Cathode Ray Tube Monitor (CRT)

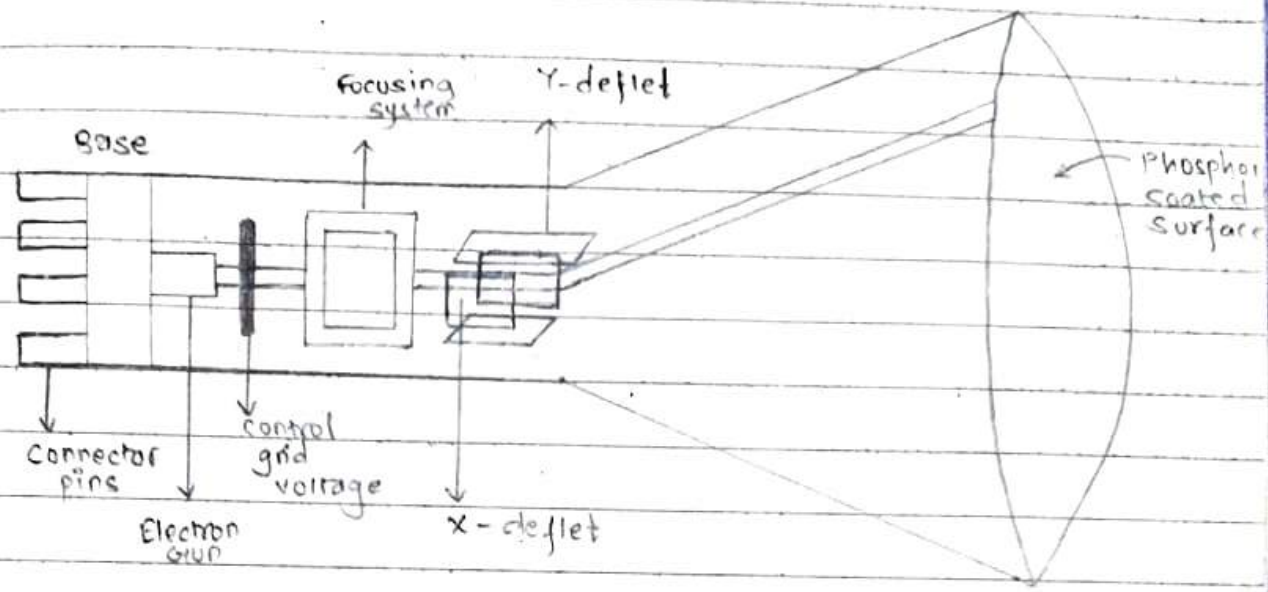


Fig: Cathode Ray Tube

- Video Display Devices
  - Cathode Ray Tube (CRT)
    - Components
      - > Electron Gun → composed of heated metal cathode and control grid.
      - > Accelerating Anode
      - > Focusing System
      - > Deflection System
      - > Phosphor Screen

* CRT operation:

> Heat generated in cathode boils off the electrons
> Electrons are accelerated towards phosphor screen with high positive voltage applied at accelerating anode.
> The negative voltage applied at cylindrical control grid controls the intensity of electron beam by repealing anode.
    - High negative voltage stops electron passing from the hole of control grid while small negative voltage decreases electron passage
> focusing system concentrates electron beam to a small spot
    - In electrostatic focusing, electrons pass through positively charged metal cylinder
    - In magnetic focusing, coils are mounted outside of CRT. Envelope which produces smallest spot.
> Deflection system deflects electron beam horizontally and vertically
    - Magnetic → two pairs of coils
    - Electrostatic → two pairs of deflection plates
> Refresh rate depends on persistance of phosphor.

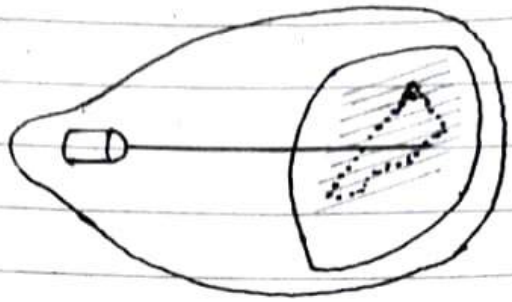
Picture Display Methodology

1) Raster Scan system: Pixel or dot based system
2) Random Scan system: Line based system

1) Raster scan Displays

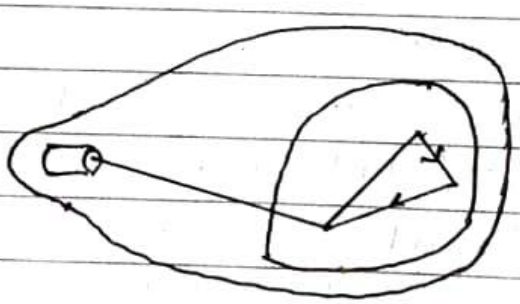
- The electron beam is swept across the screen, one row at a time from top to bottom.
- The illuminated spot pattern is created by turning on or off when electron beam moves along each scan line
- Picture definition is stored in a memory area called Refresh Buffer (Frame Buffer)
  - Refresh Buffer holds the set of intensity values for all the screen points
  - Each screen points is known as pixel (short form of picture element)
- Example: Home television sets and printers
- For bilevel system (black and white) only 1 bit memory per pixel is sufficient
- For color system more bits per pixel are needed
  - For screen with resolution 1024 by 1024, and 24 bits per pixel (8 bits each for RGB) requires 3MB of storage is needed.
- Refreshing rate for raster scan display is usually 60 to 80 frames per second (i.e. 1/80 or 1/60 seconds is taken for electron beam to scan from top left corner to bottom right corner)

→



2) Random Scan Display

- Electron beam is directed to the part of screen where picture is to be drawn.
- Picture definition is stored as set of line-drawing commands in memory are known as refresh display file or simply refresh buffer
- Also known as Vector display



Frame Buffer:

Size of frame buffer : Resolution x pixel depth
or
Refresh buffer

consider a true color CRT monitor having resolution of 640 x 480 and the refresh rate of 50Hz. Find the following

1) Size of frame buffer
2) Time required to paint one frame
3) Time required to paint one row
4) Time required to paint one pixel.

Sol^{n}

1) Frame buffer (FB) = Resolution x pixel depth.

$$= \frac{\{640 \times 480 \times 24}{8 \times 1024 \times 1024} \text{ MB}$$

2) Time required to paint one frame = $1/50$ s

3) Tim required to paint one row =

1 F = $1/50$ s

480 Row = $1/50$ s

$= 1/50 \times 480$

4) Time required to paint one pixel:

$$640 \text{ pixels} = \frac{1}{50 \times 480}$$

$$1 \text{ pixels} = \frac{1}{50 \times 480 \times 640}$$

* Color CRT Monitors

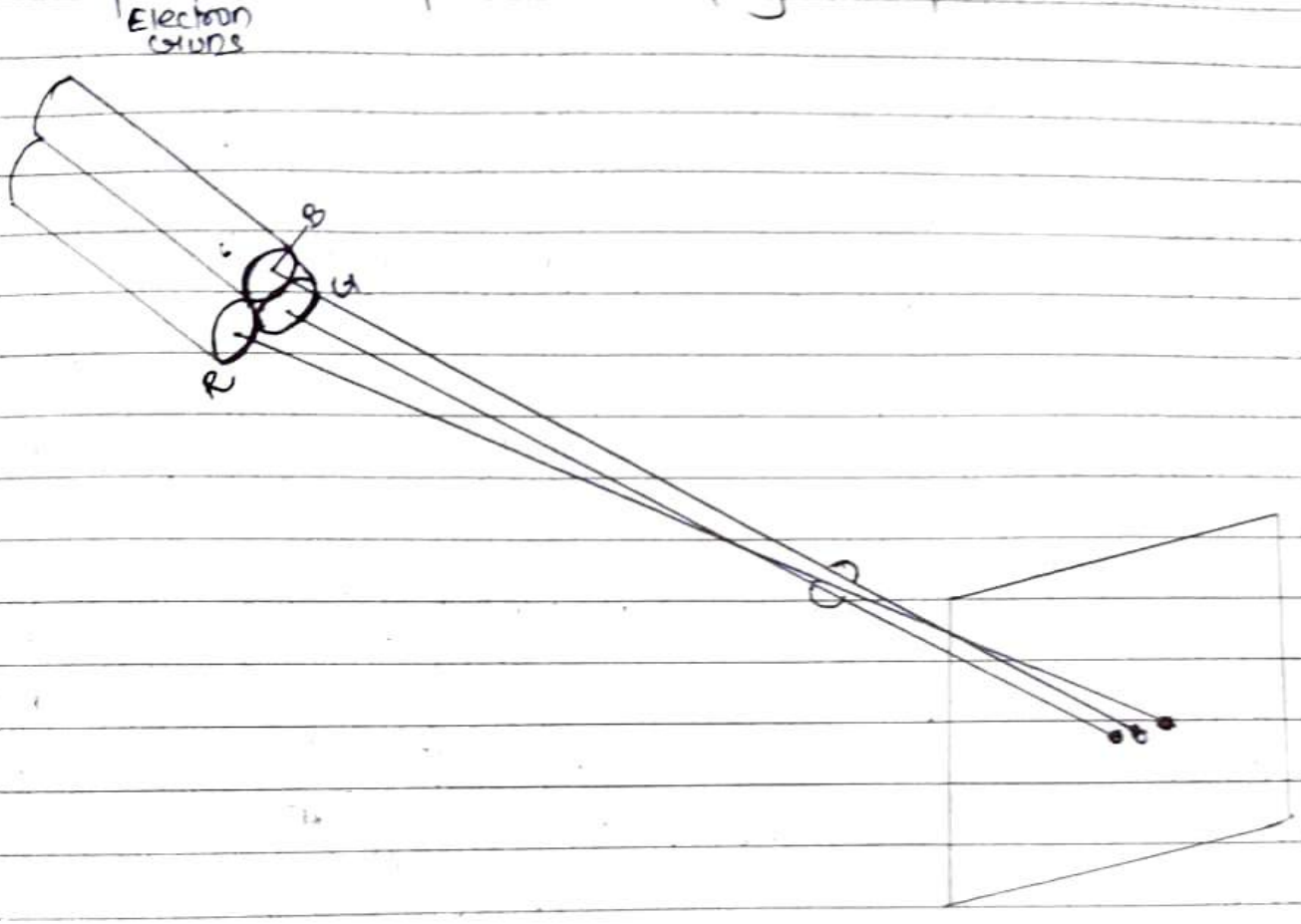
- Uses combination of phosphors that emit different colored light (usually Red, Green and Blue)

- Two basic method for color picture display

1) Beam penetration method (Random Scan):
   - Two layers of phosphor outer red layer and inner green layer
   - Slow electron strikes outer to produce red and faster strikes inner layer to produce green color while intermediate produces orange and yellow
   - Only four colors are possible

2) Shadow mask method (Raster Scan):
   - Has three phosphor color dots (RGB) for each pixel position
   - Three electron guns one for each color dot
   - A shadow mask grid with holes aligned with the phosphor dot patterns

- Electrons beams passed from a hole of shadow mask activate the phosphor dot pattern to display color picture



- Resolution: The maximum number of points that can be displayed without an overlap on a CRT

- Aspect Ratio:
This number gives the ratio of vertical points to horizontal points necessary to produce equal length line in both direction. Aspect ratio
3:4 , 16:9

* Architecture of Raster Scan Display:

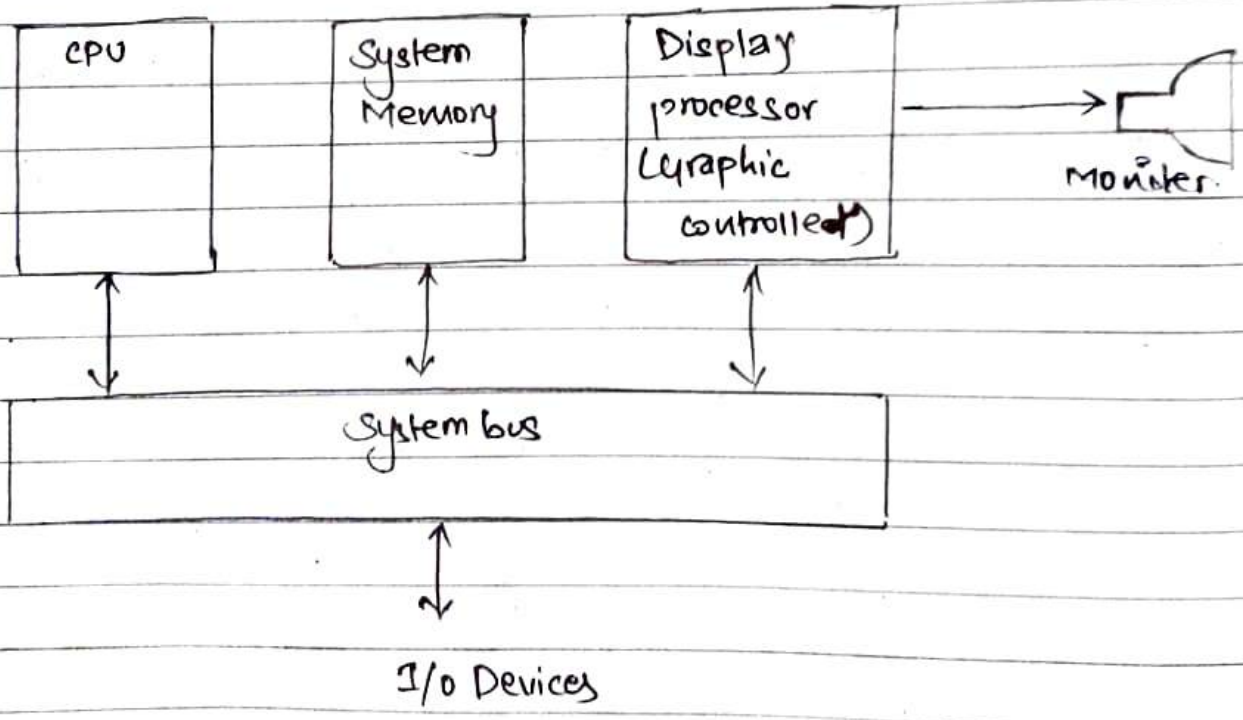


fig: Raster Scan Display.

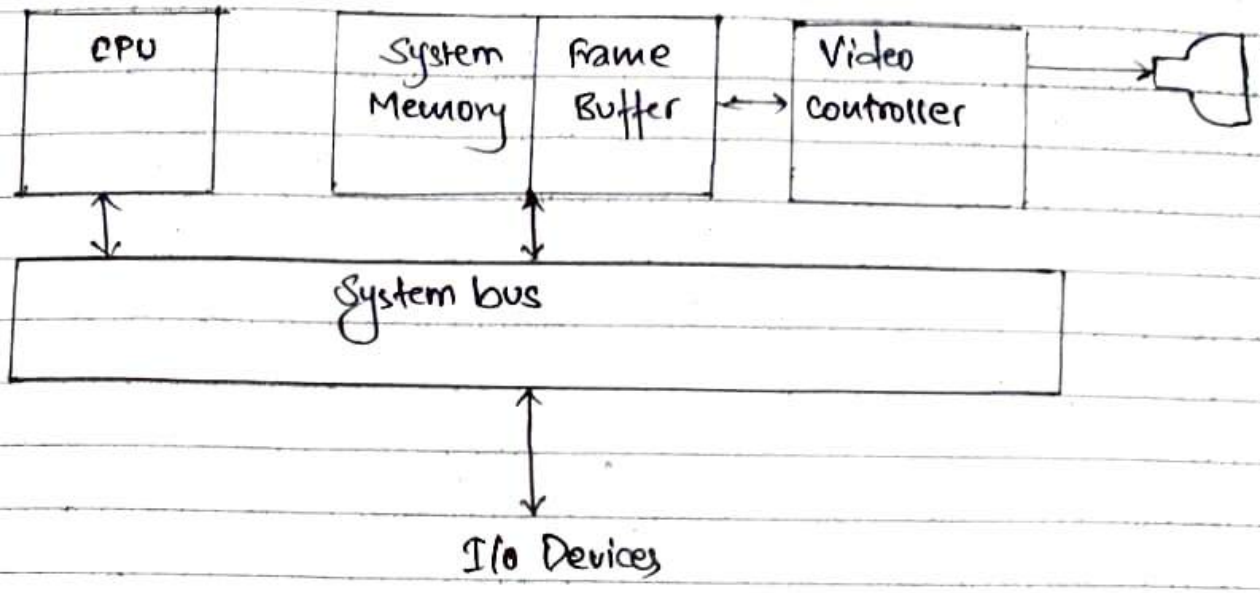


fig: Architecture of raster system with a fixed position of the system memory required for the frame buffer.

- Video controller:

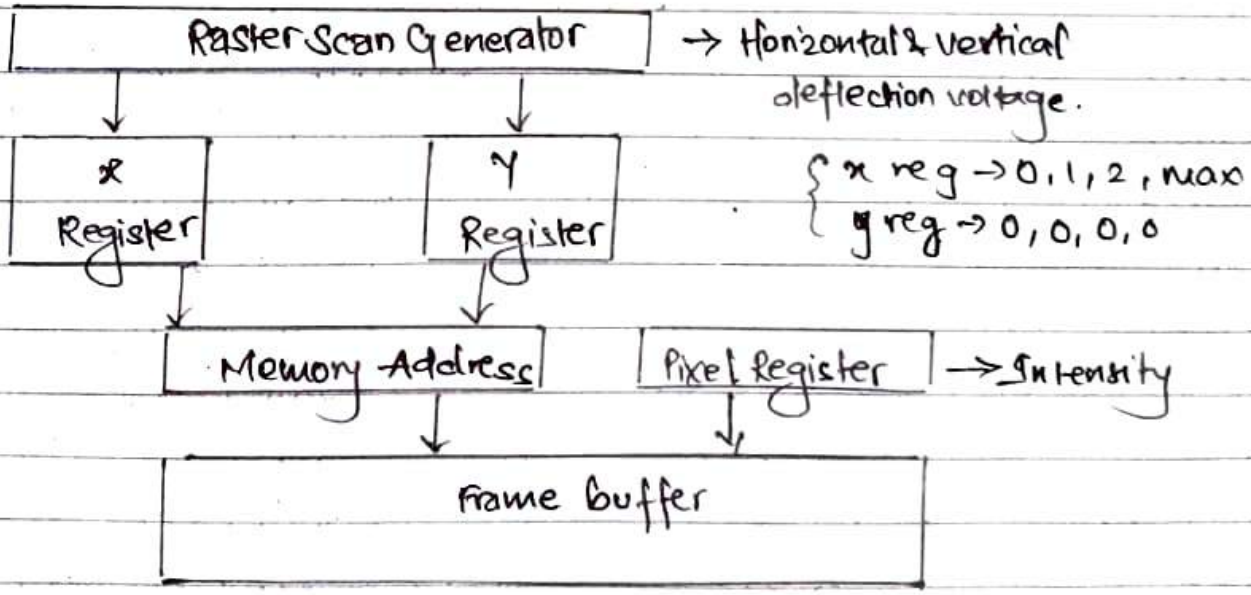


fig: Basic Video Controller ~~reference~~ refresh operation.

* Random Scan system.

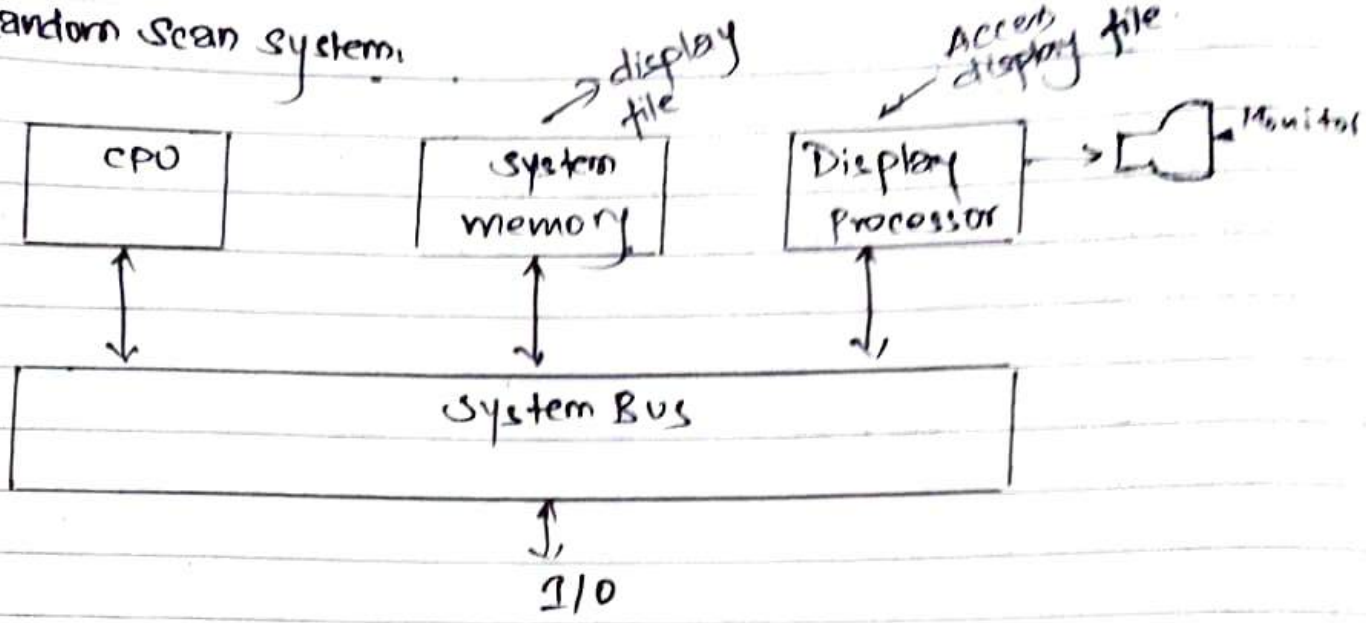
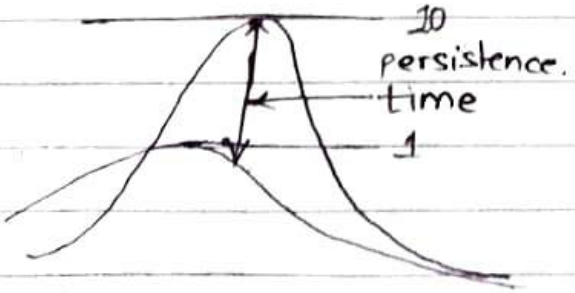


Fig: Architecture of Random Scan system

• Display file contain series of line drawing commands.

* Persistence time

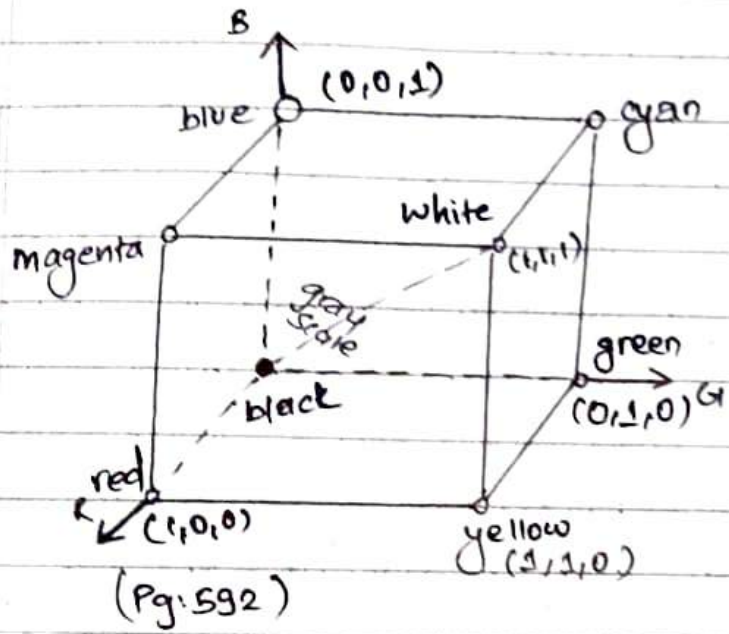
Time required to decay the original intensity to $1/10^{th}$



Color Models

1 = presence
0 = absence

a) RGB : Additive color Model.

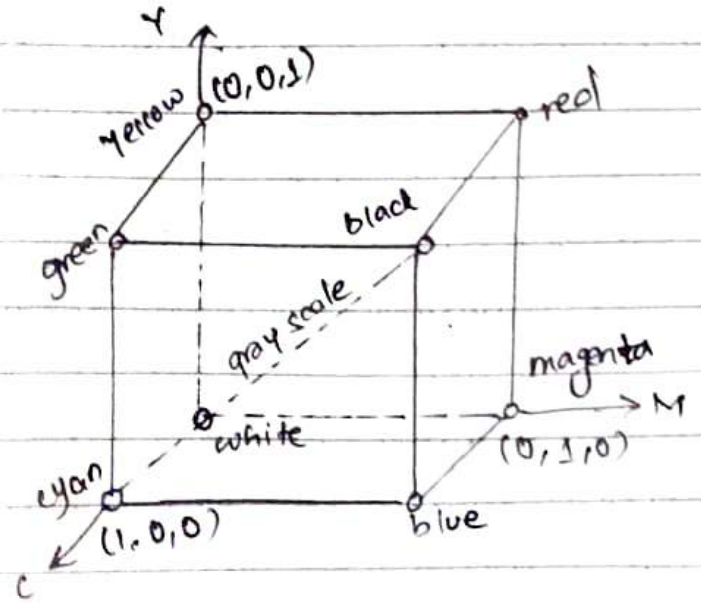


(Pg: 592)

$C\lambda = rR + gG + bB$

r, g, b are the concentration

b) CMY: Subtractive Color Model. (CMYK)



$C\lambda = \underbrace{ff + ff + ff}$

white.

Pg: 594

Conversion

- $$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix}$$

CMY → RGB

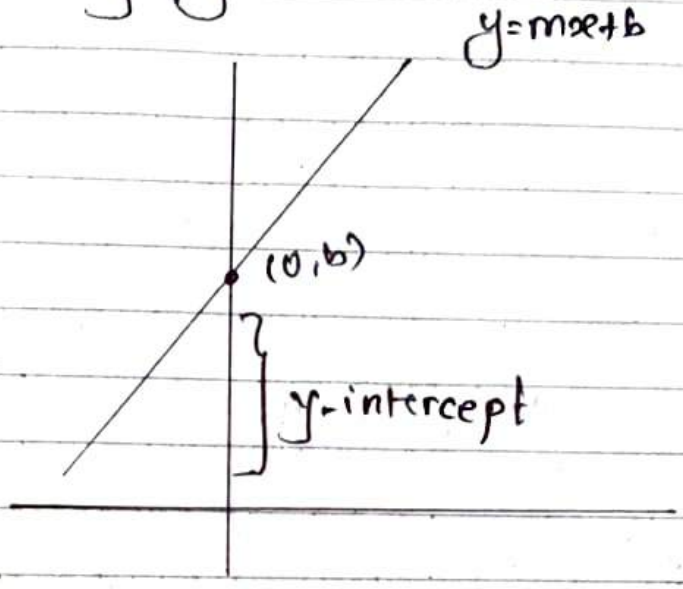
- RGB → CMYK

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Unit-3

2D Algorithm

Line drawing algorithm



m = slope

b = y-intercept

$m = \tan\theta = P/b = \frac{\text{rise}}{\text{run}}$.

X_k = current pixel.

X_{k+1} = next pixel.

(X_k, y_k) = current pixel.

(x_{k+1}, y_{k+1}) = next pixel.

Two types:

a) Digital Differential Analyzer (DDA)

b) Incremental line drawing algorithms

Chapter - 3

Digital Differential Analyzer (DDA)

1) case 1 : $|m| < 1$

$$m = \frac{\Delta y}{\Delta x} \quad \ldots (i)$$

$$\Delta x > \Delta y$$

$\left.\begin{array}{l} a/b < 1 \\ \therefore b > a \end{array}\right\}$

$$\Delta y = y_2 - y_1$$
$$\Delta x = x_2 - x_1$$

We perform sampling along x-axis

$$x_{k+1} = x_k + 1$$
$$\Delta x = 1$$

$$\therefore m = \Delta y \quad \ldots (ii)$$

$$\left.\begin{array}{l} x_{k+1} = x_k + 1 \\ y_{k+1} = y_k + m \\ \text{repeat } \Delta x \text{ times} \end{array}\right\}$$

$$y_{k+1} = y_k + \Delta y$$
$$y_{k+1} = y_k + m$$

2) Case 2: $|m| > 1$

$$m = \frac{\Delta y}{\Delta x} \quad \ldots\ldots (i)$$

$$\Delta y > \Delta x.$$

We perform sampling along y-axis

$$y_{k+1} = y_k + 1.$$
$$\Delta y = 1$$

$$m = 1/\Delta x$$

$$\therefore \Delta x = 1/m$$

$$\therefore x_{k+1} = x_k + \Delta x$$
$$x_{k+1} = x_k + 1/m$$

$$x_{k+1} = x_k + 1/m$$
$$y_{k+1} = y_k + 1$$
Repeat Δy times

3) Case 3: $|m| = 1$

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

$\therefore \Delta y = \Delta x$

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

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

Repeat Δy or Δx times.

- $(m < 1)$

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

$y_{k+1} = y_k \pm m$

Repeat Δx times

- $(m > 1)$

$x_{k+1} = x_k \pm 1/m$

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

Repeat Δy times

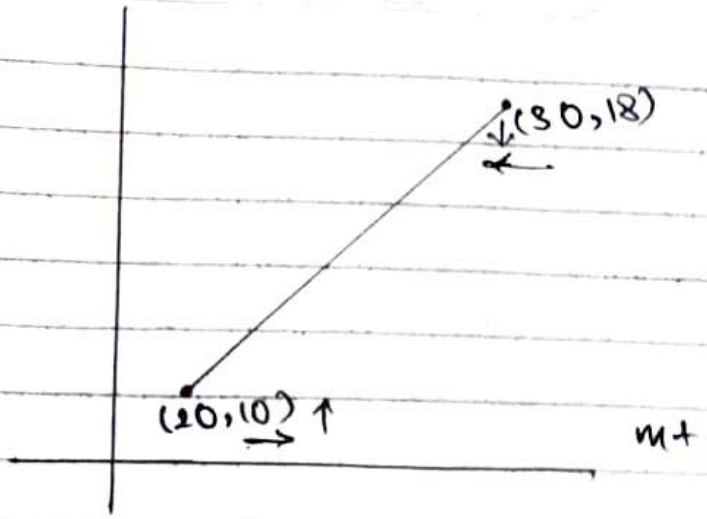
- $(m = 1)$

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

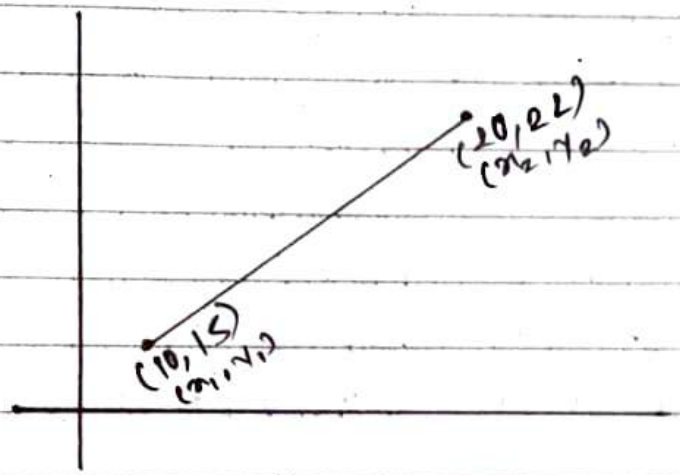
$y_{k+1} = y_k \pm 1$

Repeat Δy or Δx times.

Ex.



Q.



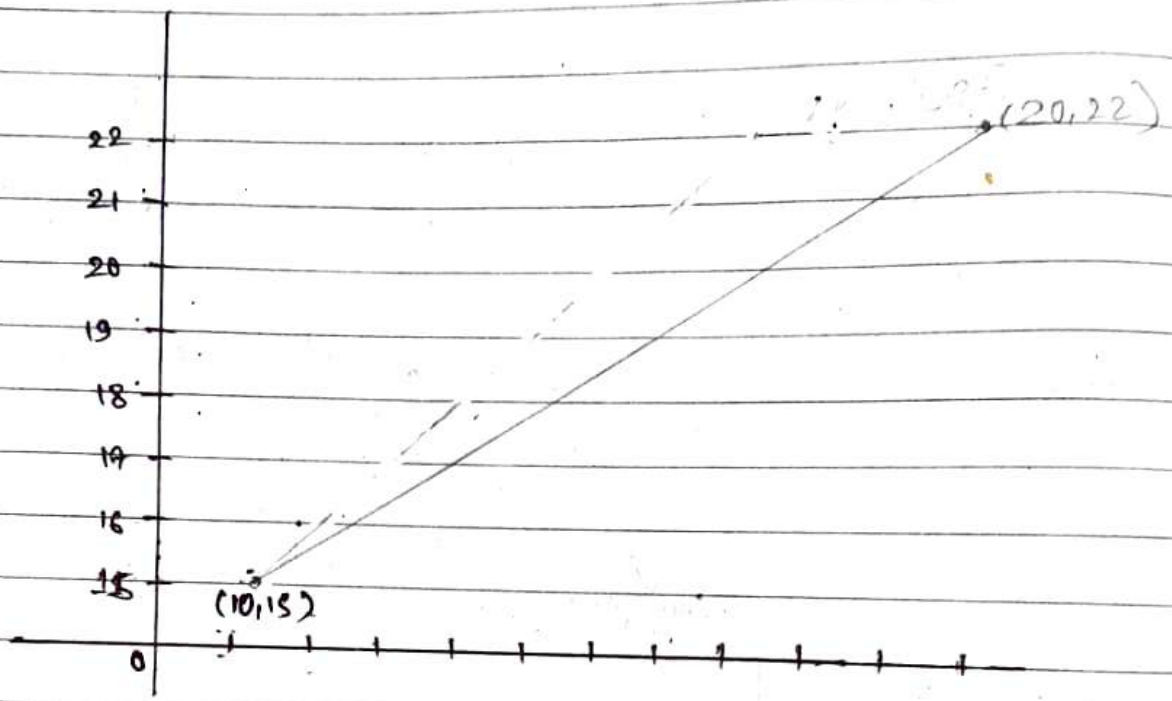
Solⁿ:

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

$$= \frac{22 - 15}{20 - 10} = 7/10 = 0.7$$

S.N'	x_{k+1} (10)	y_{k+1} (15)
0	11	15.7 ~16
1	12	16.4 ~16
2	13	17.1 ~17
3	14	17.8 ~ 18
4	15	18.5 ~ 18
5	16	19.2 ~ 19
6	17	19.9 ~ 20
7	18	20.6 ~ 21

8	19	21.3 ~21
9	20	22 ~22



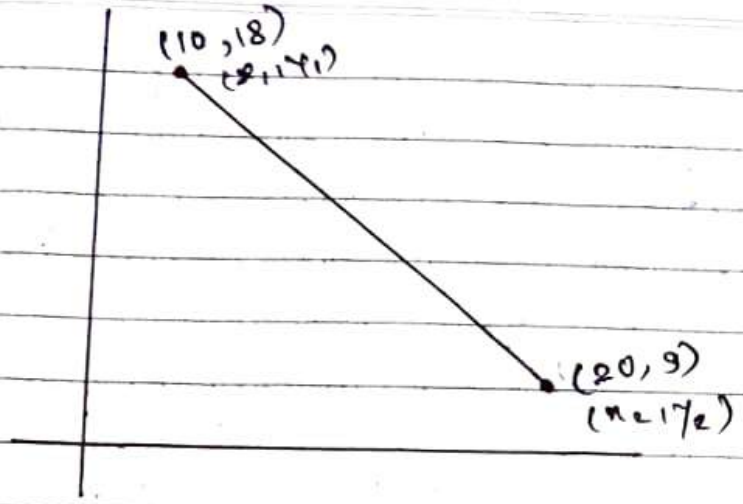
- DDA Algorithm:

1) Input the two line endpoints and store the left endpoint in (x_0, y_0)
2) Plot first point (x_0, y_0)
3) calculate constants $(\Delta x, \Delta y)$
4) If $|\Delta x| > |\Delta y|$ steps $= |\Delta x|$ else steps $= |\Delta y|$
5) Calculate XInc $= |\Delta x| /$ steps and YInc $= |\Delta y| /$ steps
6) At each x_k along the line, starting at $k=0$, plot the next pixel at $(xk + XInc, yk + YInc)$
7) Repeat Step 6. steps times.

Drawbacks:

- Transmission of round off errors resulting in to staircase effect alternatively known as Jaggering effect.

2)



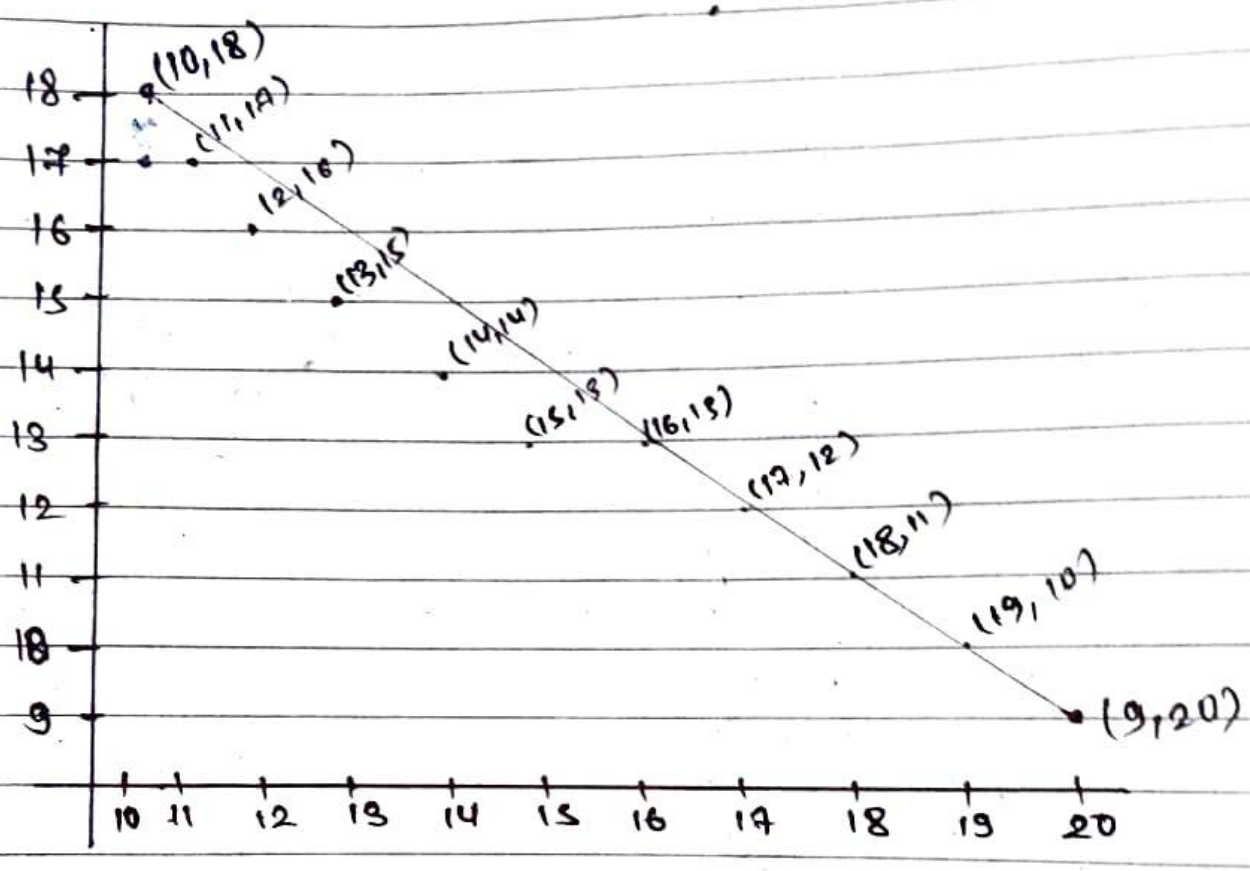
$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

$$= \frac{9 - 18}{20 - 10}$$

$$= \frac{-9}{10}$$

$$= -0.9 \qquad \therefore |m| = 0.9$$

SN	(10) x_{k+1}	(18) y_{k+1}
0	11	17.1 ~17
1	12	16.2 ~16
2	13	15.3 ~15
3	14	14.4 ~14
4	15	13.5 ~13
5	16	12.6 ~13
6	17	11.7 ~12
7	18	10.8 ~11
8	19	9.9 ~10
9	20	9 ~9


(10,18)
(11,17)
(12,16)
(13,15)
(14,14)
(15,13)
(16,13)
(17,12)
(18,11)
(19,10)
(9,20)
18
17
16
15
14
13
12
11
10
9
10 11 12 13 14 15 16 17 18 19 20

12 marks

- Bresenham's line Algorithm

Algorithm Steps ($|m| < 1$)

1) Input the two line endpoints and store the left endpoint in (x_0, y_0)

2) Plot first point (x_0, y_0)

3) Calculate constants Δx, Δy, $2\Delta y$ and $2\Delta y - 2\Delta x$, and obtain $p_0 = 2\Delta y - \Delta x$

4) At each x_k along the line, starting at $k=0$, perform the following test:

If $p_k < 0$, the next point plot is $(x_k + 1, y_k)$ and

$$p_{k+1} = p_k + 2\Delta y$$

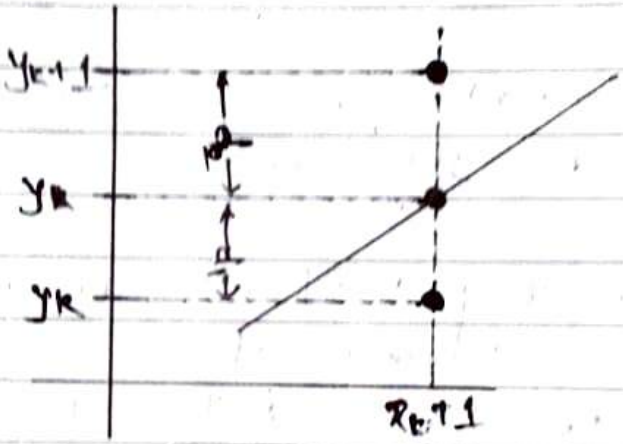
Otherwise, the next point to plot is $(x_k + 1, y_k + 1)$ and

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

5) Repeat step 4 Δx times.

Ex:-

- BLA for $|m| < 1$
- Start from left end point (x_0, y_0) step to each successive column (x samples) and plot the pixel whose scan line y value is closest to the line path
- After (x_k, y_k) the choice could be $(x_k + 1, y_k)$ or $(x_k + 1, y_k + 1)$



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

Then,

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

And

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

Difference between seperation

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

Defining decision parameter

$$P_k = \Delta x(d_1 - d_2) \quad \cdots [1]$$
$$= 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c$$

$\therefore$ constant $= 2\Delta y + \Delta x(2b - 1)$ which is independent of pixel position.

Sign of P_k is same as that of $d_1 - d_2$ for $\Delta x > 0$ (left to right sampling)

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

$$P_{k+1} - P_k = 2\Delta y (x_{k+1} - x_k) - 2\Delta x (y_{k+1} - y_k) \quad [\because c \text{ eliminated here}]$$

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

$$[y_{k+1} - y_k = 0 \text{ if } P_k < 0$$

$$[y_{k+1} - y_k = 1 \text{ if } P_k \geq 0$$

for Recursive calculation, initially

$$P_0 = 2\Delta y - \Delta x \quad [\text{Substitute } b = y_0 - m \cdot x_0 \text{ and } m = \Delta y / \Delta x \text{ in [1]}]$$

$p_0 = 2\Delta y - \Delta x$

if $p_k \leq 0$ (x_{k+1}, y_k) (x_{k+1}, y_{k+1})

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

else →

Repeat Δx times

Numerical:

Digitize a line having end points (20,10) to (30,18)

Soln:

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

$$= \frac{18 - 10}{30 - 20}$$

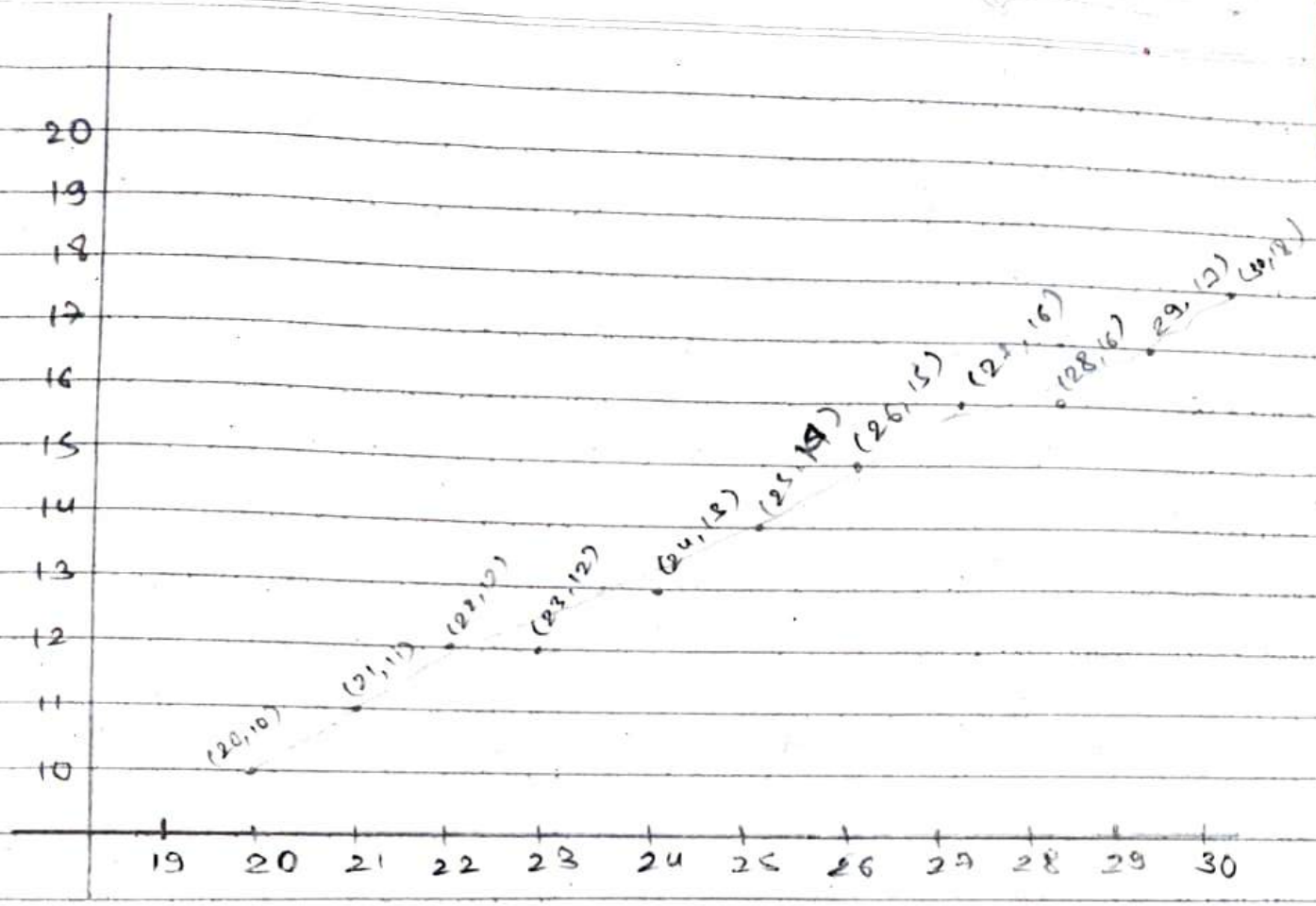
$$= 0.8$$

$|m| < 1$

[If Δy or Δx is negative, discard the sign]

$\Delta y = 8$ $\Delta x = 10$ $2\Delta y = 16$ $2\Delta y - 2\Delta x = -4$.

k	p_k	20 x_{k+1}	10 y_{k+1}	$2\Delta y - \Delta x$
0	6	21	11	$p_0 = 16 - 10 = 6$
1	2	22	12	$p_1 = 6 - 4 = 2$
2	-2	23	12	$p_2 = 2 - 4 = -2$
3	14	24	13	$p_3 = -2 + 16 = 14$
4	10	25	14	$p_4 = 14 - 4 = 10$
5	6	26	15	$p_5 = 10 - 4 = 6$
6	2	27	16	$p_6 = 6 - 4 = 2$
7	-2	28	16	$p_7 = 2 - 4 = -2$
8	14	29	17	$p_8 = -2 + 16 = 14$
9	10	30	18	$p_9 = 14 - 4 = 10$
				$p_{10} = 10 - 4 = 6$


20
19
18
17
16
15
14
13
12
11
10
(20,10)
(21,11)
(22,12)
(23,12)
(24,13)
(25,14)
(26,15)
(27,16)
(28,16)
(29,17)
(30,18)
19
20
21
22
23
24
25
26
27
28
29
30

$P_1(10,15)$, $P_2(20,22)$

Sol:

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

$$= \frac{22-15}{20-10}$$

$$= \frac{7}{10}$$

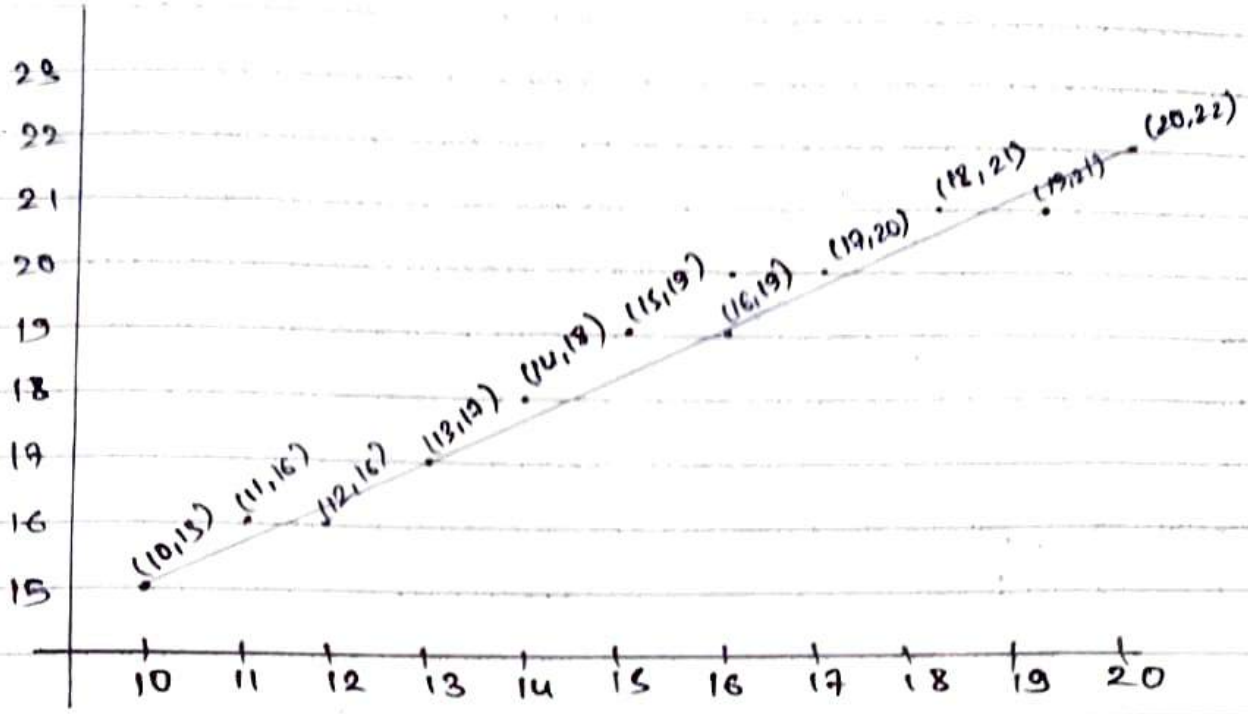
$$= 0.7$$

$|m| < 1$

$\Delta y = 7$ $\Delta x = 10$

$2\Delta y = 14$ $2\Delta y - 2\Delta x = -6$

k	P_k	x_{k+1} (10)	y_{k+1} (15)	
0	4	11	16	$P_0 = 14 - 10 = 4$
1	-2	12	16	$P_1 = 4 - 6 = -2$
2	12	13	17	$P_2 = -2 + 14 = 12$
3	6	14	18	$P_3 = 12 - 6 = 6$
4	0	15	19	$P_4 = 6 - 6 = 0$
5	-6	16	19	$P_5 = 0 - 6 = -6$
6	8	17	20	$P_6 = -6 + 14 = 8$
7	2	18	21	$P_7 = 8 - 6 = 2$
8	-4	19	21	$P_8 = 2 - 6 = -4$
9	10	20	22	$P_9 = -4 + 14 = 10$
				$P_{10} = 10 - 6 = 4$


23
22
21
20
19
18
17
16
15
(10,15)
(11,16)
(12,16)
(13,17)
(14,18)
(15,19)
(16,19)
(17,20)
(18,21)
(19,21)
(20,22)
10
11
12
13
14
15
16
17
18
19
20

- $|m| > 1$

$P_0 = 2\Delta x - \Delta y$

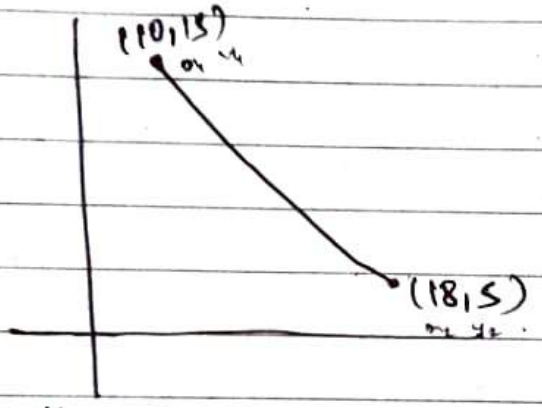
If $P_k \geq 0$ (x_{k+1}, y_{k+1})

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

otherwise (x_k, y_{k+1})

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

Repeat Δy times



Soln

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

$$= \frac{5 - 15}{18 - 10}$$

$$= \frac{-10}{8}$$

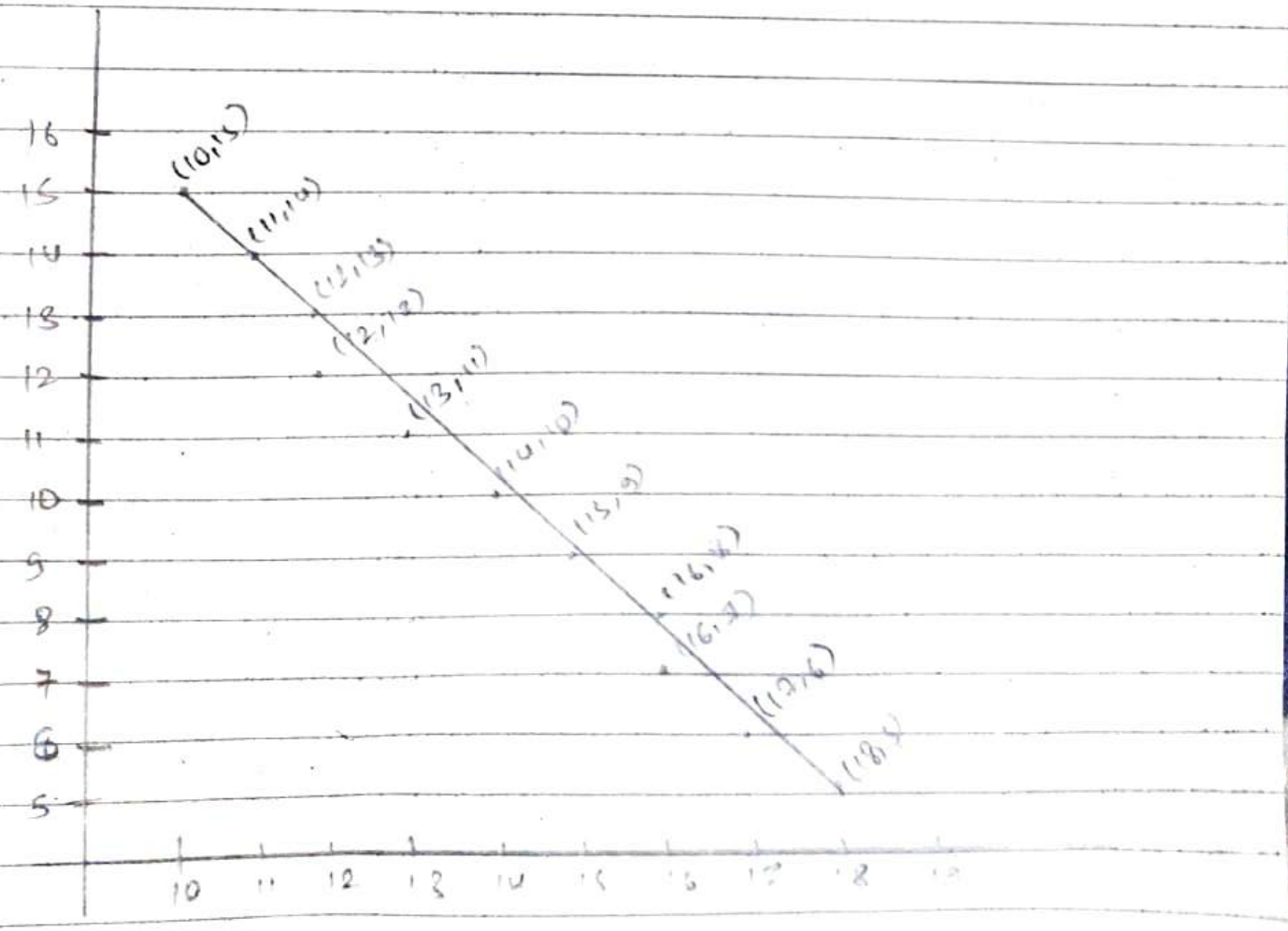
$$= -1.25$$

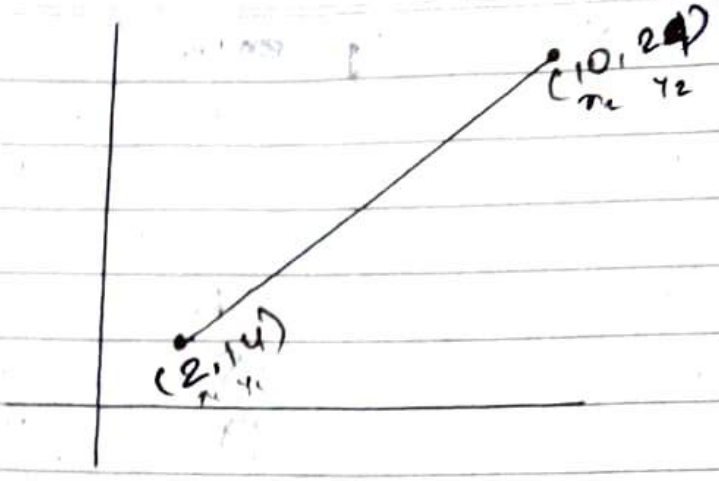
$|m| = 1.25$

$|m| > 1$

$\Delta y = 10$ (−10) $\quad \Delta x = 8 \quad 2\Delta y = 16 \quad 2\Delta x - 2\Delta y = -4$

k	P_k	x_{k+1} (10)	y_{k+1} (15)	
0	6	11	14	$P_0 = 16 - 10 = 6$
1	2	12	13	$P_1 = 6 + 16 - 20 = 2$
2	−2	12	12	$P_2 = 2 + 16 - 20 = -2$
3	14	13	11	$P_3 = -2 + 16 = 14$
4	10	14	10	$P_4 = 14 + 16 - 20 = 10$
5	6	15	9	$P_5 = 10 + 16 - 20 = 6$
6	2	16	8	$P_6 = 6 + 16 - 20 = 2$
7	−2	16	7	$P_7 = 2 + 16 - 20 = -2$
8	14	17	6	$P_8 = -2 + 16 = 14$
9	10	18	5	$P_9 = 14 + 16 - 20 = 10$
				$P_{10} = 10 + 16 - 20 = 6$





Sol:-

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

$$= \frac{24 - 14}{10 - 2}$$

$$= \frac{10}{8}$$

$$= 1.25$$

$\therefore |m| = 1.25$

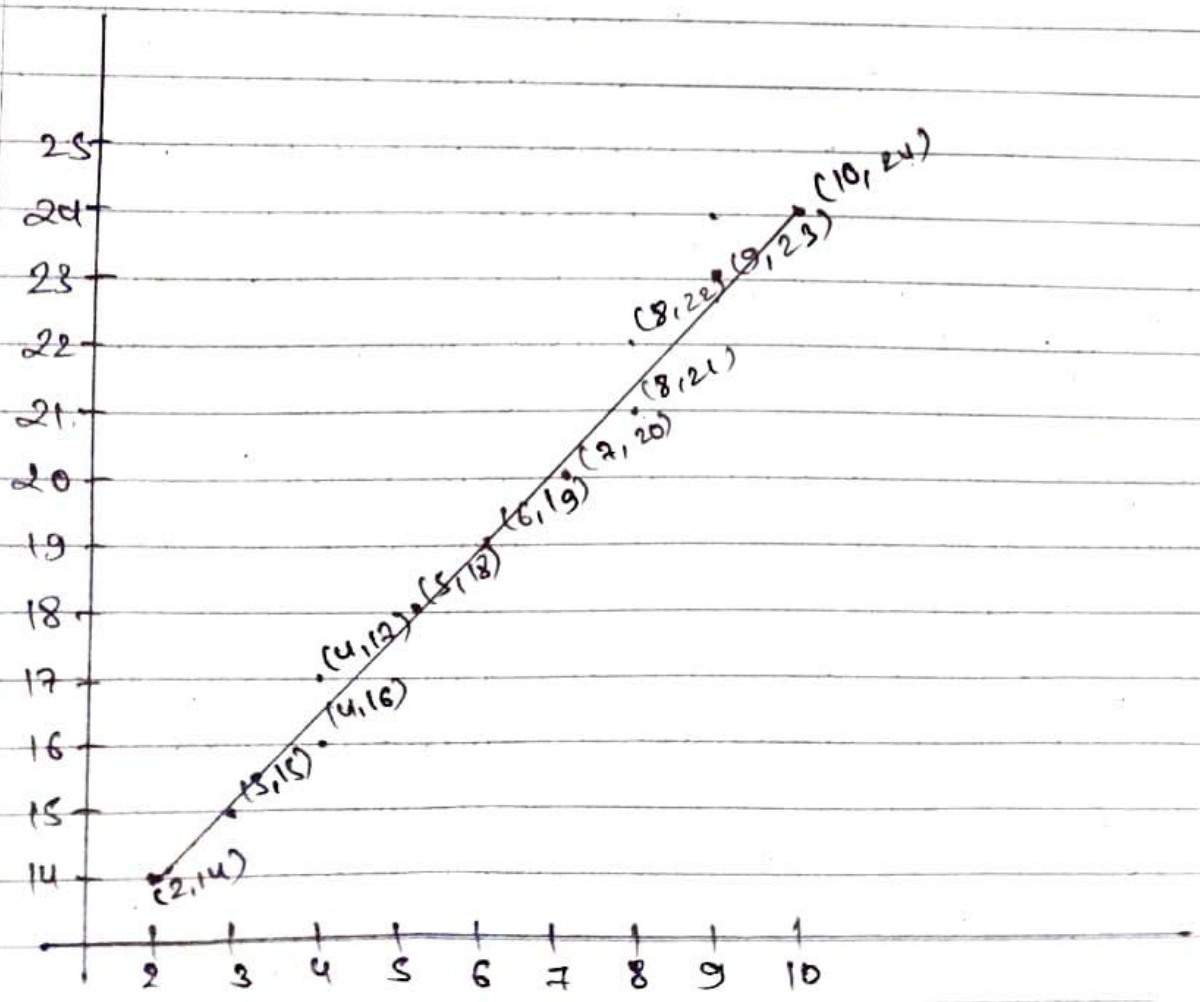
$|m| > 1$

$\Delta y = 10$ $\Delta x = 8$ $2\Delta x - 2\Delta y = 16 - 20 = -4$

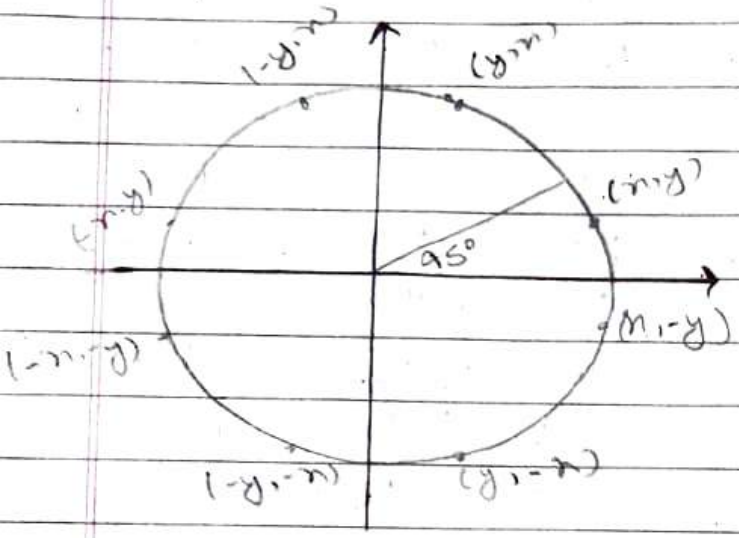
$x_0 = 2$ $y_0 = 14$

k	P_k	x_{k+1}	y_{k+1}	
0	6	3	15	$P_0 = 16 - 10 = 6$
1	2	4	16	$P_1 = 6 - 4 = 2$
2	-2	4	17	$P_2 = 2 - 4 = -2$
3	14	5	18	$P_3 = -2 + 16 = 14$
4	10	6	19	$P_4 = 14 - 4 = 10$
5	6	7	20	$P_5 = 10 - 4 = 6$
6	2	8	21	$P_6 = 6 - 4 = 2$
7	-2	8	22	$P_7 = 2 - 4 = -2$
8	14	9	23	$P_8 = -2 + 16 = 14$
9	10	10	24	$P_9 = 14 - 4 = 10$

4.



Circle symmetry / 8-way symmetry



(x, y)	$(2, 3)$
$(-x, y)$	$(-2, 3)$
$(x, -y)$	$(2, -3)$
$(-x, -y)$	$(-2, -3)$
(y, x)	$(3, 2)$
$(-y, x)$	$(-3, 2)$
$(y, -x)$	$(3, -2)$
$(-y, -x)$	$(-3, -2)$

Bresenham Midpoint Circle Drawing Algorithm.

Algorithm.

1. Input radius r and circle center (x_c, y_c) and obtain the first point on the circumference of a circle centered on the origin as

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

2. Calculate the initial value of the decision parameter as

$$P_0 = 5/4 - r$$

3. At each x_k position, starting at $k=0$, perform the following test:

If $P_k < 0$, the next point along the circle centered on $(0,0)$ is (x_{k+1}, y_k) and

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

Otherwise, the next point along the circle is $(x_k + 1, y_k - 1)$ and

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

where $2x_{k+1} = 2x_k + 2$ and $2y_{k+1} = 2y_k - 2$.

4. Determine the symmetry points in the other seven octants.

5. Move each calculated pixel position (x, y) onto the circular path centered on (x_c, y_c) and plot the co-ordinate values:

$$x = x + x_c, \quad y = y + y_c.$$

6. Repeat steps 3 through 5 until $x \geq y$.

$P_0 = 1 - r$

If $P_k < 0$ (x_{k+1}, y_k)

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

else $(x_k + 1, y_k - 1)$

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

{until $x \geq y$).

- Circle function defined as:
  $f_{circle}(x,y) = x^2 + y^2 - r^2$.

- Any point (x,y) satisfies the following conditions:

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

- Decision parameter is the circle function, evaluated as:

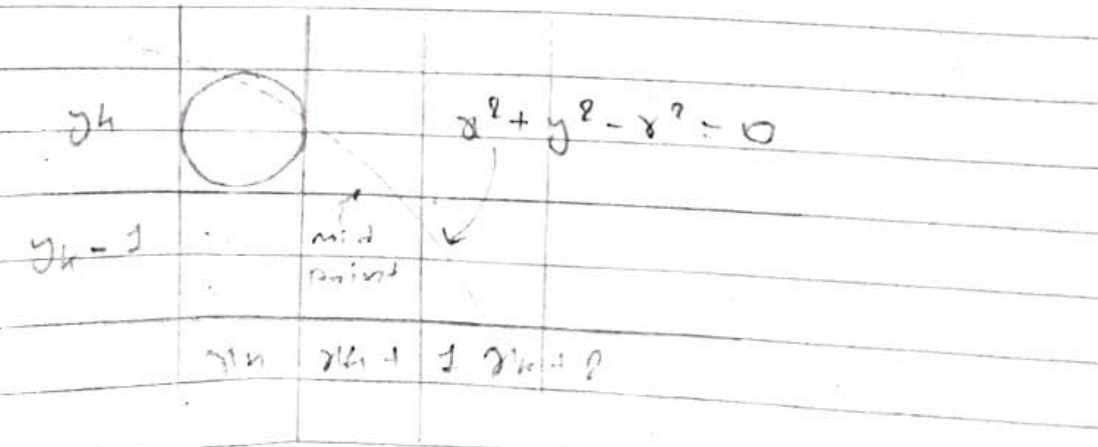
$$p_k = f_{circle}\left(x_k + 1, y_k - \frac{1}{2}\right)$$

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

$$p_{k+1} = f_{circle}\left(x_{k+1} + 1, y_{k+1} - \frac{1}{2}\right)$$

$$= \left[(x_k + 1) + 1\right]^2 + \left(y_{k+1} - \frac{1}{2}\right)^2 - r^2$$

$$p_{k+1} = p_k + 2(x_k + 1) + \left(y_{k+1}^2 - y_k^2\right) - (y_{k+1} - y_k) + 1$$



$y_{k+1} = y_k$ if $p_k < 0$

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

- Thus

$p_{k+1} = p_k + 2x_{k+1} + 1$ if $p_k < 0$

$p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$ otherwise

- Also incremental evaluation of $2x_{k+1}$ and $2y_{k+1}$

$2x_{k+1} = 2x_k + 2$

$2y_{k+1} = 2y_k - 2$ if $p_k > 0$

- At start position $(x_0, y_0) = (0, r)$

$2x_0 = 0$ and $2y_0 = 2r$

- Initial decision parameter.

$p_0 = f_{circle}(1, r - 1/2)$

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

$= 5/4 - r$

- For r specified as an integer, round p_0 to

$p_0 = 1 - r$

(because all increments are integers)

Q.

1) Digitize first octant
2) Digitize a circle centered at origin
3) " " " (h,k)

For 1, no need of 8 way symmetry
2 & 3, 8 way symmetry
Additional incase of 3
> Add center (h,k) to calculated points.

$P_0 = 1 - r$

If $P_k < 0$ (x_{k+1}, y_k)

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

else (x_{k+1}, y_{k-1})

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

until $x \geq y$

Digitize a circle having radius r = 8 and centered about origin.

or,

$x^2 + y^2 = 8^2$

Soln:

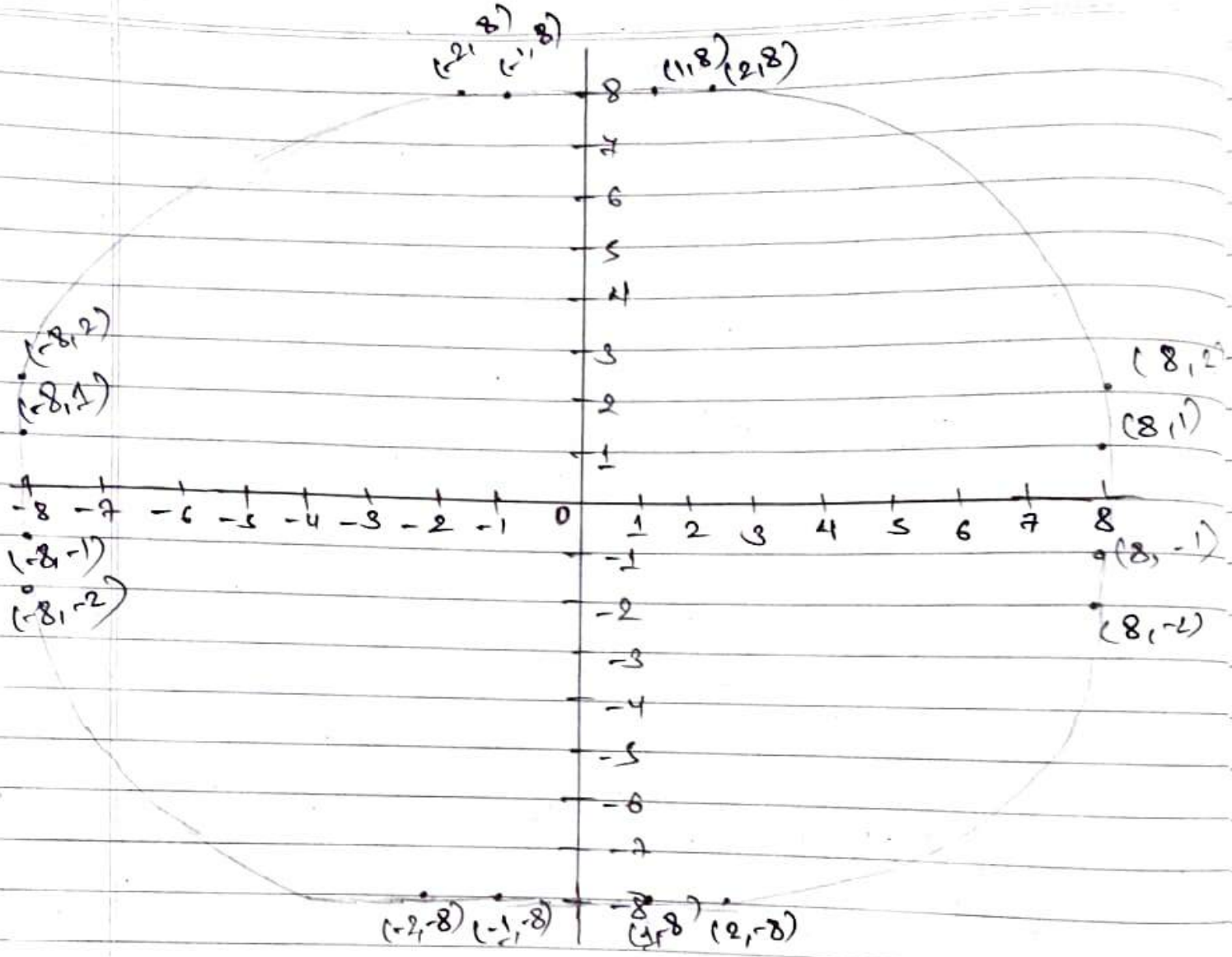
$P_0 = 1 - r$ (0) (8) $P_0 = 1 - 8 = -7$

k	P_k	x_{k+1}	y_{k+1}	$2x_{k+1}$	$2y_{k+1}$	P_0
0	-7	1	8	2	16	-7
1	-4	2	8	4	16	1
2	1	3	7	6	14	-6
3	-6	4	7	8	14	3
4	3	5	6	10	12	2
5	2	6	5	12	10	

{x = y}

8-way symmetry

(x, y)	(1, 8)	(2, 8)
(-x, y)	(-1, 8)	(-2, 8)
(x, -y)	(1, -8)	(2, -8)
(-x, -y)	(-1, -8)	(-2, -8)
(y, x)	(8, 1)	(8, 2)
(-y, x)	(-8, 1)	(-8, 2)
(y, -x)	(8, -1)	(8, -2)
(-y, -x)	(-8, -1)	(-8, -2)


(-2, 8)
(-1, 8)
(1, 8)
(2, 8)
(-8, 2)
(-8, 1)
(8, 2)
(8, 1)
(-8, -1)
(-8, -2)
(8, -1)
(8, -2)
(-2, -8)
(-1, -8)
(1, -8)
(2, -8)

Digitize a circle $(x-2)^2 + (y-3)^2 = 5^2$

Sol^n:

$P_0 = 1 - r$

$= 1 - 5 = -4.$

(0) (5)

k	P_k	x_{k+1}	y_{k+1}	$2x_{k+1}$	$2y_{k+1}$	P_0
0	-4	1	5	2	10	$-4+2+1=-1$
1	-1	2	5	4	10	$-1+4+1=4$
2	4	3	4	6	8	$4+6+1-8=3$
3	3	4	3	8	6	$3+8+1-6=6$

$x \geq y$

8-way symmetry

3 , 8

(x, y)	(1, 5)	(x+h, y+k)
(-x, y)	(-1, 5)	(1, 8) (-1+2, 5+3)
(-x, -y)	(-1, -5)	(3, -2)
(y, x)	(5, 1)	(1, -2)
(-y, x)	(-5, 1)	(7, 4)
(y, -x)	(5, -1)	(-3, 4)
(-y, -x)	(-5, -1)	(7, 2) (-3, -2)

Ellipse Generating Algorithms:

Algorithm:

1) Input r_x, r_y and the ellipse center (x_c, y_c) and obtain the first point on an ellipse centered on the origin as

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

2) Calculate the initial value of the decision parameter in region 1 as

$$p1_0 = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2$$

3) At each x_k position in region 1, starting at $k = 0$, perform the following test: If $p1_k < 0$, the next point along the ellipse centered on (0,0) is (x_{k+1}, y_k) and,

$$p1_{k+1} = p1_k + 2r_y^2 x_{k+1} + r_y^2$$

otherwise, the next point along the ellipse is $(x_k + 1, y_k - 1)$ and

$$p1_{k+1} = p1_k + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_y^2$$

with

$$2r_y^2 x_{k+1} = 2r_y^2 x_k + 2r_y^2, \quad 2r_x^2 y_{k+1} = 2r_x^2 y_k - 2r_x^2$$

and continue until.

$$2r_y^2 x \geq 2r_x^2 y$$

4) Calculate the initial value of decision parameter in region 2 using the last point (x_0, y_0) calculated in region 1 as

$$p2_0 = r_y^2 \left(x_0 + \tfrac{1}{2}\right)^2 + r_x^2 (y_0 - 1)^2 - r_x^2 r_y^2$$

5) At each y_k position in region 2, starting at $k=0$, perform the following test: If $p2_k > 0$, the next point along the ellipse centered on $(0,0)$ is $(x_k, y_k - 1)$ and

$$p2_{k+1} = p2_k - 2r_x^2 y_{k+1} + r_x^2$$

Otherwise the next point along the ellipse is $(x_k + 1, y_k - 1)$ and

$$p2_{k+1} = p2_k + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_x^2$$

Using the same incremental calculations for x and y as in region 1.

6) Determine the symmetry points in the other three quadrants.

7) Move each calculated pixel position (x, y) onto the elliptical path centered on (x_c, y_c) and plot the co-ordinate values:

$$X = x + x_c \quad , \quad Y = y + y_c$$

8) Repeat the steps for region 1 until

$$2r_y^2 x \geq 2r_x^2 y$$

First pixel is (0,6)

$$p_0 = b^2 - a^2 b + a^2/4$$
$$= 36 - 64 \times 6 + 64/4$$
$$= -332.$$

Digitize an ellipse $(x-2)^2/64 + (y-5)^2/36 = 1$ using midpoint algorithm,

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

Soln

center = (2, 5)

$a = 8$, $r_x = 8$

$b = 6$, $r_y = 6$

k	p_k	(x_{k+1}, y_{k+1})	$2b^2 x_{k+1}$ (r_y)	$2a^2 y_{k+1}$ (r_x)
0	-332	(1, 6)	72	768
1	-224	(2, 6)	144	768
2	-44	(3, 6)	216	768
3	208	(4, 5)	288	640
4	-108	(5, 5)	360	640
5	288	(6, 4)	432	512
6	244	(7, 3)	504	384

We now move out of region 1, since $2b^2x > 2a^2y$.

For region 2, the initial point is $(x_0, y_0) = (7, 3)$ and the initial decision parameter is,

$$p_0 = b^2 (x_0 + 1/2)^2 + a^2 (y-1)^2 - a^2 b^2$$
$$= 36 (7 + 1/2)^2 + 64 (3-1)^2 - 36 \times 64$$
$$= 36 \times \frac{225}{4} + 64 \times 4 - 36 \times 64$$
$$= -23$$

The remaining positions along the ellipse path in the first quadrant are then calculated as.

If $p_k > 0$

(7,3)

K	p_k	(x_{k+1}, y_{k+1})	$2b^2 x_{k+1}$	$2a^2 y_{k+1}$
0	~~-23~~	(8,2)	576	256
1	~~-233~~ 361	(8,1)	576	128
2	~~-1465~~ 297	(8,0)	-	-

Rough. (Plot)

$p_r = b^2 ($

2D Transformation.

Transformation: changing co-ordinate description of an object is called transformation.

Types:
- Rigid body transformation (transformation without definition in shape.)
- Non rigid body transformation (transformation with change in shape)

Basic Transformations:
1. Translation
2. Rotation
3. Scaling

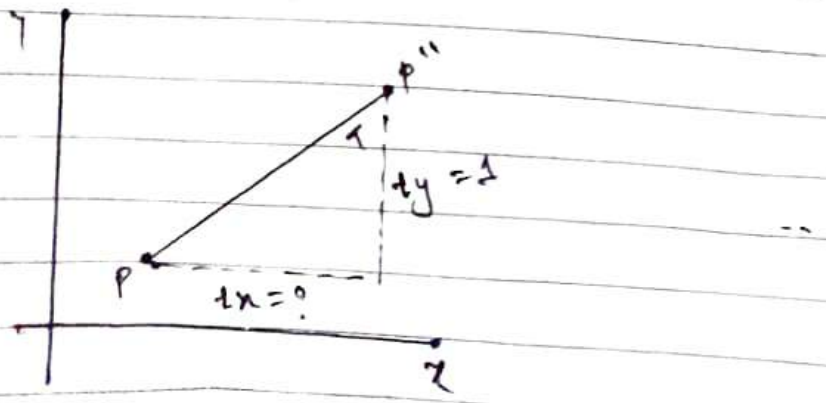
Other Transformation.
1. reflection
2. shearing.

Translation:

$$x' = x + t_x \quad , \quad y' = y + t_y$$

$$P = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \quad , \quad P' = \begin{bmatrix} x_0' \\ y_0' \end{bmatrix} \quad T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$P' = P + T$$



ACW = +ve
C.W = -ve

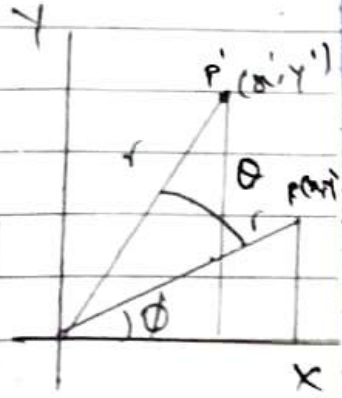
(Rotation about origin)

Rotation: (anti-clockwise)

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

$x = r\cos\phi$, $y = r\sin\phi$ — ii)

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



$$P' = R \cdot P$$

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

If co-ordinates represented as row vector, then: $P'^T = (R \cdot P)^T = P^T \cdot R^T$

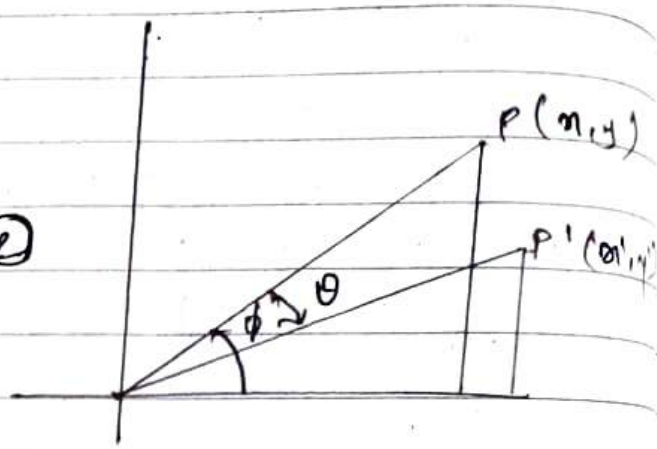
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

* In homogeneous co-ordinate.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow P' = R(\theta) \cdot P$$

Rotation (clockwise rotation)

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



$$x' = r\cos(\phi - \theta) \quad \} \quad -(2)$$
$$y' = r\sin(\phi - \theta) \quad \}$$

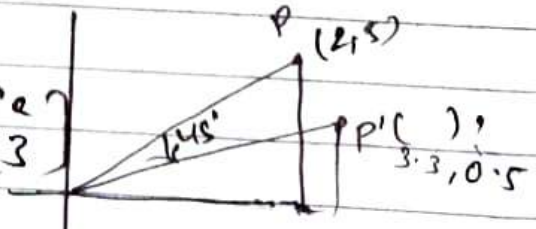
$$x' = r\cos\phi \cdot \cos\theta + r\sin\phi \cdot \sin\theta.$$
$$y' = r\sin\phi \cdot \cos\theta - r\cos\phi \cdot \sin\theta.$$

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

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

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

Ex.



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos 45' & \sin 45' \\ -\sin 45' & \cos 45' \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

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

$$= \begin{bmatrix} 1/\sqrt{2} \times 2 + 1/\sqrt{2} \times 3 \\ -1/\sqrt{2} \times 2 + 1/\sqrt{2} \times 3 \end{bmatrix}$$

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

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

$$= \begin{bmatrix} 5/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

$$= \begin{bmatrix} 3.3 \\ 0.5 \end{bmatrix}$$

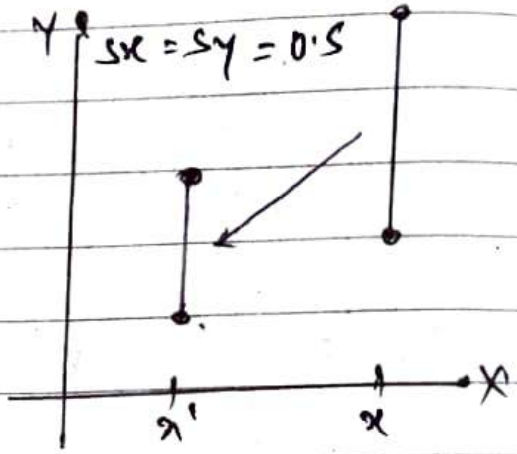
$$= (3.3, 0.5)$$

Scaling

$x' = x \cdot s_x \;,\; y' = y \cdot s_y$

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

$P' = S \cdot P$



Inverse Transformation

$$T^{-1}(t_x, t_y) = T(-t_x, -t_y)$$
$$R^{-1}(\theta) = R(-\theta)$$
$$S^{-1}(s_x, s_y) = S(1/s_x, 1/s_y)$$

Homogenous co-ordinate form

Q. (Why we need homogenous co-ordinate form?)

To treat all the transformation in a common way,

$$(x, y) \rightarrow (x, y, h)$$

Taking $h = 1$

$$(x, y) \rightarrow (x, y, 1)$$

- Translation

$$x' = x + t_x$$
$$y' = y + t_y$$
$$1 = 1$$

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

- Scaling

$x' = x * S_x$

$y' = y * S_y$

$1 = 1$

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

- Rotation.

* Anticlockwise (ACW)

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

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

$1 = 1$

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

* clockwise (CW)

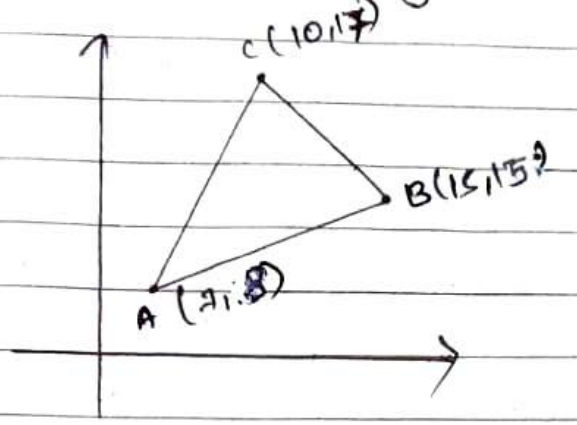
$x' = x\cos\theta + y\sin\theta$

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

$1 = 1$

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

Rotate a ΔABC by 45° in CW and scale to double if its original size, both about origin



Sol^n

T1 : Rotate by 45° in CW about origin i.e R(45) CW

T2 : Scale to double about origin i.e S(2,2)

Net Transformation (T) : $T_2 \times T_1$

$T = S(2,2) \times R(45)$ CW

$$T = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

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

Transformed points

$$[A'\ B'\ C'] = \begin{bmatrix} \sqrt{2} & \sqrt{2} & 0 \\ -\sqrt{2} & \sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 7 & 15 & 10 \\ 8 & 15 & 17 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \sqrt{2}\times 7 + \sqrt{2}\times 8 & \sqrt{2}\times 15 + \sqrt{2}\times 15 & \sqrt{2}\times 10 + \sqrt{2}\times 17 \\ -\sqrt{2}\times 7 + \sqrt{2}\times 8 & -\sqrt{2}\times 15 + \sqrt{2}\times 15 & -\sqrt{2}\times 10 + \sqrt{2}\times 17 \\ 0+0+1 & 0+0+1 & 0+0+1 \end{bmatrix}$$

$$= \begin{bmatrix} 15\sqrt{2} & 30\sqrt{2} & 27\sqrt{2} \\ \sqrt{2} & 0 & 7\sqrt{2} \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 21 & 42 & 38 \\ 1 & 0 & 10 \\ 1 & 1 & 1 \end{bmatrix}$$

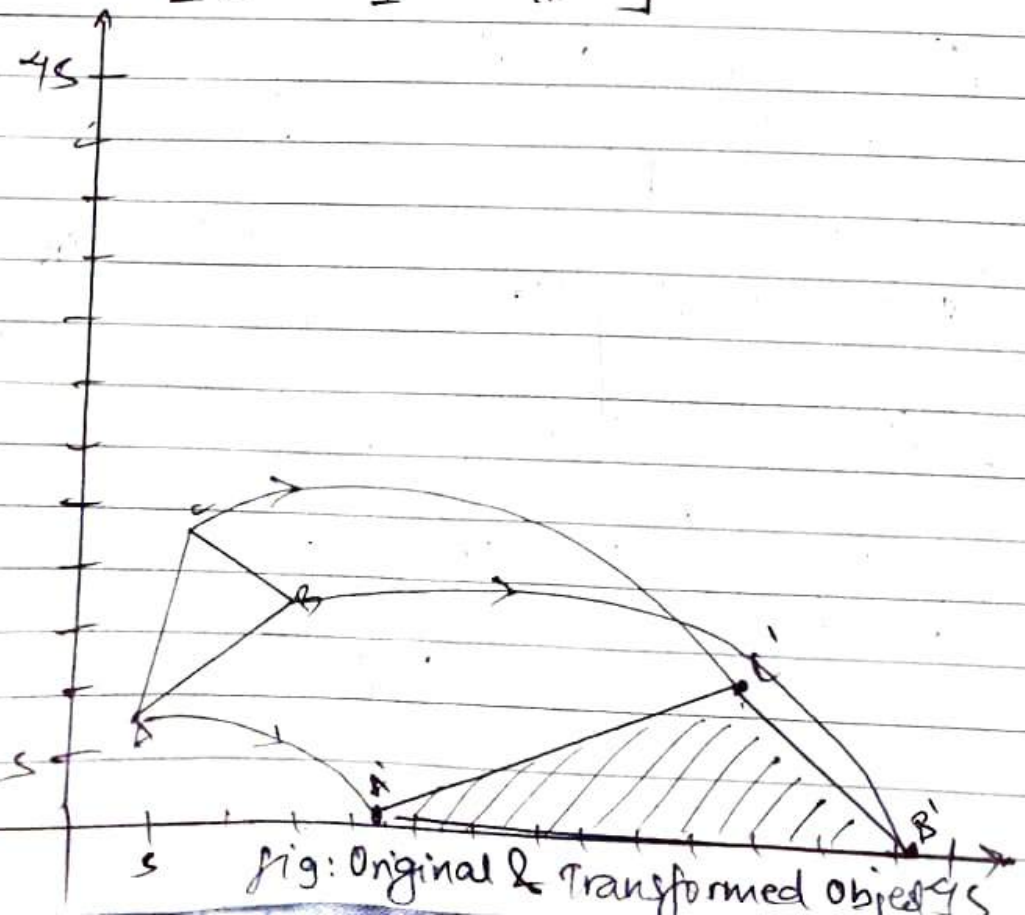
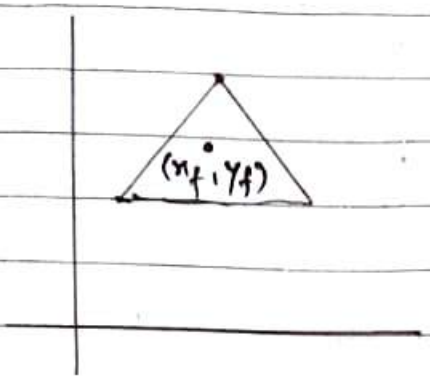


Fig: Original & Transformed Objects

derivation.

4.4 Pivot point scaling / Fixed point scaling



fig(A): original object

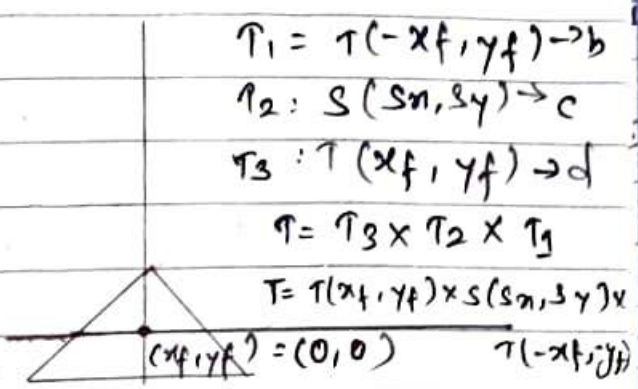
$T_1 = T(-x_f, y_f) \rightarrow b$

$T_2 : S(s_x, s_y) \rightarrow c$

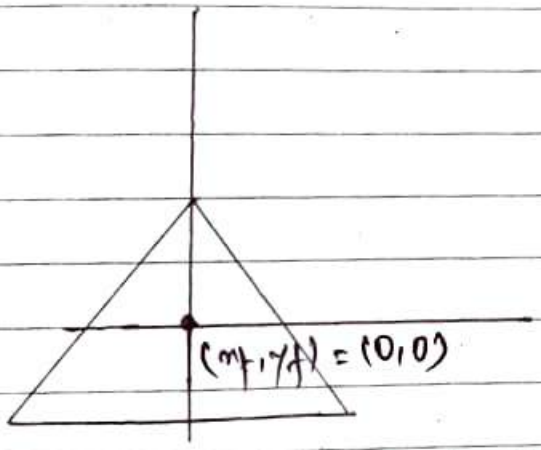
$T_3 : T(x_f, y_f) \rightarrow d$

$T = T_3 \times T_2 \times T_1$

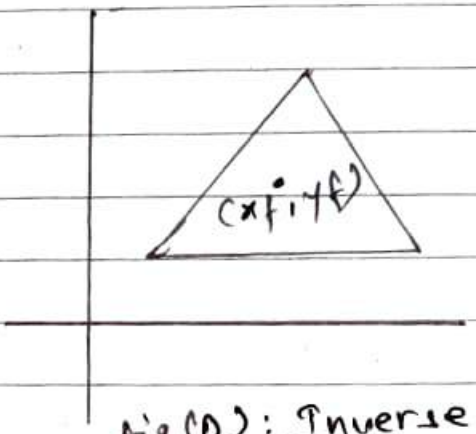
$T = T(x_f, y_f) \times S(s_x, s_y) \times$



fig(B): Translation of pivot point to origin



fig(C): Scaling about origin



fig(D): Inverse translation of fixed point

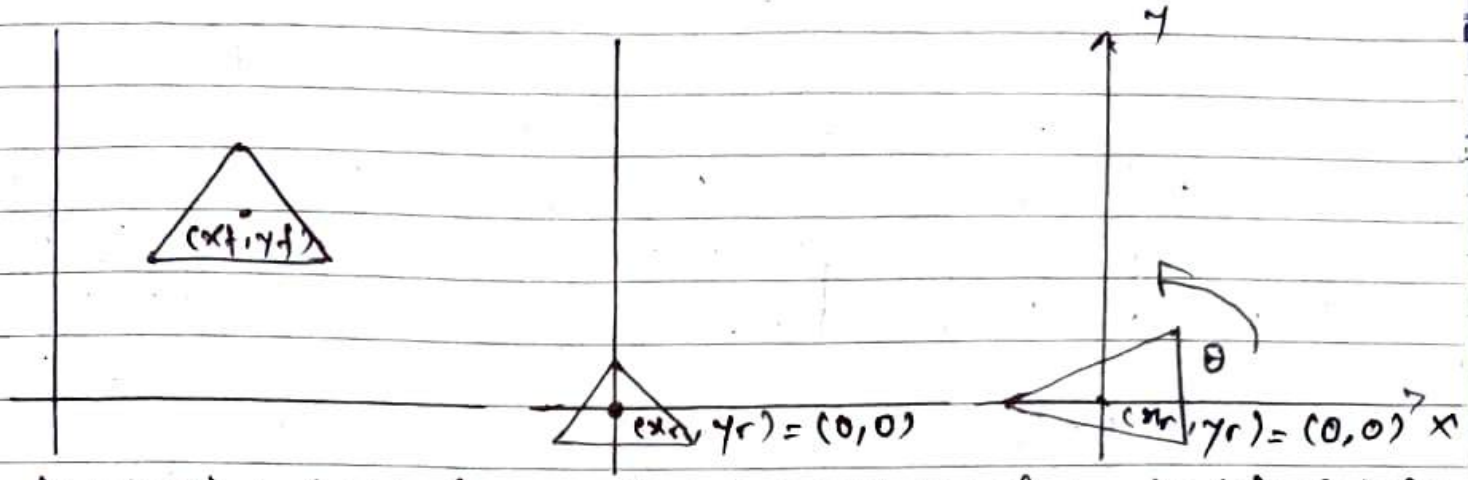
$T_4 \times T_3 \times T_2 \times T_1$

$$T = \begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} s_x & 0 & s_x * -x_f \\ 0 & s_y & s_y * -y_f \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} s_x & 0 & x_f(1-s_x) \\ 0 & s_y & y_f(1-s_y) \\ 0 & 0 & 1 \end{bmatrix}$$

Pivot point rotation.



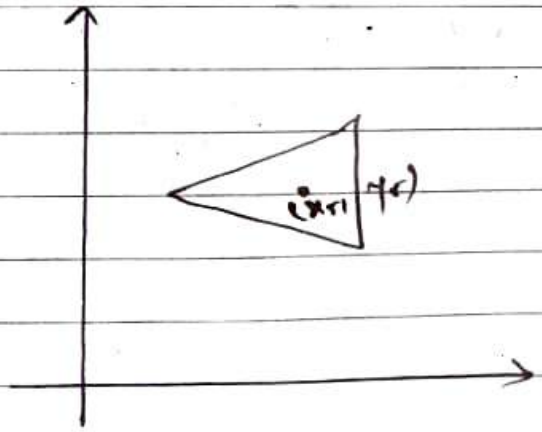
fig(a): Original object

fig(b): translation of fixed point to origin

fig(c): Rotate about origin by θ.

$$T = T(x_r, y_r)\, R(\theta)\, T(-x_r, -y_r)$$

ACW



fig(d): Inverse translation of fixed point

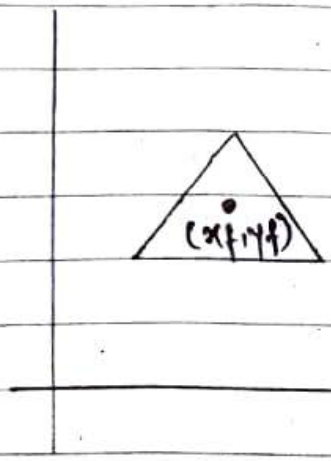
$$T = \begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos\theta & -\sin\theta & \cos\theta x - x_r + \sin\theta \cdot y_r \\ \sin\theta & \cos\theta & \sin\theta x - x_r + \cos\theta \cdot y_r \\ 0 & 0 & 1 \end{bmatrix}$$

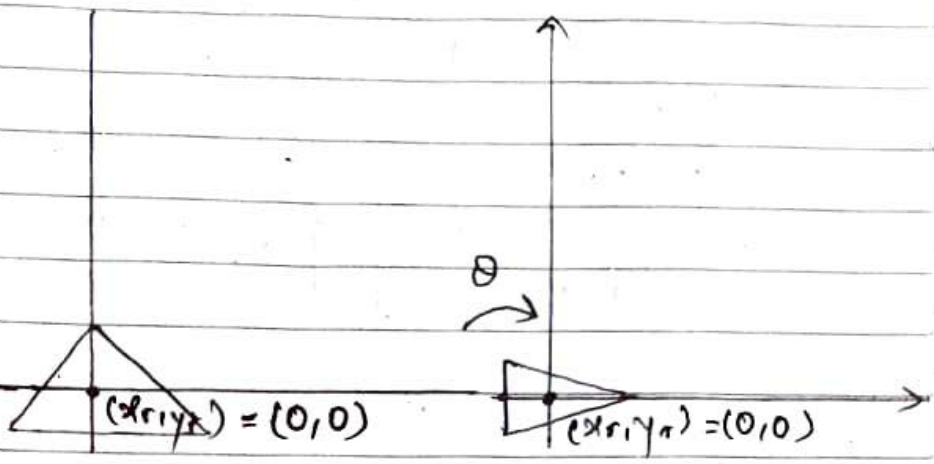
$$= \begin{bmatrix} \cos\theta & -\sin\theta & -\cos\theta x x_r + \sin\theta \cdot y_r + x_r \\ \sin\theta & \cos\theta & \sin\theta x - x_r + \cos\theta \cdot -y_r + y_r \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta & x_r(1 - \cos\theta) + \sin\theta y_r \\ \sin\theta & \cos\theta & \cdot y_r(1 - \cos\theta) - \sin\theta x_r \\ 0 & 0 & 1 \end{bmatrix}$$

Pivot point rotation (clockwise)



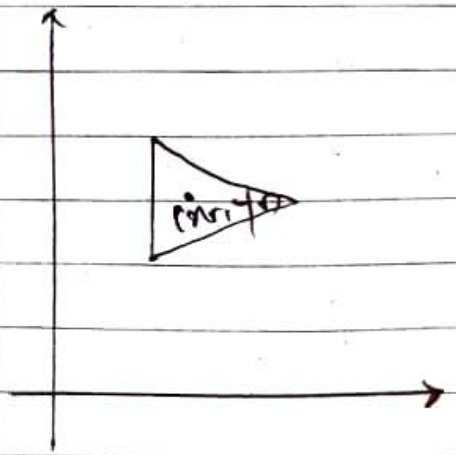
fig(a): Original object.



fig(b): Translation of fixed point to origin

fig(c): Rotate about origin by θ.

$$T = T(x_r, y_r) \times R(\theta) \text{ CW} \times T(-x_r, -y_r)$$



fig(d): Inverse translation of fixed point.

$$T = \begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix}$$

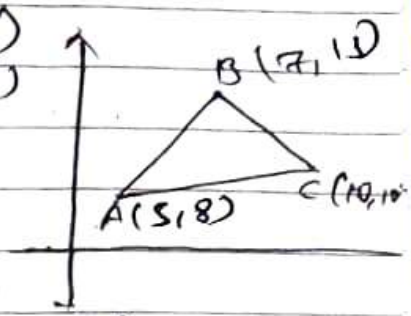
$$= \begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos\theta & \sin\theta & -x_r \cdot \cos\theta - y_r \cdot \sin\theta \\ -\sin\theta & \cos\theta & -x_r \cdot \sin\theta - y_r \cos\theta \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta & \sin\theta & -x_r \cos\theta - y_r \sin\theta + x_r \\ -\sin\theta & \cos\theta & +x_r \sin\theta - y_r \cos\theta + y_r \\ 0 & 0 & 1 \end{bmatrix}$$

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

Rotate △ ABC by 45° ACW about (5,8) & scale to double of its original size about (10,10)

(x_r, y_r) θ $(s_x, s_y) = (2, 2)$ (x_f, y_f)



T_1: Rotate by 45° in ACW about (5,8)

T_2: Scale to double about (10,10)

Net transformation (T) = T(10,10) S(2,2) T(−10,−10)

Fixed point scaling.

T(5,8) R(45)ACW T(−5,−8)

Fixed point rotation.

$$= \begin{bmatrix} s_x & 0 & x_f(1-s_x) \\ 0 & s_y & y_f(1-s_y) \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos\theta & -\sin\theta & x_r(1-\cos\theta) + y_r\sin\theta \\ \sin\theta & \cos\theta & y_r(1-\cos\theta) - x_r\sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} s_x\cdot\cos\theta & -\sin\theta\, s_x & s_x\cdot x_r(1-\cos\theta) + y_r\sin\theta + x_f(1-s_x) \\ s_y\cdot\sin\theta & s_y\cos\theta & s_y\cdot y_r(1-\cos\theta) - x_r\sin\theta + y_f(1-s_y) \\ & & \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 0 & 10(1-2) \\ 0 & 2 & 10(1-2) \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 5(1-1/\sqrt{2}) + 8\times\frac{1}{\sqrt{2}} \\ 1/\sqrt{2} & 1/\sqrt{2} & 8(1-1/\sqrt{2}) - 5\times 1/\sqrt{2} \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 0 & -10 \\ 0 & 2 & -10 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 7 \\ 1/\sqrt{2} & 1/\sqrt{2} & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2\times 1/\sqrt{2} & 2\times -1/\sqrt{2} & 2\times 7 + 0 - 10 \\ 2\times 1/\sqrt{2} & 2\times 1/\sqrt{2} & 0 - 2 - 10 \\ 0 & 0 & 1 \end{bmatrix}$$

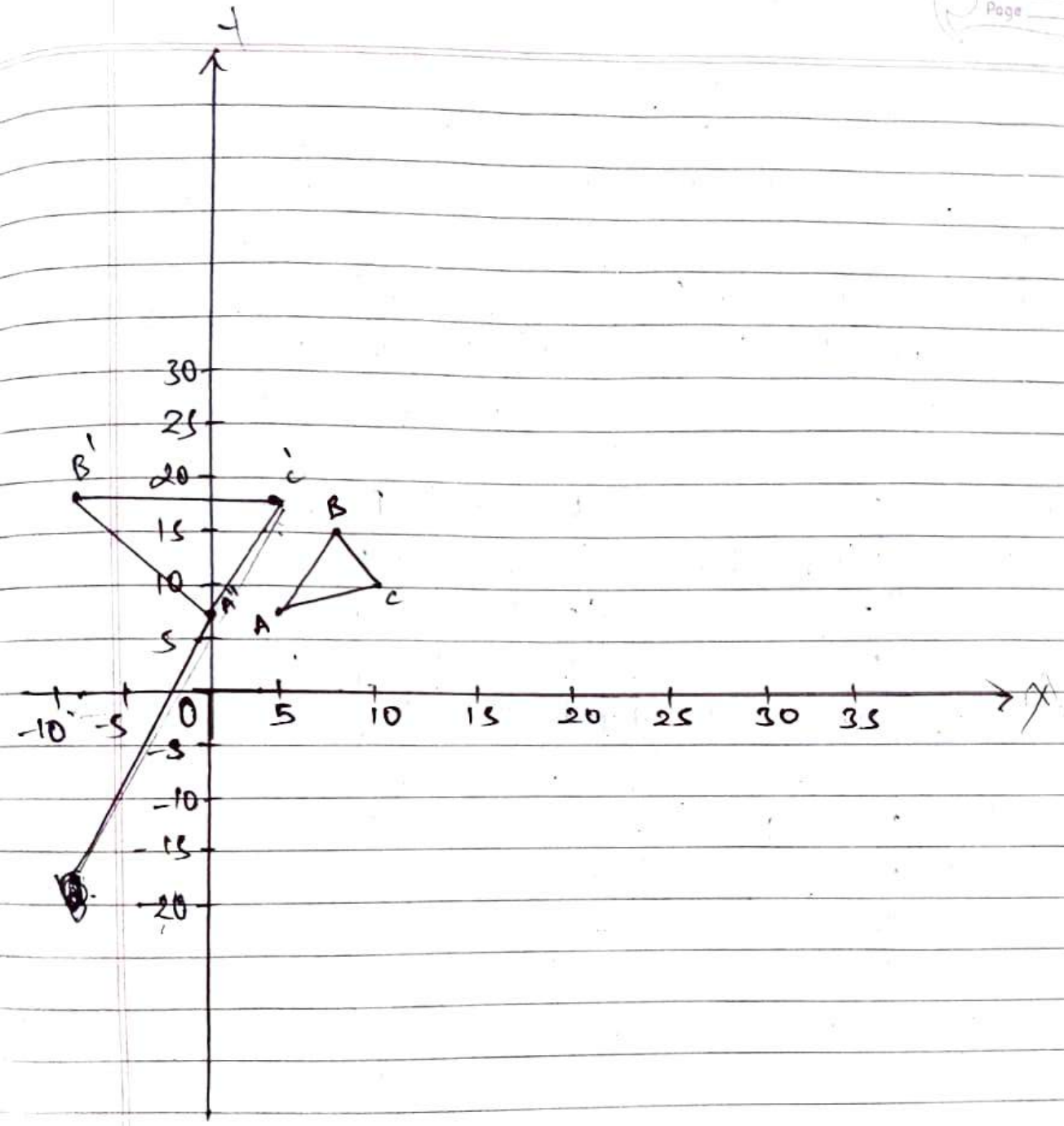
$$= \begin{bmatrix} \sqrt{2} & -\sqrt{2} & 4 \\ \sqrt{2} & \sqrt{2} & -12 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & -1 & 4 \\ 1 & 1 & -12 \\ 0 & 0 & 1 \end{bmatrix}$$

$$[A'B'C'] = \begin{bmatrix} \sqrt{2} & -\sqrt{2} & 4 \\ \sqrt{2} & \sqrt{2} & -12 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 5 & 7 & 10 \\ 8 & 15 & 10 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \sqrt{2}\times 5 - \sqrt{2}\times 8 + 4\times 1 & \sqrt{2}\times 7 - \sqrt{2}\times 15 + 4 & \sqrt{2}\times 10 - \sqrt{2}\times 10 + 4 \\ \sqrt{2}\times 5 + \sqrt{2}\times 8 - 12\times 1 & \sqrt{2}\times 7 + \sqrt{2}\times 15 - 12\times 1 & \sqrt{2}\times 10 + \sqrt{2}\times 10 - 12 \\ 0+1 & 1 & 1. \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -7 & 4 \\ 6 & 19 & 16 \\ 1 & 1 & 1 \end{bmatrix}$$


Y
X
30
25
20
15
10
5
0
-5
-10
-15
-20
-10
-5
5
10
15
20
25
30
35
B'
C'
A'
B
C
A

composite transformation

Imp. 1) Prove that sucessive translation is additive.

A point P is translated by Tc (tx_1, ty_1) and the result is again translated by Tc (tx_2, ty_2) and mapped to P'

$$P' = T(tx_2, ty_2)\{T(tx_1, ty_1) \cdot P\}$$

$$P' = \{T(tx_2, ty_2) \cdot T(tx_1, ty_1)\} P$$

$$= \begin{bmatrix} 1 & 0 & tx_2 \\ 0 & 1 & ty_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & tx_1 \\ 0 & 1 & ty_1 \\ 0 & 0 & 1 \end{bmatrix} \cdot P$$

$$P' = \begin{bmatrix} 1 & 0 & tx_1 + tx_2 \\ 0 & 1 & ty_1 + ty_2 \\ 0 & 0 & 1 \end{bmatrix} \cdot P$$

Hence, proved ~~#~~

2) Prove that successive scaling is multiplicative.

A point P is scaled by $S(S_{x_1}, S_{y_1})$ and the result is again scaled by $S(S_{x_2}, S_{y_2})$ and mapped to P'

$$P' = S(S_{x_2}, S_{y_2}) \cdot \{S(S_{x_1}, S_{y_1}) \cdot P\}$$

$$= \{S(S_{x_2}, S_{y_2}) \cdot S(S_{x_1}, S_{y_1})\} \cdot P$$

$$= \begin{bmatrix} S_{x_2} & 0 & 0 \\ 0 & S_{y_2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot P$$

$$P' = \begin{bmatrix} S_{x_2} \cdot S_x & 0 & 0 \\ 0 & S_{y_2} \cdot S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot P$$

Hence proved.

3) Prove that successive rotation is additive.

A point P is rotated by $R(\theta_1)$ and the result is again rotated by $R(\theta_2)$ and mapped to P'.

ACW

$$P' = R(\theta_1)\{R(\theta_2) \cdot P\}$$

$$= \{R(\theta_1) \cdot R(\theta_2)\} \cdot P$$

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

$$= \begin{bmatrix} \cos^2\theta - \sin^2\theta & -\cos\theta\cdot\sin\theta - \sin\theta\cdot\cos\theta & 0 \\ \sin\theta\cdot\cos\theta + \cos\theta\cdot\sin\theta & -\sin^2\theta + \cos^2\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot P$$

$$= \begin{bmatrix} \cos^2\theta - \sin^2\theta & -2\sin\theta\cdot\cos\theta & 0 \\ 2\sin\theta\cdot\cos\theta & \cos^2\theta - \sin^2\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot P$$

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

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

Other transformation

1) Reflection
2) Shearing

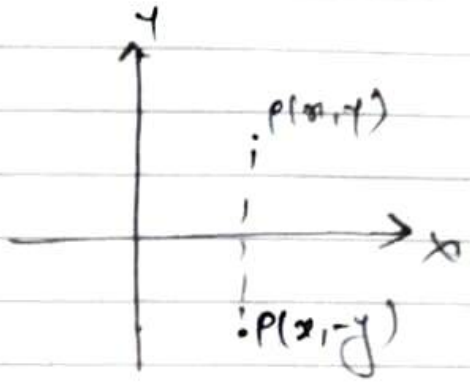
1) Reflection

1.1) Reflect about X-axis

$x' = x$

$y' = -y$

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

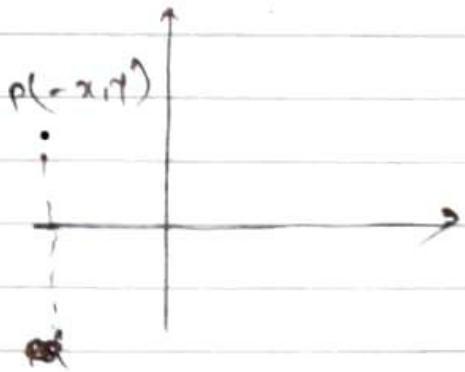


1.2) Reflect about Y-axis

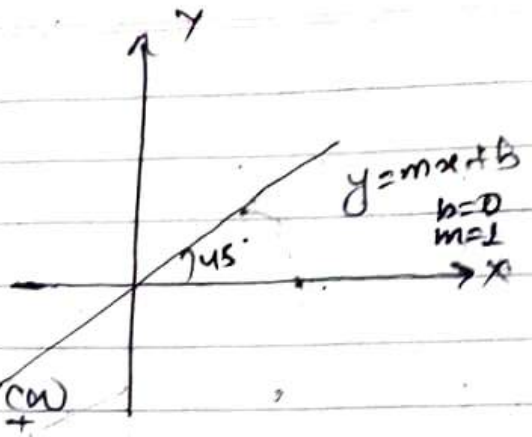
$x' = -x$

$y' = y$

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



- Reflect about $y=x$.



R1, ~~R(45)CW~~

$T = R(45)_{ACW}\ R_{fx}\ R(45)_{CW}$

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

R2,

$T = R(45)_{CW}\ R_{fy}\ R(45)_{ACW}$

Now,

$R_{2\times 2}$ for R_2

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

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

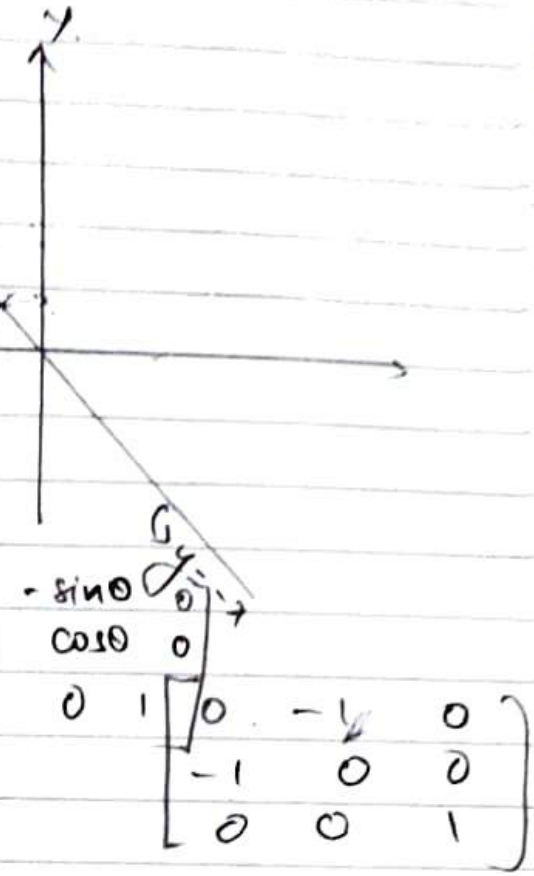
$$= \begin{bmatrix} \cos\theta \cdot -\cos\theta + \sin\theta \cdot +\sin\theta & \cos\theta \cdot +\sin\theta + \sin\theta \cdot \sin\theta & 0 \\ -\sin\theta \cdot -\cos\theta + \cos\theta \cdot \sin\theta & -\sin\theta \cdot \sin\theta + \cos\theta \cdot \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

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

. for R_1 also same.

* Reflect about $y = -x$



R_y,

$T = R(45)\ ACW\ \ Rf_y \qquad R(45)CW$

$x' = -y$

$y' = -x$

R_x

$T = R(45)CW\ \ Rf_x\ \ R(45)ACW$

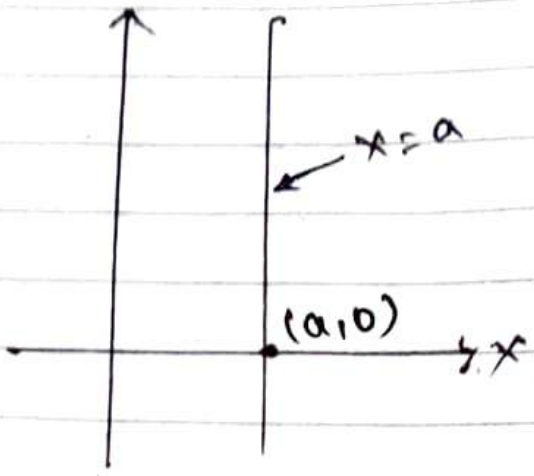
Now,

for R_x

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

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

Reflect about $x = a$



$T(a,0)$ Rfy $T(-a,0)$

Shear: Shearing Transformation

Shearing along x-axis

$x' = x + shx \cdot y$

$y' = y$

shx = shearing constant

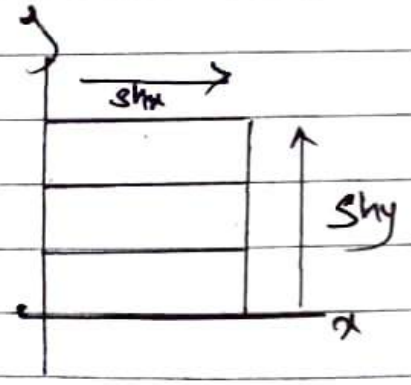
$$\begin{bmatrix} 1 & shx & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

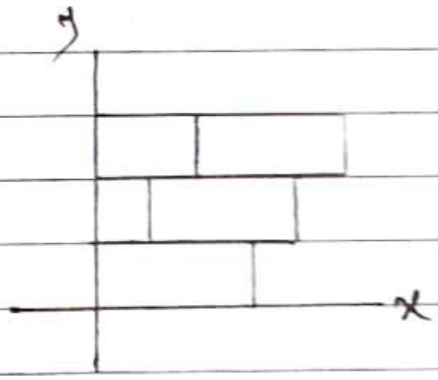
Shearing along y-axis

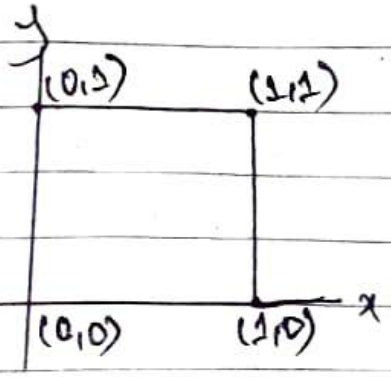
$x' = x$

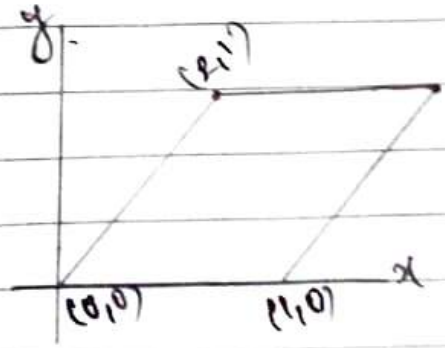
$y' = y + shy \cdot x$

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









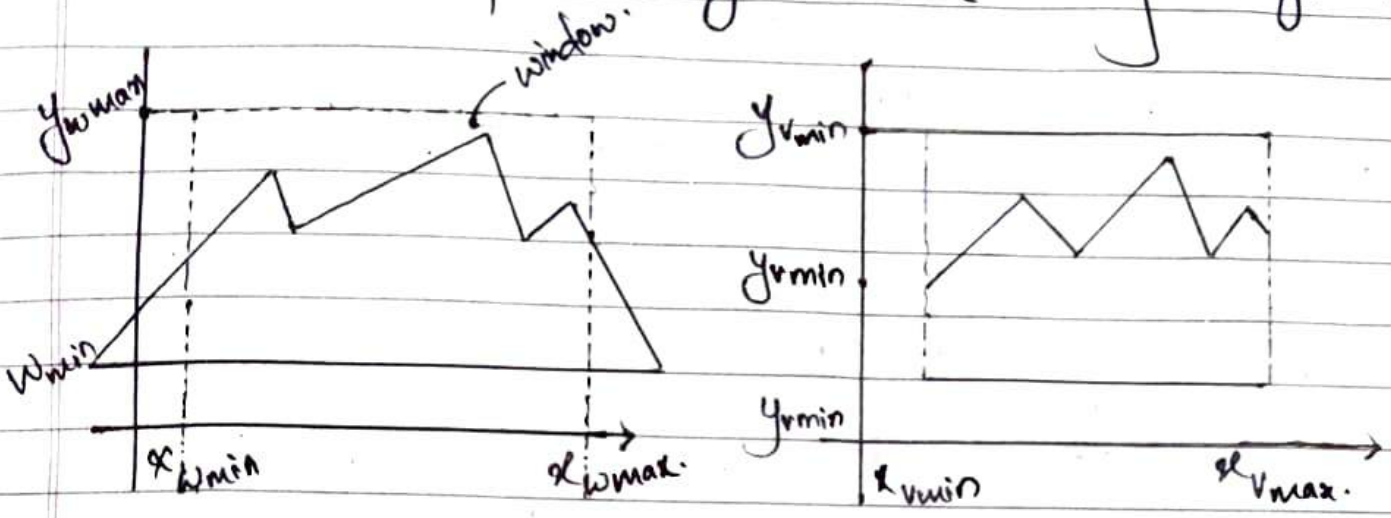
$$= \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\begin{matrix} O' & A' & B' & C' \end{matrix}$$

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

$T(0,b) \cdot R(\theta) \cdot A_{CW} \; R_{fx} \; R(\theta)_{CW} \; T(0,-b)$

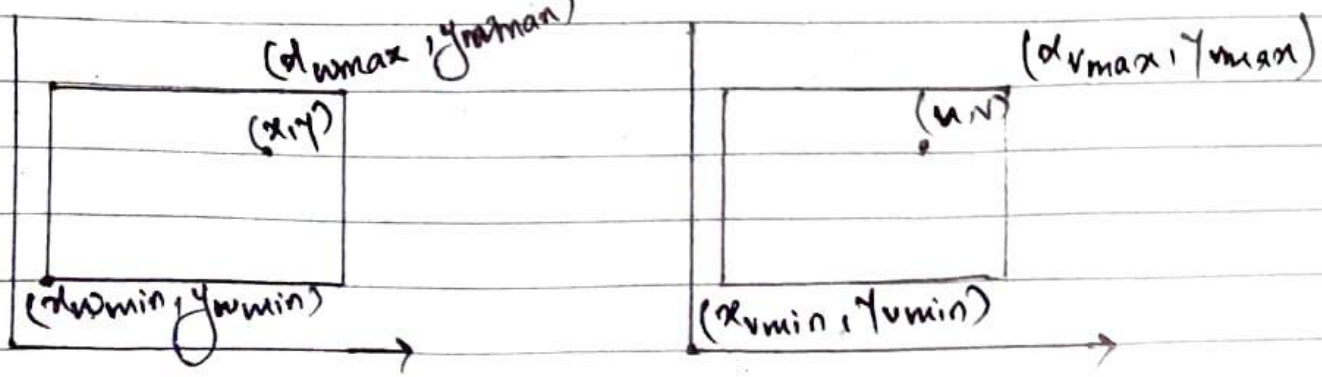
Window to viewport transformation: (Viewing Transformation)



World Co-ordinates

Device Co-ordinate.

It is the mechanism for displaying view of a picture of a picture on an output device. The world co-ordinate selected for display is called window. The area on the display device to which window is mapped is called viewport. So, window defines where it is to be displayed. The mapping of part of world co-ordinate scene to device co-ordinate is called viewing transformation or window-to-viewport transformation.



Window-to-viewport transformation can be explained as:
- Choose the world co-ordinate in a rectangle
- Transform it to origin
- Scale it with appropriate value.
- Transform it to the relative position in viewport.

Step 1: $T(-x_{wmin}, -y_{wmin})$
Step 2: $S(S_x, S_y)$
Step 3: $T(x_{wmin}, y_{wmin})$

∴ Net transformation

$$T_{wv} = T(x_{vmin}, y_{vmin}) \cdot S(S_x, S_y) \cdot T(-x_{wmin}, -y_{wmin})$$

where,

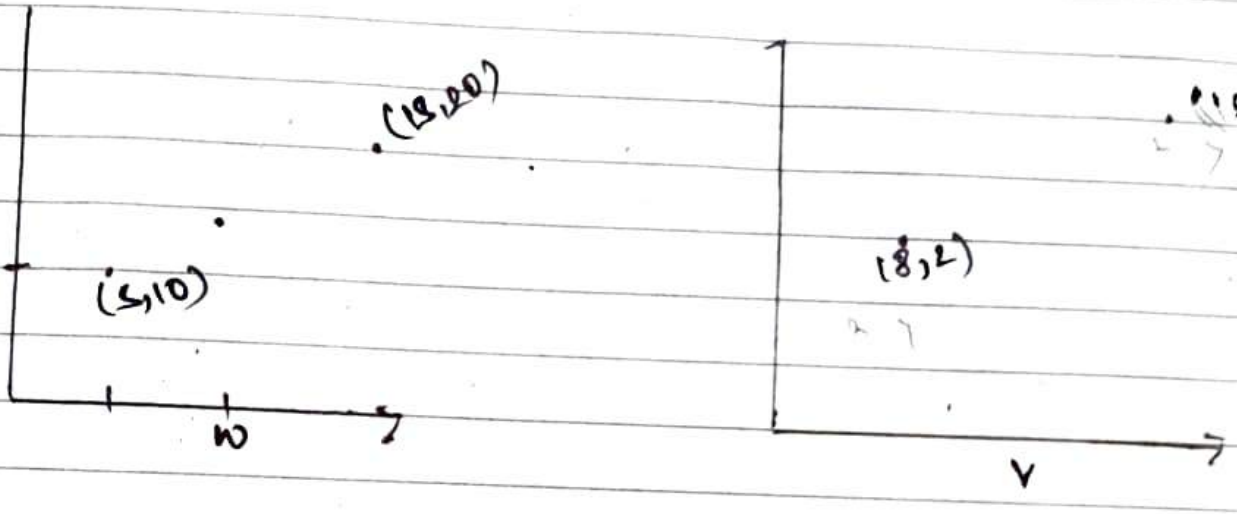
$$S_x = \frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}}$$

$$S_y = \frac{y_{vmax} - y_{vmin}}{y_{wmax} - y_{wmin}}$$

$$\begin{bmatrix} 1 & 0 & x_{vmin} \\ 0 & 1 & y_{vmin} \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & -x_{wmin} \\ 0 & 1 & -y_{wmin} \\ 0 & 0 & 1 \end{bmatrix}$$

9) Determine window to viewport transformation for the following dimension of windows and viewpoint.

		window	viewpoint
lower	left corner	(5, 10)	(8, 2)
upper	right corner	(15, 20)	(12, 18)



Soln:

$$S_x = \frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}}$$

$$= \frac{12-8}{15-5}$$

$$= 0.4$$

$$S_y = \frac{y_{vmax} - y_{vmin}}{y_{wmax} - y_{wmin}}$$

$$= 0.6$$

$$T = T(8, 12) \times S(0.4, 0.6) \times T(-15, -10)$$

$$\begin{bmatrix} 1 & 0 & 8 \\ 0 & 1 & 12 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.4 & 0 & 0 \\ 0 & 0.6 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & -5 \\ 0 & 1 & -10 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 8 \\ 0 & 1 & 12 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0.4 & 0 & -2 \\ 0 & 0.8 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

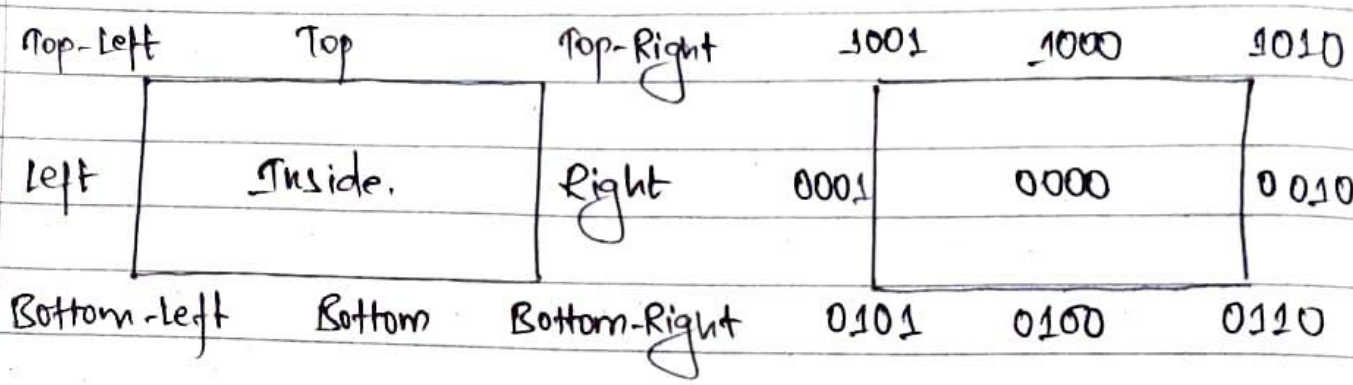
Clipping (Line Clipping)

Cohen-Sutherland Line Clipping (Solve Numerical or Explain 2Q)

Algorithm Steps:

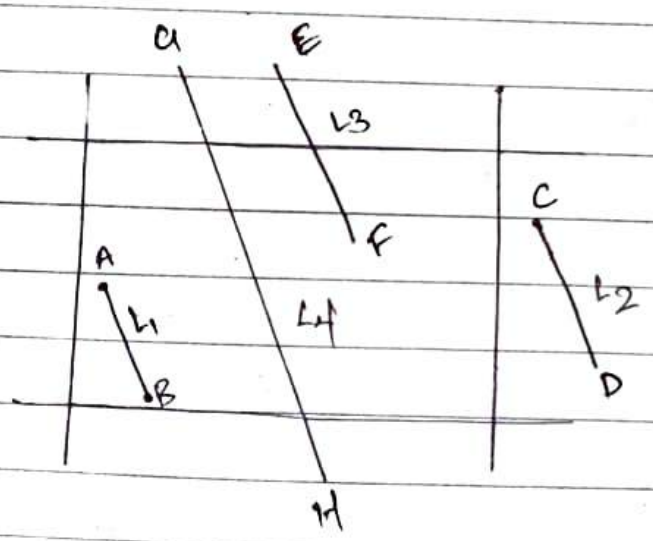
1) Assign a region code for each endpoints.
2) If both endpoints have a region code 0000 ---> trivially accept these line.
3) Else, perform the logical AND operation for both region codes.
   - 3.1. if the result is NOT 0000 → trivially reject the line.
   - 3.2. else (i.e result = 0000, need clipping)
     - 3.2.1. Choose an endpoint of the line that is outside the window.
     - 3.2.2. Find the intersection point at the window boundary (base on region code)
     - 3.2.3 Replace endpoint with the intersection point and update the region code.
     - 3.2.4 Repeat step 2 until we find a clipped line either trivially accepted or trivially rejected.
4) Repeat step 1 for other lines.

1) Cohen - Sutherland Line clipping



LRBT

Region Codes	T	B	R	L
Bit position	1	2	3	4



* Line segment L_1, AB
Region code of A; 0000
" " " B: 0000
Trivially accept

* Line segment L_2, CD
Region code of C: 0010
" " D: 0010
C & D : 0010
Trivially reject.

* Line segment I3 , EF
Region code of E : 1000
" " " F : 0000
E && F : 0000 (Non-trivial case)

start from the point outside the window boundary & find intersection point with window boundary.

Line segment E F is reduced to E'F.

Line segment E'F
Region code of E' = 0000
" " F : 0000
Trivially accept.

* Line segment GH
Region code of G : 1000
" " H : 0100
G && H : 0000 (Non-trivial case)
Start from the point outside the window boundary & find intersection point with window boundary.

Line segment GH is reduced to G'H.
Line segment G'H
Region code of G' : 0000
" " " H : 0100
Trivially accept.

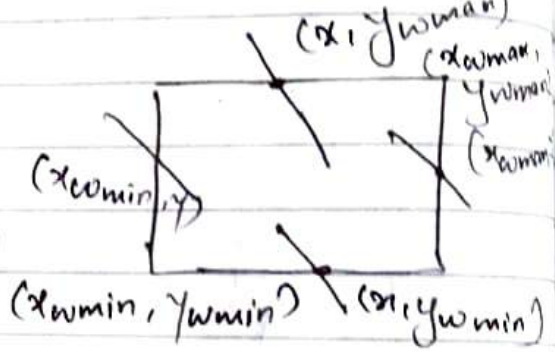
Line segment GH is reduced to GH'

Line segment GH'

Region code of G: 0000
" " H': 0000

Trivally accept.

Cohen-Sutherland Line clipping



Intersection calculations:

Intersection with vertical boundary

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

where,

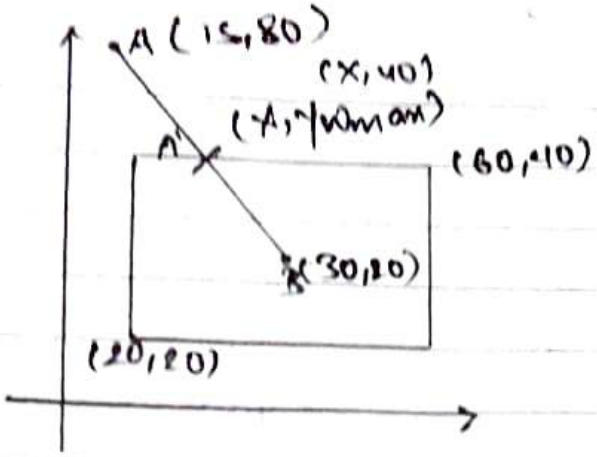
$x = x_{wmin}$ or x_{wmax}

Intersection with horizontal boundary

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

where,

$y = y_{wmin}$ or y_{wmax}



clip a line having end points A(15,80) and B(30,20) against the window having lower left corner (20,20) & upper right corner (60,40).

Line segment L_1, AB
Region code of A : 1000
Region code of B : 0000
A & B : 0000 (Non-trivial case)

start from the point outside the window boundary & find intersection point with window boundary.

Line segment AB is reduced to A'B.
Region code of A' = 0000
Region code of B = 0000
Trivially accept

Now,

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

(AB) $$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{20 - 80}{80 - 15} = \frac{-60}{15} = -4.$$

$40 - 20 = -4(x - 30)$

$20 = -4x + 120$

$4x = 100$

$x = \frac{100}{4}$

$\therefore x = 25$

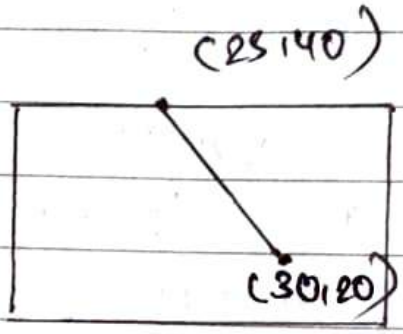


fig: After clipping

Sutherland Hodgeman Polygon Cupping Algorithm

We can clip a polygon by processing the polygon boundary as a whole against each window edge. This could be accomplished by processing all polygon vertices against each ehij clip rectangle boundary in turn. Beginning with initial set of vertices (polygon) we clip the polygon in this order left clip → right clip → bottom clip → top clip where new set of vertices are generated and passed to next window boundary clipper.

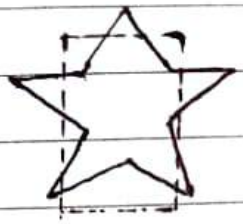


fig: Original polygon

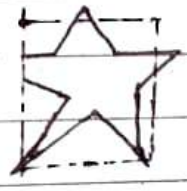


fig: clip left

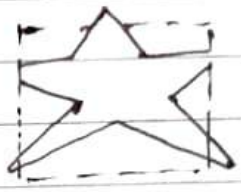


fig: clip right

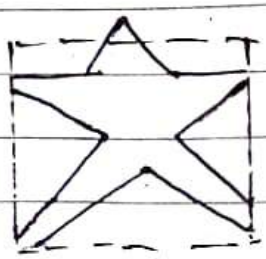
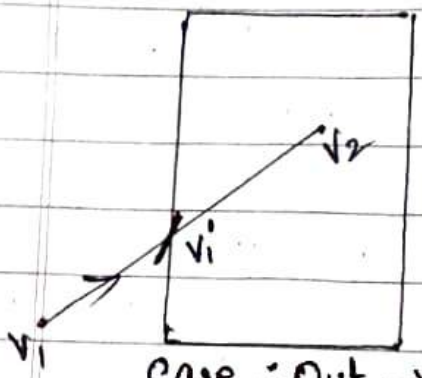


fig: bottom clip

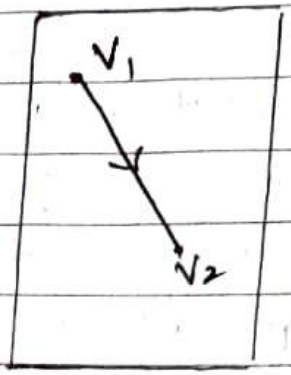
fig: Top clip

There are four cases w.r.t to polygon edge and window boundary



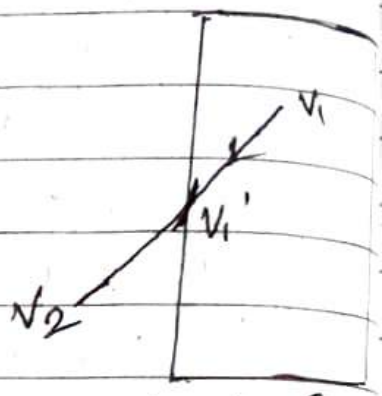
case : Out → IN

Save: V_1' & V_2



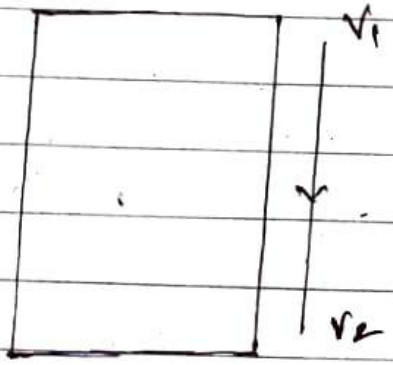
case: IN → IN

save : V_2



case : IN → O

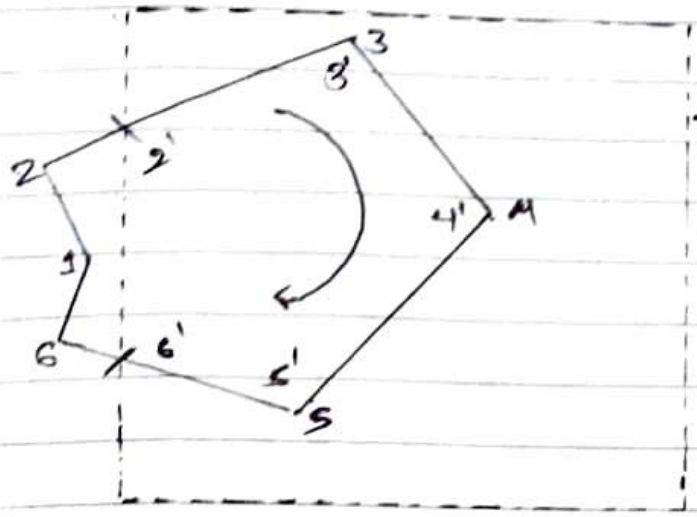
save : V_1'



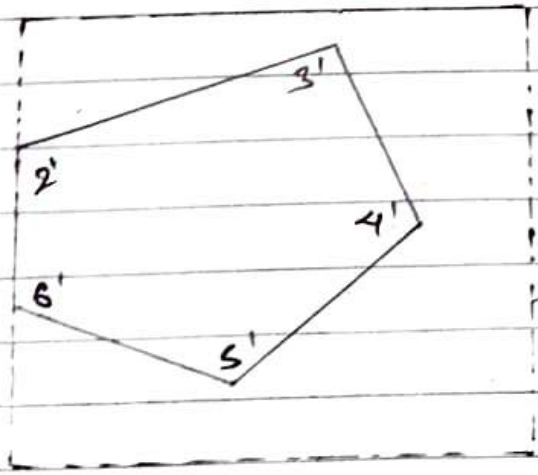
case: OUT → OUT

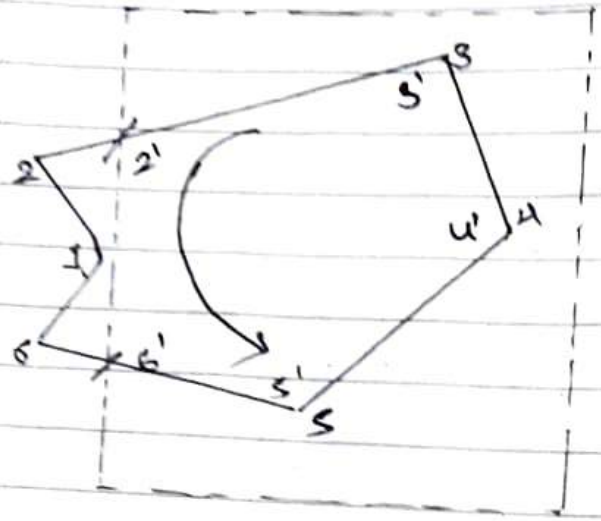
Save: None.

1.



Edge ~~case~~	Case	Output Vertex list
1 - 2	Out → Out	$\emptyset$
2 - 3	Out → In	{2', 3'}
3 - 4	In → In	{2', 3', 4'}
4 - 5	In → In	{2', 3', 4', 5'}
5 - 6	In → Out	{2', 3', 4', 5', 6'}
6 - 1	Out → Out	{2', 3', 4', 5', 6'}

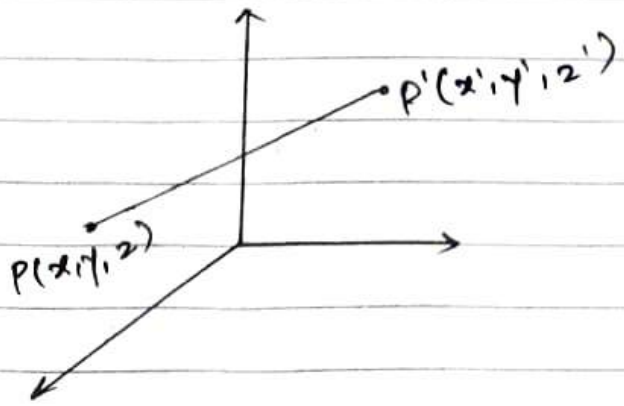




Edge	Case	Output Vertex List
~~1 — 2 — 1~~	~~Out — Out~~	$\emptyset$
1. 1 - 6	Out - Out	$\emptyset$
2. 6 - 5	Out - In	$\{6', 5'\}$
3. 5 - 4	In - In	$\{6', 5', 4'\}$
4. 4 - 3	In - In	$\{6', 5', 4', 3'\}$
5. 3 - 2	In - Out	$\{6', 5', 4', 3', 2'\}$
6. 2 - 1	Out - Out	$\{6', 5', 4', 3', 2'\}$

Unit-5.

3D-Graphics



Translation:

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

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

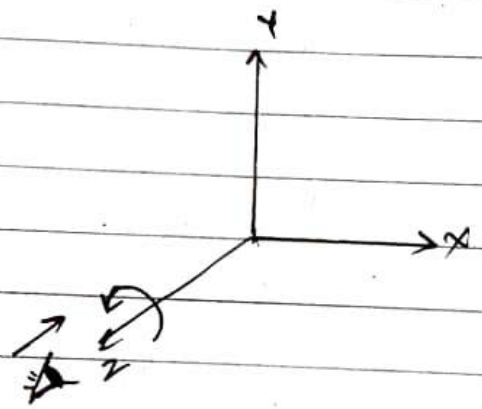
$$P' = T \cdot P$$

Rotation (about co-ordinate axes)

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

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

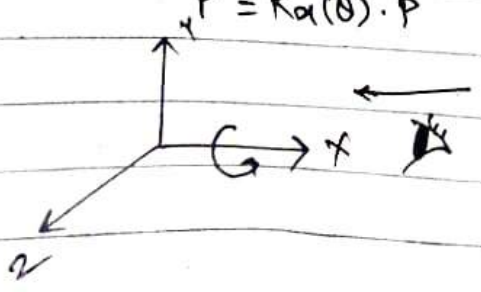
$$P' = R_z(\theta) \cdot P$$



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

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

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



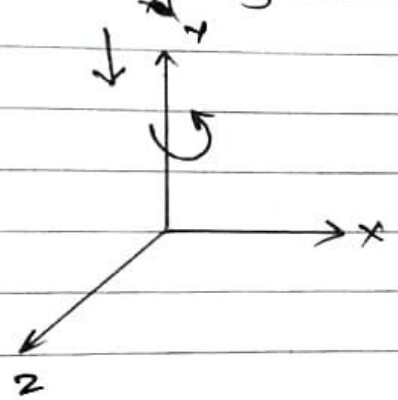
- $z' = z\cos\theta - x\sin\theta$

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

$y' = y$

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

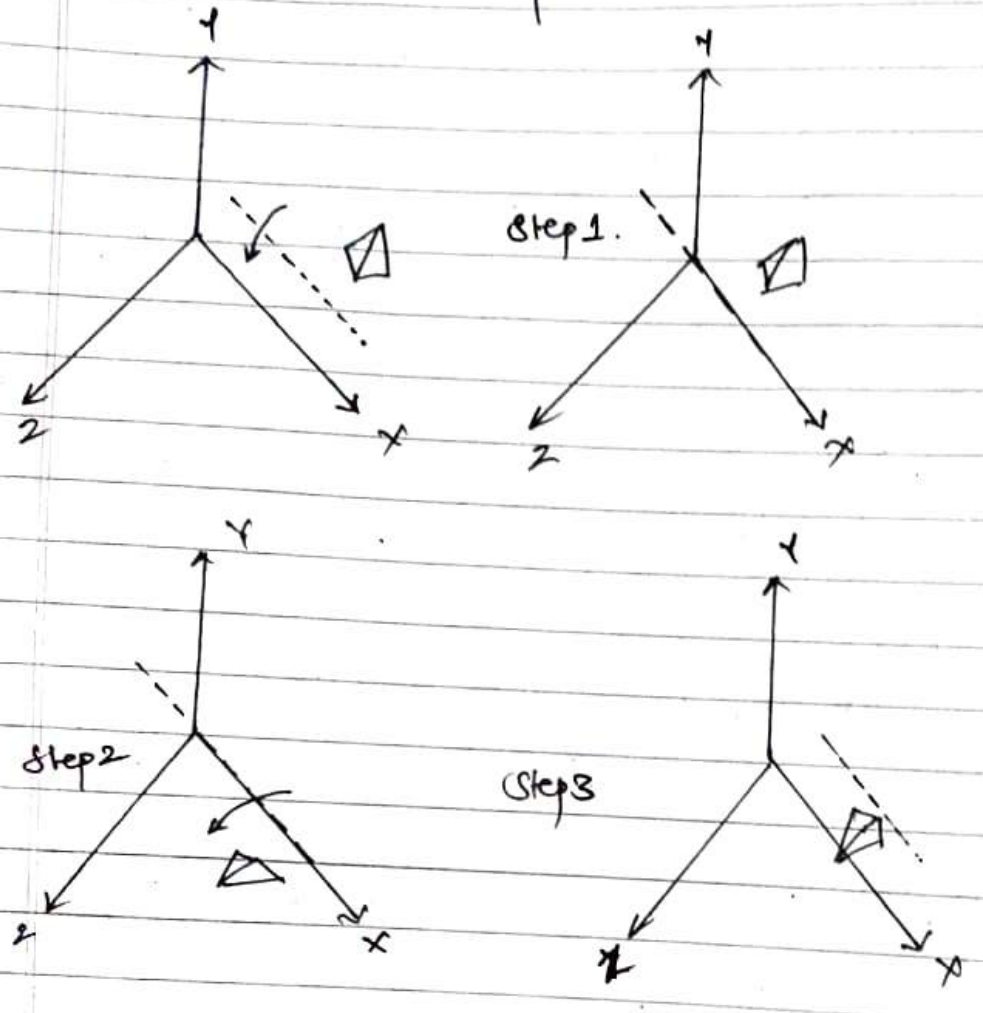
$P' = R_y(\theta) \cdot P$



Remember cyclic order

$x \to y \to z \to x$

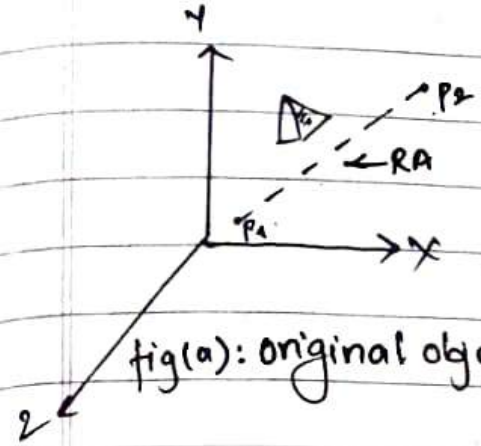
Rotation about axis parallel to co-ordinate axis.



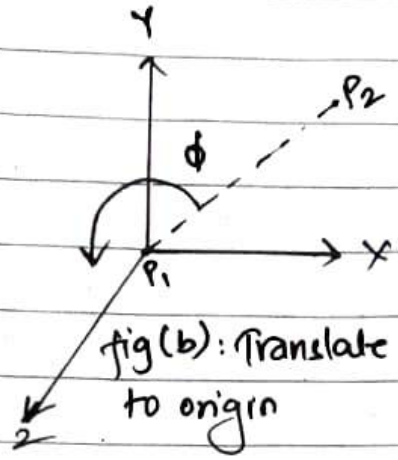
Steps

- Translate the object so as to coincide rotation axis to parallel co-ordinate axis
- Perform the rotation about the axis
- Translate back the object so as to move rotation axis to original position.

General 3D rotation



fig(a): original object



fig(b): Translate RA to origin

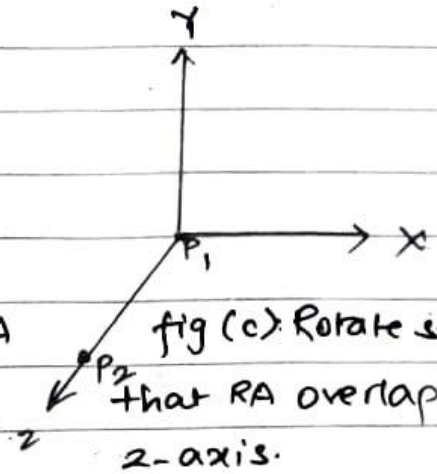
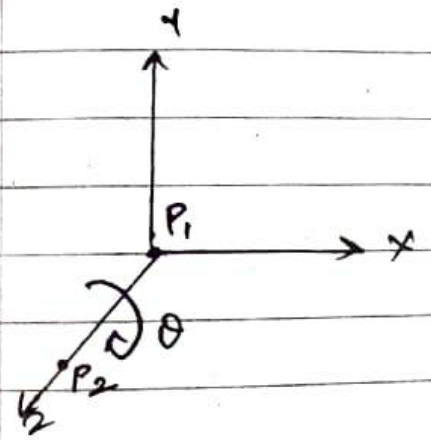
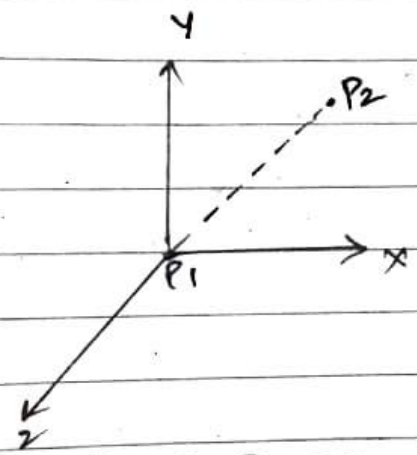


fig (c): Rotate such that RA overlap with z-axis.



fig(d): Rotation about z-axis



fig(e): Inverse Rotation.

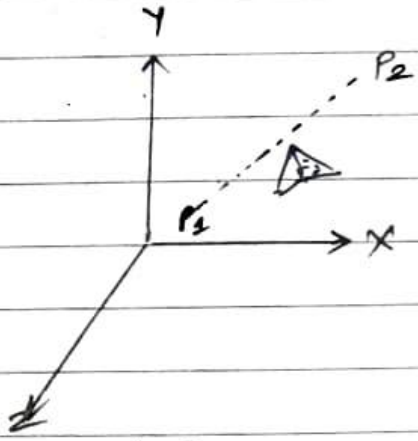


fig (f): Inverse translation.

- 3-D Scaling

Scaling about origin

$x' = x * S_x$

$y' = y * S_y$

$z' = z * S_z$

$$s(S_x, S_y, S_z) = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Fixed point scaling

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

$$\left[T(x_f, y_f, z_f) \times S(S_x, S_y, S_z) \times T(-x_f, -y_f, -z_f) \right]$$

- 3-D Reflection

* Performed relative to a reflection axis or reflection plane.
* Axis reflection → equivalent to 180 degree relation about the axis in 3-D space.
* Reflection about a plane converts right-handed co-ordinate system to left handed co-ordinate system and vice versa.

- Reflection in xy plane

$x' = x$

$y' = y$

$z' = -z$

- Matrix is RF_z:

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

- Reflection in yz plane

$x' = -x$

$y' = y$

$z' = z$

- Matrix is

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

- Reflection in zx plane

$x' = x$

$y' = -y$

$z' = z$

- Matrix is

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

- 3-D Shear

- 2-axis shear

$x' = x + a \cdot z$

$y' = y + bz$

$z' = z$

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

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

- x-axis shear

$$x' = x$$
$$y' = y + cx$$
$$z' = z + dx$$

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

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

- y-axis shear

$$x' = x + ay$$
$$y' = y$$
$$z' = z + by$$

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

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

Projection

This is a techuique to map a 3D object on to a 2D screen. Projection transform points in a coordinate-system of dimension 'n' into points in a coordinate system of dimension less than 'n'.

* Types of projection:

1) Parallel projection.
   - 1.1) Orthogonal projection
   - 1.2) Oblique

2) Perspective Projection.
   - 2.1) One point perspective projection
   - 2.2) Two " " "
   - 2.3) Three " " "

Parallel projection.

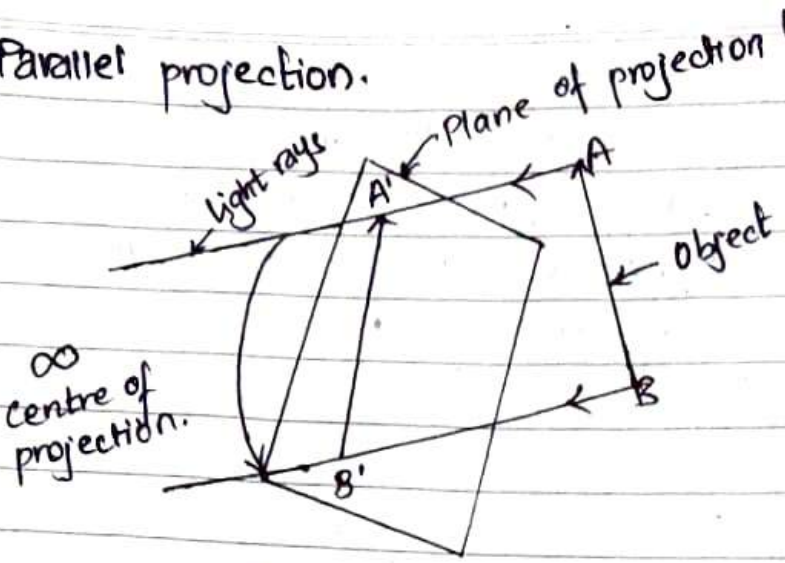


fig: parallel projection.

If the distance between COP and POP is ∞, then projection is called parallel projection i.e. The rays from COP are paralleled. ∞ it maintains relative proportion of the object. In this projection we extend parallel lines from each vertex to the view plane.

Equation of line 2D is

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

In parametric form,

$$X = X_1 + (X_2 - X_1)u$$
$$Y = Y_1 + (Y_2 - Y_1)u \quad \text{where } u \in [0,1]$$

Similarly in 3D

$$X = X_1 + (X_2 - X_1)u$$
$$Y = Y_1 + (Y_2 - Y_1)u$$
$$Z = Z_1 + (Z_2 - Z_1)u$$

Let the direction of projection is given by $[x_p, y_p, z_p]$ and the image is to be projected on X-Y plane. If

we have point on the object at (x_1, y_1, z_1) and the projected point (x_2, y_2) then eqn of the line is

$$x = x_1 + x_p u$$
$$y = y_1 + y_p u$$
$$z = z_1 + z_p u$$

Since $z = 0$

$$0 = z_1 + z_p u$$
$$u = -z_1 / z_p$$
$$x_2 = x_1 - z_1 (x_p / z_p)$$
$$y_2 = y_1 - z_1 (y_p / z_p)$$

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -x_p/z_p & 0 \\ 0 & 1 & -y_p/z_p & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}$$

Perspective projection:

If the distance POP and COP is finite and converges to a single point C vanishing point) then the project is called perspective projection i.e. it transforms point of 3D object along project line that meet a vanishing point. The size of image will differ depending on the distance of object & the view plane.

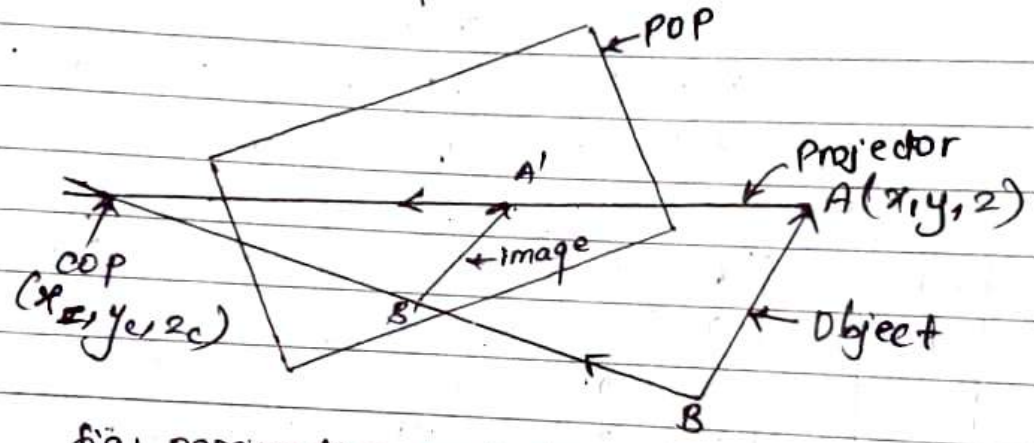


fig: perspective projection.

If the COP at $[x_c, y_c, z_c]$ and the point on the object is $[x, y, z]$ then projection ray will be joining these points. Equation of the line is

$$x = x_c + (x_1 - x_c)u$$
$$y = y_c + (y_1 - y_c)u$$
$$z = z_c + (z_1 - z_c)u$$

The image is projected on xy plane therefore $z = 0$

$$0 = z_c + (z_1 - z_c)u$$
$$u = \frac{-z_c}{(z_1 - z_c)}$$

$$x_2 = (x_c z_1 - x_1 z_c) / (z_1 - z_c)$$
$$y_2 = (y_c z_1 - y_1 z_c) / (z_1 - z_c)$$

$$[x_2 \quad y_2 \quad z_2] = [(x_c z_1 - x_1 z_c)/(z_1 - z_c), (y_c z_1 - y_1 z_c)/(z_1 - z_c), 0]$$

In homogeneous form,

$$[x_2 \quad y_2 \quad z_2 \quad 1] = [(x_c z_1 - x_1 z_c), (y_c z_1 - y_1 z_c), 0, (z_1 - z_c)]$$

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} -z_c & 0 & x_c & 0 \\ 0 & -z_c & y_c & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -z_c \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}$$

~~Visual~~

5.3 Visible Surface Detection (Hidden Surface removal)

Two approaches:

- Depends upon object definition or projected image
- Object-space Method (OSM)
  - > compares objects and parts of objects to each other within the scene.
- Image-space Method (ISM)
  - > visibility is decided point by point at each pixel position on projected plane.

Back face Detection Method

- The test could be simplified by considering normal vector $N = (A, B, C)$ to polygon surface and vector V in viewing direction as..

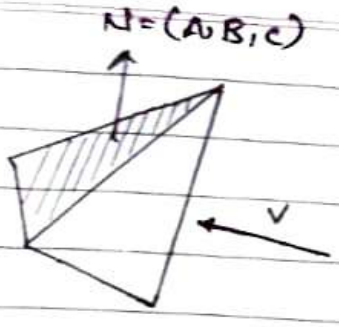
  the polygon is back face if: $V \cdot N > 0$

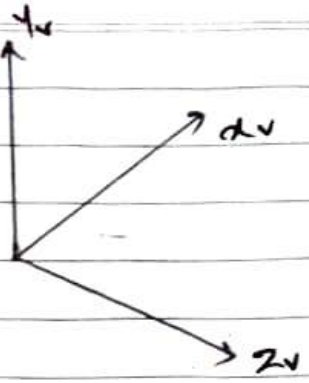
- If object description is converted to viewing co-ordinates then, V is along Z_v axis i.e. $V = (0, 0, V_z)$ then,

  $V \cdot N = V_z C$

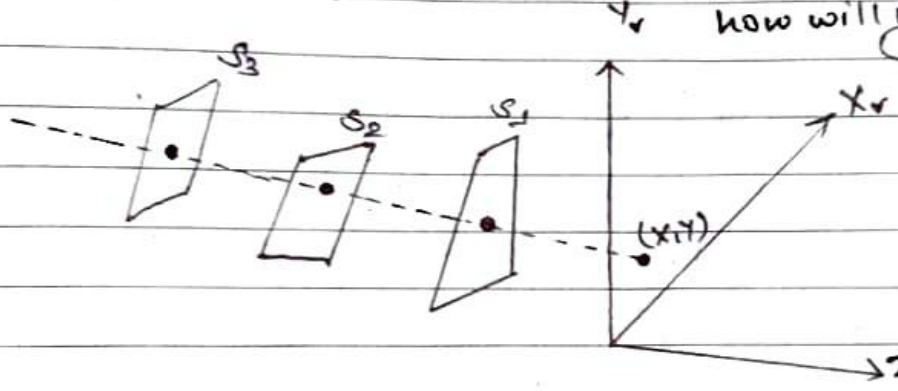
  ie. $V \cdot N > 0 \rightarrow$ @ $V_z C$

  So we need to consider only sign of C
- For right handed viewing system with viewing direction along negative Z_v axis, $C < 0 \rightarrow$ back face.





Depth Buffer (Z-Buffer) Method: (What are its limitation & how will you correct it)



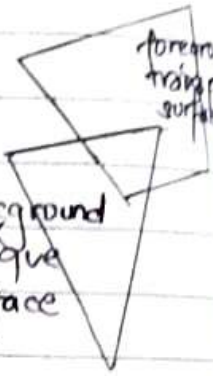
to with A buffer method

- Image-space method
- Surface depth is compared at each pixel position on the projection plane
- Usually applied to scene containing only polygon surfaces, but possible to apply for non-planar surfaces.
- For each pixel position (x, y) on projection plane, object depths can be compared by comparing z-values as each (x, y, z) position on a polygon surface corresponds to the orthographic projection point (x, y) on projection plane.
- Two buffers required
  - Depth buffer: to store depth value for each position
  - Refresh buffer: to store intensity values of each position

- Drawback: Deals only with opaque surface but not with transparent surface.

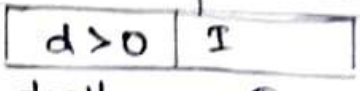
A - Buffer Method.

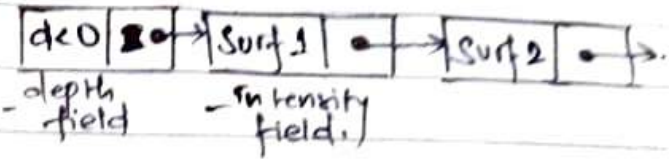
- Extension of depth-buffer Method.
- Antialised area-average, accumulation-buffer
- The depth buffer is expanded so that: each position in the buffer can reference a linked list of surfaces - thus enabling more than one surface intensity consideration.



- The object boundary could be antialised.
- Each pixel position in the A-buffer has two fields
  - Depth field → stores a positive or negative real number
    - Positive → single surface contributes to pixel intensity
    - Negative → multiple surfaces contribute to pixel intensity.
  - Intensity Field → Stores surface-intensity information or a pointer value
    - Surface intensity if single surface → stores the RGB components of the surface color at that point and percent of pixel coverage
    - Pointer value if multiple surfaces.

- In case of surface linked list, the data for each surface in the linked list includes.
  - RGB intensity
  - Opacity parameter
  - Depth
  - Percent of area coverage
  - Surface identifier
  - other surface-rendering parameters
  - Pointer to next surface.



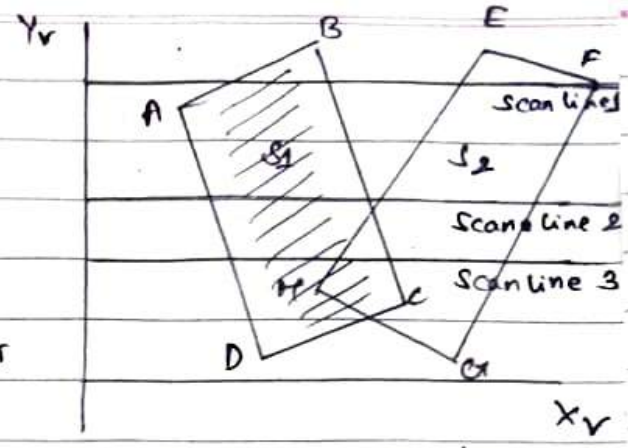


- Scanlines are processed to determine surface overlaps of pixels across the individual scanlines.
- Surfaces are subdivided into a polygon mesh and clipped against the pixel boundaries.
- The opacity factors and percent of surface overlaps are used to determine the pixel intensity as an average of the contribution from the overlapping surfaces.

Scan-Line Method.

- Extension of scanline algorithm for filling polygon interiors
- Instead of filling just one surface, we deal with multiple surfaces
- Construct edge-tables and polygon tables
  - Edge table contains → end points of each edge in the scene, inverse slope of each line and pointers to the polygon table to identify its corresponding surface.
  - Polygon table contains → plan equation coefficients, surface intensity, pointers to the edge table (optional)
- Define flags for each surface to indicate whether a position is inside or outside the surface.
  - At leftmost boundary of surface flag == on and at rightmost boundary flag == off
- For scan Line 1
  - The active edge list contains edges AB, BC, EH, FG
  - Between edges AB and BC, only flags for S_1 == on and between edges EH and FG, only flags for S_2 == on
    - no depth calculation needed and corresponding surface intensities are entered in refresh buffer.

- For Scan Line 2

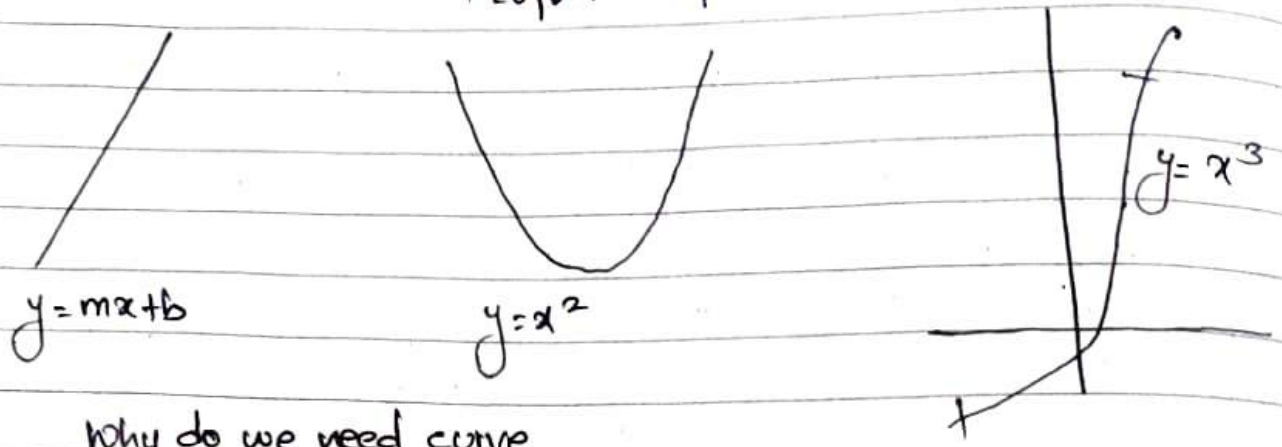


- The active edge list contains edges AD, EH, BC and FG
- Between edges AD and EH, only the flag for surface S1 == On
- Between edges EH and BC flags for both surfaces == On
  - Depth calculation (using plan coefficients) is needed.
  - In this example depth for S1 is less than for S2, so intensities for surface S1 are loaded into the refresh buffer until boundary BC is encountered
- Between edges BC and FG flag for S1 == off and flag for S2 == On
  - Intensities for S2 are loaded on refresh buffer.

- For scan line 3
  - Same coherent property as scan line 2 as noticed from active list, so no depth calculation needed between edges BC and EH.

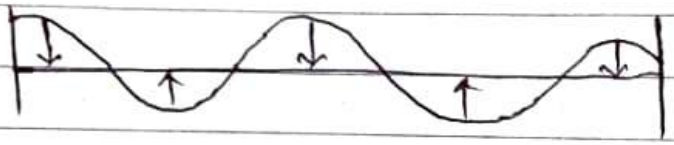
Spline Representation.

↳ Loftman Spline.



Why do we need curve

> Trace the path of animation
> Represent complex objects.



Spline representation:

In industries like ship building, automobile, and aircraft, it is a common practice to layout the model's shapes curve in full or near full on a large drafting floor. This is done using a long narrow strip of wood or plastic known as loftman spline several small lead weights are distributed along the length of such physical splines. By varying the number and positions of lead different curve can be ~~gen~~ generated.

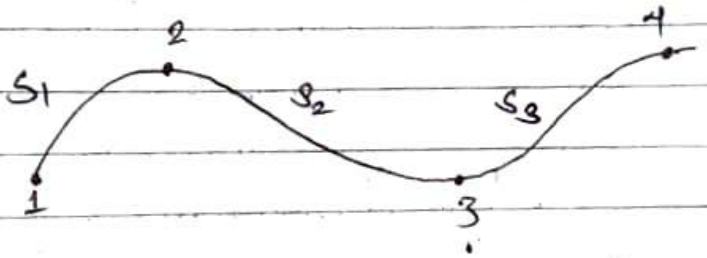
Types of spline

1) Piecewise cublic spline
2) Bezier Spline

3) B-Spline.

Types of spline
1) Interpolated spline
2) Approximated spline

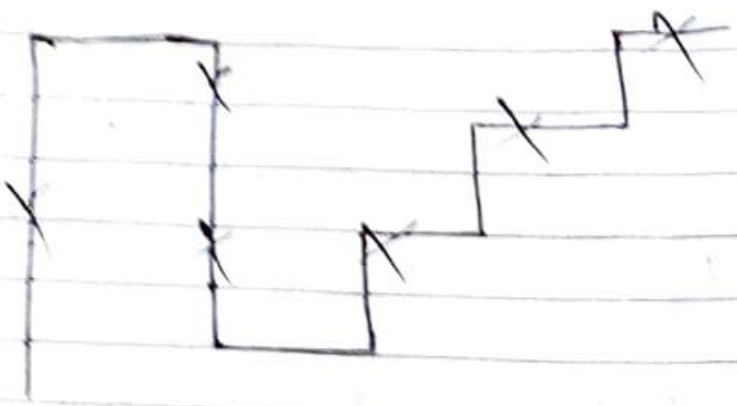
1) Interpolated spline: We Specify spline curve by giving a set of co-ordinate positions, called 'control points' which indicate the general shape of the curve. These control points are then fitted with piecewise continuous parametric polynomial function.



Three polynomial curve sections S_1, S_2 & S_3 interpolated between 1-2, 2-3, & 3-4

2) Approximated Spline
When polynomial are so fitted that without necessarily passing through any of the control points, it approximated the shape of the control polygon, the resulting curve is said to have approximated the set of control points.

→



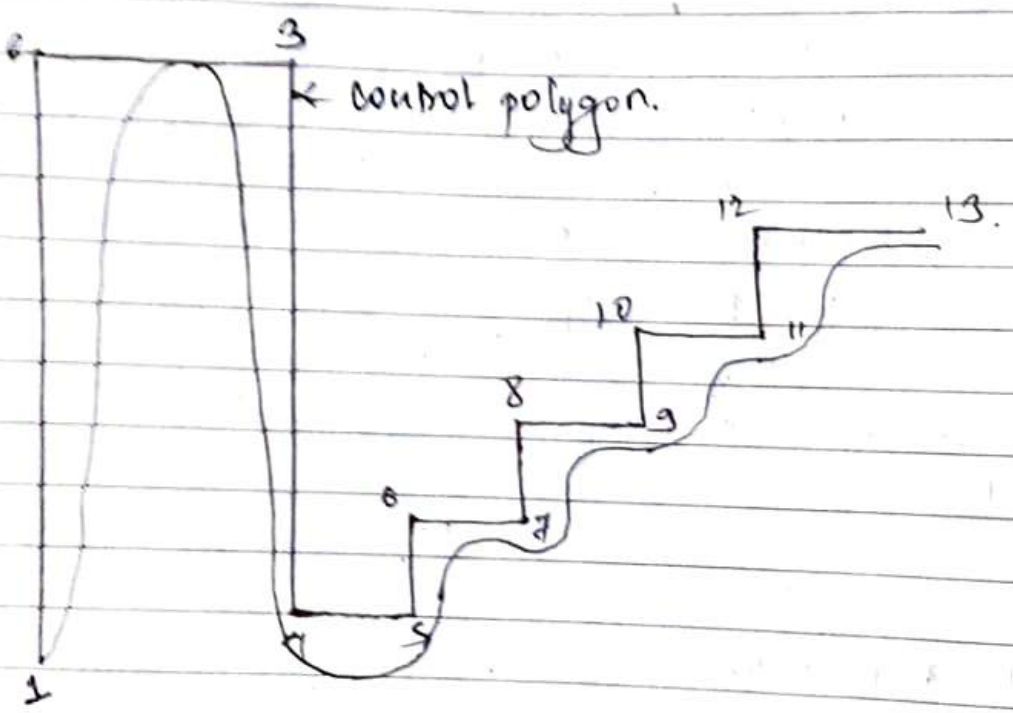


fig: Approximated Spline

Cubic Bezier Curve:

Unlike the piecewise spline curve, a Bezier spline curve can be fitted to any number of control points. However the degree of Bezier curve segment is determined by the number of control points to be fitted with that curve segment

Given a set of n+1 control points $P_0, P_1, P_2 \ldots P_n$ a parametric Bezier curve segment that will fit to those points is mathematically defined by

$$P(u) = \sum_{i=0}^{n} P_i B_{i,n}(u) \quad \text{where } 0 \le u \le 1$$

where,

$B_{i,n}(u)$ is Bezier blending function, also known as Bernstein Basis fn defined as

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

and $C(n,i)$ is the Binomial coefficient

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

Expanding the equation, we get

$$P(u) = \sum_{i=0}^{3} P_i B_{i,3}(u) \quad \text{where } 0 \le u \le 1$$

For Bezier curve of order 3 we need to have four control points

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

$$B_{0,3}(u) = \left\{ \frac{3!}{(3-0)!\,0!} \right\} u^0 (1-u)^{3-0} = (1-u)^3$$

$$B_{1,3}(u) = \left\{ \frac{3!}{(3-1)!\,1!} \right\} u^1 (1-u)^{3-1} = 3u(1-u)^2$$

$$B_{2,3}(u) = \left\{ \frac{3!}{(3-2)!\,2!} \right\} u^2 (1-u)^{3-2} = 3u^2(1-u)$$

$$B_{3,3}(u) = \left\{ \frac{3!}{(3-3)!\,3!} \right\} u^3 \times (1-u)^0 = u^3$$

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

$$P(u) = (-u^3 + 3u^2 - 3u + 1) P_0 + (3u^3 - 6u^2 + 3u) P_1 + (-3u^3 + 3u^2) P_2 + u^3 P_3$$

This expression can be translated into matrix form i.e.

$$P(u) = [F][B]$$

$$P(u) = [T][M][B]$$

Here, the basis fn $[F] = [B_{0,3}(u)\ B_{1,3}(u)\ B_{2,3}(u)\ B_{3,3}(u)]$

• Bezier Geometry matrix.

$$[B] = \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

$$[T] = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}$$

- Bezier Basis Matrix $[M] = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$

Such that $[F] = [T][M]$

Therefore a cubic Bezier curve controlled by the points P_0, P_1, P_2, P_3 is

$$P(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

Bezier surface & its properties.

A bezier surface is a mathematical representation of a smooth surface that is defined by a set of control points. The control points determine the shape and curvature of the surface, and the smoothness of the surface is determined by the degree of polynomial equation used to define it.

These are species of mathematical spline used in computer graphics and finite element modeling

Properties:

- A bezier surface will transform in the same way as its control points under all linear transformations and translations:

- ~~An~~ A bezier surface will lie completely within the convex hull of its control points and therefore also completely within the bounding box of its control points in any given cartesian co-ordinate system.

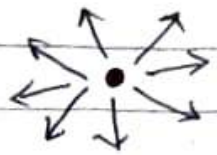
- The points in the patch corresponding to the corners of the deformed unit square coincide with four of the control points.

- However, a Bezier surface doesnot generally pass through its control points.

Chapter-6 Light Color Shading or Illumination Models & Surface Rendering Methods.

Light Sources.

- Luminous objects
  - Light Emitting sources
  - Reflecting sources



point

Distributed

- Point source: Simplest model of light emitter where light is emitted radially

- Distributed Light Source: modeled as accumulated illumination effects of the points over the surface of the source.

Types of reflection

1) Diffuse reflection (sandy / dull surface)

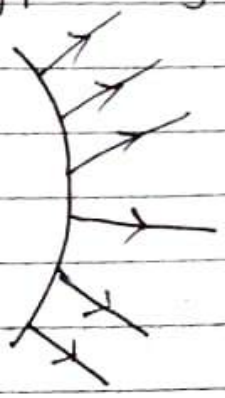


fig: Diffuse reflection.

2) Specular reflection (glossy / shiny)

fig: Specular reflection.

Basic Illumination Model

- ~~Ambient Light~~ → Bo
- Diffuse Reflection
  - Reflected Light intensity are constant over each surface in a scene independent of viewing direction → ideal diffuse reflectors

- the diffuse-reflection coefficient (diffuse reflectivity)
  - sets the fraction of light intensity that is reflected from each surface
  - k_d is a the function of surface color, but for our purpose we assume k_d to be constant.

- Diffuse reflection intensity when scene illuminated only with ambient light:

$$I_{ambdiff} = k_d I_a$$

- Lambertian reflectors
  - follow lambart's cosine law → radiant energy from a small surface area dA is proportional to ~~the~~ cosine of angle θ between surface normal and incident light direction.
    - For a point source with intensity I_L, the diffuse reflection intensity is

$$I_{L,diff} = k_d I_L \cos\theta$$

$$I_{L,diff} = k_d I_L (N \cdot L)$$

where N and L are unit vectors

- Diffuse reflection for ambient light source + a point light source.
  - $I_{diff} = K_d I_a + K_d I_L (N \cdot L)$
    - I_a: amb
    - I_L: point light

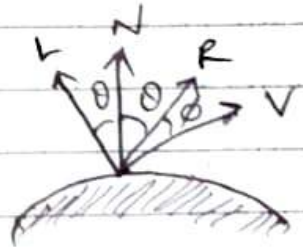
Fig:
- ~~sphere~~ fig

Empirical formula → no theoretical derivation.

Specular Reflection or Phong Model.

- Bright spot seen at an illuminated shiny surface when viewed at certain direction
  - Polished metal surface, person's forehead, apple etc. exhibit specular reflection.

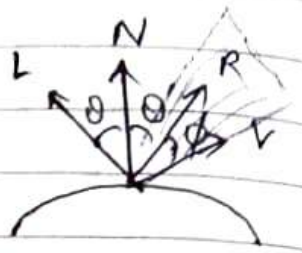
- In fact an image of light source
- Result of total or near total reflection of incident light in a concentrated region around the specular reflection angle θ.



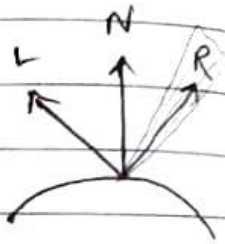
- Fig:
  - L → unit vector pointing to light source
  - N → unit surface normal vector
  - R → unit vector in direction of specular reflection
  - V → unit vector pointing viewer

- Ideal reflector exhibit specular reflection in the direction of R only (i.e. $\phi = 0$) but for non-ideal case specular reflection is seen over finite range of viewing positions.

- Intensity of specular reflection: proportional to $\cos^{n_s}\phi$
  - $n_s \rightarrow$ specular reflection parameter (depends on surface)
  - ϕ ranges from 0 to 90° (i.e $\cos\phi$ varies from 0 to 1)

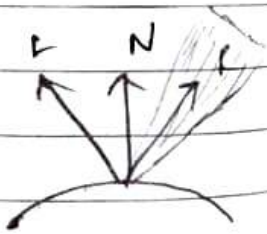


- Intensity of specular reflection depends on:
  - Material properties of surface
  - Angle of incidence θ
  - other factors such as polarization and color of the incident light



- Monochromatic specular intensity variations can be approximated using specular-reflection coefficient, $w(\theta)$ for each surface

$$I_{spec} = w(\theta) I_L \cos^{n_s}\phi$$



- At $\theta = 90°$, $w(\theta) = 1 \rightarrow$ all incident light is reflected.

- Simplied form : assume $w(\theta) = k_s$ = constant.

$$I_{spec} = k_s I_L (V \cdot R)^{n_s}$$

Polygon Rendering Methods

- Illumination model is applied to fill the interior of polygons
- curved surfaces are approximated with polygon meshes
  - But polyhedra that are not curved surfaces are also modeled with polygon meshes.
- Two ways of polygon surface rendering
  - single intensity for all points in a polygon.
  - Interpolation of intensities for each point in a polygon
- Methods:
  - Constant Intensity Shading / Flat shading.
  - Gouraud shading / Intensity Interpolation
  - Phong Shading / Normal Vector Interpolation.

for polygon rendering.

- Constant Intensity Shading

★ Flat shading:
- Each polygon shaded with single intensity calculated for the polygon.

□ Useful for displaying general appearance of a curved surface

□ Accurate rendering conditions:
- Object is a polyhedron and not an curved surface approximation.
- All light sources should be sufficiently '

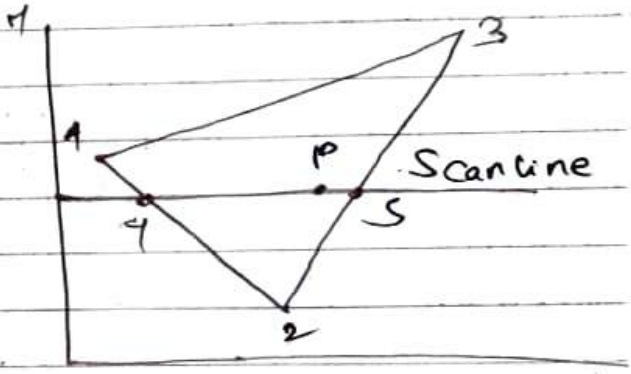
□ Drawback: Intensity discontinuity at the edges of polygons!

★ Gouraud shading:
- Calculation steps:
  - Determine the average unit normal vector at each polygon vertex.
  - Calculate each of the vertex intensities by applying an illumination model.
  - Linearly interpolate the vertex intensities over the polygon surface.
- Intensity discontinuity at the edges of polygons is eliminated

- Drawback:
  - Mach bands: bright and dark intensity streaks caused by linear interpolation of intensities
    - could be reduced by dividing the surface into large number of polygons or by using other methods, such as Phong shading.

- Average Unit Normal: obtained by averaging the surface normals of all polygons sharing the vertex.

$$N_v = \frac{\sum_{k=1}^{N} N_k}{\left| \sum_{k=1}^{N} N_k \right|}$$



- Intensity interpolation:
  - Along the polygon edges are obtained by interpolating intensities at the edge ends.

$$I_4 = \frac{y_4 - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y_4}{y_1 - y_2} I_2$$

[Recursive calculation along the edge

$$I' = I + \frac{I_2 - I_1}{y_1 - y_2}$$

]

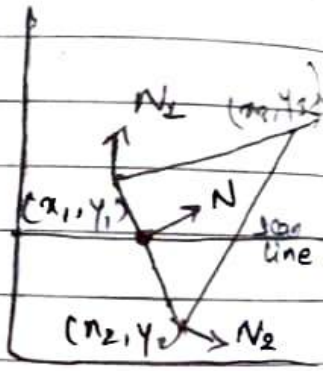
  - Along the scan line between the polygon edges are obtained by interpolating intensities at the intersection of scan line and polygon edges.

$$I_p = \frac{x_5 - x_p}{x_5 - x_4} I_4 + \frac{x_p - x_4}{x_5 - x_4} I_5$$ Recursive calculation along the scan line!!

Phong shading:

- More accurate method for rendering.
- Fundamental: Interpolate normal vectors and apply illumination model to each surface point.

- Calculation steps:
  - Determine average unit normal vectors at each polygon vertex.
  - Linearly interpolate the vertex normals over the surface of the polygon
  - Apply an illumination model along each scan line to calculate projected pixel intensities for the surface points.



$$N = \frac{y - y_2}{y_1 - y_2} N_1 + \frac{y_1 - y}{y_1 - y_2} N_2$$

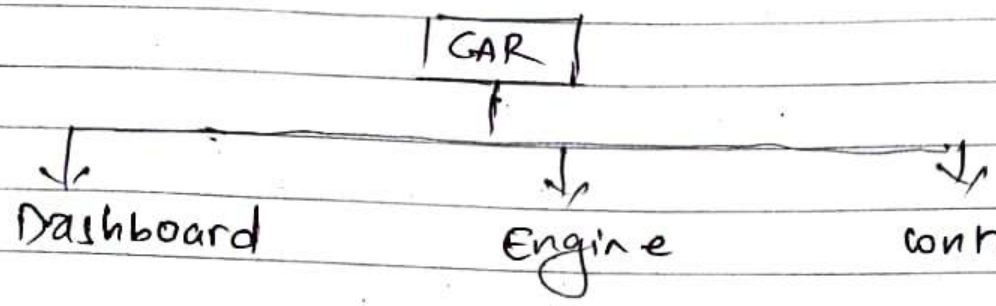
- Trade-off: requires considerably more calculations.

1. Graphical Language.

Need for machine independent graphical language

- PHIGS: Programmer Hierarchical Interactive graphics System.
- GKS: Graphics Kernel System.

(Structure)



File format

- RASTER
  - → JPEG
  - → BMP
  - → JPG
- VECTOR
  - ↳ PDF
  - ↳ AI

Audio
- .mp3
- .wav
- .ogg

Video
- .mp4
- .WEMP.

Unit-8.

1. OpenGL, use, block-diagram.

Why OpenGL?

- Device independence
- Platform independence
  - SGI Irix, Linux, Windows
- Abstractions (GL, GLU, GLUT ← library)
- Open source
- Hardware-independent software interface
- Support of client-server protocol
- Other APIs
  - OpenInventor (Object-oriented toolkit)
  - DirectX (Microsoft), Java3D (Sun)

- OpenGL is a software interface that allows the programmer to create 3D and 2D graphics images. OpenGL is both a standard API and implementation of that API. You can call the functions that comprise OpenGL from a program you write and expect to see the same results no matter where your program is running.

OpenGL is independent of the hardware, operating and windowing systems in use. The fact that is windowing-system independent, makes it portable. OpenGL program must interface with the windowing system of the platform where the graphics are to be displayed. Therefore, a number of windowing toolkits have been developed for use openGL

OpenGL functions is a client/server environment. That is, the application program producing the graphics may run on a machine other than the one on which the graphics are displayed. The server part of OpenGL, which runs on the workstation where the graphics are displayed, can access whatever physical graphics device or framebuffer is available on that machine.

OpenGL Operation