & 17.09 \\ -4.18 & 2.91 & 10 & 2.91 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

original co-ordinates	Transformed co-ordinates
P (0,0)	P' (10, -4.18)
Q (0, 10)	Q' (2.91, 2.91)
R (10, 10)	R' (10, 10)
S (10, 0)	S' (17.09, 2.91)

Q.2 consider a triangle with vertices A(1,1) B(5,2) & C(3,4) find out the transformation matrix which rotates given triangle about point C(3,4) by an angle 30° clockwise. find the co-ordinates of the rotated triangle.

→ 30° clockwise (-ve direction)

Ref point (3,4)

$$P' = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 30 & \sin 30 & 0 \\ -\sin 30 & \cos 30 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

$$= \begin{bmatrix} \frac{\sqrt{3}}{2} & 1/2 & \frac{2-3\sqrt{3}}{2} \\ -1/2 & \frac{\sqrt{3}}{2} & \frac{11-4\sqrt{3}}{2} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 5 & 3 \\ 1 & 2 & 4 \\ 1 & 1 & 1 \end{bmatrix}$$

$$P = M \cdot P = \begin{bmatrix} \frac{3-2\sqrt{3}}{2} & \frac{2\sqrt{3}+4}{2} & 3 \\ \frac{10-3\sqrt{3}}{2} & \frac{6-2\sqrt{3}}{2} & 4 \\ 1 & 1 & 1 \end{bmatrix}$$

original coordinates	Transformed coordinates
$A(1,1)$	$A'\left(\frac{3-2\sqrt{3}}{2}, \frac{10-3\sqrt{3}}{2}\right)$
$B(5,2)$	$B'\left(\frac{2\sqrt{3}+4}{2}, \frac{6-2\sqrt{3}}{2}\right)$
$C(3,4)$	$C'(3,4)$

Page No.
Date:

Scaling

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

$$\boxed{P' = S \cdot P}$$

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

$S_x \cdot S_y$ are any +ve values.

Ex:- A(1, 1) & B(1, 4).

$S_x = 3, \quad S_y = 2$

$$A' = S \cdot A = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

$$B' = S \cdot B = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \end{bmatrix} = \begin{bmatrix} 3 \\ 8 \end{bmatrix}$$

* Scaling with Respect to Reference point.

1. Translate the ref point to origin

$$T = \begin{bmatrix} -x_r \\ -y_r \end{bmatrix}$$

2. Apply scaling on the translated obj.

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

3) Translate ref point back to its actual location.

$$T^{-1} = \begin{bmatrix} x_r \\ y_r \end{bmatrix}$$

triangle

(2) scale a triangle with vertices A(2,2) B(6,2) & C(4,4) with respect to a ref point (3,4) with $S_x = 2$ & $S_y = 3$

→ scaling is with respect to a ref point.

first translate to origin
then apply scaling
then translate back

$$x' = S_x \cdot x + x_r (1 - S_x)$$

$$y' = S_y \cdot y + y_r (1 - S_y)$$

$x_r (1 - S_x)$ & $y_r (1 - S_y)$ are constant for all points.

$$x_r (1 - S_x) = 3 (1 - 2) = -3$$
$$y_r (1 - S_y) = 4 (1 - 3) = -8$$

$$A' = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \end{bmatrix} + \begin{bmatrix} -3 \\ -8 \end{bmatrix} = \begin{bmatrix} 4 \\ 6 \end{bmatrix} + \begin{bmatrix} -3 \\ -8 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

$$B' = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 6 \\ 2 \end{bmatrix} + \begin{bmatrix} -3 \\ -8 \end{bmatrix} = \begin{bmatrix} 9 \\ -2 \end{bmatrix}$$

$$C' = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 4 \\ 4 \end{bmatrix} + \begin{bmatrix} -3 \\ -8 \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \end{bmatrix}$$

original coordinates	Transformed co-ordinates
A (2, 2)	A' (1, -2)
B (6, 2)	B' (9, -2)
C (4, 4)	C' (5, 4)

Q. scaling on triangle (1,1) (8,1) & (1,9) with scaling factor 2 in both x & y direction.

→ $P' = S \cdot P$

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

$$= \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 8 & 1 \\ 1 & 1 & 9 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 16 & 2 \\ 2 & 2 & 18 \\ 1 & 1 & 1 \end{bmatrix}$$

original	
A (1, 1)	A' (2, 2)
B (8, 1)	B' (16, 2)
C (1, 9)	C' (2, 18)

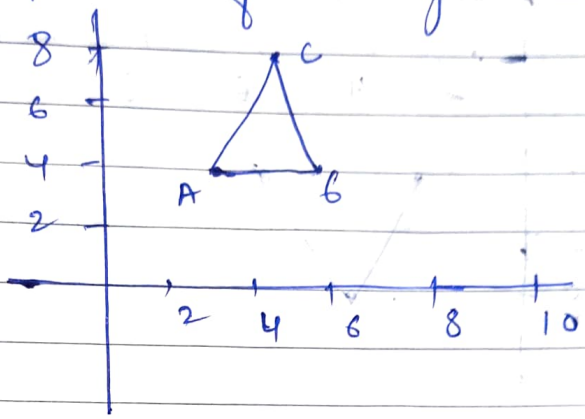
Q. Scale a triangle A(4,4) B(12,4) & C(8,10) with scaling factor $S_x = 2$ & $S_y = 1$ [Ex - map23]

* Reflection

- Produces a mirror image of an object about a given axis.
- Reflection is achieved by rotation an obj by 180° around ref axis, perpendicular to the xy plane.
- It does not alter the shape & size of the object

Page No.:
Date:

ex:- A triangle ABC is given
A (3,4) B(6,4) C (4,8) find Reflected position of triangle to the x-axis



→ Ref to the x-axis

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

A point after Reflection.

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

B pt

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

C pt

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

* Ref about origin

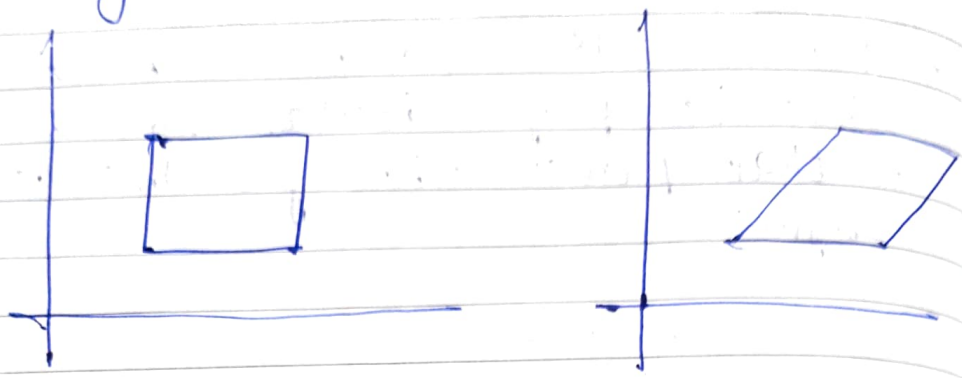
- This is 180° rotation about z axis that is perpendicular to the xy plane & that passes through the coordinate origin.

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

$$\text{Ref (origin)} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Shearing

x-direction



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

Shx is shear parameter in x-direction & y is hight of the internal layer from the base.

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

$$P' = SHx \cdot P$$

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

ex: Apply shearing on cube with vertices A(0,0) B(2,0), C(2,2) & D(0,2) with Shx = 3

→ for x-direction shearing with y=0 as ref line

$x' = x + shx \cdot y$

$y' = y$

$A' = 0 + 3(0) = 0$ $\quad (0, 0)$

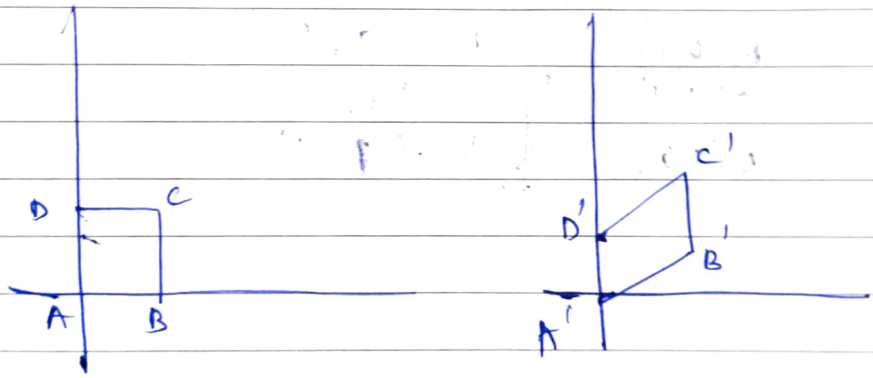
$B' = 2 + 3(0) = 2$ $\quad (2, 0)$

$C' = 2 + 3(2) = 8$ $\quad (2, 2)$

$D' = 0 + 3(2) = 6$ $\quad (0, 2)$

A (0,0)	A' (0, 0)
B (2, 0)	B' (2, 0)
C (2, 2)	C' (8, 0)
D (0, 2)	D' (6, 0)

y - direction



$x' = x$

$y' = y + x \cdot shy$

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

Ex: Apply y direction shearing on unit square with the base on $x=0$ line & one corner at the origin with $shy = 1/2$ A(0,0) B(1,0), C(1,1) & D(0,1). For y direction shear.

→ $x' = x$

$y' = y + x \cdot shy$

$A' = 0 + (0)\,1/2 = 0$

$B' = 0 + (1)\,1/2 = 1/2$

$C' = 1 + (1)\,1/2 = 3/2$

$D' = 1 + (0)\,1/2 = 1$

A(0,0)	A'(0,0)
B(0,1)	B'(0, 1/2)
C(1,1)	C'(1, 3/2)
D(1,0)	D'(1, 1)

Page No.
Date

Q. Translate ABCD with coordinates A(0,0) B(5,0) C(5,5) D(0,5) by 2 units x-direc & 3 unit in y-direc.

$$P' = M \cdot P = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \begin{matrix} \leftarrow t_x \\ \leftarrow t_y \\ \\ \end{matrix} \begin{bmatrix} 0 & 5 & 5 & 0 \\ 0 & 0 & 5 & 5 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$P' = \begin{bmatrix} 2 & 7 & 7 & 2 \\ 3 & 3 & 8 & 8 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Q) Triangle (10,10) (40,10) (30,30) Apply scaling with scale factor 5 in x & y direc draw triangle

⇒ $S_x = S_y = 5$

$$P' = S \cdot P = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 & 40 & 30 \\ 10 & 10 & 30 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 50 & 200 & 150 \\ 50 & 50 & 150 \\ 1 & 1 & 1 \end{bmatrix}$$

Q) Rotate a triangle ABC by 30°. Where the triangle is A(0,0) B(10,2) & C(7,8)

$$P' = M \cdot P = R(\theta = 30) \times P = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \cos 30 & -\sin 30 & 0 \\ \sin 30 & \cos 30 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 10 & 7 \\ 0 & 2 & 4 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & \frac{10\sqrt{3}-2}{2} & \frac{7\sqrt{3}-4}{2} \\ 0 & \frac{10+2\sqrt{3}}{2} & \frac{7+4\sqrt{3}}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

Q. (1) Find transformation matrix to rotate the obj abt the origin by 45° in counter clockwise direction.

(2) Find new coordinates of the point (8,4) after rotation.

$$\Rightarrow M = \begin{bmatrix} \cos 45 & -\sin 45 \\ \sin 45 & \cos 45 \end{bmatrix} \cdot \begin{bmatrix} 8 \\ 4 \end{bmatrix} = \begin{bmatrix} 2\sqrt{2} \\ 6\sqrt{2} \end{bmatrix}$$

Q. Explain the steps for 2D reflection w.r.t. line $y = mx$ & also derive a composite transformation matrix. — Exam Dec 19

- Reflection abt any arbitrary line $y = mx + c$ can be accomplished by the combination of translation, scaling, rotation & reflection
- First translate the line which passes through origin.
- Then rotate the line so that it aligns with one of the principal axes.

- Perform reflection operation.
- Finally, restore the line to its actual position by performing inverse rotation & inverse translation.

- The composite transformation matrix for this seq of operation is obtained by multiplying following matrices.

$$M = T^{-1} \cdot R^{-1} \cdot Ref \cdot R \cdot T$$

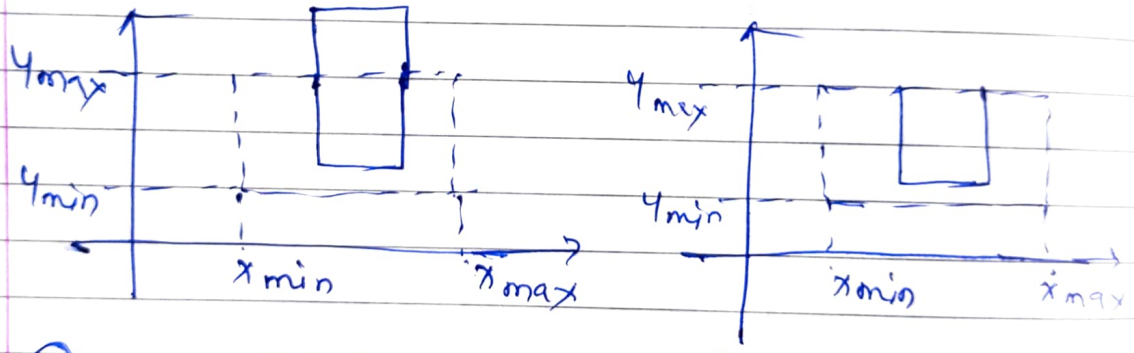
Chap 4: 2D viewing & clipping

* 2D viewing

- Graphics packages allow the user to specify which part of the scene is to displayed

- process of selecting the part of the real-world scene to display it on devices called windowing.

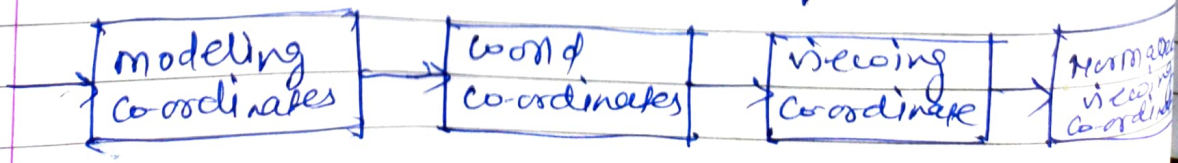
- clipping is the process of deciding & removing the portion of the obj which is outside the clipping window.



(a) word co-ordinates (b) viewport co-ordinates

The process of mapping the part of world co-ordinate scene to device co-ordinate is called viewing transformation or windowing transformation

* window to viewport Transformation.



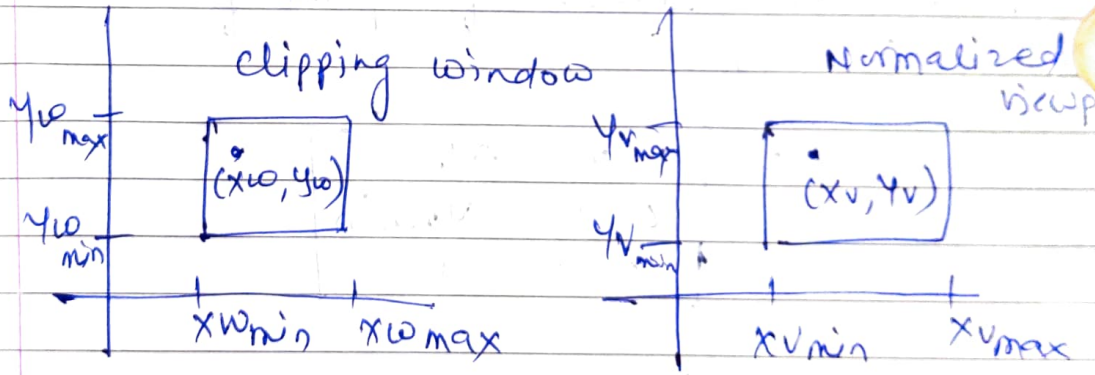
* window to viewport co-ordinate transformation

- window to the viewport transformation is necessary because the size of the window & viewport may not be the same all the time.

So actual picture selected by window needs to be rescaled to fit it in the viewport.

Let (xw_{min}, yw_{min}) & (xw_{max}, yw_{max}) point of clipping window.

- (xv_{min}, yv_{min}) & (xv_{max}, yw_{max}) point of viewport.



$$\frac{xv - xv_{min}}{xv_{max} - xv_{min}} = \frac{xw - xw_{min}}{xw_{max} - xw_{min}}$$

$$xv - xv_{min} = (xv_{max} - xv_{min}) \cdot \frac{xw - xw_{min}}{xw_{max} - xw_{min}}$$

$$xv = xv_{min} + (xw - xw_{min}) \cdot sx$$

Similary for sy.

where the scaling factors are -

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

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

Matrix

If the lower left corner of the window is not at the origin, we translate it to the origin before we map the point in window to viewport.

(1) Translate lower-left point of window to the origin.

$$T = \begin{bmatrix} 1 & 0 & -XW_{min} \\ 0 & 1 & -YW_{min} \\ 0 & 0 & 1 \end{bmatrix}$$

(2) Apply scaling to viewport.

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

(3) Inverse translation in the viewport.

$$T^{-1} = \begin{bmatrix} 1 & 0 & x_{Vmin} \\ 0 & 1 & y_{Vmin} \\ 0 & 0 & 1 \end{bmatrix}$$

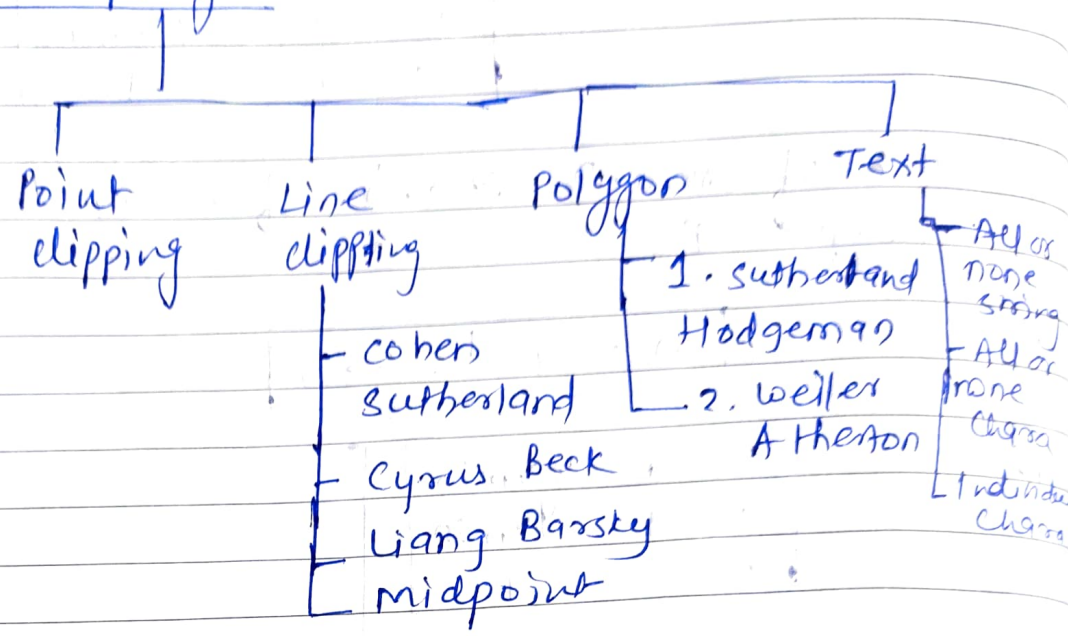
Transformation Matrix for window to viewport is -

$$\boxed{M = T^{-1} \cdot S \cdot T}$$

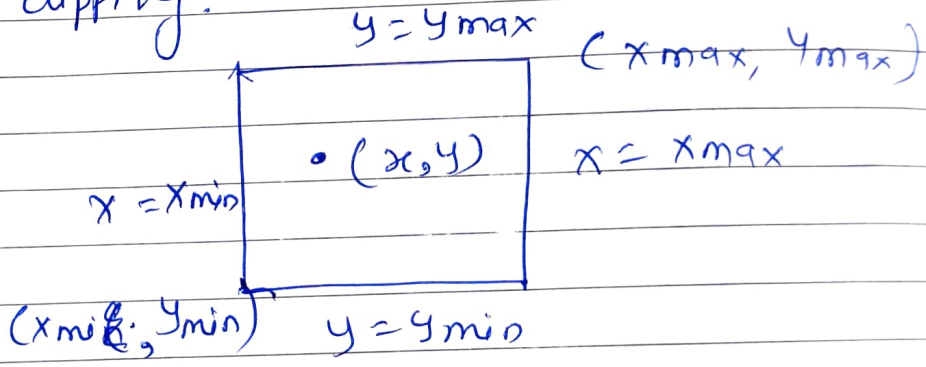
$$= \begin{bmatrix} 1 & 0 & x_{Vmin} \\ 0 & 1 & y_{Vmin} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

* 2D clipping

-



① Point clipping.



- The point can be either fully inside or fully outside the clipping region.

- ~~The point can be either fully inside or fully outside the clipping region.~~

The point on the clipping window boundary is considered inside.

Let (x_{min}, y_{min}) & (x_{max}, y_{max}) represents the lower-left & top-right corner of the clipping window.

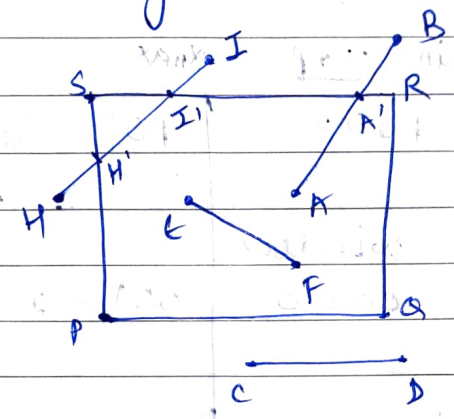
Point (x, y) is inside the region only if the following four inequalities are true,

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

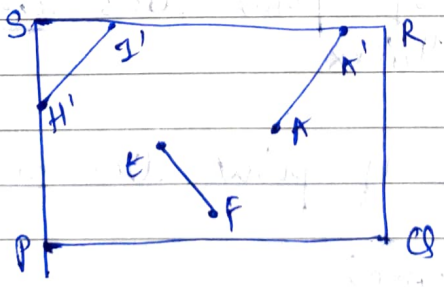
$$y_{min} \leq y \leq y_{max}$$

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

(2) Line clipping

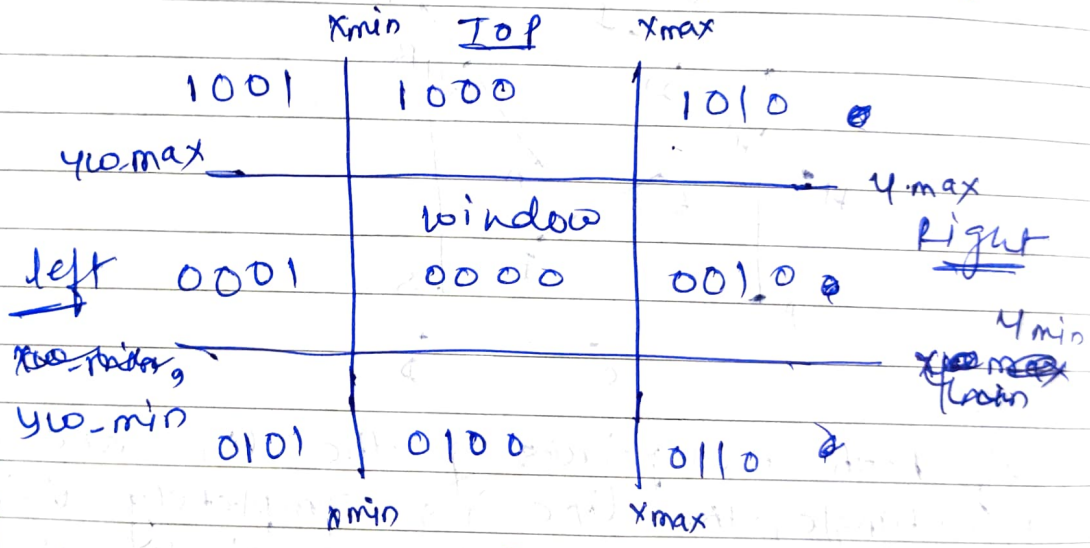


- If both end points of the line are within a rectangle, the line is completely visible. otherwise line may be partially visible or completely outside the window.



Cohen-Sutherland Line clipping Algorithm

- In this algo, we are given 9 regions on the screen out of which one region is of the window & the rest 8 regions are around it given by 4 digit binary
- The division of regions are based on (x_max, y_max) & (x_min, y_min)
- The central part is the viewing region or window. All the lines which lie within this region are completely visible



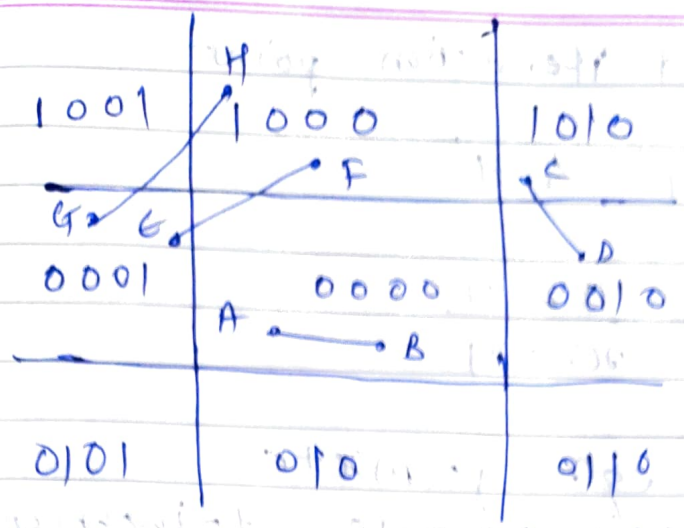
① Set first bit if point lies above window i.e. $y > y_{max}$

② Set second bit if point lies below $y < y_{min}$

③ Right $x > x_{max}$

④ Left $x < x_{min}$

TBRL code.



① If both endpoints of a line inside window line is visible. i.e. 0000 logical OR-AND conditions are evaluated.

② If logical OR of the region code is not 0000 there are two possibilities, the line may be partially visible or it may be completely outside.

if $x < x_{min}$ → left of window

$x > x_{max}$ → Right

$y < y_{min}$ - Bottom

$y > y_{max}$ - Top.

Ex:- Line GH

(0001) . (1000)

G H

0 0 0 1

1 0 0 0

0 0 0 0

Perform logical AND

If result ≠ 0000 line is outside

line partially invisible

- Find intersection point

G' & H'

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

line eqn $y = mx + c$
we can compute y-intercept of line as $c = y - mx$

- y-coordinate of an intersection with a vertical window boundary can be cal as
$y = mx + c$ where x is either x_{min} or x_{max} depends on for which vertical line we are cal. y.

$$y = m \cdot x_{min} + c$$
or. $$y = m \cdot x_{max} + c$$

x coordinate of an intersection with a horizontal window boundary is computed as,

$$x = (y - c)/m$$

where y is either y_{min} or y_{max} depends on for which horizontal line we are cal x.

$$x = \frac{y_{min} - c}{m}$$

$$\text{or } x = \frac{y_{max} - c}{m}$$

Algo

1. Read (x_{min}, y_{min}) & (x_{max}, y_{max})
2. Read A (x_1, y_1) & B (x_2, y_2) end pts of line
3. Compute outcode of A & B

if $y_1 > y_{max}$ then outcode A1 = 1 else 0

if $y_1 < y_{min}$ then outcode A2 = 1 else 0

if $x_1 > x_{max}$ then outcode A3 = 1 else 0

if $x_1 < x_{min}$ then outcode A4 = 1 else 0

if $y_2 > y_{max}$ — B1 = 1 else 0

$y_2 < y_{min}$ — B2 = —

$x_2 > x_{max}$ — B3 = —

$x_2 < x_{min}$ — B4 = —

4. If outcode_A OR outcode_B == 0000 then disp entire line & goto Step 5

else

if outcode_A AND outcode_B ≠ 0000 then Reject the entire line

else

compute the intersection point with window boundaries

Repeat step 4

5. Stop

Cohen Sutherland Algo.

1. Cal position of both endpoints of the line.
2. Perform OR operation on both these end points.
3. If OR gives 0000
   Then
   line is visible
   else
   Perform AND operation
   If AND ≠ 0000
   then line is invisible
   else
   AND = 0000
   Line is considered the clipped case
4. If a line is clipped case, find an intersection with boundaries of the window.

$$m = (y_2 - y_1) / (x_2 - x_1)$$

(a) If bit 1 is "1" line intersect with left boundary of rectangle window

$$y_3 = y_1 + m(x - x_1)$$

where $x = x_{wmin}$

(b) If bit 2 is "1" line intersect with Right boundary

$$y_3 = y_1 + m(x - x_1)$$

where $x = x_{wmax}$

(c) If bit 3 is "1" line intersects with bottom boundary

$$x_3 = x_1 + (y - y_1) / m$$

where

$$y = y_{wmin}$$

(d) If bit 4 is "1" line intersects with top boundary

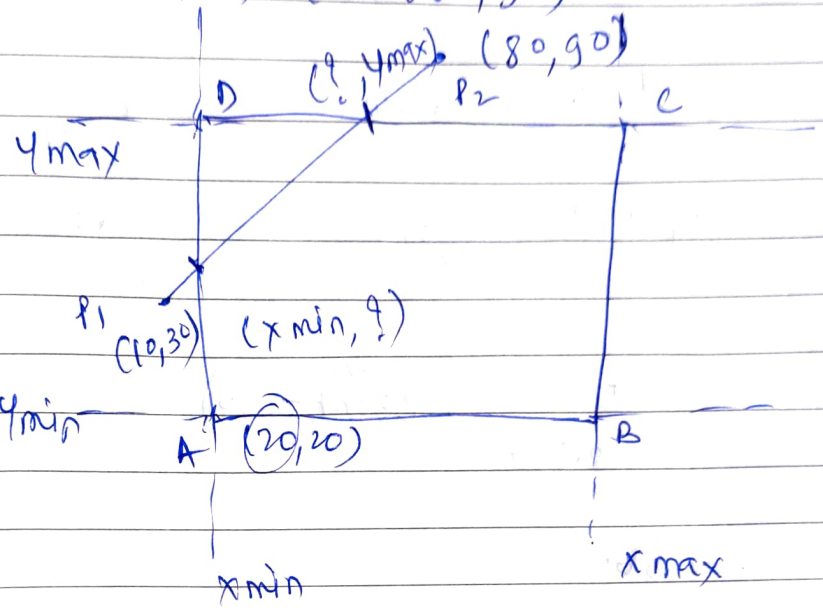
$$x_4 = x_1 + (y - y_1) / m$$

where

$$y = y_{wmax}$$

freedom

Q. Let ABCD be rectangle window with A(20,20) B(90,20) C(90,70) & D(20,70) find region code for the endpoints & use cohen sutherland line clip algo to clip the line P_1P_2 with $P_1(10,30)$ & $P_2(80,90)$



$P_1 = 0001$ TBRL

$P_2 = 1000$

AND 0000

OR 1001

} partially visible.

Slop, $m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{90-30}{80-10} = \frac{6}{7} = 0.857$

$P_1(10,30)$ $P_2(80,90)$

(x_1, y_1) (x_2, y_2)

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

$= 0.857(20-10) + 30$

$y = 38.57$

1st intersection point

$i_1 = (20, 38.57) \cong (20, 39)$

$(x_{min}, 4)$

2nd intersection.

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

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

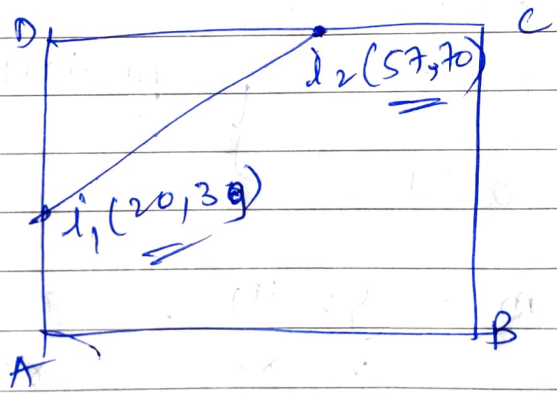
$= 56.67$

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

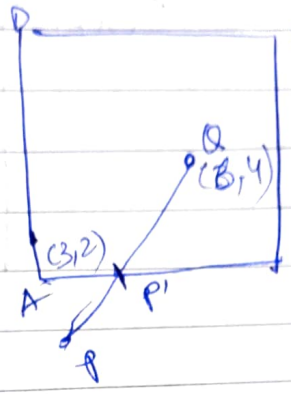
$y = y_1 + m(x_{min} - x_1$

else

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



Q. clip the line PQ, P(4,1) Q(6,4) against the clip window A(3,2) B(7,2) C(7,6) & D(3,6) using cohen sutherland

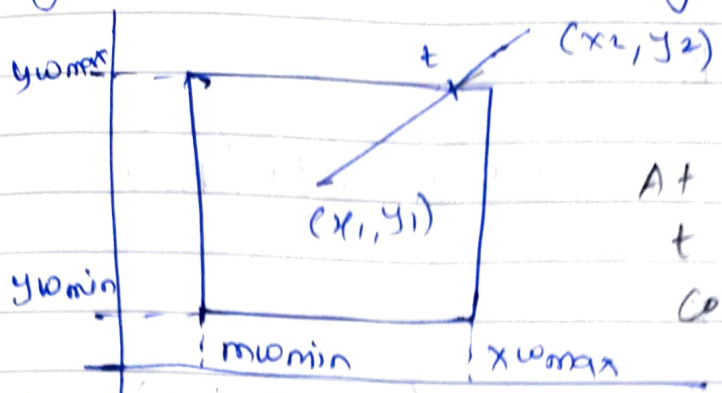


$y = (2, y_{min})$

$x = (4.67, 2)$

↑ new

Liang Barsky Line clipping Algo



At time interval t what are the co-ordinates?

(1) $x = x_1 + t\,\Delta x$

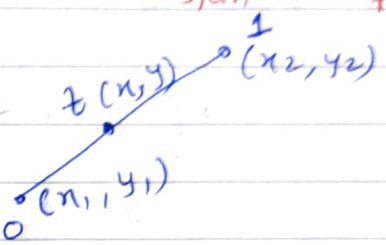
$y = y_1 + t\,\Delta y$

$0 \le t \le 1$

start — final

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

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



(2) $xw_{min} <= x_1 + t(x_2 - x_1) <= xw_{max}$

$yw_{min} <= y_1 + t\underbrace{(y_2 - y_1)}_{\Delta y} <= yw_{max}$

$t\Delta x \le xw_{max} - x_1$ $\qquad$ $t\Delta x \ge xw_{min} - x_1$

$t\Delta y \ge yw_{min} - y_1$ $\qquad$ $t\Delta y \le yw_{max} - y_1$

(3) $tp_k <= q_k$ $\qquad \{k = 1, 2, 3, 4\}$

$p_1 = -\Delta x$, $q_1 = x_1 - xw_{min}$ (Left Boundary)

$p_2 = \Delta x$, $q_2 = xw_{max} - x_1$ (Right B)

$p_3 = -\Delta y$, $q_3 = y_1 - yw_{min}$ (Bottom B)

$p_4 = \Delta y$, $q_4 = yw_{max} - y_1$ (Top B)

(4) If $p_k = 0$ then the line is parallel to window

if $q_k < 0$ — Line completely outside.

if $p_k < 0$

$$t_1 = \max(0, -q_k/p_k)$$

if $p_k \geq 0$

$$t_2 = \min(1, q_k/p_k)$$

if $t_1 > t_2$ → Line completely outside & it can be rejected.

if $t_1 < t_2$ then

$$x = x_1 + t\Delta x \qquad x_{wmin}, x_{wmax}$$
$$y = y_1 + t\Delta y \qquad y_{wmin}, y_{wmax}$$

} 4 times

---

condition

1. $p_k = 0$ — parallel to clipping boundary
2. $p_k = 0$ & $q_k < 0$ — completely outside
3. $p_k = 0$ & $q_k >= 0$ — inside the parallel clipping
4. $p_k < 0$ — line proceeds from outside to
5. $p_k > 0$ — line proceeds from inside to outside.

$y = y_1 + t\Delta y$

* Algorithm

1. Read 2 endpoints $P_1(x_1, y_1)$ & $P_2(x_2, y_2)$

2. Read 2 corners (left top & right Bottom) of the clipping window as -

   $(x_{wmin}, y_{wmin}, x_{wmax}, y_{wmax})$

3. cal values of parameter p_i & q_i for $i = 1, 2, 3, 4$

- $p_1 = -dx$ $\quad q_1 = x_1 - x_{wmin}$
- $p_2 = dx$ $\quad q_2 = x_{wmax} - x_1$
- $p_3 = -dy$ $\quad q_3 = y_1 - y_{wmin}$

$p_4 = dy$, $q_4 = y_{max} - y_1$

(4) If $p_i = 0$ line is parallel

If $q_i < 0$ line is completely outside so discard

else

check line is horizontal or vertical

(5) Initialize t_1 & t_2 as $t_1 = 0$ $t_2 = 1$

Conditions.

(6) Parameters t_1 & t_2 can be calculated that define the part of line that lies within the clip rectangle.

when

1. $p_k < 0$, $\max(0, q_k/p_k)$

2. $p_k > 0$, $\min(1, q_k/p_k)$

If $(t_1 > t_2)$ — line completely outside

otherwise the endpoints of the clipped line are calculated from the two values of parameter t.

(7) If $(t_1 < t_2)$

{

$xx_1 = x_1 + t_1 dx$

$xx_2 = x_1 + t_2 dx$

$yy_1 = y_1 + t_1 dy$

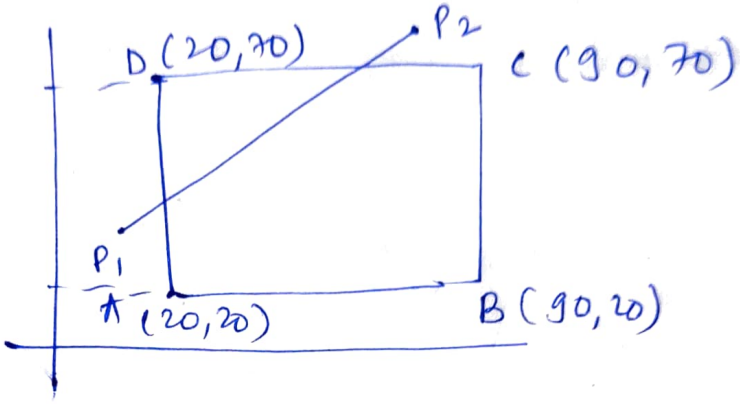
$yy_2 = y_1 + t_2 dy$

line (xx1, yy1, xx2, yy2)

}

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

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



$p_1 = -\Delta x$
$= -(80-10)$
$= -70$

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

q_1/p_1
$= -70/-70$
$= 1$

$p_2 = \Delta x$
$= 70$

$q_2 = x_{wmax} - x_1$
$= 90 - 10$
$= 80$

q_2/p_2
$= 80$
70

$p_3 = -\Delta y$
$= -60$
$=$

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

$p_4 = \Delta y$
$= 60$

$q_4 = y_{wmax} - y_1$
$= 70 - 30$
$= 40$

$(P_K < 0)$

$$t_1 = \max\left(0, \frac{q_1}{p_1}, \frac{q_3}{p_3}\right)$$

$$= \max\left(0, \frac{-10}{-70}, \frac{10}{-60}\right)$$

$$t_1 = 1/7$$

$(P_K > 0)$

$$t_2 = \min\left(1, \frac{q_2}{p_2}, \frac{q_4}{p_4}\right)$$

$$= \left(1, \frac{80}{70}, \frac{40}{60}\right)$$

$$t_2 = 2/3$$

$(t_1 < t_2)$ then find (x, y) for t_1 & t_2 ⇒

$$x = x_1 + t_1 \Delta x$$

$$y = y_1 + t_1 \Delta y$$

$$x = 10 + 1/7 \times 70 = 20$$

$$y = 30 + 1/7\ 60 = 38.57$$

for t_1

A $(20, 38.57)$

$$y = y_1 + t_2 \Delta y$$

$$= 30 + 2/3 \times 60$$

$$= 70$$

$$x = x_1 + t_2 \Delta x$$

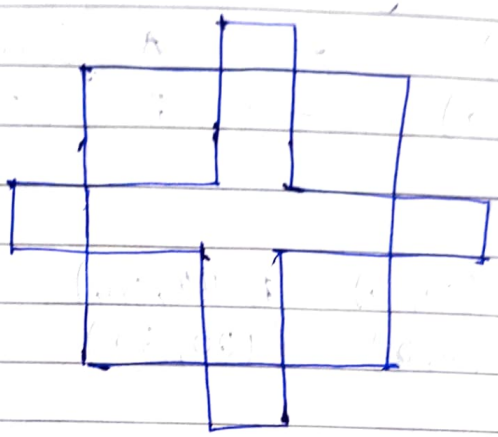
$$= 10 + 2/3 \times 70$$

$$= 56.67$$

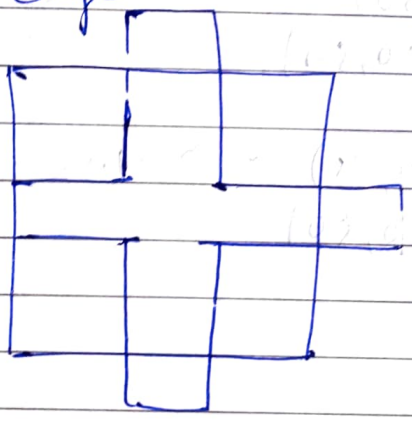
$B = (56.67, 70)$ ⇒

* Polygon clipping

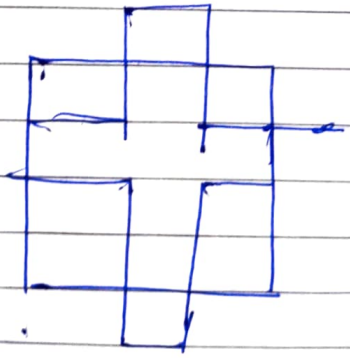
① Sutherland - Hodgeman Polygon clipping



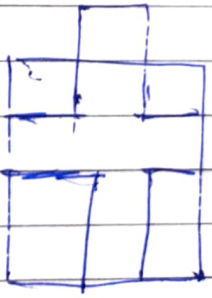
① clip Lef edge



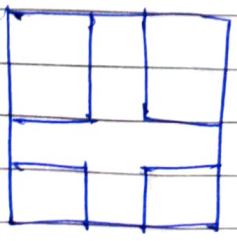
② Right



③ Bottom



④ Top

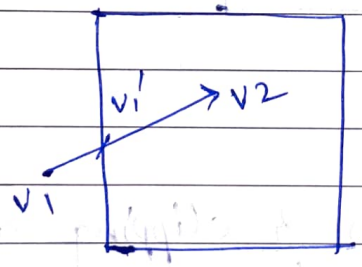


The sutherland Hodgman algo performs a clipping of a polygon against each window edge in turn

(1) clipping against the left side of the clip window
(2) clipping against ~~Right~~ top side
(3) clipping against right side
(4) clipping against bottom side.

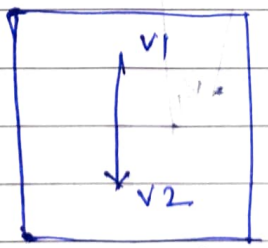
It consist of Four cases:-

1. If the first vertex of the edge is outside of window & second vertex is inside then the intersection point of the polygon edge with the window boundary & the second vertex are added to the O/P vertex list.



save { v_1' & v_2 }

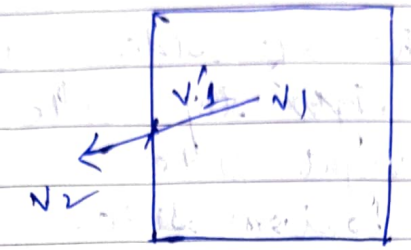
2. If both vertices of the edge are inside the window. Then only second vertex is added to the o/p vertex list.



save { v_2 }

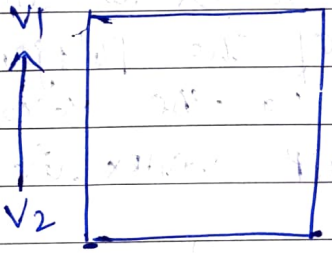
3. If the first vertex of the edge is inside the window & second vertex is outside

then only the intersection point of the polygon edge with the window boundary is added to the o/p vertex list.



save {V1'}

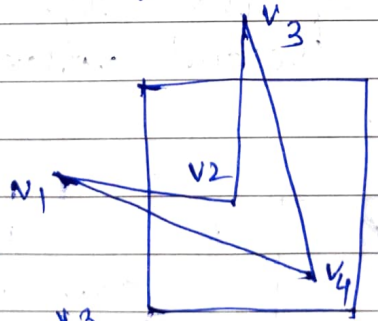
4. If both vertices of the edge are outside the window. Then nothing is added to the o/p vertex list.



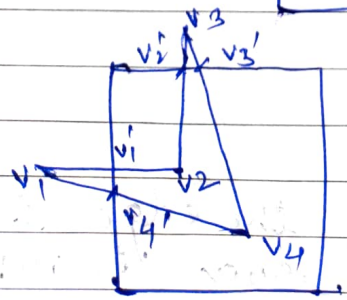
save {ϕ}

Ex:-

Q1. For a polygon & clipping window give the list of vertices after boundary clipping.

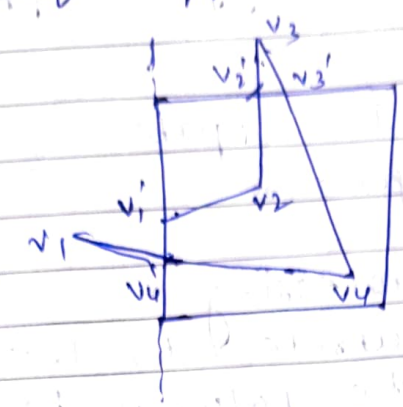


⇒



v_1v_2	out-in	v_1'
$v_1'v_2$	in-in	v_2
v_2v_3	in-out	v_2'

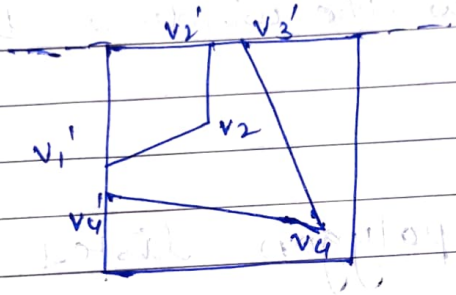
1. Left clipped:



∴ O/P vertex =

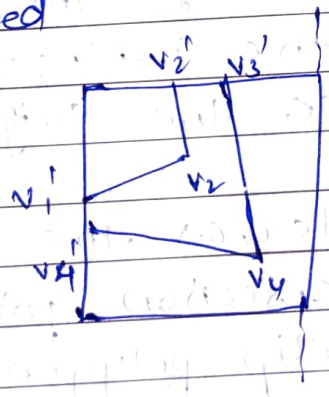
$\{v_1', v_2, v_2', v_3, v_3', v_4, v_4'\}$

3. Top clipped



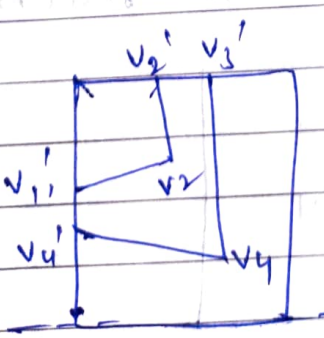
O/P = $\{v_1', v_2, v_2', v_3', v_4, v_4'\}$

2. Right clipped



O/P = $\{v_1', v_2, v_2', v_3', v_4, v_4'\}$

4. Bottom clipped



O/P = $\{v_1', v_2, v_2', v_3', v_4, v_4'\}$

Finally clipped polygon: $\{v_1', v_2, v_2', v_3', v_4, v_4'\}$

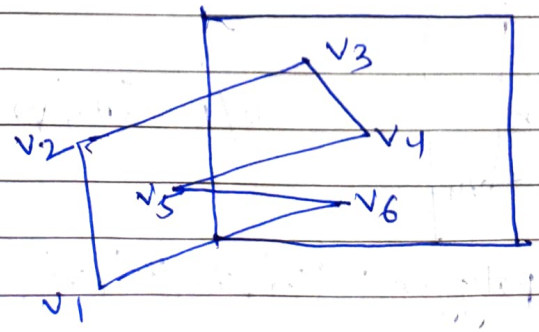
* Weiler - Atherton Polygon clipping Algo

- This Algorithm is capable of clipping of a concave polygon.
- For clockwise processing :
  - For an outside - to - Inside pair of vertices follow the polygon boundary.
  - For an Inside - to - outside pair of vertices, follow the window boundary.

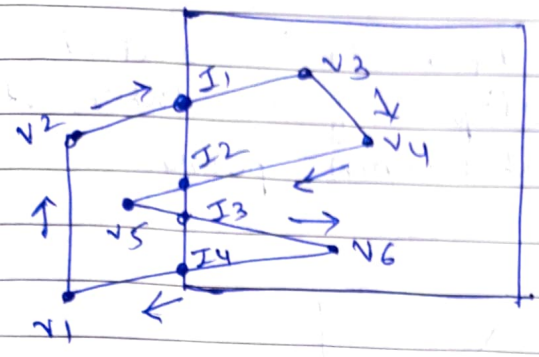
Algorithm :-

- Assume the polygon listed in clockwise order.
- If the edge enter the clip ~~window~~ polygon, record the intersection points & continue to trace the subject polygon.
- If the edge leaves the clip polygon, record the intersection point & make a right follow the clip polygon in same manner.
- Continue until vertex reach visited vertex

Ex :-



⇒ 1. clockwise notation in subject polygon.



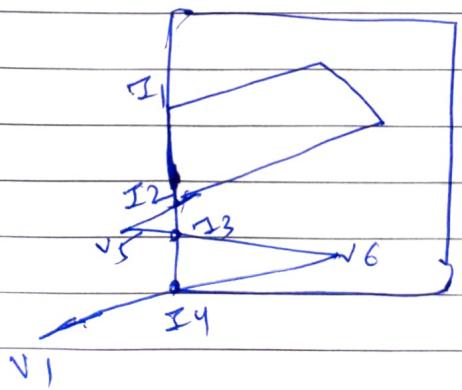
Here Intersection points are I_1, I_2, I_3 & I_4

2. Start from V_1 vertex to V_2 vertex in clockwise direction, both outside vertices then leave.

3. From V_2 to V_3 in clkwise, Here V_2 is outside & V_3 in inside, so record Intersection point I_1 & continue to V_3 vertex.

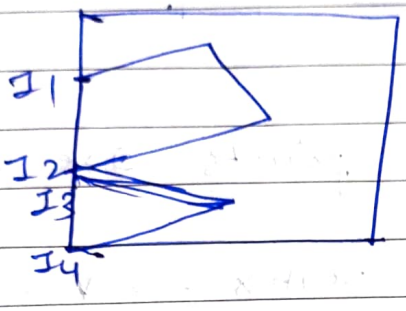
4. From V_3 to V_4 both are inside, so continue to subject polygon to V_4 vertex

5. From V_4 vertex to V_5 both V_4 in inside point & V_5 is outside so continue to clip polygon to intersection point from I_2 to I_1



6. From V_5 to V_6 : follow V_6

7. From V_6 to V_1 . V_1 is outside so continue to clip from intersection point I_4 to I_3 intersection point

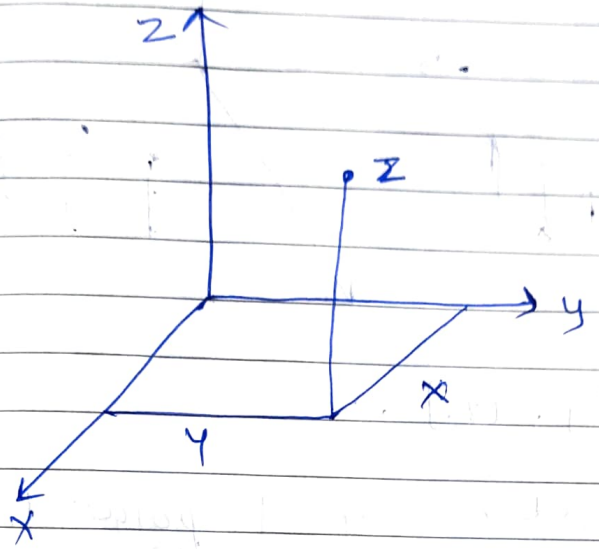


Page No.
Date

Chap 5.

3D Transformation.

- In 2D, the point is described by (x, y) co-ordinates in xy plane, whereas in 3D the point (x, y, z) is defined.



2D geomery divides the co-ordinate system into four quadrants.
3D divides the co-ordinate sys into eight octants.

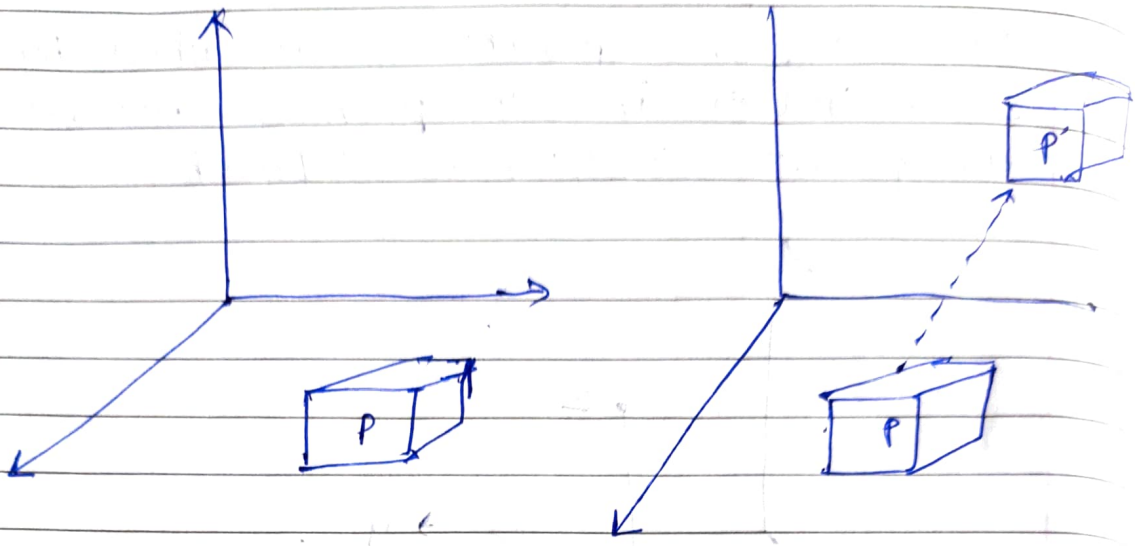
- Distance between two points A (x_1, y_1, z_1) & B (x_2, y_2, z_2) is computed by -

$$d(A,B) = \sqrt{(x_1-x_2)^2 + (y_1-y_2)^2 + (z_1-z_2)^2}$$

- When the point is projected on any plane using a perpendicular projection, the 3rd co-ordinate becomes zero.

- Any point on the xy plane will be $(x, y, 0)$ & any point on x-axis will be $(x, 0, 0)$ Similar observation can be made for other planes.

① Translation



$T = [t_x, t_y, t_z]$.

- Let us consider original point $P(x, y, z)$ which becomes $P'(x', y', z')$

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

Homogeneous Matrix —

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

$$P' = T \cdot P$$

Inverse Matrix

$$T^{-1} = \begin{bmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Q. Translate a triangle with vertices A(2,2,2) B(3,4,7) & C(8,9,12) by translation vector T[2,4,5].

$$P' = T \cdot P = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 3 & 8 \\ 2 & 4 & 9 \\ 2 & 7 & 12 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 5 & 10 \\ 6 & 8 & 13 \\ 7 & 12 & 17 \\ 1 & 1 & 1 \end{bmatrix}$$

A (2, 2, 2) — A' (4, 6, 7)
B (3, 4, 7) — B' (5, 8, 12
C (8, 9, 12) — C' (10, 13, 17)

Page No.:
Date:

* Scaling

- Multiplying all co-ordinates of the object by some constants called scaling parameters : S

$$S = [S_x, S_y, S_z]$$

- Scaling alters the shape & size of the obj.

- scaling can be performed in either two ways

① With Respect to the origin.
② With Respect to reference point.

① With Respect to origin.

$$P' = S \cdot P$$

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

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

2. With Respect to Reference point.

(x_r, y_r, z_r)

- Translate ref point to the origin
- Apply scaling with respect to origin.
- Perform inverse translation to move the ref point back to the original position.

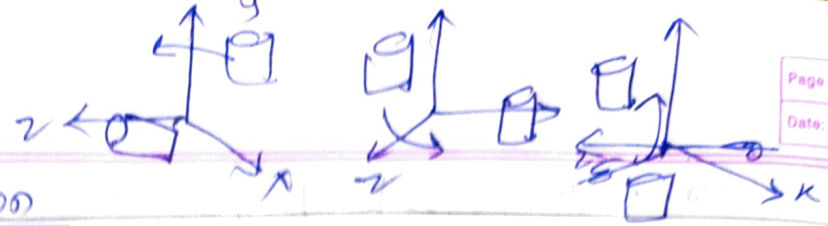
$$P' = T^{-1} \cdot S \cdot T \cdot P$$

$$P' = T(x_r, y_r, z_r) \cdot S(s_x, s_y, s_z) \cdot T(-x_r, -y_r, -z_r) \cdot P$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_r \\ 0 & 1 & 0 & y_r \\ 0 & 0 & 1 & z_r \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & -x_r \\ 0 & 1 & 0 & -y_r \\ 0 & 0 & 1 & -z_r \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & (1-s_x) \cdot x_r \\ 0 & s_y & 0 & (1-s_y) \cdot y_r \\ 0 & 0 & s_z & (1-s_z) \cdot z_r \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Rotation

- For anticlockwise rotation, angle is considered +ve
- For clockwise rotation, angle is -ve

① Rotation about x-axis

- It does not alter x-coordinate.
- It only affects y & z co-ordinates.

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

$$P' = R_x(\theta) \cdot P$$

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

② Rotation about y-axis

- It does not alter the y-co-ordinate. It only affects z & x co-ordinates.

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

$$P' = R_y(\theta) \cdot P$$

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

(3) Rotation about z-axis

- It does not alter z-coordinate. It only affects x & y co-ordinates

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

$$P' = R_z(\theta) \cdot P$$

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

Inverse Rotation Matrix

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

$\therefore \cos(-\theta) = \cos\theta$

$\sin(-\theta) = -\sin\theta$

Page No.:
Date:

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

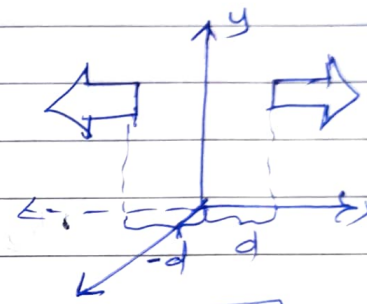
Page No.
Date

* Reflection

- It can be performed either with respect to the selected axis or with respect to the selected plane.

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

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



$$Ref_{(z=0)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P' = M \cdot P = Ref_{(x=0)} \cdot P$$

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

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

* Shearing

- Alter the shape of obj.
- It is used in 3D viewing to derive projection.

$$SH_x = \begin{bmatrix} 1 & 0 & b & 0 \\ a & 1 & 0 & 0 \\ b & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Sh_y = \begin{bmatrix} 1 & c & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & d & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Sh_z = \begin{bmatrix} 1 & 0 & e & 0 \\ 0 & 1 & f & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$SH = \begin{bmatrix} 1 & c & e & 0 \\ a & 1 & f & 0 \\ b & d & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Off diagonal values specify the shear amount in a particular direction.

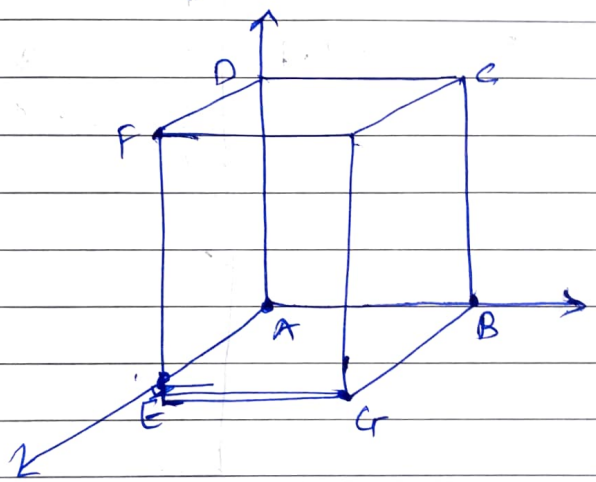
$$P' = M \cdot P = SH \cdot P$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ \end{bmatrix} = \begin{bmatrix} 1 & c & e & 1 \\ a & 1 & f & 1 \\ b & d & 1 & 1 \\ \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$P' = [(x + cy + ez) \; (ax + y + fz) \; (bx + dy + z)]$$

Q. A cube is defined by 8 vertices
A (0,0,0) B (2,0,0) C (2,2,0) E (0,0,2)
F (0,2,2) G (2,0,2) H (2,2,2) D (0,2,0)
perform 3D Transformation on the cube.

(i) Translation [5, 3, 4]
(ii) Scaling [1, 2, 0.5]
(iii) Rotation abt x-axis by 90° in clkwise direction.



$$P' = T \cdot P = \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & 2 & 0 & 0 & 2 & 2 & 0 \\ 0 & 0 & 2 & 2 & 0 & 2 & 0 & 2 \\ 0 & 0 & 0 & 0 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$P' = \begin{bmatrix} 5 & 7 & 7 & 5 & 5 & 5 & 7 & 7 \\ 3 & 3 & 5 & 5 & 3 & 5 & 3 & 5 \\ 4 & 4 & 4 & 4 & 6 & 6 & 6 & 6 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

(2) Scaling

$$P' = S \cdot P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}$$

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

(3) Rotation

$$P' = R_x(\theta = -90^\circ) \cdot P_2 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(-90) & -\sin(-90) & 0 \\ 0 & \sin(-90) & \cos(-90) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

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

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

Q. Rotate a triangle ABC with vertices A (2, 2, 2) B (3, 4, 7) & C (8, 9, 12) about y axis in anticlockwise direction by angle 90°

→

$$P' = R_y(\theta) \cdot P = \begin{bmatrix} \cos 90 & 0 & \sin 90 & 0 \\ 0 & 1 & 0 & 0 \\ -\sin 90 & 0 & \cos 90 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

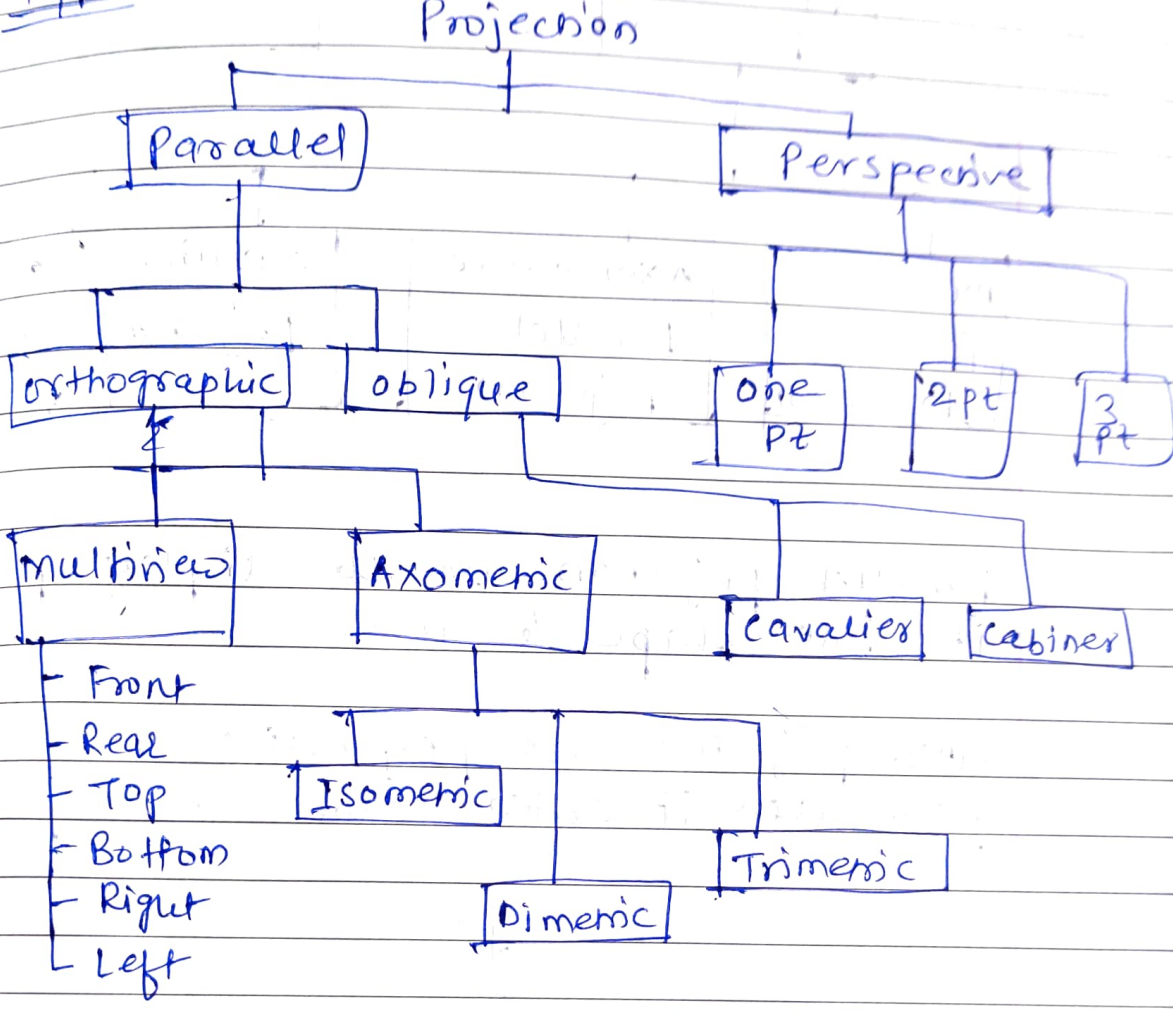
$$\begin{bmatrix} 2 & 3 & 8 \\ 2 & 4 & 9 \\ 2 & 7 & 12 \\ 1 & 1 & 1 \end{bmatrix}$$

$$P' = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 3 & 8 \\ 2 & 4 & 9 \\ 2 & 7 & 12 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 7 & 12 \\ 2 & 4 & 9 \\ -2 & -3 & -8 \\ 1 & 1 & 1 \end{bmatrix}$$

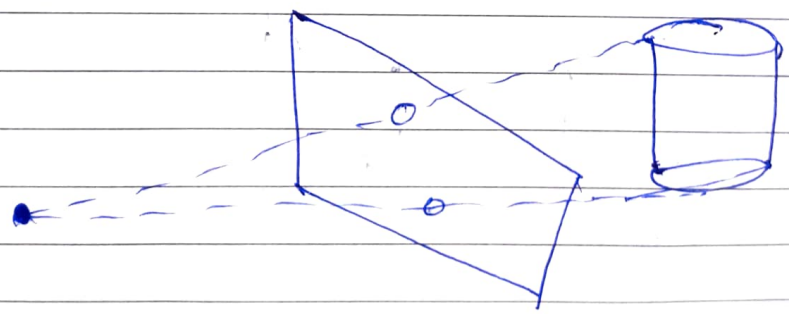
* Projection

- It is the process of converting a 3D object into 2D object.

Types

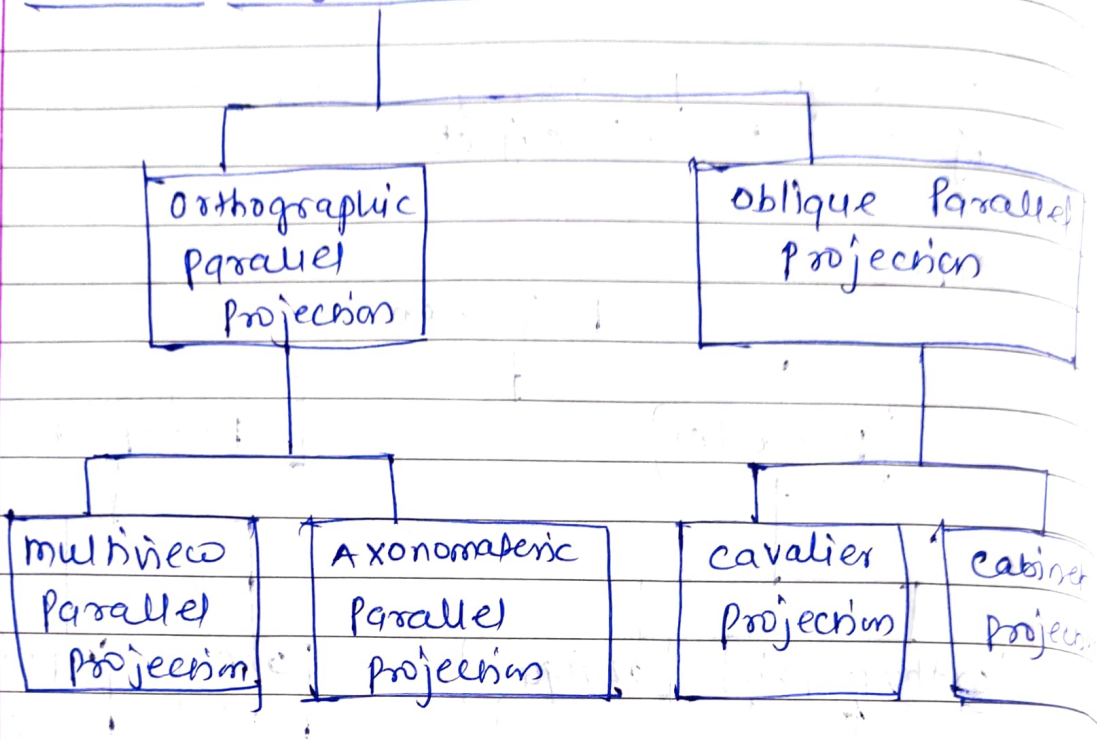


* Ray.



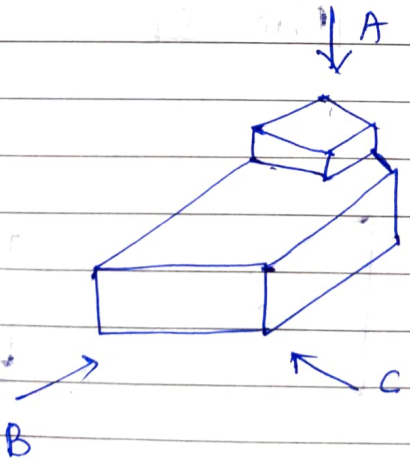
In 3D we map points from 3-space to the project" plane along projectors from center of projection (c.op)

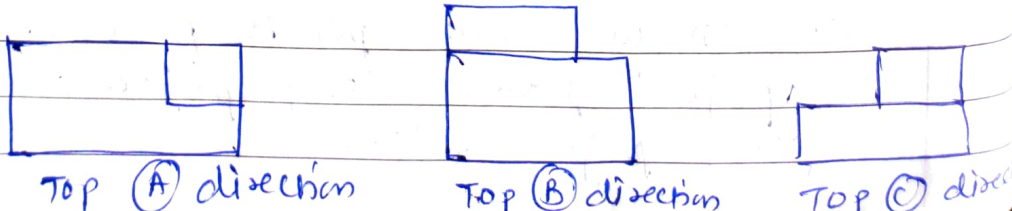
* Parallel Projection

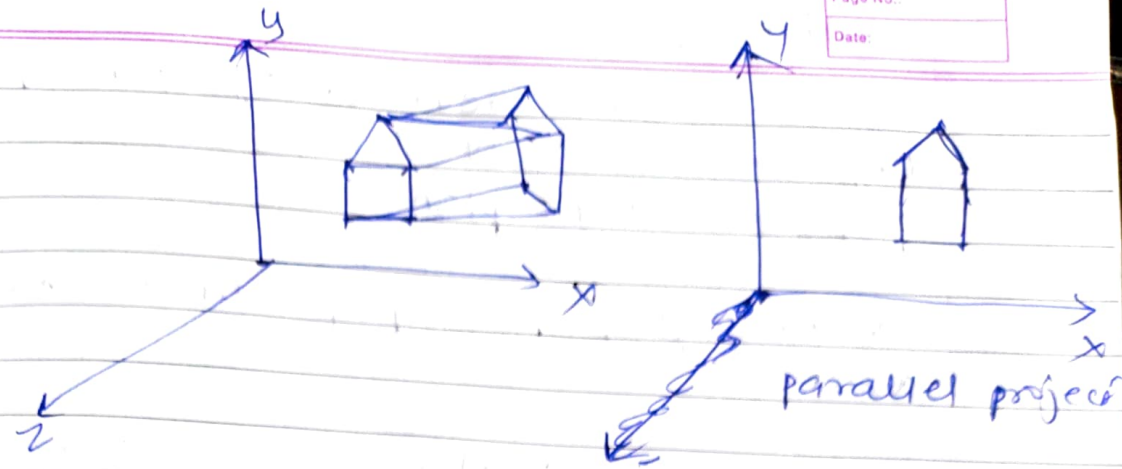


- Parallel projection use to display picture its true shape & size.

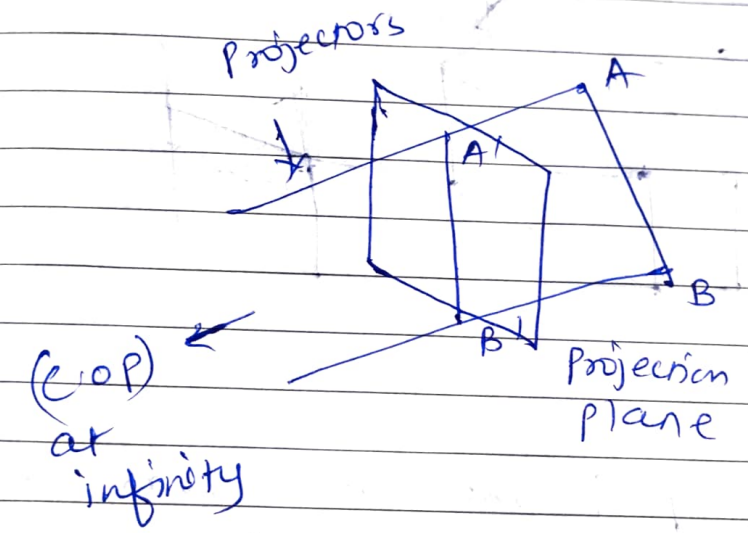
- When projectors are perpendiculars to view plane then it called orthographic projection





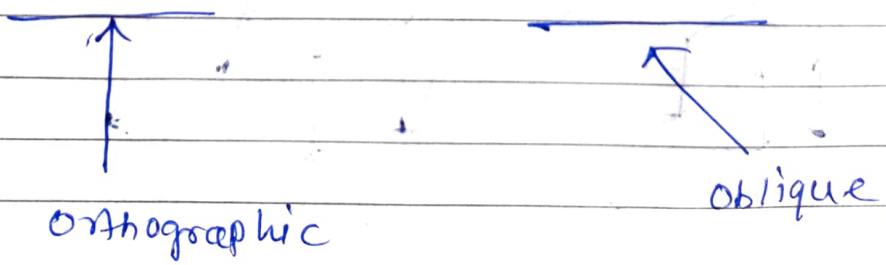


In parallel proj, coordinate positions are transformed to the view plane along parallel lines.



① Orthographic - when the projection is perpendicular to the view plane.

② Oblique - when proj is not perpendicular to the view plane.

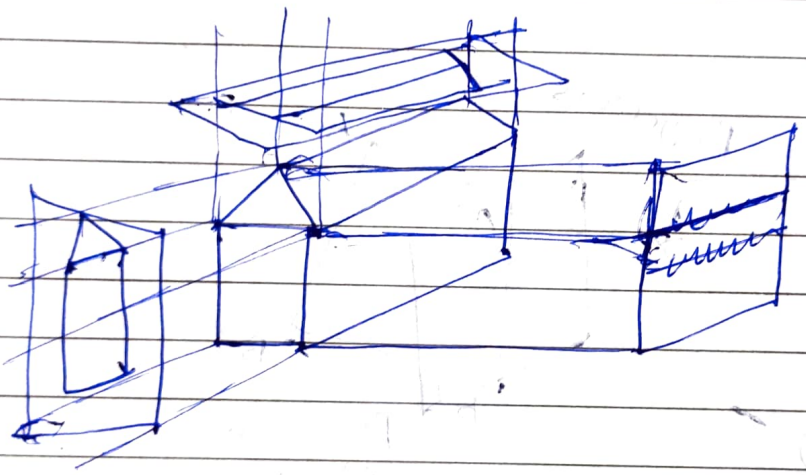


Orthographic

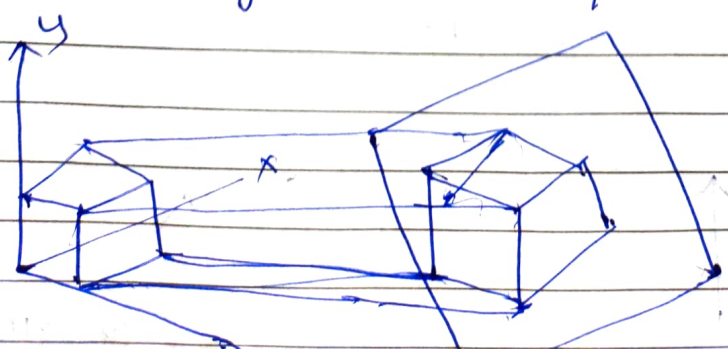
- Front, side and rear orthographic proj of obj are called as elevations & the top proj is called plan view.

- All have proj plan perpendicular to a principal axies.

- Here length & angles are accurately depicted & measured. So engg & architectural drawings its commonly use.



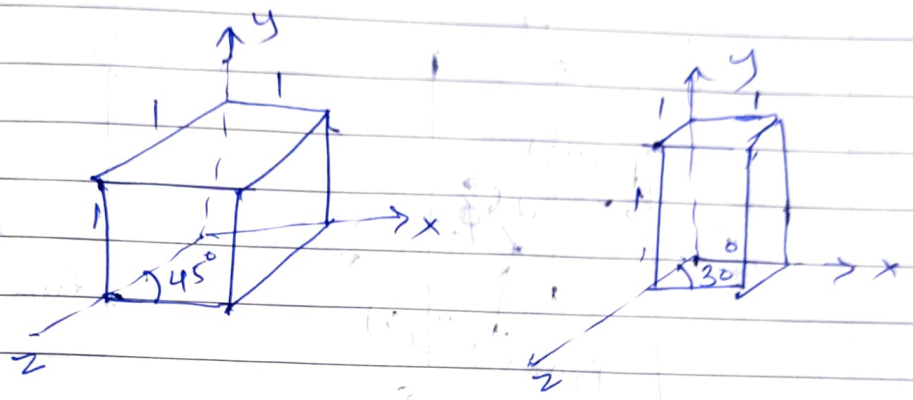
- Orthographic proj show more than one face of an obj are called axonometric orthographic projects.
most common axonometric proj is an isometric proj where the proj plane intersects each coordinate axis in the model coordinate sys at an equal distance.



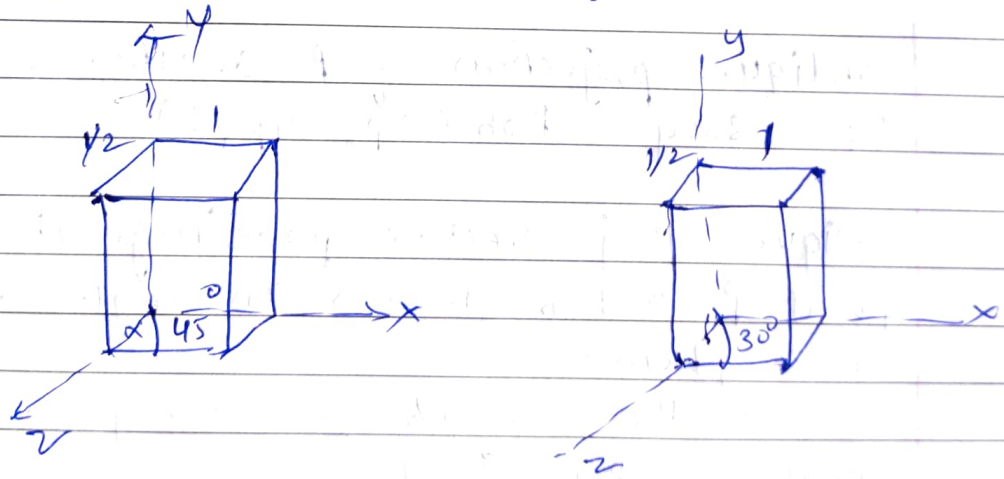
Oblique

- common oblique parallel projs - cavalier & cabinet.

- **Cavalier** -
  All lines ^not perpendicular to the projection plane are projected with no change in length.



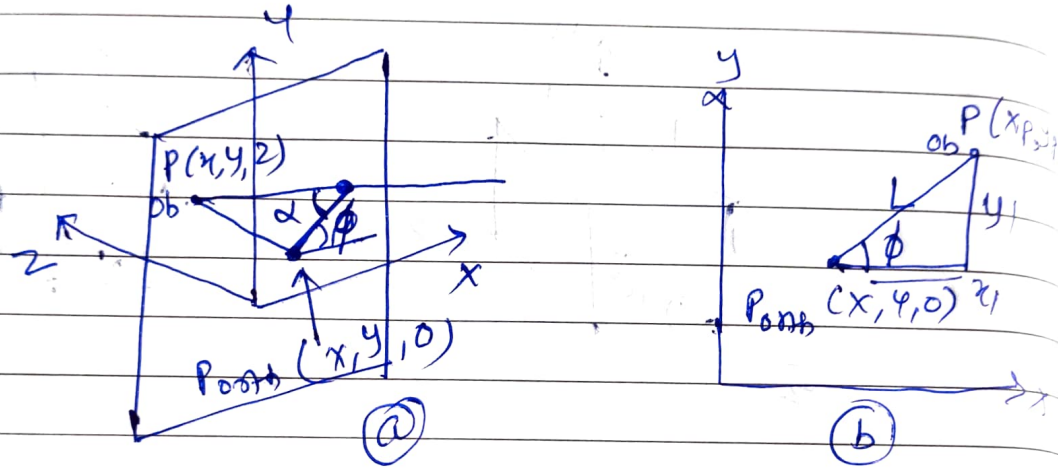
- **Cabinet** -
  Lines are ^not perpendicular to proj plane are projected 1/2 the length.



Oblique parallel projection

- Projectors are parallel to each other & they are not perpendicular to the view plane.

- view in oblique projection is controlled by two parameters ϕ & & and α.



- Let P(x, y, z) be the point in space. Orthographic projection of point P on view plane xy is P(x, y, 0)

- oblique projection of P on the same plane is say Pob(x_p, y_p, 0).

- Oblique proj vector passing through point P & Pob makes an angle α with view plane.
  Let the length of line joining points Portho & Pob is L

- $x_p = x + L\cos\phi$
  $y_p = y + L\sin\phi$

$\tan\alpha = \frac{z}{L}$ -- from fig (a)

$\therefore L = \frac{z}{\tan\alpha}$ e zL_1

$= zL_1$

by solving the eqn of x_p & y_p for L

$$x_p = x + zL_1\cos\phi$$

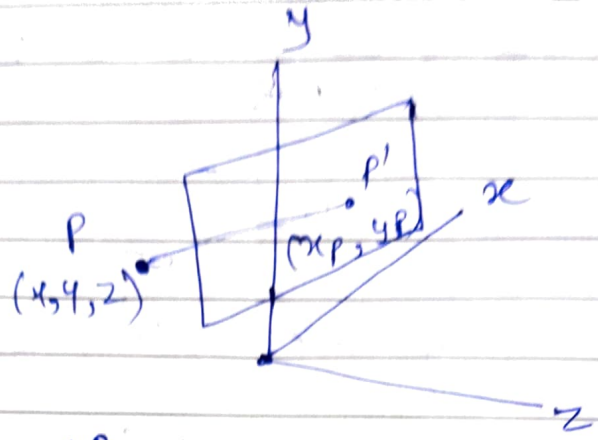
$$y_p = y + zL_1\sin\phi$$

So the transformation matrix for any parallel projection on xy view plane xv, yv is -

$$M = \begin{bmatrix} 1 & 0 & L_1\cos\alpha & 0 \\ 0 & 1 & L_1\sin\alpha & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

L_1 is foreshortening factor.

* Parallel Projection Derivation



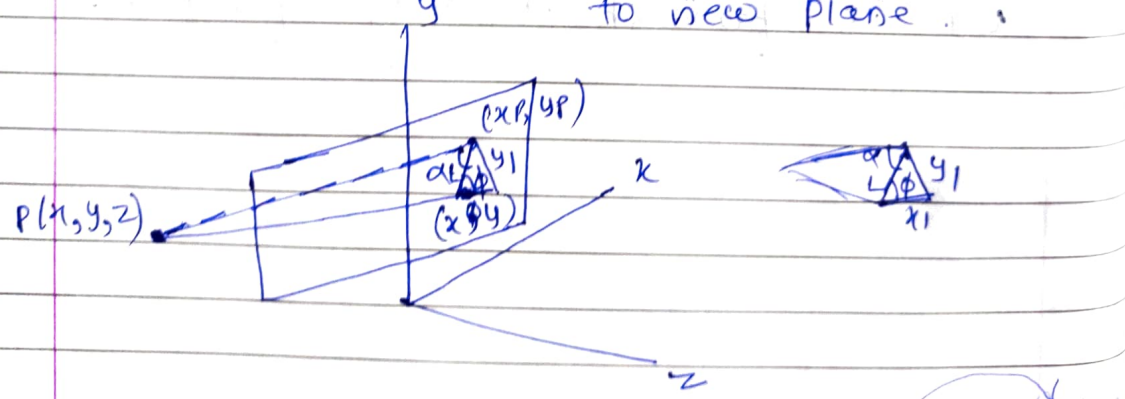
① Orthographic

$z_p = 0$
$x_p = x$
$y_p = y$

In xy plain z axis is also as zero. ∴ $z = 0$

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

② Oblique projection – Line is not perpendicular to view plane.



$x_p = x + L\cos\phi$
$y_p = y + L\sin\phi$

$\cos\phi = \frac{b}{H} = \frac{x_1}{L}$ (b = base, H = hypotenuse)

$x_1 = L\cos\phi$

Perpendicular → opp to angle.
Hypotenious

Page No.:
Date:

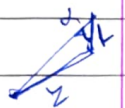
$\sin\phi = \frac{P}{H} = \frac{y_1}{L}$ $\quad \therefore y_1 = L\sin\phi$

$x_p = x + x_1$
$y_p = y + y_1$

Distance of x_1 & y_1 are equal. So x_p will be plot using $x+x_1$ & $y_p = y+y_1$

$\therefore x_p = x + L\cos\phi$ ✓
$y_p = y + L\sin\phi$ ✓

(90° angle) $\tan\alpha = \frac{P}{B} \rightarrow$ opp to angle. $= \frac{z}{L}$ $\quad \therefore L = \frac{z}{\tan\alpha}$



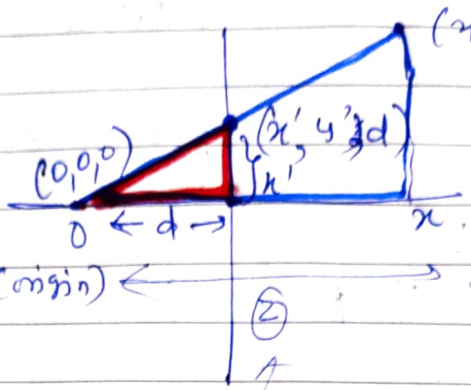
$L = zL_1$ where $L_1 = \frac{1}{\tan\alpha}$

$x_p = x + L\cos\phi$
$= x + zL_1\cos\phi$

$y_p = y + L\sin\phi$
$= y + zL_1\sin\phi$

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

Prospective Projection.



Two Triangles.

$$\Rightarrow \frac{x'}{d} = \frac{x}{z}$$

(when two triangles are similar then their ratio is equal)

$$\therefore x' = \frac{xd}{z} = \frac{x}{z/d}$$

$$\frac{y'}{d} = \frac{y}{z}$$

$$\therefore y' = \frac{yd}{z} = \frac{y}{z/d}$$

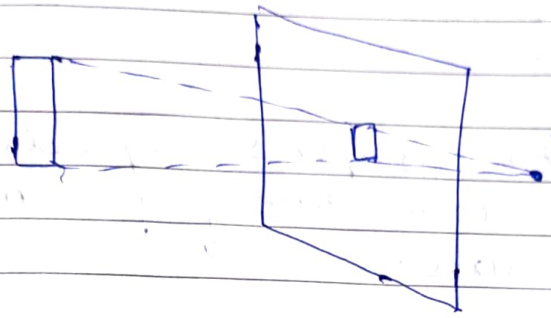
$$z' = z$$

$$x' = \frac{x}{z/d}, \quad y' = \frac{y}{z/d}, \quad z' = z$$

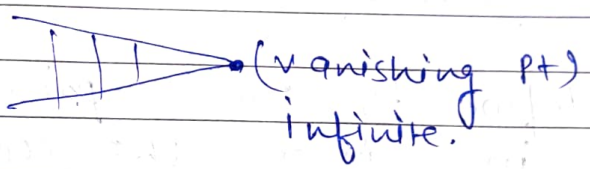
$$\begin{bmatrix} x' \\ y' \\ z' \\ z/d \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Homogeneous coordinate $1/d$.

* Perspective projection.

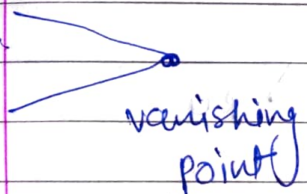


- No Relative proportion presence.

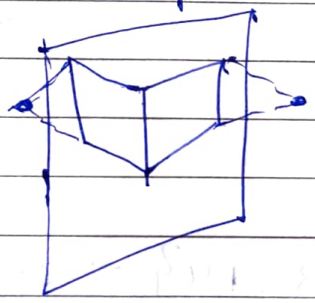


Ex:- Railway track.

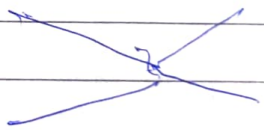
① 1 pt



② 2 pt



③ 3 pt.



* Curves & Fractals

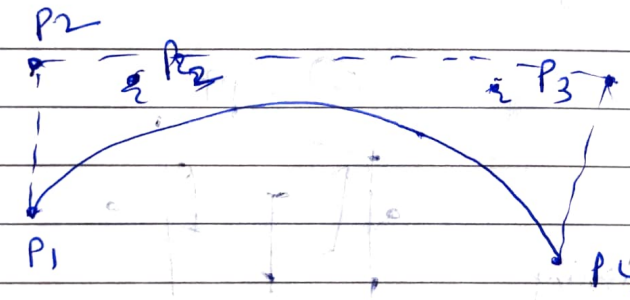
① Implict
② Explict
③ Parameric
} Types of curves

Most computer graphics depends on parameric curve. one of parameric curve is Bezier curve.

* Bezier Curve

- Is define by control points. By using this point we draw the curve.

ex:-

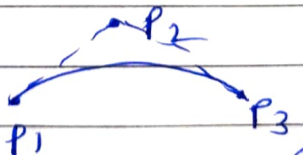


{P1, P2, P3, P4} - control points.

2 pt curve -

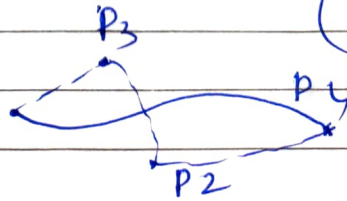
2 pt (liner curve)

3 pt curve -



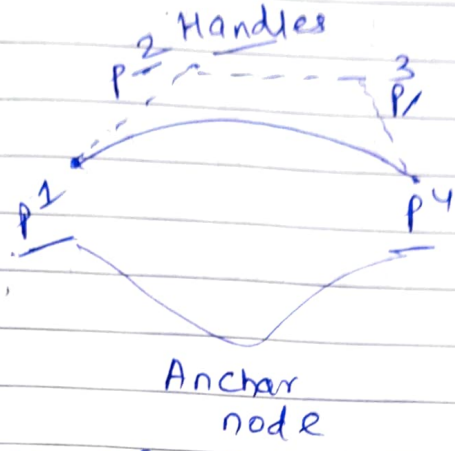
(quadratic curve
Parabolic curve
(3 control points
degree is 2
(n-1)

4 pt curve



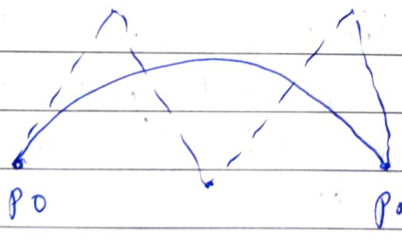
Cubic

- points are not always on curve.
- (n-1) degree on curve.



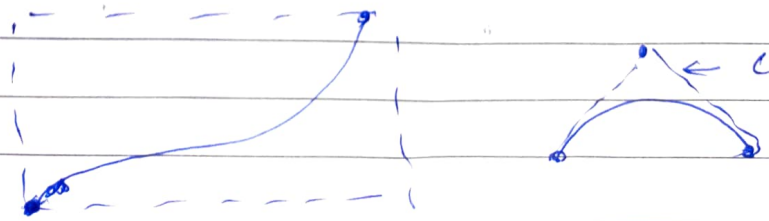
n-1
4-1 = 3
↑ degree.
(cubic curve)

* Properties of Bezier curve
- They always pass through the first & last control points.
- They are contained in the convex hull of their defining control points.
- The degree of polynomial defining the curve segments is one less than the no. of defining polygon points.

① 

It pass through first & last control points

② Convex hull - boundary outside to control point



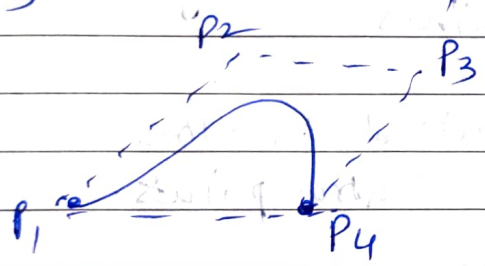
(3) If we having 4 control points
∴ degree is 3 (n-1) so it is
cubic polygon.

If degree is 1 - linear poly

" " 2 - quadratic poly

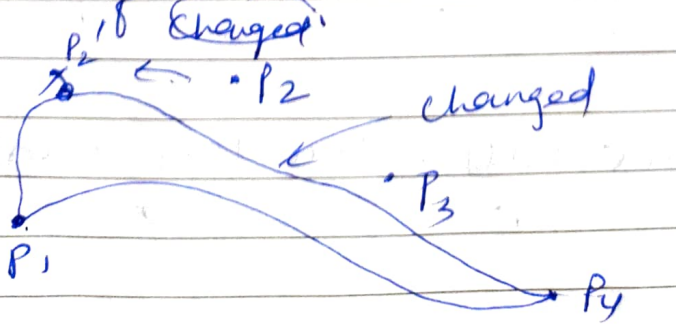
" " 3 - cubic poly

(4) The shape of defining polygon is usually followed by Bezier curve.



(5) Bezier curves are tangent to their first & last edges of control polyline

(6) Bezier curves exibit global control point means moving a control point alters the shape of the whole curve.



Page No.
Date

Derivation of Bezier curve.

- Blending or basis function.

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

$0 \le u \le 1$

u = control points. $n+1 \Rightarrow$ control point.

where

$$B_{i,n}(u) = c(n,i)\, u^i \, (1-u)^{n-i}$$

$$\Downarrow$$

$$c(n,i) = \frac{n!}{i!\,(n-i)!}$$

$$B_{i,n}(u) = \frac{n!}{i!\,(n-i)!}\, u^i \, (1-u)^{n-i}$$

Cubic Bezier – Degree is (3) $n = 3$

control pt = 4

$$P(u) = P_0 B_{0,3}(u) + P_1 B_{1,3}(u) + P_2 B_{2,3}(u) + P_3 B_{3,3}(u)$$

(P_0: control point)

- Put values.

$$\Rightarrow B_{0,3}(u) = c(3,0) \cdot u^0 (1-u)^{3-0}$$

$$= \frac{3!}{0!\,3!} (1-u)^3 \Rightarrow (1-u)^3 \text{ — (1)}$$

$$\Rightarrow B_{1,3}(u) = c(3,1) \cdot u^1 \cdot (1-u)^{3-1}$$

$3u \cdot (1-u)^2$ — (2)

$\Rightarrow B_{2,3}(u) = C(3,2) \cdot u^2 \cdot (1-u)^{3-2}$

$= 3u^2 \cdot (1-u)$ — (3)

$\Rightarrow B_{3,4}(u) = C(3,3) \cdot u^3 \cdot (1-u)^{3-3}$

$= u^3$ — (4)

so, blending fn is -
substitute values

$$P(u) = P_0 \cdot (1-u)^3 + P_1 \cdot 3u(1-u)^2 + P_2 \cdot 3u^2(1-u) + P_3 \cdot u^3$$

x - cordinates

$$P(ux) = P_{x0}(1-u)^3 + P_{x1} \cdot 3u(1-u)^2 + P_{x2} \cdot 3u^2(1-u) + P_{x3} \cdot u^3$$

y - cordinates

$$P(uy) = P_{y0}(1-u)^3 + P_{y1} \cdot 3u(1-u)^2 + P_{y2} \cdot 3u^2(1-u) + P_{y3} \cdot u^3$$

Page No.:
Date:

2 Co-ordinates.

$$P(u_2) = P_{20}(1-u)^3 + P_{21} \cdot 3u(1-u)^2 + P_{23} \cdot 3u^2(1-u) + P_{23} \cdot u^3.$$

For Three ~~control points~~ dimensional take 3 co-ordinates.

Ex:- Design a Bezier Curve controlled by 4 points A(1,1) B(2,3) C(4,3) D(6,4)

→ control point = 4

$$n = 4-1 = 3$$

A(1,1) = P_0 C(4,3) = P_2

B(2,3) P_1 D(6,4) = P_3

eqn of bezier curve.

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

$$B_{i,n}(u) = \frac{n!}{i!(n-i)!} \cdot u^i (1-u)^{n-i}$$

n=3

$$P(u) = P_0 B_{0,3}(u) + P_1 B_{1,3}(u) + P_2 \cdot B_{2,3}(u) + P_3 B_{3,3}(u)$$

$= P_0(1-u)^3 + P_1 \cdot 3u(1-u)^2 + P_2 3u^2(1-u) + P_3 u^3$

To find $x(u)$ & $y(u)$

$P_0(1,1)$
$P_1(2,3)$
$P_2(4,3)$
$P_3(6,4)$

$x(u) = 1 \cdot (1-u)^3 + 2 \times 3u(1-u)^2 + 4 \times 3u^2(1-u) + 6 \cdot u^3$

$y(u) = (1-u)^3 + 3u(1-u)^2 + 3u^2(1-u) + 4u^3$

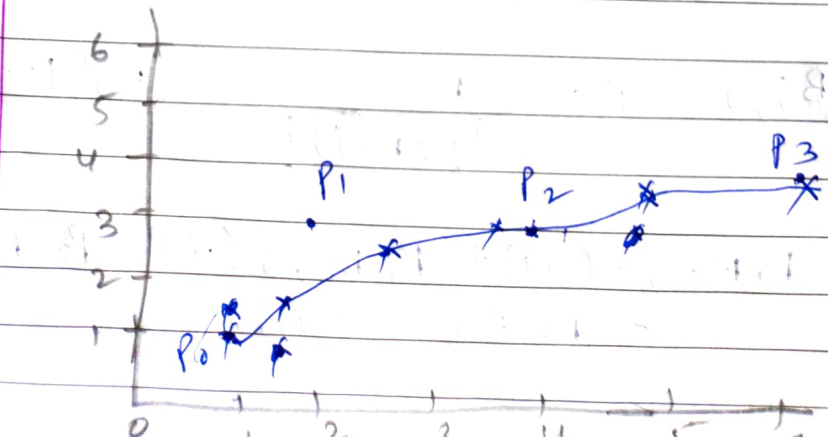
$0 \le u \le 1$

0 to 1

u take 0.2 interval

u	$x(u)$	$y(u)$
0	1	1
0.2	1.71	1.98
0.4	2.616	2.63
0.6	3.66	3.088
0.8	4.80	3.49
1	6	4

Substitute in $x(u)$ & $y(u)$

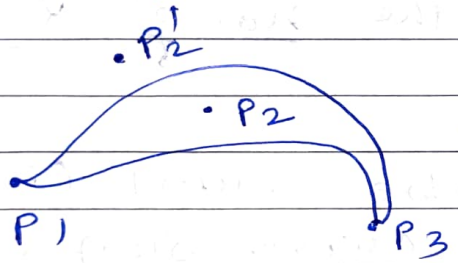


* B-Spline Curve.

- Bezier curve (global control) — Depend on control points
- B-Spline curve (local control)

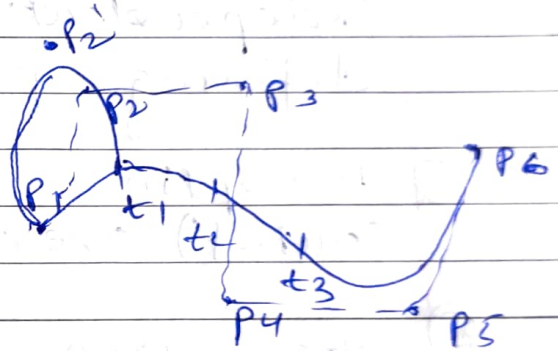
Degree is depends on control points in Bezier curve.
Where in B-spline curve it does not depends on control points. It depends on order of polynomial. i.e. based on the no. of control points if order varies then blending fn degree will be diff.

eg:- Bezier (global)



If P2 change shape of curve will change

B-Spline (Local)

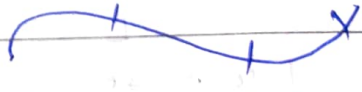


- only some part of portion will change.

Properties of B-Spline curve

(1) The sum of B-Spline basis function at any parameter 'u' equal to 1 i.e.

$$\sum_{i=1}^{n+1} Ni,k(u) = 1$$



equal segments.

n+1 = no. of control point
K = order of B-Spline curve.

(2) The basis fn is +ve or zero for all parameter values i.e. $N_{i,K}(u) \geq 0$ Except for K=1 each basis fn has one max value.

(3) The maximum order of the curve is equal to the no. of vertices of defining polygon.

(4) The degree of B-Spline polynomial is independent on the no. of vertices defining polygon.

(5) B-Spline allow the control (i.e. local control) over the curve surface.

(6) The curve lies within the convex hull of its defining polygon.

(7) The curve generally follow the shape of defining polygon.

* It is Represented as - Blending fn.

$$P(u) = \sum_{i=1}^{n+1} P_i N_{i,K}(u)$$

where $u_{min} \leq u \leq u_{max}$

~~$2 \leq k \leq n+1$~~

$2 \leq k \leq n+1$

↳ Two partition.

$N_{i,k}(u) = (u - x_i) \cdot N_{i,k-1}(u) + (x_{i+k} - u) \cdot N_{i+1,k-1}(u)$

$x_{i+k-1} - x_i$	$x_{i+k} - x_{i+1}$

$$N_{i,k}(u) = \begin{cases} 1 & x_i \leq u \leq x_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$x_i = 0$ if $i < k$

$x_i = i - k + 1$ if $k \leq i \leq n$

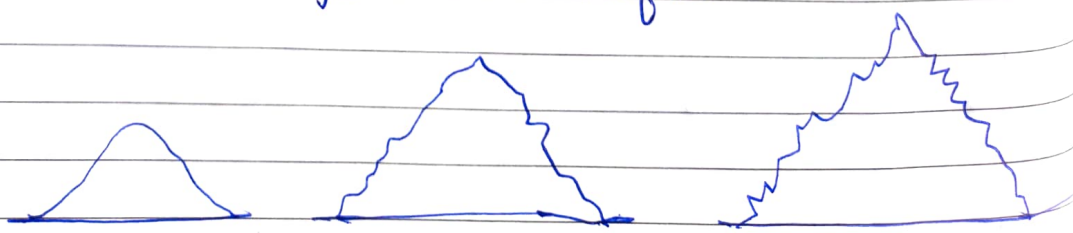
$x_i = n - k + 2$ if $i > n$.

* Fractals

- Shapes like line, circle, ellips, curves defined using Euclidean geometry or mathematical eqn.
- Natural objs like plants, tree, mountains, waves etc cannot described by such mathematical notion.
- Natural shapes have many irregularities & self-similarities.
- Shapes generated using euclidean geometry have a smooth surface. but these are not suitable for generating a realistic display of the natural scene.
- Natural objs are well described using fractal geometry methods.

* Properties of Fractal

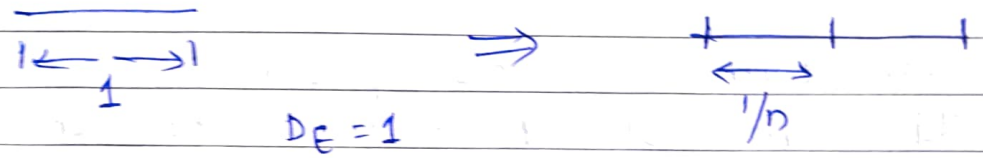
1. Infinite detail at every point.
2. Self-similarity betn obj parts.
3. Natural objs have infinite details.



Distant mountain closer view Even more closer

Fractal Dimension

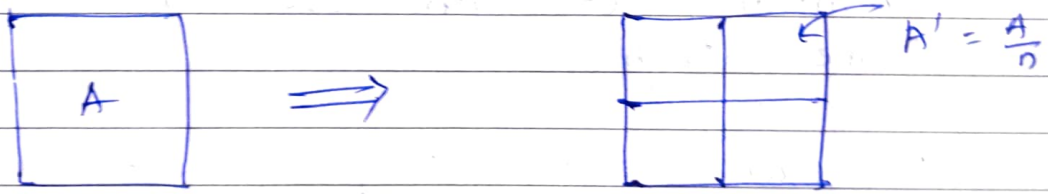
- Amount of variation in obj detail is described by a number called fractal dimension
- D can be measure of the roughness of the obj
- obj with jaggy boundary have larger D value.
- Sef-Semilar fractals are obtained by recursively applying the same scaling factor to Euclidd shapes.



$D_E = 1$

$S = 1/n, \quad n = 2$

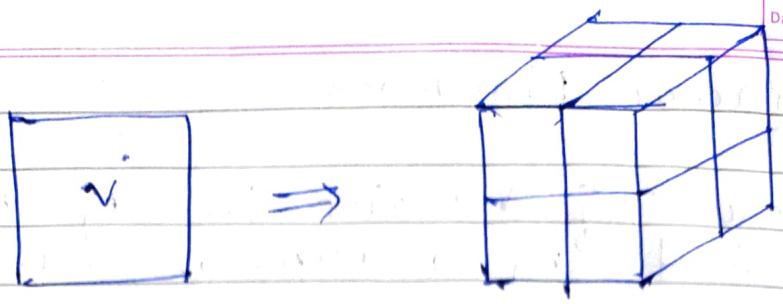
$$ns^1 = 1$$



$D_E = 2$

$S = \frac{1}{\sqrt{D}}, \quad n = 4$

$$ns^2 = 1$$

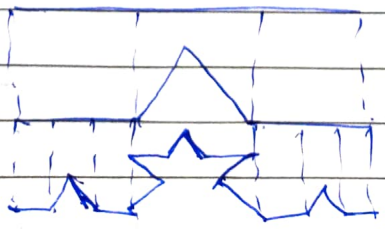


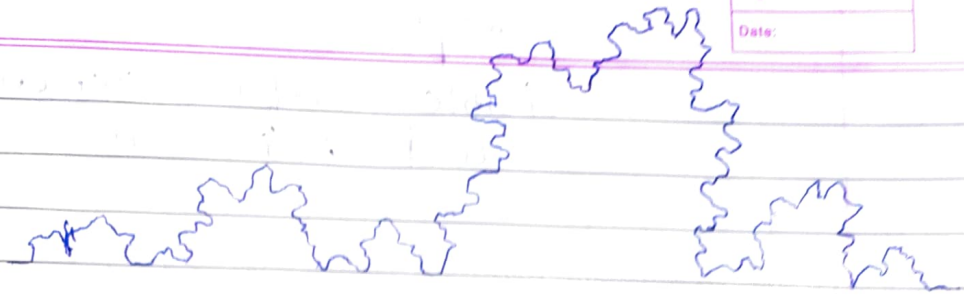
$D_E = 3$, $s = \frac{1}{3\sqrt{n}}$, $n = 8$

$$n s^3 = 1$$

* Koch curve.

- Iterative construction is used to create this fractal shape.
- Start with a section of straight line.
- This line should be divided into 3 equal parts.
- Without the base, an equilateral triangle is formed by removing the middle section & substituting it with two segments of the same length.
- For each segment, keep going through these stages indefinitely.



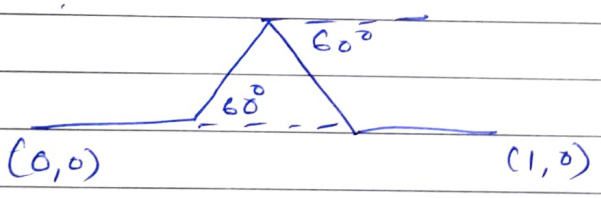


$$\sum_{K=1}^{4} r^d = 1 \Rightarrow d = \frac{\log(1/4)}{\log(1/3)} = \frac{\log(4)}{\log(3)}$$

$$= 1.2618$$

The koch curve is self similar with 4 non-overlapping copies of itself, each scaled by the factor $r < 1$. Therefore the similarity dimension, d, of the attractor of the IFS is the solution.

$r = 1/3$



It scaled by $r = 1/3$. Two segments must be rotated by 60°, one counterclkwise & one clockwise.

Chapter 6

Visible Surface Detection & Animation

Visible surface detection :

- classification of visible surface detection algo
- Back surface detection method
- Depth Buffer mtd,
- Area sub division Mtd.

Animation : Introduction to Ani

- Traditional Ani Tech.
- Principles of Ani
- Key framing
- Character & facial Ani
- Deformation
- Motion Capture.

1. Explain Z Buffer algo for hidden surface removal.
2. Explain the concept of halftoning with example.
3. Short note on i) Sweep representation & octree
   Gouraud & Phong shading
   Dithering.
4. What is shading? Explain Gouraud & Phong shading with their pros & cons.
5. Write short on i) Halftone & Dithering ii) Area Subdivision

Classification of Visible Surface Detection Algorithms

This algos are classified a/c to whether they deal with object definitions directly or with their projected images.

These 2 approaches are called object space methods and image space methods resp.

An object-space method compares objects & parts of object to each other within the scene to determine which surfaces, as a whole are visible.

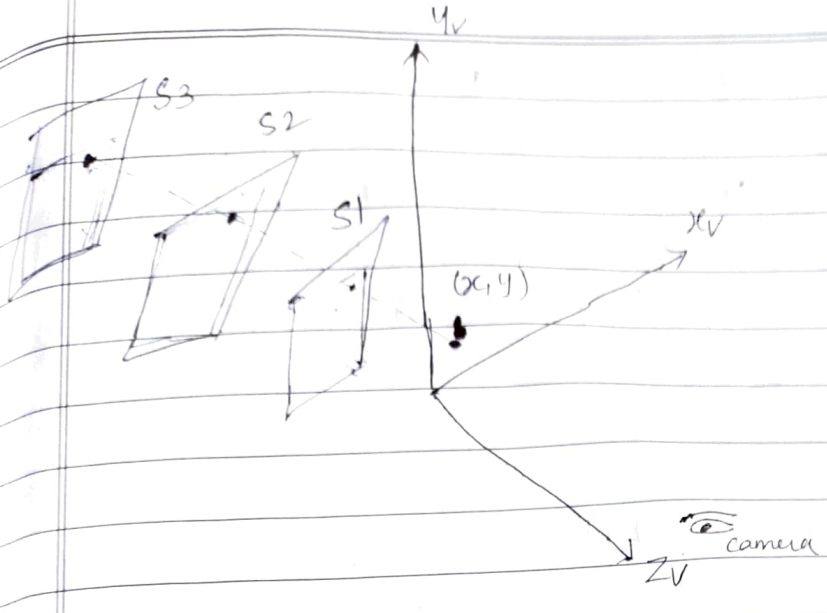
In an image-space algorithm, visibility is decided point by point at each pixel position on projection plane.

Depth-Buffer Method

This uses image space approach to detect visible surfaces, which compares surface depths at each pixel position on the projection plane. This is also referred as z-buffer method. Since object depth is usually measured from view plane along z-axis of the viewing system.

Each surface of a scene is processed separately, one point at a time across the surface.

When object descriptions are converted to projection coordinates, each (x, y) position on a polygon surface corresponds to orthographic project point (x, y) on view plane. Therefore for each pixel (x, y) on view plane, object depths can be compared by comparing z-values.



In the above diagram 3 surfaces are taken at different positions along orthographic projection line from (x, y) position in view plane.

So here from where you are observing the object is important. If u are at Z axis then S1 is closest at the position

But if eye position is along -Z axis then S_3 is the closest & accordingly save the intensity value (x, y).

In this method, 2 buffers are required. A depth buffer is used to store depth values for each (x, y).

Depth is nothing but distance bet^n object & the eye position (or camera).

Refresh buffer stores the intensity values for each position

Initially, all depth of all positions are set to 0, & refresh buffer is initialized to background intensity

Each surface in the list of polygon tables is then processed, one scan line at a time, calculating the depth (z) at each (x, y).

The calculated depth is compared to the

stored in depth buffer at that position.

If the calculated depth is greater than the value stored in depth buffer, the new depth value is stored & intensity of that pixel position i.e (x,y) is placed at xy location in refresh buffer.

Algo

1) Initialize the depth buffer & refresh buffer so that for all buffer positions (x,y)

depth(x,y) = 0 refresh(x,y) = $I_{bkground}$

2) For each position on each polygon surface compare depth value with the stored values in the depth buffer to determine visibility

- Calculate depth z for each (x,y)

if z > depth(x,y) then

depth(x,y) = z

refresh(x,y) = $I_{suface}(x,y)$

where $I_{bkground}$ – background intensity

$I_{suface}(x,y)$ is projected intensity value for pixel position (x,y).

After all surfaces have been processed, the depth buffer contains depth values for the visible surfaces & refresh buffer contains corresponding intensity values for those surfaces.

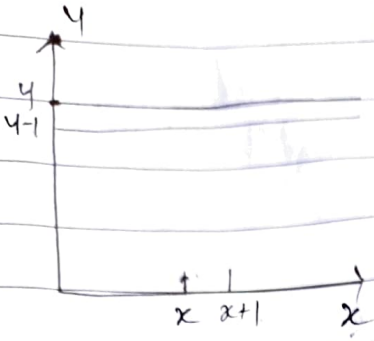
A polygon surface with plane parameters A, B, C & D is

$$Ax + By + Cz + d < 0 \quad (1)$$

Depth values for surface position (x,y) are calculated from plane eqn for each surface

$z =$

$$Z = \frac{-Ax - By - D}{C} \quad \text{---} \cdot (2)$$



For each scan line adjacent horizontal & vertical positions across the line differ by 1

If the depth is Z then z' is the depth of next position ~~(x, y)~~ (x+1, y) along scan line is

$$Z' = \frac{-A(x+1) - By - D}{C}$$

$$\therefore Z' = Z - \frac{A}{C}$$

Here $-A/C$ is constant for each surface. So next depth values are obtained from ~~previous~~ preceding values with single addition

Advantage

It is very easy to implement & it doesnot requires sorting of the surfaces.

Disadv

It requires 2 buffers.

If system has resolution (1024 x 1024) then it require a million positions in depth buffer. So large amount of memory is required. This algo works only for opaque objects & cannot accumulate intensity values for more than one surface.

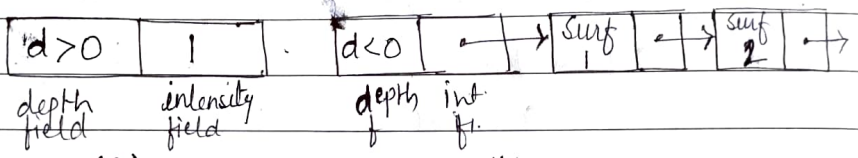
A- Buffer Method.

To overcome the drawbacks of Z- buffer. A- Buffer mtd is used.

In this mtd, it expands the depth buffer so that each position in the buffer can reference a linked list of surfaces. So more than one surface intensity can be taken into consideration at each pixel position.

A - buffer mtd represents an antialiased, area averaged, accumulation buffer mtd developed by Lucasfilm in surface rendering systems called REYES (Render Everything YOU EVER Saw)

Each position in A - buffer has 2 fields
depth field - stores positive or -ve real nos.
intensity field - stores surface intensity information



(a) (b)

Single surface Overlap of corresp. pixel area — Multiple surface overlap.

If the depth field is +ve, the no. stored at that position is the depth of single surface overlapping the corresponding pixel area. The intensity field stores RGB components of the surface color at that pt. & percent of pixel coverage.

If the depth field -ve, this means multiple surface contributions to pixel intensity. The intensity field stores a pointer to a linked list of surface data. Data for each surface in linked list includes.

a) RGB intensity components b) opacity parameter (% of transparency). c) depth d) percent of area coverage e) surface identifier f) other surface rendering parameters g) pt to next surface.

Scan lines are processed to determine surface overlaps of pixels across individual scanlines.

Surfaces are subdivided into polygon mesh & clipped against pixel boundaries.

Using opacity & % of surface overlap, we can calculate intensity of each pixel as an avg. of the contributions from overlapping surfaces.

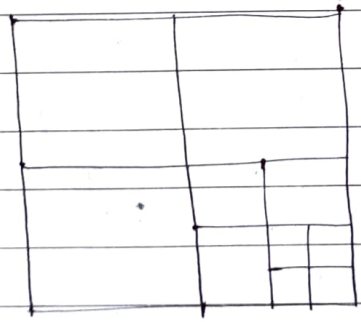
Area-Subdivision Method

This is a hidden surface removal technique which uses image space method but object space operations can be used to find depth ordering of surfaces.

The area-subdivision method takes advantage of area coherence in a scene by locating those view areas that represent part of a single surface.

We apply this method by successively dividing the total viewing area into smaller & smaller rectangles until each small area is the projection of part of a single visible surface or no surface at all.

To implement this method, first step is to quickly identify the area as a part of a single surface or find out whether the area is too complex to analyze. If the tests indicate that the view is sufficiently complex then subdivide it. Next, apply the test to each of the smaller areas, subdivide these if the tests indicate that the visibility of single surface is uncertain. Continue this until the subdivisions are easily analyzed as belonging to a single surface or until they are reduced to the size of single pixel. Easiest way to do this is to successively divide the area into 4 equal parts at each step.



Tests to determine the visibility of a single surface within a specified area are made by comparing surfaces to the boundary of a area. There are 4 possible rel^{n}ships that a surface can have with a specified area boundary.

1) Surrounding surface - One that completely encloses the area
2) Overlapping surface - One that is partly inside & partly outside the area.

3) Inside surface - one that is completely inside the area
4) Outside surface - One that is " " outside "

No further subdivisions of a specified area are needed if one of the following cond^n is true

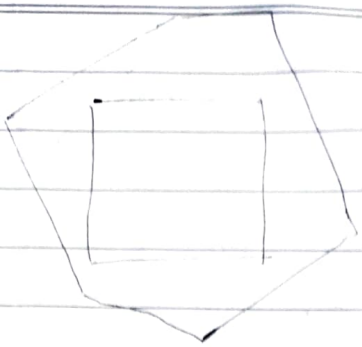
1) All surfaces are outside surfaces with respect to the area
2) Only one inside, overlapping or surrounding surface is in the area.
3) A surrounding surface obscures all other surfaces within the area boundaries.

Test1 can be carried out by checking the bounding rectangles of all surfaces against the area boundaries.

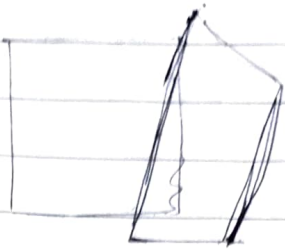
Test2 can also use bounding rectangles in xy plane to identify an inside ~~check~~ surface.

For other types of surfaces, bounding rectangles can be used as an initial check. If a single bounding rectangle intersects the area then additional tests are required to determine whether the surface is surrounding, overlapping or outside. Once a single inside, overlapping or surrounding surface has been identified, its pixel intensities are transferred to the appropriate area within frame buffer.

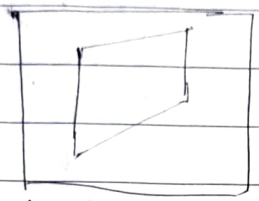
One method for implementing test 3 is to order surfaces acc to their minimum depth from view plan. For each surround surface, we then compute maximum depth within the area under consideration. If the maximum depth of one of these surrounding surfaces is closer to view plane then the minimum depth of all other surfaces within the area, test 3 is satified.



Surrounding surface



Overlapping surface



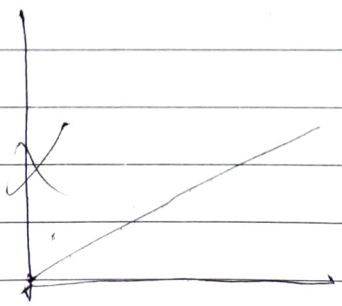
inside surface

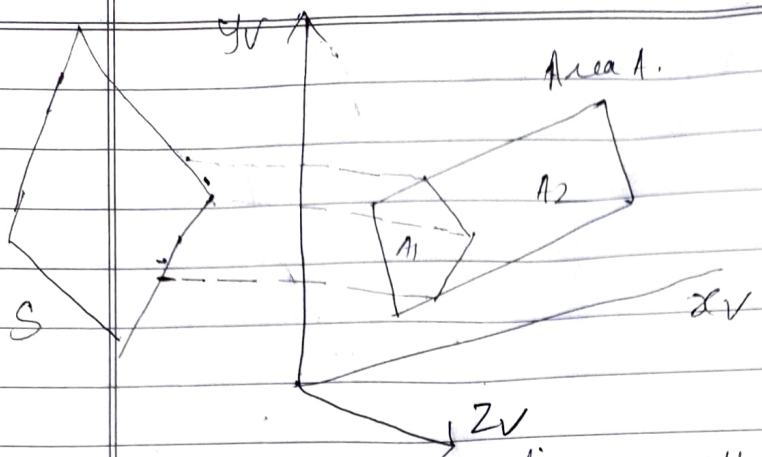
outside surface

Rela^n ship bet^n polygon surfaces & a rectangular area.

Another ~~test~~ method for test 3 ~~is to sort~~ is to use plane equations to calculate depth values at 4 vertices of the area for all surrounding, overlapping & inside surfaces. If the calculated depth for one of the surrounding surfaces is less than the calculated depth for all other surfaces, test 3 is true. Then the area can be filled with intensity values of the surrounding surface.

As a variation on basic subdivision process, we could subdivide areas along surface boundaries instead of dividing them in half. If the surfaces are sorted according to minimum depth, we can use surface with smallest depth value to subdivide a given area.





In the above diagram, the projections of boundary surface S is used to partition ~~the~~ the original area into the subdivisions A1 & A2. Surface S is then surrounding surface for area A1 & visibility tests 2 & 3 can be applied to determine whether further subdivision is necessary. So in this method, fewer subdivisions are required but more processing is needed to subdivide areas and to analyze the relation of surfaces to subdivision boundaries.

Back-Face-Detection

A fast & simple object space method for identifying the back faces of polyhedron is based on inside outside tests.

A point (x, y, z) is inside polygon surface with plane parameters A, B, C & D if

$$Ax + By + Cz + D < 0 \quad - (1)$$

when an inside pt is along the line of sight to the surface, the polygon must be a back-face.

To test this, consider a normal vector N to a polygon surface, which has cartesian components (A, B, C).

In general, if V is a vector in the viewing direction from the eye, then this polygon is a back face if

$$V \cdot N > 0 \quad - (2)$$

If the object descriptions are converted to projection coordinates & our viewing direction is ||el to Z_v then $V = (0, 0, \pm V_z)$

& $V \cdot N = V_z C$

Here confirm the sign of C.

In a right handed viewing system with viewing direction along $-ve\ Z_v$ axis., the polygon is a back face if $C < 0$

Thus in general, we can label any polygon as a back face if its normal vector has z-component value:

$$C \leq 0 \quad - (3)$$

In left handed viewing system,

In Hidden surface, we assume specific viewing direction.
bcoz surface hidden from a particular position may not be so if we change the viewing posi.

Right handed coord. system with viewer looking at the scene along -Z in scene.
Objects which represent with polygonal surfaces.

HSR has 2 types
1) Object space
↓
Compare objects or parts of objects with each other.
For
Advantage is
device independent & work for any resolution
Disadv.
Computation intensive
Sometimes if scene is complex then - is infeasible
Suitable for simple scenes & having small no. of obj.

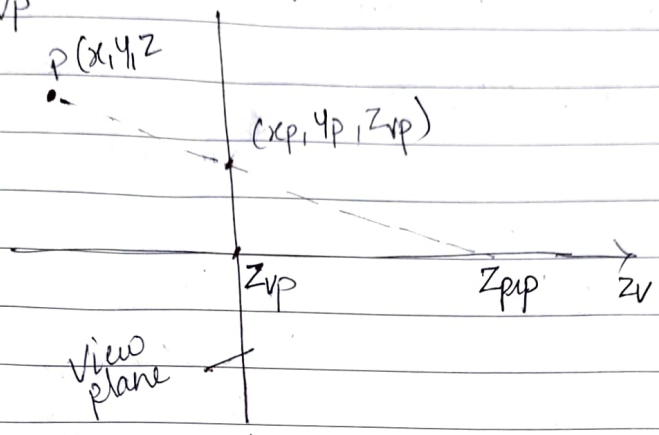
2) Image space
↓
Visibility decided pt-by-pt at each pixel
In this, determine objects closest to viewer & then draw the pixel with object color.
Computations are less
change in resolution requires re-computation of pixels.

Coherence - one way to reduce computation by exploit local similarities - making use of calculated result for nearby objects.

Perspective Projections.

To obtain this of a 3D object, we transform point along projection lines that meet at the projection reference point.

Suppose the projection reference point at position z_{prp} along z_v axis & view plane is z_{vp}



perspective projection of a pt $P(x,y,z)$ to position (x_p, y_p, z_{vp}) on view plane.

Eqns describing coordinate positions along this prospective projection line in parametric form as

$$\left.\begin{aligned} x' &= x - xu \\ y' &= y - yu \\ z' &= z - (z - z_{prp})u \end{aligned}\right\} \quad (1)$$

u takes values from 0 to 1

coordinate position (x', y', z') represents any pt along projection line.

when $u = 0$, we are at position $P = (x,y,z)$

At the end of line, $u = 1$

projection reference pt coordinates are $(0, 0, z_{prp})$

On view plane

$$z' = z_{vp}$$

$$u = \frac{z_{vp} - z}{z_{prp} - z} \quad - (2)$$

Sustitute z into eqns for x' & y'
we obtain perspective transformation eqns

$$x_p = x\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right) = x\left(\frac{d_p}{z_{prp} - z}\right)$$

$$y_p = y\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right) = y\left(\frac{d_p}{z_{prp} - z}\right) \quad -(3)$$

where $d_p = z_{prp} - z_{vp}$ is the distance of view plane from projection reference point.

3D homogeneous coordinate representation is

$$\begin{bmatrix} x_h \\ y_h \\ z_h \\ h \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -z_{vp}/d_p & z_{vp}(z_{prp}/d_p) \\ 0 & 0 & -1/d_p & z_{prp}/d_p \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad -(4)$$

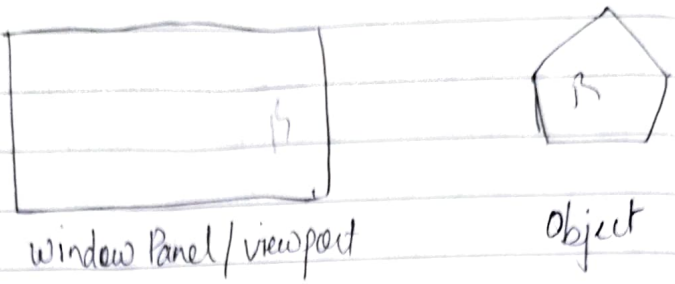
homogenous factor is

$$h = \frac{z_{prp} - z}{d_p} \quad -(5)$$

projection coordinates on view plane are

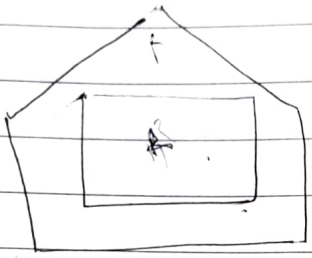
$$x_p = x_h/h \quad \& \quad y_p = y_h/h \quad -(6)$$

When we subdivide the panel aganist the polygon we may come through the following cases which are as follows :

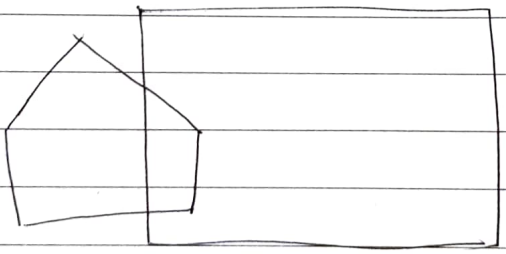


Window Panel / viewport Object

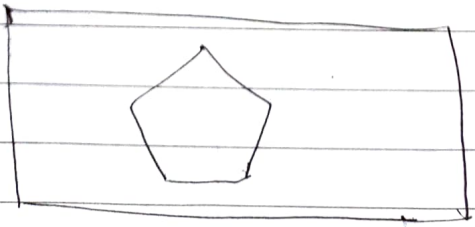
1) Surrounding surface
It's the case in which the viewing polygon surface completely surrounds the whole window panel



2) Overlapping surface -
It's the case in which the window panel & viewing polygon surface both intersect each other

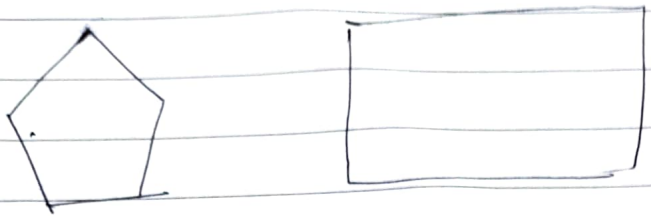


3) Inside surface -
In this case, in which polygon surface is inscribed inside the window panel.



4) Outside surface -

Here the entire surface is completely outside the window panel



Algorithm

1. Initialize the viewing area or window panel
2. Enlist all polygon & sort them a/c to Zmin (depth value) w.r. to window panel
3. Categorize all polygons a/c to their corresponding cases in which they are falling.
4) Now perform visible surface detection test :

i) if a polygon is surrounded the window panel then set the viewing area color to corresponding polygon color that is stored in frame buffer.

ii) if the polygon is outside the viewpot then background color of window plane will be done & polygon will be ignored.

iii) if the polygon is inside the window panel then the color the polygon from its corresponding color & color the rest of surface with bkcolor.

iv) If poly & viewpot intersect each other then following cases are there

a) Fill the overlapped region with poly. color & is set in the frame buffer

b) If we are given more than one polygon, with overlapped surfaces w.r. to viewpot then first find Zmin (depth) in order to find surface which is closer to viewpot & will fill the overlapped region with the color of that polygon that has

5 Repeat all steps for all given polygons then exit.