Unit - 1 Introduction

→ Advantage of Computer Graphics and Area of Application
→ Hardware and Software for Computer Graphics: (Hard copy, Display Technologies)
→ Random Scan Display System, Video Controller, Random Scan Display processor
→ Raster Graphics
→ Scan Conversion Algorithms (Line, Circle, Ellipse)
→ Area Filling (Rectangle, Ellipse), Clipping (Lines, Circle, Ellipse), Clipping Polygons

Computer Graphics

It is the field of study which deals with pictures or images.
It synthesis pictures from mathematical or geometrical models.

Application Area of Computer Graphics

i) User Interface
ii) plotting
iii) Office-Automation and electronic publishing
iv) Computer Aided Drafting and Design
v) Scientific and Business Visualization
vi) Simulation and Modeling
vii) Entertainment
viii) Art and Commerce
ix) Cartography

Advantages of Computer Graphics

Display Technologies

i) Cathode Ray Tube (CRT) → Monochrome and Color CRT

Advantages:

→ High quality graphics provides the best way to communicate with computer.
→ It is possible to produce animation. → Can be used to control animation such as speed of total scene in view etc → provides facility of update dynamic which can be used to change shape, color and other properties of object in view. → Used to present data or information in the form of bar diagram, pie charts etc. which makes visualization better.

Display Terminology

i) Persistance – how long phosphor continues to emit light after the electro beam is removed.

ii) Refresh Rate

iii) Resolution

iv) Aspect Ratio → [Width/Height]

24 bits per pixel

Screen resolution = 1024 x 1024

Frame buffer size = ? MB.

Total Resolution = 1024 x 1024 pixel

1 pixel = 24 bit

Total bits = 1024 x 1024 x 24

Total MB required = $\frac{1024 \times 1024 \times 24}{8 \times 1024 \times 1024}$

= 3 MB

Raster Scan System Vs Vector (Random) Display System

2080-08-15

Line Drawing Algorithm
a) DDA (Digital Differential Analyzer)
b) Bresenham's Algorithm

a) DDA Algorithm :-

① Given a line with two end points (x_0, y_0) and (x_1, y_1)
② Calculate the Difference between two end-points.

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

③ Calculate the Number of steps required

```
if dx > dy
    steps = |dx|
else
    steps = |dy|
```

④ Calculate the increment in x and y coordinate.

x increment = dx / (float) steps
y increment = dy / (float) steps

⑤ Put the pixel successfully incrementing x and y coordinate accordingly to draw a line.

```
for(int v=0; v<steps; v++) {
    x = x + x increment;
    y = y + y increment;
    putpixel(x,y);
}
```

* Digitize a line with end-points (1, 1) and (8, 5) using DDA Algorithm.

Soln

Here:

Given $(x_0, y_0) = (1, 1)$ $(x_1, y_1) = (8, 5)$

$dx = x_1 - x_0 = (8-1) = 7$

$dy = y_1 - y_0 = (5-1) = 4$

Here,

$dx > dy$, So, Steps $(dx) = (7|) = 7$

$x_{increment} = dx/(float)steps = 7/7 = 1.0$

$y_{increment} = dy/(float)steps = 4/7 = 0.57$

Iteration	v	x_{new}	y_{new}	x_{new}, y_{new}
1	0	$x = 1+1 = 2$	$y = 1+0.57 = 1.57$	(2, 2)
2	1	$x = 2+1 = 3$	$y = 1.57+0.57 = 2.14$	(3, 2)
3	2	$x = 3+1 = 4$	$y = 2.14+0.57 = 2.71$	(4, 3)
4	3	$x = 4+1 = 5$	$y = 2.71+0.57 = 3.28$	(5, 3)
5	4	$x = 5+1 = 6$	$y = (3.28+0.57) = 3.85$	(6, 4)
6	5	$x = 6+1 = 7$	$y = 3.85+0.57 = 4.42$	(7, 4)
7	6	$x = 7+1 = 8$	$y = 4.42+0.57 = 4.99$	(8, 5)

∴ So, the resulting points are

(1, 1), (2, 2), (3, 2), (4, 3), (5, 3), (6, 4), (7, 4), (8, 5)

Ans:

* Digitize a line with end-points (25,20) and (15,10) using DDA algorithm.

Here,

Given $(x_0, y_0) = (25,20)$, $(x_1, y_1) = (15,10)$

$dx = x_1 - x_0 = 15 - 25 = -10$

$dy = y_1 - y_0 = 10 - 20 = -10$

Here,

dx is not > dy so, steps = $(dy) = |-10| = 10$

$x_{increment} = \frac{dx}{(float)\ steps} = \frac{-10}{10} = -1$

$y_{increment} = dy/float\ steps = -10/10 = -1$

Iteration	v	x_{new}	y_{new}	(x_{new}, y_{new})
1	0	$x = 25 - 1 = 24$	$y = 20 - 1 = 19$	(24, 19)
2	1	$x = 24 - 1 = 23$	$y = 19 - 1 = 18$	(23, 18)
3	2	$x = 23 - 1 = 22$	$y = 18 - 1 = 17$	(22, 17)
4	3	$x = 22 - 1 = 21$	$y = 17 - 1 = 16$	(21, 16)
5	4	$x = 21 - 1 = 20$	$y = 16 - 1 = 15$	(20, 15)
6	5	$x = 20 - 1 = 19$	$y = 15 - 1 = 14$	(19, 14)
7	6	$x = 19 - 1 = 18$	$y = 14 - 1 = 13$	(18, 13)
8	7	$x = 18 - 1 = 17$	$y = 13 - 1 = 12$	(17, 12)
9	8	$x = 17 - 1 = 16$	$y = 12 - 1 = 11$	(16, 11)
10	9	$x = 16 - 1 = 15$	$y = 11 - 1 = 10$	(15, 10)

∴ So, the resulting points are

(25,20), (24,19), (23,18), (22,17), (21,16), (20,15), (19,14), (18,13), (17,12), (16,11), (15,10)

Ans //

b) Bresenham's Line Drawing Algorithm

There are four cases

i) Line with +ve slope less than or equal to 1 (i.e. $m \leq 1$)

Algorithm

1. Given a line with two end-points (x_0, y_0) and (x_1, y_1)
2. Calculate the value of initial decision parameter P_0 as
   $$P_0 = 2 * dy - dx$$
   where $dy = y_1 - y_0$ and $dx = x_1 - x_0$.
3. Perform the following test, starting at $k = 0$
   If $P_k < 0$
   - the next point to plot is (x_{k+1}, y_k);
   - the value of next decision parameter P_{k+1} is
     $$P_{k+1} = P_k + 2 * dy$$

   Otherwise
   - the next point to plot is (x_{k+1}, y_{k+1});
   - the value of next decision parameter P_{k+1} is
     $$P_{k+1} = P_k + 2 * dy - 2 * dx$$
4. Repeat Step (3) dx times.

* Digitize a line with end-points (20, 10) and (30, 18) using Bresenham's line drawing algorithm.

Here:

Given: $(x_0, y_0) = (20, 10)$

$(x_1, y_1) = (30, 18)$

$dy = y_1 - y_0 = 18 - 10 = 8$

$dx = x_1 - x_0 = 30 - 20 = 10$

$$\text{Slope}(m) = \frac{dy}{dx} = \frac{8}{10} = 0.8$$

Here, Slope(m) is +ve and less than 1.

K	P_K	x_{k+1}	y_{k+1}	(x_{k+1}, y_{k+1})
0	$P_0 = 2dy - dx = 2\times8 - 10 = 6$	20+1 = 21	10+1 = 11	(21, 11)
1	$P_1 = 6 + 2\times8 - 2\times10 = 2$	21+1 = 22	11+1 = 12	(22, 12)
2	$P_2 = 2 + 2\times8 - 2\times10 = -2$	22+1 = 23	12	(23, 12)
3	$P_3 = -2 + 2\times8 = 14$	24	13	(24, 13)
4	$P_4 = 14 + 2\times8 - 2\times10 = 10$	25	14	(25, 14)
5	$P_5 = 10 + 2\times8 - 2\times10 = 6$	26	15	(26, 15)
6	$P_6 = 6 + 2\times8 - 2\times10 = 2$	27	16	(27, 16)
7	$P_7 = 2 + 2\times8 - 2\times10 = -2$	28	16	(28, 16)
8	$P_8 = -2 + 2\times8 = 14$	29	17	(29, 17)
9	$P_9 = 14 + 2\times8 - 2\times10 = 10$	30	18	(30, 18)

So, the required points are (20, 10), (21, 11), (22, 12), (23, 12), (24, 13), (25, 14), (26, 15), (27, 16), (28, 16), (29, 17), (30, 18) Ans//

11) Line with +ve Slope Greater than 1 (i.e. $m > 1$)

Algorithm

① Given a line with two end-points (x_0, y_0) and (x_1, y_1)

② Calculate the value of initial decision parameter p_0 as

$p_0 = 2 * dx - dy$ where

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

③ Perform the following test, starting at $k = 0$

If $p_k < 0$

— the next point to plot is (x_k, y_{k+1})

— the value of next decision parameter p_{k+1} is

$p_{k+1} = p_k + 2 * dx$

Otherwise

— the next point to plot is (x_{k+1}, y_{k+1})

— the value of next decision parameter p_{k+1} is

$p_{k+1} = p_k + 2 * dx - 2 * dy$

④ Stop Repeat Step ③ dy times.

* Digitalize a line with end-points (1,0) and (3,3) using Bresenham's line drawing algorithm.

Soln

Given $(x_0, y_0) = (1,0)$

$(x_1, y_1) = (3,3)$

$dy = y_1 - y_0 = 3 - 0 = 3$

$dx = x_1 - x_0 = 3 - 1 = 2$

$\text{Slope}(M) = \frac{y_1 - y_0}{x_1 - x_0} = \frac{dy}{dx} = \frac{3}{2} = 1.5$

Here, Slope (M) is +ve and greater than 2.

K	P_K	x_{K+1}	y_{K+1}	(x_{K+1}, y_{K+1})
0	$P_0 = 2\times2 - 3 = 1$	$0+2=2$	$0+1=1$	(2,1)
1	$P_1 = 1 + 2\times2 - 2\times3 = -1$	2	2	(2,2)
2	$P_1 = -1 + 2\times2 = 3$	3	3	(3,3)

∴ So the required points are (1,0), (2,1), (2,2), (3,3)

Any 1,

iii) Line with -ve slope less than or equal to 2.

Algorithm

(1) Given a line with two ~~each~~ end-point (x_0, y_0) and (x_1, y_1)

(2) Calculate the value of initial decision parameter P_0 as

$$P_0 = 2 * |dy| - |dx|$$

where, $|dx| = |x_1 - x_0|$ and $|dy| = |y_1 - y_0|$

(3) Starting at $k = 0$, perform the following test

If $P_k < 0$

- the next point to plot is $(x_k - 1, y_k)$
- $P_{k+1} = P_k + 2 * (dy)$

Otherwise,

- the next point to plot is $(x_k - 1, y_k + 1)$
- $P_{k+1} = P_k + 2 * (dy) - 2 * (dx)$

(4) Repeat step (3) (dx) times

* Digitize a line with end-points (5,10) and (10,7) using Bresenham's algorithm.

Soln

$(x_0, y_0) = (10, 7)$, $(x_1, y_1) = (5, 10)$

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

$|dy| = |y_1 - y_0| = |10-7| = |3| = 3$

$\text{Slope } (m) = \frac{y_1 - y_0}{x_1 - x_0} = \frac{10-7}{5-10} = \frac{3}{-5} = -0.6$

Here Slope m is -ve and less than 1.

k	P_k	x_{k+1}	y_{k+1}	(x_{k+1}, y_{k+1})
0	$P_0 = 2\times3-5 = 1$	10-1=9	7+1=8	(9,8)
1	$P_1 = 1+2*3-2*5 = -3$	9-1=8	8	(8,8)
2	$P_2 = -3+2*3 = 3$	8-1=7	8+1=9	(7,9)
3	$P_3 = 3+2*3-2*5 = -1$	7-1=6	9	(6,9)
4	$P_4 = -1+2\times3 = 5$	6-1=5	9+1=10	(5,10)

∴ So the required points are (10,7), (9,8), (8,8), (7,9), (6,9), (5,10)

Ans

Practice Set:-

* (15, 18) and (10, 18) :-

Here, let $y_0 = 15$, $y_1 = 18$

$y_1 - y_0 = 18 - 15 = 3$ +ve so leave as it is

$(x_0, y_0) = (15, 18)$

$(x_1, y_1) = (10, 18)$

* (10, 10) and (20, 5)

$y_0 = 10$, $y_1 = 5$

$y_1 - y_0 = 5 - 10 = -5$ -ve So change it

$(x_0, y_0) = (20, 5)$, $(x_1, y_1) = (10, 10)$

iv) Line with -ve slope greater than 1

Algorithm

(1) Given a line with two end-points (x_0, y_0) and (x_1, y_1)

(2) Calculate the value of initial decision parameter P_0 as

$$P_0 = 2 * |dx| - |dy|$$

where, $|dx| = |x_1 - x_0|$ $|dy| = |y_1 - y_0|$

(3) Starting at $k = 0$, perform the following test

If $P_k < 0$

— the next point to plot is (x_k, y_{k+1})

— $P_{k+1} = P_k + 2 * |dx|$

Otherwise

— the next point to plot is $(x_k - 1, y_{k+1})$

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

(4) Repeat Step (3) $|dy|$ times

* Digitize a line with end-points (5,6) and (6,3) using Bresenham's algorithm.

Soln

$(x_0, y_0) = (6,3)$, $(x_1, y_1) = (5,6)$

$|dx| = |x_1 - x_0| = |5-1| = 1$

$|dy| = |y_1 - y_0| = |6-3| = 3$

$\text{Slope } (M) = \frac{y_1 - y_0}{x_1 - x_0} = \frac{6-3}{5-6} = \frac{3}{-1} = -3$

Here Slope M is -ve and greater than 1.

k	P_k	x_{k+1}	y_{k+1}	(x_{k+1}, y_{k+1})
6	$P_0 = 2x1 - 3 = -1$	6	3+1=4	(6,4)
1	$P_1 = -1 + 2x1 = 1$	6-1=5	4+1=5	(5,5)
2	$P_2 = 1 + 2*1 - 2x3 = 3$	5	5+1=6	(5,6)

∴ So, the required points are (6,3), (6,4), (5,5) and (5,6) Ans//

* (25,20) and (15,10)

$(x_0, y_0) = (15,10)$, $(x_1, y_1) = (25,20)$

∴ $y_1 - y_0$ = -ve take 2nd point as starting points.

2080-08-24

Circle Drawing Algorithm

Mid-point circle drawing algorithm

(1) Input radius 'r' and centre (x_c, y_c) and obtain the first point on the circle centered at origin as,

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

(2) Calculate the value of initial decision parameter P_0 as,

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

(3) At each point x_k, starting at $k=0$ perform the following test:

If $P_k < 0$

- the next point to plot centered at origin is (x_{k+1}, y_k)
- $P_{k+1} = P_k + 2x_{k+1} + 1$

else

- the next point to plot centered at origin is (x_{k+1}, y_{k-1})
- $P_{k+1} = P_k + 2x_{k+1} + 1 - 2y_{k+1}$

(4) Determine the symmetry point on the other seven octants.

(5) Move each calculated point (x, y) into the circle centered at (x_c, y_c) as $x = x + x_c$, $y = y + y_c$

(6) Repeat step 3 to 5 until $x \geq y$.

* Symmetry of circle

→ Division of circle into equal parts.

2 Way Symmetry → division of circle into 2 equal points.

8 Way Symmetry → Division of circle into 8 equal points.

eg:- $(x,y) \rightarrow (-x,y), (x,-y), (-x,-y)$

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

* Calculate the required points to draw a circle using mid-point circle drawing algorithm having radius 10 and centre (3,4)

Soln

Given, $r = 10$

$(x_c, y_c) = (3, 4)$

The 1st point of the circle centred at origin is $(x_0, y_0) = (0, r) = (0, 10)$

Other 7 octants are of 1st point are

$(-0, 10)$, $(0, -10)$, $(-0, -10)$, $(10, 0)$ $(-10, 0)$ $(10, -0)$ $(-10, -0)$

After Adding $(x_c, y_c) = (3, 4)$ to all octants we get

$(3, 14)$ $(3, 14)$, $(3, -6)$ $(3, -6)$, $(13, 4)$, $(-7, 4)$, $(13, 4)$, $(-7, 4)$

k	P_k	(x_{k+1}, y_{k+1})	other 7 octants	All octants after adding $(x_c, y_c) = (3, 4)$
0	$P_0 = 1 - r = 1 - 10 = -9$	$(1, 10)$	$(-1, 10)$, $(1, -10)$, $(-1, -10)$ $(10, 1)$, $(-10, 1)$ $(10, -1)$ $(-10, -1)$	$(4, 14)$, $(2, 14)$ $(4, 6)$ $(2, -6)$, $(13, 5)$ $(-7, 5)$, $(13, 3)$, $(-7, 3)$
1	$-9 + 2 \times 1 + 1 = -6$	$(2, 10)$	$(-2, 10)$ $(2, -10)$ $(-2, -10)$ $(10, 2)$ $(-10, 2)$, $(10, -2)$ $(-10, -2)$	$(5, 14)$, $(1, 14)$, $(5, -6)$, $(1, -6)$, $(13, 6)$, $(-7, 6)$ $(13, 2)$ $(-7, 2)$
2	$-6 + 2 \times 2 + 1 = -1$	$(3, 10)$	$(-3, 10)$, $(3, -10)$, $(-3, -10)$, $(10, 3)$, $(-10, 3)$, $(10, -3)$ $(-10, -3)$	$(6, 14)$, $(0, 14)$, $(6, -6)$ $(0, -6)$, $(13, 7)$, $(-7, 7)$, $(13, 1)$ $(-7, 1)$

3	$-1+2\times3+1=6$	(4, 9)	(-4, 9), (4, -9), (-4, -9), (9, 4), (-9, 4), (9, -4), (-9, -4)	(7, 13), (-1, 13), (7, -5), (-1, -5), (12, 8), (-6, 8), (12, 0), (-6, 0)
4	$6+2\times4+1=-3$	(5, 9)	(-5, 9), (5, -9), (-5, -9), (9, 5), (-9, 5), (9, -5), (-9, -5)	(8, 13), (-2, 13), (8, -5), (-2, -5), (12, 9), (-6, 9), (12, -1), (-6, -1)
5	$-3+2\times5+1=8$	(6, 8)	(-6, 8), (6, -8), (-6, -8), (8, 6), (-8, 6), (8, -6), (-8, -6)	(9, 12), (-3, 12), (9, -4), (-3, -4), (11, 10), (-5, 10), (11, -2), (-5, -2)
6	$8+2\times6+1=5$	(7, 7)	(-7, 7), (7, -7), (-7, -7), (7, 7), (-7, 7), (7, -7), (-7, -7)	(10, 11), (-4, 11), (10, -3), (-4, -3), (10, 11), (-4, 11), (10, -3), (-4, -3)

∴ Here, $x = y$. So algorithm stops.

Ellipse Drawing Algorithm

Mid-Point Ellipse Drawing Algorithm :-

Algorithm :-

1. Input centre (x_c, y_c) and radi along x-axis r_x, radius along y axis r_y and obtain the first point of the ellipse centered at origin as.

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

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

$$p_{10} = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2$$

3. At the each position x_k, starting at $k=0$, perform the following test in region 1.

If $p_{1k} < 0$

— The next point to plot ellipse centred at origin is (x_{k+1}, y_k)

— The $p_{1k+1} = p_{1k} + 2r_y^2 x_{k+1} + r_y^2$

else

— The next point to plot ellipse centred at origin is (x_{k+1}, y_{k-1})

— $p_{1k+1} = p_{1k} + 2r_y^2 x_{k+1} + r_y^2 - 2r_x^2 y_{k+1}$

4. Calculate the value of initial decision parameter at region 2 using last calculated point on the region 1, say (x_0, y_0)

(5) Calculate initial decision parameter value in region 2 as

$$p_{20} = ry^2(x_0 + 1/2)^2 + r^2(y_0 - 1)^2 - r^2r^2 + r^2y^2$$

(6) At each position y_k, starting at $k = 0$, perform the following test on the region 2.

If $p_{2k} > 0$

— The next point to plot ellipse centered at origin is $(x_k, y_k - 1)$

$$- p_{2k+1} = p_{2k} - 2r^2(y_k - 1) + r^2$$

else

— The next point to plot ellipse centered at origin is $(x_k + 1, y_k - 1)$

$$- p_{2k+1} = p_{2k} + 2ry^2x_{k+1} - 2r^2y_{k+1} + r^2$$

(7) Determine the symmetric points on other 3 quadrants.

(8) Move each calculated point (x, y) on the ellipse centre (x_c, y_c) as

$$x = x + x_c$$

$$y = y + y_c$$

(9) Repeat the process for region 1 until $2ry^2x_k \geq 2r^2y_k$ and region 2 until $(x, y) = (r, 0)$.

* Given ellipse parameters: $r_x = 8$ and $r_y = 6$. Using Mid-Point ellipse drawing algorithm. Determine raster position using along the ellipse path in 1st quadrant.

Given:-

For Region 1: $r_x = 8$, $r_y = 6$

$(x_c, y_c) = (0,0)$

$(x_0, y_0) = (0, r_y) = (0, 6)$

$\therefore p_{10} = 6^2 - 8^2 \times 6 + \frac{1}{4} \times 8^2$

$= 36 - 64 \times 6 + \frac{1}{4} \times 64 = -332$

k	p_{1k}	(x_{k+1}, y_{k+1})	$2r_y^2 x_{k+1}$	$2r_x^2 y_{k+1}$
0	-332	$(1,6)$	72	768
1	$-332 + 2\times6^2\times1 + 6^2 = -224$	$(2,6)$	144	768
2	$-224 + 2\times6^2\times2 + 6^2 = -44$	$(3,6)$	216	768
3	$-44 + 2\times6^2\times3 + 6^2 = 208$	$(4,5)$	288	640
4	$208 + 2\times6^2\times4 + 6^2 - 2\times8^2\times5$	$(5,5)$	360	640
5	$-108 + 2\times6^2\times5 + 6^2 = 288$	$(6,4)$	432	512
6	$288 + 2\times6^2\times6 + 6^2 - 2\times8^2\times4$ $= 224$	$(7,3)$	504	384

This point is greater than $2r_x^2 y_{k+1}$: so it stops here

Here, $2r_y^2 x_{k+1} > 2r_x^2 y_{k+1}$

So, Stop in region 1.

(Or

For region 2, take $(x_0, y_0) = (7, 3)$

k	p_{2k}	(x_{k+1}, y_{k+1})
0	-151	(8, 2)
1	233	(8, 1)
2		(8, 0)

Here, $(x_n, 0)$. So, algorithm stops in region 2.

Raster Scan Numerical

① Consider a raster scan system with resolution 640 by 640. What size frame buffer (in bytes) is needed for the system to store 12 bits per pixel? How much storage is required if 24 bits per pixels are to be stored?

→ Given:-

Resolution = 640 × 640 pixels.

For 12-bits per pixels :-

To store 1 pixel, it requires 12 bit

To store 640×640 pixel, it requires = 12×640×640 bit

$= \frac{12 \times 640 \times 640}{8}$ bytes

= 38400 bytes

For 24-bits per pixels :-

To store 1 pixel, it requires 24 bits

To store 640×640 pixel, it requires = 24×640×640 bit

$= \frac{24 \times 640 \times 640}{8}$ bytes

= 768000 bytes.

② How long would it take to load a 640 by 480 frame buffer with 12 bit per pixel if 10^5 bit can be transferred per second.

→ Given,

Resolution = 640×480 pixel

Bits per pixel = 12 bits

∴ Total no. of bits = 640 × 480 × 12 bits

10^5 bits can be transmitted in 1 second

1 bits can be transmitted in $1/10^5$ second

640 bits i.e. (640 × 480 × 12) can be transmitted in

$\frac{1}{10^5} \times (640 \times 480 \times 12)$ second

= 36.864 second.

(3) Calculate the size of frame buffer required to store 640 × 480 B and W video of length 5 minutes without compression

Given:

Resolution (m×n) = 640 × 480 pixel

Refresh Rate (R) = 50 Hz

Time (t) = 5 minutes

= 5 × 60 = 300 seconds

Size of frame buffer required = m × n × R × t

= 640 × 480 × 50 × 300

$= \frac{640 \times 480 \times 50 \times 300}{8}$ bytes

(4) Find out the aspect ratio of the raster system using 8×10 inches screen and 100 pixel/(inch).

Given

Aspect Ratio = $\frac{\text{Width}}{\text{Height}} = \frac{8 \times 100}{10 \times 100} = \frac{4}{5}$ 4 : 5

2080-08-22

Date
Page

Unit:-2 Two Dimensional and Three Dimensional Transformation

→ 2-Dimensional Transformation
→ 2-D Translation, Rotation, Scaling
→ Homogeneous Coordinates, Reflection, Shear Transform
→ 3-Dimensional Transformation
→ 3-D Translation, Rotation Scaling, Reflection, Shear

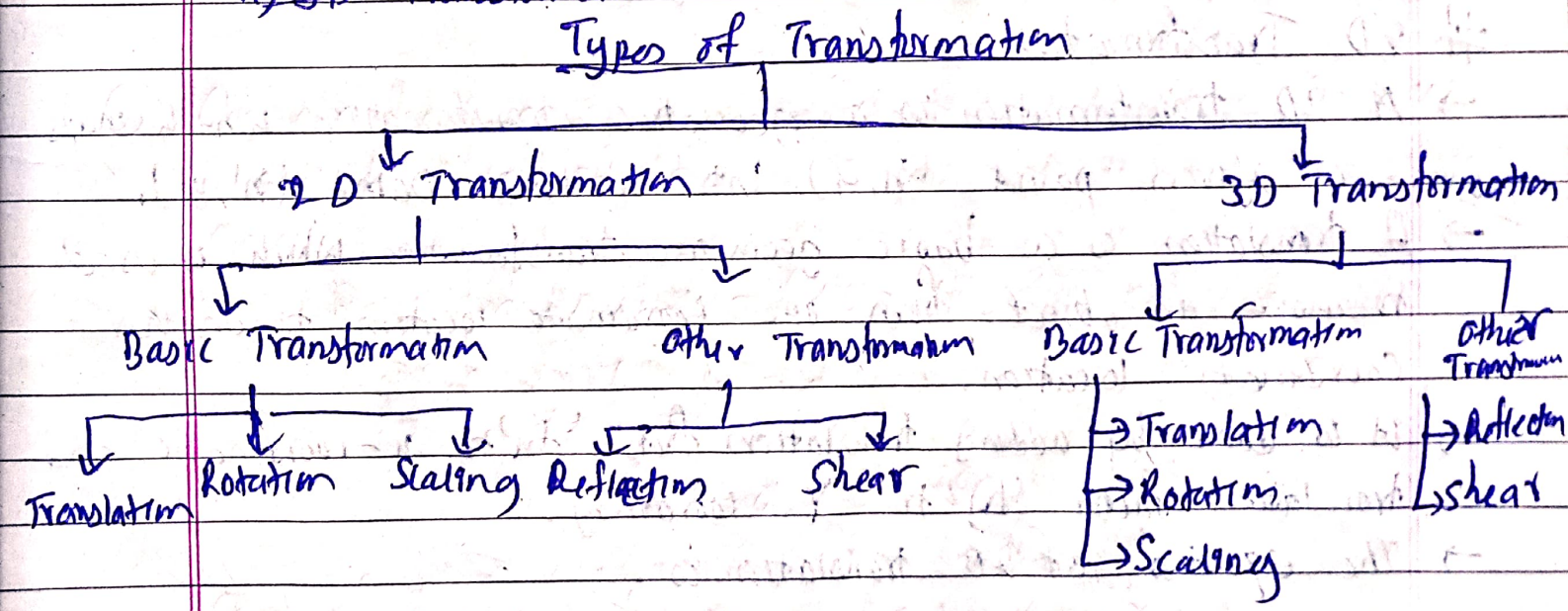
Geometric Transformation

→ A geometric transformation is an operation that changes the position, shape and orientation of an object

* Types of Geometric Transformation

~~i) 2D Transformation~~

~~ii) 3D Transformation~~



* Rigid Transformation

→ A transformation without deformation (changes) in shape is called Rigid body translation.

→ Rigid body transformation includes

i) Translation

ii) Rotation

iii) Reflection

* Non-Rigid Body Transformation
→ A transformation with deformation (change) in shape is called non-rigid body transformation.
→ ~~Includes~~ It includes :-
    i) Scaling
    ii) Shear

2D Transformation
→ A 2D transformation is a geometric transformation which maps a given point (x, y) into a new point (x', y').
→ A translation is a basic geometric transformations which is used to move an object from one coordinate location to another coordinate location.
→ It is done by adding translation factor 'tx' to x-coordinate and translation factor 'ty' to Y-coordinate.
→ The equation of 2D translation is.

$x' = x + tx$ where, (x, y) is given point.
$y' = y + ty$ (x', y') is resultant point after applying translation.

tx = translation factor along x-axis.
ty = translation factor along y-axis.

→ When the translation factor tx and ty are positive then the object moves away from the origin and when the translation factor tx and ty are negative then the object moves towards the origin.

→ Note :- i) While translating a line, the translation factor is apply to both the endpoint of the line.

ii) While translating a circle, the translation factor is apply to the centre of the circle.

iii) While translating a polygon, the translation factor is apply to all the vertices of the polygon.

→ The translation of the object is possible in one of the following three directions:-

i) Translation of an object in horizontal direction i.e. translation parallel to x-axis.

$$x' = x + tx$$
$$y' = y$$

ii) Translation of an object in vertical direction i.e. translation parallel to y-axis.

$$x' = x$$
$$y' = y + ty$$

iii) Translation of an object in both horizontal and vertical direction i.e. translation parallel to both x and y-axis.

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

* Example :-

Given a triangle with vertices A(1,2), B(2,3) and C(3,2). obtain the new vertices of the triangle after applying translation factor along x-axis by 2 and translation factor along y-axis by 3.

Soln

Given, $t_x = 2$, $t_y = 3$

vertices of A(1,2), B(2,3), C(3,2) triangle

we know, the equation of translation is

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

For point A(1,2) :-

$$x' = x + t_x = (1+2) = 3$$
$$y' = y + t_y = (2+3) = 5$$

$\therefore A'(3,5)$

For point B(2,3) :-

$$x' = x + t_x = 2+2 = 4$$
$$y' = y + t_y = 3+3 = 6$$

$\therefore B'(4,6)$

For point C(3,2) :-

$$x' = x + t_x = 3+2 = 5$$
$$y' = y + t_y = 2+3 = 5$$

$\therefore C'(5,5)$

$\therefore$ the resultant vertices of triangle are A'(3,5), B'(4,6) and C'(5,5)

2D Rotation

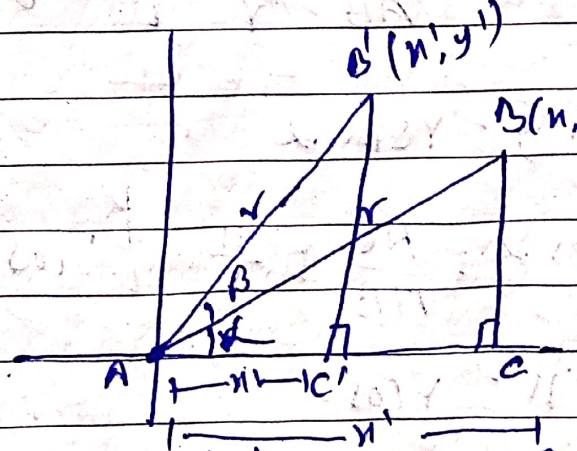
→ A rotation is a basic geometric transformation which is used to move an object on a circular path about an origin.

→ Rotation requires two major parameters : the direction of rotation and Angle of rotation.

→ Any given object can be rotated in either in clockwise or anti-clockwise direction.

→ A clockwise rotation indicates the negative angle and anti-clockwise rotation indicates the positive.

Rotation When Pivot Point is at Origin :-



Here the point B(x,y) is rotated by an angle β to get new point B'(x',y').

In triangle $\Delta A'B'C'$:-

$$\cos(\alpha+\beta) = \frac{AC'}{AB'} = \frac{x'}{r} \quad \left[\because \cos\theta = \frac{b}{h}\right]$$

$$x' = r\cos(\alpha+\beta)$$

$$x' = r(\cos\alpha\cos\beta - \sin\alpha\sin\beta) \quad \left[\because \cos(A+B) = \cos A\cos B - \sin A\sin B\right]$$

$\therefore x' = r\cos\alpha\cos\beta - r\sin\alpha\sin\beta$ ---- eqn (i)

Similarly

$$\sin(\alpha+\beta) = \frac{BC'}{AB'} = \frac{y'}{r} \quad \left(\because \sin\theta = \frac{P}{h}\right)$$

or $y' = r\sin(\alpha+\beta)$

$\therefore y' = r(\sin\alpha\cos\beta + \cos\alpha\sin\beta)$ [$\because \sin(A+B) = \sin A\cos B + \cos A\sin B$]

$\therefore y' = r\sin\alpha\cos\beta + r\cos\alpha\sin\beta$ ---- eqn (ii)

In Triangle ΔABC :-

$$\sin\alpha = \frac{BC}{AB} = \frac{y}{r} \quad \left[\sin\theta = \frac{P}{h}\right]$$

$\therefore y = r\sin\alpha$

$$\cos\alpha = \frac{AC}{AB} = \frac{x}{r} \quad \left[\because \cos\theta = \frac{b}{h}\right]$$

$\therefore x = r\cos\alpha$

Now Substituting the value of x and y in eqn (i) and (ii), We get :-

$x' = r\cos\alpha\cos\beta - r\sin\alpha\sin\beta$

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

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

This is the equation of rotation when direction of rotation is anticlockwise.

When the direction of rotation is clockwise, the angle is negative:-

i.e. $x' = x\cos(-\beta) - y\sin(-\beta)$

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

$y' = y\cos(-\beta) + x\sin(-\beta)$

$= y\cos\beta - x\sin\beta$

$\therefore x' = x\cos\beta + y\cos\beta$

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

Example:-

* Rotate a point (2,3) by an angle 30°.

Given

$(x, y) = (2, 3)$, $\beta = 30°$

We know equation of rotation is

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

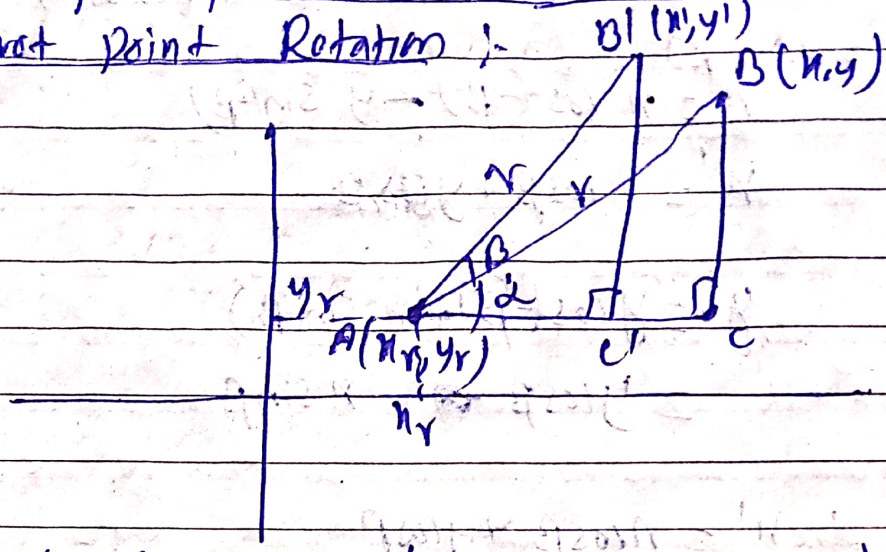
$y' = y\cos\beta + x\sin\beta$

$x' = x\cos\beta - y\sin\beta = 2\times\cos 30 - 3\times\sin 30 = 0.2320$

$y' = y\cos\beta + x\sin\beta = 3\times\cos 30 + 2\times\sin 30 = 3.5981$

$\therefore (x', y') = (0.2321, 3.5981)$

Rotation when pivot point is other than origin / Fixed point rotation / Pivot Point Rotation :-



Here point B(x,y) is rotated by an angle β to obtain new point B'(x',y') by keeping pivot point (x_r, y_r) fixed.

In Triangle ΔABC :-

$$\sin\alpha = \frac{BC}{AB} = \frac{y-y_r}{r}$$

$$\therefore y - y_r = r\sin\alpha$$

Similarly

$$\cos\alpha = \frac{AC}{AB} = \frac{x-x_r}{r}$$

$$\therefore x - x_r = r\cos\alpha$$

In Triangle ΔAB'C' :-

$$\sin(\alpha+\beta) = \frac{B'C'}{AB'} = \frac{y'-y_r}{r}$$

$$\Rightarrow y' - y_r = r\sin(\alpha+\beta)$$

Date
Page

or, $y' - y_r = r\sin\alpha\cos\beta + r\cos\alpha\sin\beta$

or, $y' = y_r + (y - y_r)\cos\beta + (x - x_r)\sin\beta$

Similarly

$$\cos(\alpha+\beta) = \frac{AC'}{AB'} = \frac{x' - x_r}{r}$$

or, $(\cos\alpha\cos\beta - \sin\alpha\sin\beta) = \frac{x' - x_r}{r}$

or, $x' - x_r = r\cos\alpha\cos\beta - r\sin\alpha\sin\beta$

or, $x' = x_r + (x - x_r)\cos\beta - (y - y_r)\sin\beta$

∴ The equation of the rotation when pivot point is other than origin i.e. (x_r, y_r) is

$\therefore x' = x_r + (x - x_r)\cos\beta - (y - y_r)\sin\beta$

$\therefore y' = y_r + (y - y_r)\cos\beta + (x - x_r)\sin\beta$

When the direction of rotation is clockwise, the angle is negative

i.e. $x' = x_r + (x - x_r)\cos(-\beta) - (y - y_r)\sin(-\beta)$

$= x_r + (x - x_r)\cos\beta + (y - y_r)\sin\beta$

$y' = y_r + (y - y_r)\cos(\beta) + (x - x_r)\sin(\beta)$

$= y_r + (y - y_r)\cos\beta - (x - x_r)\sin(\beta)$

Example:-

* Rotate a point (2,3) by an angle 60° by keep pivot point at (1,2)

Soln

Given $(x,y) = (2,3)$

$\beta = 60°$, $(x_r, y_r) = (1,2)$

We know the equation is

$$x' = x_r + (x - x_r)\cos\beta - (y - y_r)\sin\beta$$

$$= 1 + (2-1)\cos 60 - (3-2)\sin 60°$$

$$= 0.634$$

$$y' = y_r + (y - y_r)\cos\beta + (x - x_r)\sin\beta$$

$$= 2 + (3-2)\cos 60 + (2-1)\sin 60$$

$$= 2.366$$

$$\therefore (x', y') = (0.634, 2.366)$$

2D Scaling:

→ A scaling is a basic geometric transformation which is used to alter (change) the ~~Shape~~ size of an object.

→ It is achieved by multiplying x-coordinate by scaling factor S_x and multiplying y-coordinate by scaling factor S_y.

i.e.

The equation of scaling is

$x' = x * S_x$ where, (x, y) = original point

$y' = y * S_y$ (x', y') = Resultant point

S_x = Scaling factor along x-axes

S_y = Scaling factor along y-axes

→ To increase the size of an object, the scaling factor must be greater than 1 and to decrease the size of an object, the scaling factor must be less than 1.

→ When $S_x = S_y$ then the scaling is called Uniform Scaling otherwise (i.e. $S_x \neq S_y$) the scaling is called differential Scaling.

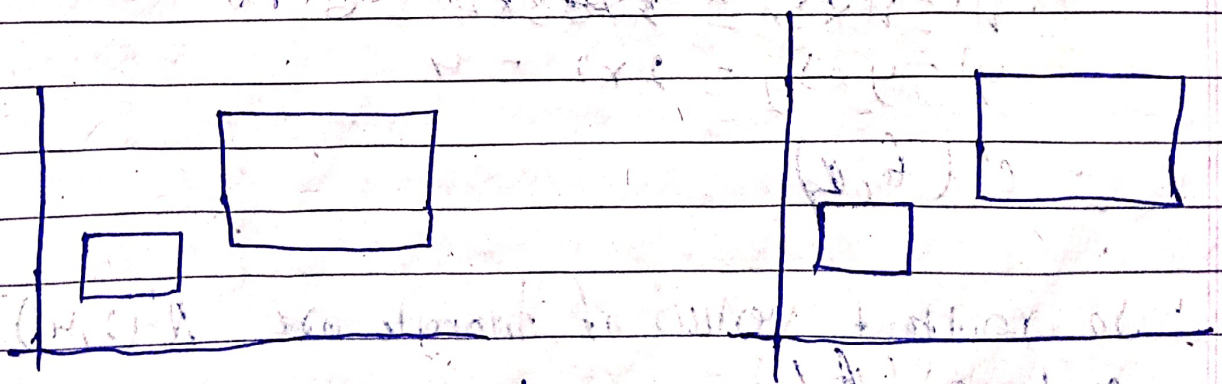


Fig 1:- Uniform Scaling

Fig 2:- Differential Scaling

Example:-

* Given a triangle with vertices A(1,2), B(2,3), C(3,2). obtain a new vertices of the triangle after magnified the given triangle twice.

Soln

Given, $S_x = S_y = 2$ (magnified twice)

For point A(1,2):

$x' = x \times S_x = 1 \times 2 = 2$

$y' = y \times S_y = 2 \times 2 = 4$

$\therefore A'(x', y') = A(2,4)$

For point B(2,3):

$x' = x \times S_x = 2 \times 2 = 4$

$y' = y \times S_y = 3 \times 2 = 6$

$\therefore (x', y') = (4, 6)$

For point C(3,2):-

$x' = x \times S_x = 3 \times 2 = 6$

$y' = y \times S_y = 2 \times 2 = 4$

$\therefore C'(6, 4)$

So resultant vertices of triangle are A'(2,4), B'(4,6) and C'(6,4).

Matrix Representation of 2D basic Geometric transformation

i) 2D Transformation :-

$x' = x + t_x$

$y' = x + t_y$

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

ii) 2D Rotation :-

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

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

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

iii) 2D Scaling :-

$x' = x * S_x$

$y' = y \times S_y$

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

Homogeneous Coordinate System

→ The representation of N coordinate System with N+1 coordinate System is known as homogeneous coordinate System.

eg:-

Representation of 2 coordinate System with 3 coordinate System

→ In Homogeneous coordinate System, we introduce one additional coordinate that allows us to represent all geometric transformation in the form of Matrix Multiplication.

$$(x, y) \xrightarrow[\text{representation}]{\text{Homogeneous coordinate}} (x, y, 1)$$

Representing translation as Homogeneous Coordinate

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

Here:-

$$x' = x + t_x$$
$$y' = y + t_y$$
$$1 = 1$$
$$p' = T(t_x, t_y) \cdot p$$

Where, p' = New Point

p = Original Point

$T(t_x, t_y)$ = Translation Matrix

Representing Rotation as Homogeneous Coordinate

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

$$P' = R(\beta) \cdot P$$

where, P' = resultant point
P = Original point
$R(\beta)$ = Rotation Matrix

In above :-

$$x' = x\cos\beta - y\sin\beta$$
$$y' = x\sin\beta + y\cos\beta$$
$$1 = 1$$

Representing Scaling as Homogeneous Coordinate

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

Here, ~~P = Scaling Matrix~~ $\cdot P' = S(S_x, S_y) \cdot P$

where, P' = Resultant Point
P = Original Point
$S(S_x, S_y)$ = Scaling Matrix

In above,

$$x' = x * S_x$$
$$y' = y * S_y$$
$$1 = 1$$

Composite Transformation

→ A composite transformation is a transformation which is obtained by multiplying matrices in order from right to left.

→ The composite transformation is required because it helps to identify the nature of successive translation, successive rotation and successive scaling.

* Two Successive Translation

→ If the successive translation vector (t_{x_2}, t_{y_2}) and (t_{x_1}, t_{y_1}) are applied to a coordinate position P then the final transformed location P' is calculated as,

$$P' = \{T(t_{x_2}, t_{y_2}) \cdot \{T(t_{x_1}, t_{y_1}) \cdot P\}$$

$$= T(t_{x_2}, t_{y_2}) \cdot T(t_{x_1}, t_{y_1}) \cdot P$$

→ The composite transformation matrix for this sequence of translation is:-

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{x_2} \\ 0 & 1 & t_{y_2} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & t_{x_1} \\ 0 & 1 & t_{y_1} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & (t_{x_2} + t_{x_1}) \\ 0 & 1 & (t_{y_2} + t_{y_1}) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

→ From above, it is seen that the two successive translation is additive

* Two Successive Scaling

→ If the successive scaling vector (s_{x_2}, s_{y_2}) and (s_{x_1}, s_{y_1}) are applied to the coordinate position p then the final transformed location p' is calculated as

$$p' = S(s_{x_2}, s_{y_2}) \{S(s_{x_1}, s_{y_1}) \cdot p\}$$

$$= S(s_{x_2}, s_{y_2}) \cdot S(s_{x_1}, s_{y_1}) \cdot p$$

→ The composite transformation matrix of this sequence of scaling is

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_{x_2} & 0 & 0 \\ 0 & s_{y_2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_{x_1} & 0 & 0 \\ 0 & s_{y_1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} s_{x_1} \cdot s_{x_2} & 0 & 0 \\ 0 & s_{y_1} \cdot s_{y_2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

→ From above, it is seen that the two successive scaling is Multiplicative.

* Two Successive Rotation

→ If the rotation angle β_1 and β_2 are applied to a coordinate position p then the final transformed location p' is calculated is

$$p' = R(\beta_2) \cdot \{R(\beta_1) \cdot p\}$$

$$= R(\beta_2) \cdot R(\beta_1) \cdot p$$

→ The composite transformation matrix for this sequence of rotation is

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\beta_2 & -\sin\beta_2 & 0 \\ \sin\beta_2 & \cos\beta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta_1 & -\sin\beta_1 & 0 \\ \sin\beta_1 & \cos\beta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\beta_1 \cdot \cos\beta_2 - \sin\beta_1 \cdot \sin\beta_2 & -\cos\beta_1 \cdot \sin\beta_2 - \sin\beta_1 \cdot \cos\beta_2 & 0 \\ \sin\beta_1 \cdot \cos\beta_2 + \cos\beta_1 \cdot \sin\beta_2 & -\sin\beta_1 \cdot \sin\beta_2 + \cos\beta_1 \cdot \cos\beta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(\beta_1+\beta_2) & -\sin(\beta_1+\beta_2) & 0 \\ \sin(\beta_1+\beta_2) & \cos(\beta_1+\beta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

* Scale the triangle (1,1), (1,2) and (2,3) first with 0.5 and then with 4.

Soln Given, $S_{x_1} = S_{y_1} = 0.5$

$S_{x_2} = S_{y_2} = 4$

$S_x = S_{x_1} * S_{x_2} = 0.5 \times 4 = 2$

$S_y = S_{y_1} * S_{y_2} = 0.5 \times 4 = 2$

} we know 2 successive scaling is multiplicative.

1st coordinate (1,1):

$x' = x * S_x = 1 \times 2 = 2$

$y' = y * S_y = 1 * 2 = 2$

$\therefore (x', y') = (2, 2)$

2nd Coordinate (1,2):-

$x' = x * s_x = 1 \times 2 = 2$

$y' = y \times s_y = 2 \times 2 = 4$

$\therefore (x', y') = (2, 4)$

3rd Coordinate (2,3):-

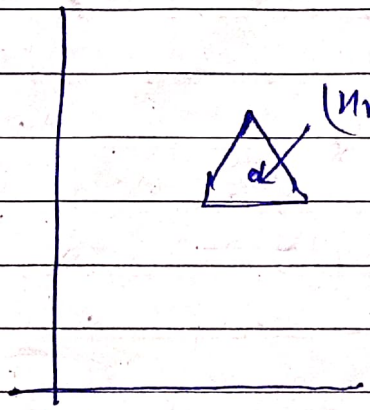
$x' = x * s_x = 2 \times 2 = 4$

$y' = y \times s_y = 2 \times 3 = 6$

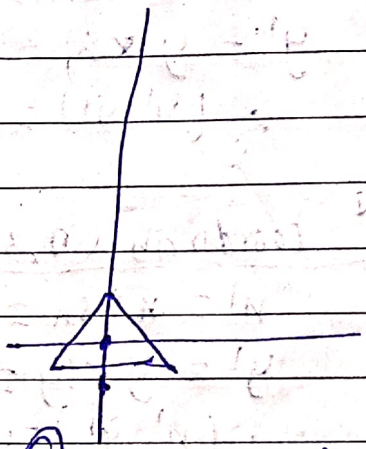
$\therefore (x', y') = (4, 6)$

∴ So resultant vertices of triangle are (2,2), (2,4) and (4,6).

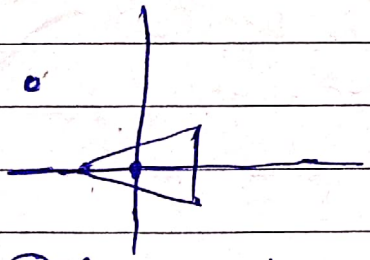
Pivot Point Rotation :-



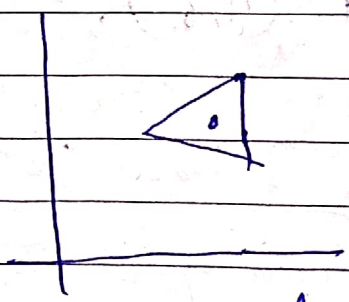
(a) Original position of an object and pivot point (x_r, y_r)



(b) Translation of an object such that pivot point is at origin



(c) Rotation about an origin



d) Translation of an object so the pivot point is at original position (x_r, y_r)

→ The rotation about any selected pivot point (x_r, y_r) by performing the following sequence of translate, rotate and reverse translate is called pivot point rotation.

→ Steps :-

i) Translate the object so that pivot point lies at origin
ii) Rotate the object about an origin
iii) reverse translate an object back to the original position

→ The Composite transformation matrix for this sequence is obtained with the concatination of :-

$$\begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & -\sin\beta & 0 \\ \sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix}$$

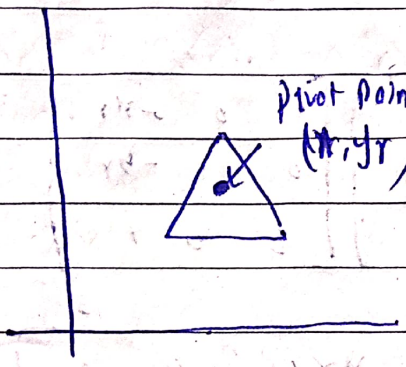
$$= \begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & -\sin\beta & -x_r\cos\beta + y_r\sin\beta \\ \sin\beta & \cos\beta & -x_r\sin\beta - y_r\cos\beta \\ 0 & 0 & 1 \end{bmatrix}$$

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

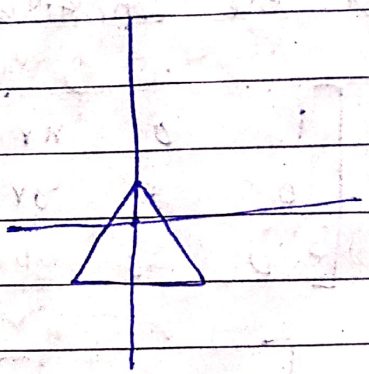
∴ Note :- New point = Composite transformation Matrix X orginal point

i.e. $$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\beta & -\sin\beta & x_r(1-\cos\beta) + y_r\sin\beta \\ \sin\beta & \cos\beta & y_r(1-\cos\beta) - x_r\sin\beta \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

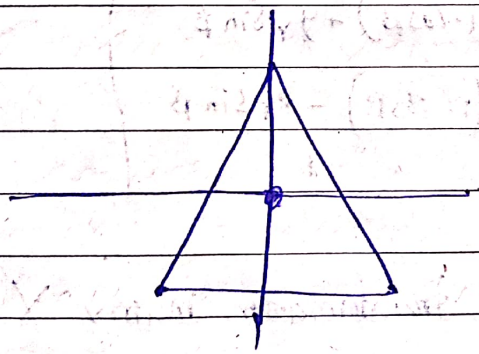
Pivot Point Scaling



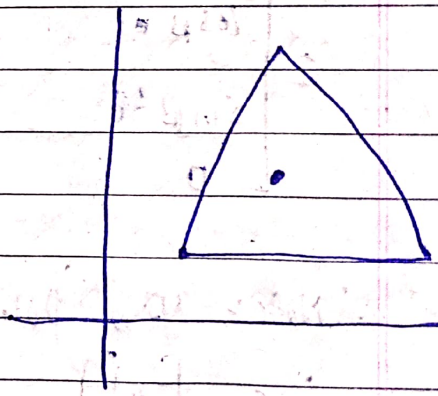
∴ Fig(a) Original position of an object



Fig(b) Translate an object such that the pivot point lies in origin



∴ Fig(c) Scale an object around origin



∴ Fig(d):- Translate an object back to it's original position

→ The transformation formed by following the sequence of translate scale and reverse (Inverse) translate operation is called pivot point Scaling.

→ Steps :-

i) Translate an object such that the pivot point lies on origin.
ii) Scale an Object around origin.
iii) Reverse translate an object back to its original position.

→ The Composite transformation matrix for this sequence of transformation is obtained as follows:-

$$= \begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & -x_r s_x \\ 0 & s_y & -y_r s_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} s_x & 0 & x_r s_x - x_r \\ 0 & s_y & y_r s_y - y_r \\ 0 & 0 & 1 \end{bmatrix}$$

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

Note:- New point = Composite transformation matrix X Original point

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

Other Transformation

→ Inspite of Basic transformation (Translation, Rotation and Scaling), there are some other type of transformation which are as follows:-

1. Reflection
2. Shear

Reflection

→ A reflection is a geometric transformation that produces the mirror image of an object.

→ The reflection is generated ~~relating~~ relative to an axies of reflection by rotating the object about 180° b about the reflection axis.

→ The axes of relection can be choose either along x-axis, either along y-axis or either along both x-axis and y-axis.

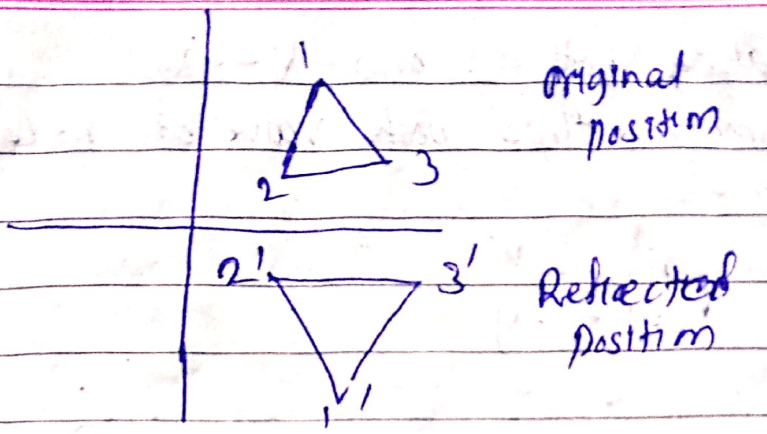
Reflection along x-axis:

→ This transformation keeps the x-value same and flips the y-coordinate position value.

i.e. $x' = x$

$y' = 0 - y$

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



∴ Fig:- Reflection of an object along X-axis.

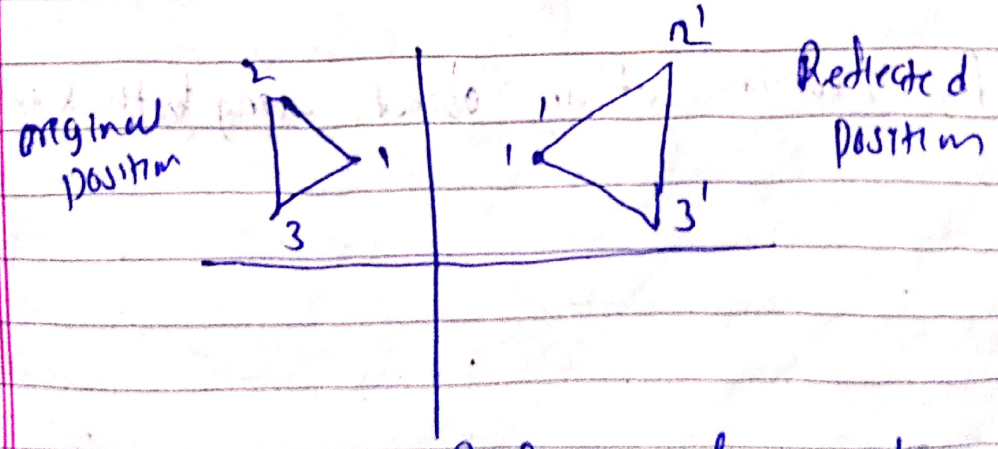
Reflection Along Y-axis:-

→ This transformation keeps the Y-value same and flips the X-coordinate position value.

i.e. $x' = -x$

$y' = y$

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



∴ Fig:- Reflection of an object along Y-axis

Date
Page

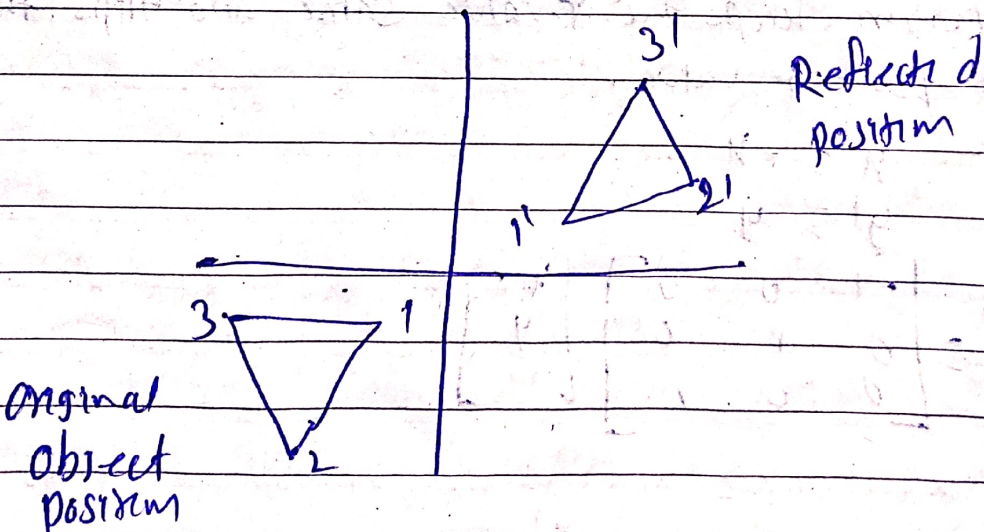
Reflection along both x and y-axis:

→ This transformation flips both value of x-coordinate and y-coordinate.

i.e. $x' = -x$

$y' = -y$

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



∴ fig: Reflection of an object along both x and y-axis

2D Shear

→ A shear is a geometric transformation which is used to produce the distorted image of an object.

→ Shearing can be chosen either along x-axis or y-axis.

Shearing about x-axis

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

$$y' = y$$

Where Sh_x = Shearing factor along x-axis

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

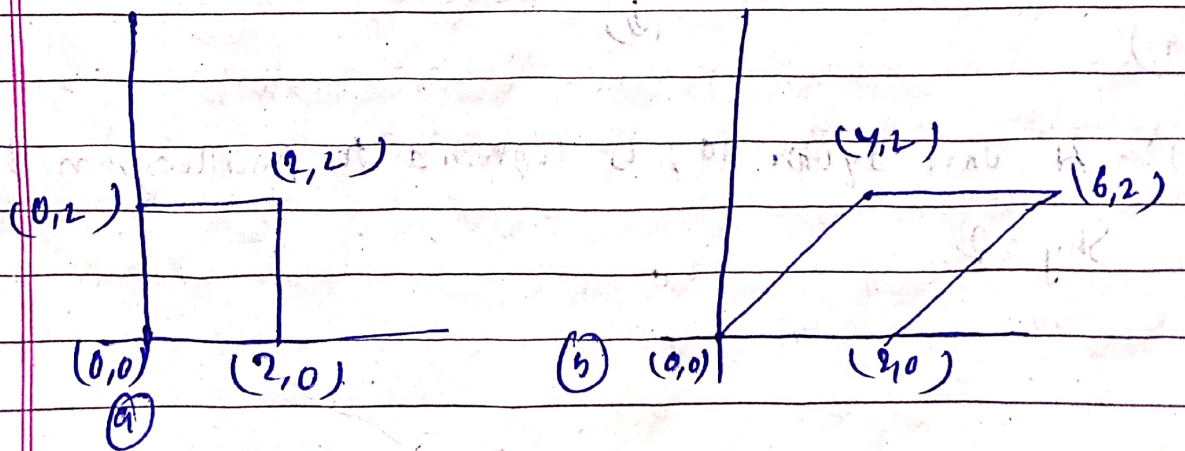


Fig:- (A) Unit Square (a) is converted to parallelograms (b) using $Sh_x = 2$.

Shearing about X-axis:-

$x' = x$

$y' = y + Shy \cdot x$

where Shy = Shearing factor along X-axis.

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

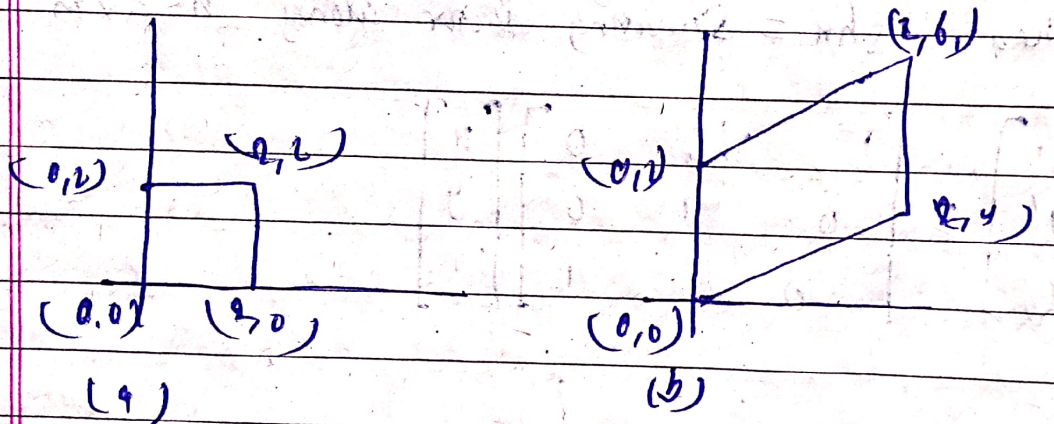


Fig:- A unit square (a) is converted to parallelogram (b) using Shy = 2

3D Transformation

→ A 3-D geometric transformation is a transformation in which a given point (x, y, z) is mapped into a new point (x', y', z').

3D Basic Transformation

i) 3D Translation
ii) 3D Rotation
iii) 3D Scaling

3D Translation

→ A translation is a basic geometric transformation which is used to move an object from one coordinate location to another coordinate location.

→ It is done by adding translation factor t_x to x-coordinate, t_y to y-coordinate and t_z to z-coordinate.

i.e.
$$x' = x + t_x$$
$$y' = y + t_y$$
$$z' = z + t_z$$

where (x, y, z) = Given point
(x', y', z') = Resultant point
(t_x, t_y, t_z) = translation factor along x, y and z axes.

→ If the translation factors t_x, t_y and t_z are positive, then the object moves away from the origin. If the ~~beam~~ t_x, t_y and t_z are negative, then the object moves towards the origin.

→ The homogeneous coordinate representation of 3-D Translation is as follows:-

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

3D Rotation

→ A rotation is a basic geometric transformation which is used to move an object on a circular path about an origin.

→ Rotation requires two major parameters :- Direction of rotation and Angle of Rotation.

→ The direction of rotation can be either clockwise or anti-clockwise

→ The clockwise rotation indicates negative angle and anti-clockwise rotation indicates positive angle.

→ In 3-D, the rotation can be done either along x-axis (yz-plane) or along y-axis (xz plane) or along z-axis (xy plane)

* Rotation Along x-axis (yz plane) :-

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

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

* **Rotation Along Y-Axis (XZ plane)**

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

$y' = y$

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

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

* **Rotation Along Z-Axis (XY plane)**

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

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

$z' = z$

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

3-D Scaling

→ A scaling is a basic geometric transformation which is used to alter (change) the size of an object.

→ It is achieved by multiplying scaling factor (S_x) by X-coordinate, Scaling factor (S_y) by Y-coordinate and Scaling factor (S_z) by Z-coordinate respectively.

i.e.

$x' = x * S_x$
$y' = y * S_y$
$z' = z * S_z$

where :- (x, y, z) = Given point in 3D
(x', y', z') = Resultant point
S_x = Scaling factor along X-axis
S_y = " " " Y-axis
S_z = " " " Z-axis

→ If the Scaling factor S_x, S_y and S_z are positive and greater than 1, then the size of object increases and if S_x, S_y and S_z are positive and less than 1, then the size of object decreases.

→ If $S_x = S_y = S_z$ then the Scaling is called Uniform Scaling otherwise the scalling is called differential Scaling.

→ The homogeneous coordinate representation of 3D Scaling is follows:-

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

3D Other Transformation

i) Reflection
ii) Shear

3D Reflection

→ A reflection is a geometric transformation which is used to produce the mirror image of an object.

→ In 3D, the reflection can be chosen either along x-axis, y-axis and z-axis.

* Reflection Along x-axis (yz plane) :-

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

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

* Reflection Along y-axis (xz plane)

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

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

* **Reflection Along Z-axis (XY-plane)**

$x' = x$

$y' = y$

$z' = -z$

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

2080-09-04

3-D Shear

→ A shear is a geometric transformation which is used to produce the distorted image of an object.

→ In 3D, the shearing can be done either along YZ plane (x-axis) or XZ plane (y-axis) or XY plane (z-axis).

* **Shearing Along YZ plane (x-axis)**

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

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

* **Shearing along XZ plane (y-axis)**

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

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

* Shearing Along XY plane (2-axis)

$$x' = x + shx \cdot z$$

$$y' = y + shy \cdot z$$

$$z' = z$$

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

Numericals problems and Solutions

1) Perform a (anticlockwise) counterclockwise 90° rotation of a triangle A(2,2), B(5,5) and C(4,3) about point(1,1).

Given:

$\beta = 90°$

$(x_r, y_r) = (1,1)$

The composite transformation matrix for fixed point rotation is

$= T(x_r, y_r) \cdot R(\beta) \cdot T(-x_r, -y_r)$

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

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

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

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

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

Now

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

For A(2,3)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$$

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

$\therefore A'(-5, 2)$

For B(5,5)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 5 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} -7 \\ 5 \\ 1 \end{bmatrix}$$

$\therefore B'(-7, 5)$

For C (4,3)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 4 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} -5 \\ 4 \\ 1 \end{bmatrix}$$

$\therefore C'(-5,4)$

∴ The resultant point are $A'(-5,2)$, $B'(-7,5)$, $C'(-5,4)$

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

Soln

Given: $S_x = S_y = 0.5$

$(x_r, y_r) = (2,2)$

The composite transformation of matrix for fixed point scaling $= T(x_r, y_r) \cdot S(S_x, S_y) \cdot T(-x_r, -y_r)$

$$= \begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} S_y & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix}$$

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

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

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

Now

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0.5 & 0 & -1 \\ 0 & 0.5 & -1 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

For A(1,1)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0.5 & 0 & -1 \\ 0 & 0.5 & -1 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.5 \\ -0.5 \\ 1 \end{bmatrix}$$

A'(-0.5, -0.5)

For B(3,1)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0.5 & 0 & -1 \\ 0 & 0.5 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.5 \\ -0.5 \\ 1 \end{bmatrix}$$

$B'(0.5, -0.5)$

For C(3,3) :-

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0.5 & 0 & -1 \\ 0 & 0.5 & -1 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 3 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \\ 1 \end{bmatrix}$$

$C'(0.5, 0.5)$

For D(1,3)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0.5 & 0 & -1 \\ 0 & 0.5 & -1 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.5 \\ 0.5 \\ 1 \end{bmatrix}$$

$D'(-0.5, 0.5)$

The resultant point $A'(0.5, -0.5)$, $B'(0.5, -0.5)$, $C'(0.5, 0.5)$ and $D'(-0.5, 0.5)$.

* Find out the final coordinates of a figure bounded by the coordinates (4,1), (3,4), (5,7), (10,3) when rotated about a point (8,8) by 30° in clockwise direction and Scaled by 2 unit in x-direction and 3 unit in Y-direction.

Soln

Given: $(x_r, y_r) = (8,8)$

$\beta = 30°$

$S_x = 2$, $S_y = 3$

The Composite transformation matrix is = ~~$T(x_r, y_r) \cdot R(\beta) \cdot T(-x_r, -y_r)$~~

$= S(S_x, S_y) \cdot T(x_r, y_r) \cdot R(-\beta) \cdot T(-x_r, -y_r)$

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

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

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

$$= \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0.866 & 0.5 & 2.928 \\ -0.5 & 0.866 & -5.072 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1.732 & 1.5 & 2.928 \\ -1 & 2.598 & -5.072 \\ 0 & 0 & 1 \end{bmatrix}$$

Now

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1.732 & 1.5 & 2.928 \\ -1 & 2.598 & -5.072 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

For A(1,1)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1.732 & 1.5 & 2.928 \\ -1 & 2.598 & -5.072 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 6.16 \\ -3.474 \\ 1 \end{bmatrix}$$

$\therefore A'(6.16, -3.474)$

For B(3,4)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1.732 & 1.5 & 2.928 \\ -1 & 2.598 & -5.072 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 3 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 14.124 \\ 2.32 \\ 1 \end{bmatrix}$$

$B'(14.124, 2.32)$

For C(5,7)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1.732 & 1.5 & 2.928 \\ -1 & 2.598 & -5.072 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 5 \\ 7 \\ 1 \end{bmatrix} = \begin{bmatrix} 22.088 \\ 8.114 \\ 1 \end{bmatrix}$$

$\therefore C'(22.088, 8.114)$

For D(10,3)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1.732 & 1.5 & 2.928 \\ -1 & 2.598 & -5.072 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 16 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 24.748 \\ -7.278 \\ 1 \end{bmatrix}$$

$\therefore D'(24.748, -7.278)$

∴ The resultanted coordinates are $A'(6.16, -3.474)$, $B'(14.124, 2.32)$, $C'(22.088, 8.114)$, $D'(24.748, -7.278)$

For point C(3,7,2)

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 3 \\ 7 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 9 \\ 5 \\ 1 \end{bmatrix}$$

$\therefore C'(4,9,5)$

$\therefore$ The resultant coordinates are $A'(2,4,5)$, $B'(5,8,5)$, $C'(4,9,5)$

* Given a triangle ABC with vertices A(1,2,3), B(4,6,2,) and C(3,7,2) perform reflection along xy plane

Soln

We know that the homogeneous coordinates representation for 3D reflection is

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

For point A(1,2,3)

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ -3 \\ 1 \end{bmatrix}$$

$\therefore A'(1,2,-3)$. And so on.

Unit :- 3 Clipping

→ Window to view port transformation
→ Clipping, line clipping
→ Cohen - Sutherland line clipping
→ Polygon clipping
→ Sutherland and Hodgman polygon clipping Algorithm

Window to view port Transmission

* Window to view port Transmission Formula

$$x_v = x_{vmin} + S_x (x_w - x_{wmin})$$

where,

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

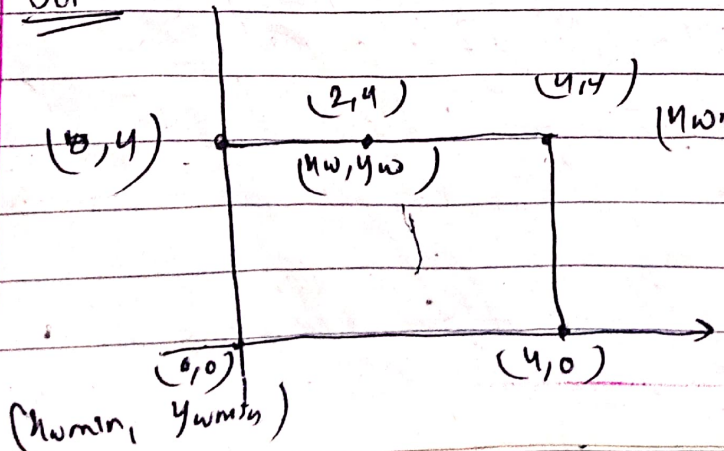
$$y_v = y_{vmin} + S_y (y_w - y_{wmin})$$

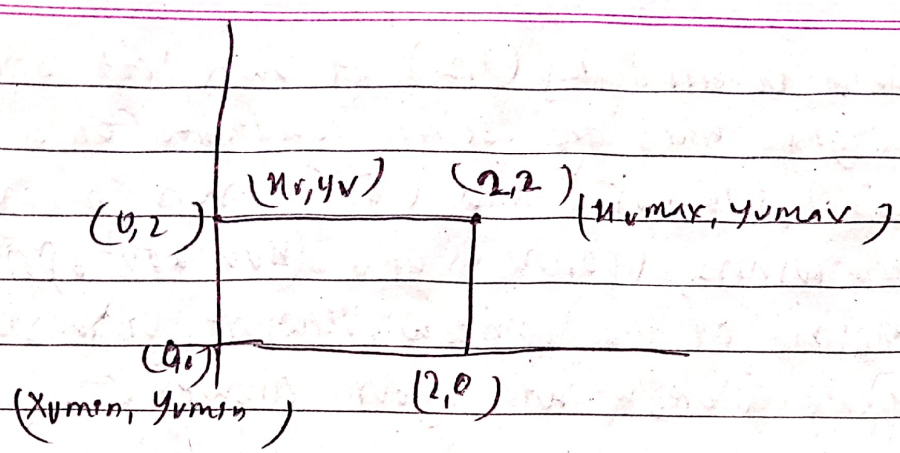
where,

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

* A window having lower lefthand corner (0,0) and Upper righthand corner (4,4) is mapped to the view port having lower lefthand corner (0,0) and upper righthand corner (2,2). If we place the window at position, (2,4), then at position we have to place the view port to maintain the same relative placement as in the window.

Soln





Given

$(x_{wmin}, y_{wmin}) = (0,0)$ $(x_{wmax}, y_{wmax}) = (4,4)$

$(x_{vmin}, y_{vmin}) = (0,0)$ $(x_{vmax}, y_{vmax}) = (2,2)$

$(x_w, y_w) = (2,4)$

$x_v = ?$, $y_v = ?$

We know that

$$x_v = x_{vmin} + S_x (x_w - x_{wmin})$$

$$x_v = x_{vmin} + \left(\frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}}\right)(x_w - x_{wmin})$$

$$= 0 + \left(\frac{2-0}{4-0}\right) \times (2-0) = 0.5 \times 2 = 1$$

Similarly

$$y_v = y_{vmin} + S_y (y_w - y_{wmin})$$

$$= 0 + \left(\frac{2-0}{4-0}\right) \times (4-0) = 0.5 \times 4 = 2$$

$\therefore x_v = 1$

$y_v = 2$

Imp

Given a window bordered by (1,2) at lower left and (16,12) at the upper right, given the screen coordinate of a triangle with vertices (3,2), (10,7.5) and (5,5) when mapped into a viewport with corners (100,100) and (400,200). provide accurate illustration of the window, viewport, untransformed and transformed triangle with your answer.

Soln

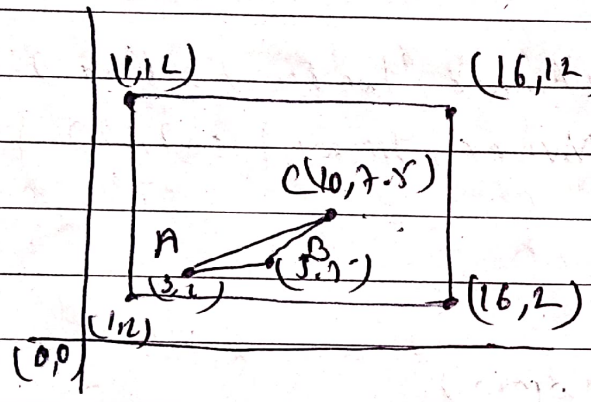


Fig:- Window port

Here,
$(x_{wmin}, y_{wmin}) = (1,2)$
$(x_{wmax}, y_{wmax}) = (16,12)$

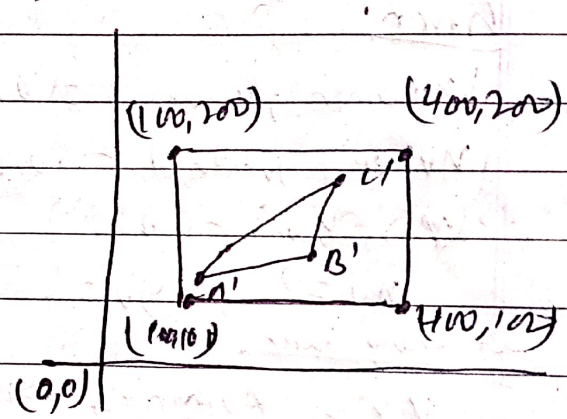


Fig:- View port

Here,
$(x_{vmin}, y_{vmin}) = (100,100)$
$(x_{vmax}, y_{vmax}) = (400,200)$

We know:-
In Windows to viewport transformation

$$x_v = x_{vmin} + S_x (x_w - x_{wmin})$$

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

Similarly

$$y_v = y_{vmin} + S_y (y_w - y_{wmin})$$

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

For A(3,2)

$(x_w, y_w) = (3, 2)$

$x_v = 100 + \frac{400-100}{16-1}(3-1) = 100 + \frac{300}{15} \times 2 = 140$

$y_v = 100 + \frac{200-100}{12-2}(2-2) = 100 + \frac{100}{10} \times 0 = 100$

$\therefore A'(140, 100)$

For B(5,5)

$(x_w, y_w) = (5, 5)$

$x_v = 100 + \frac{(400-100)}{16-1}(5-1) = 100 + \frac{300}{15} \times 4 = 180$

$y_v = 100 + \frac{(200-100)}{(12-2)}(5-2) = 100 + \frac{100}{10} \times 3 = 130$

$\therefore B'(180, 130)$

For C(10,7.5)

$(x_w, y_w) = (10, 7.5)$

$x_v = 100 + \frac{(400-100)}{(16-1)}(10-1) = 100 + \frac{300}{15} \times 9 = 280$

$y_v = 100 + \frac{(200-100)}{(12-2)}(7.5-2) = 100 + \frac{100}{10} \times 5.5 = 155$

$\therefore C'(280, 155)$

$\therefore$ The result :- $A'(140, 100)$, $B'(180, 130)$ and $C'(280, 155)$

2-D Viewing Pipeline

i) Modeling Coordinate
ii) Word Coordinate
iii) Viewing Coordinate
iv) Normalized Viewing Coordinate
v) Device Coordinate

Prob

Reflection about line Y-axis

Composite Transformation Matrix = $T(0,c) \cdot R(\beta) \cdot Reflex \cdot R(-\beta) \cdot T(0,$

Where :- $\beta = \tan^{-1}(m)$.

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & c \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos\beta & -\sin\beta & 0 \\ \sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} -\cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

Pivot point Rotation

* $[A * B * C]$
  $[A * (B * C)]$
  $[A * M]$

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

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

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

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

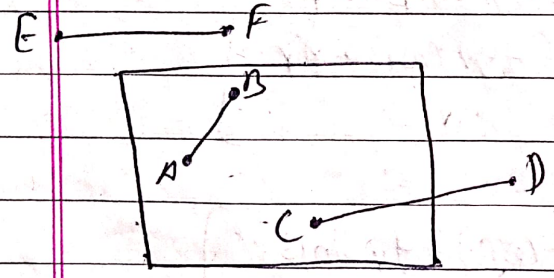
Clipping

→ Any procedure that identifies those portions of a picture that are either inside or outside of a specified region of space is referred to as clipping algorithm or simply Clipping.

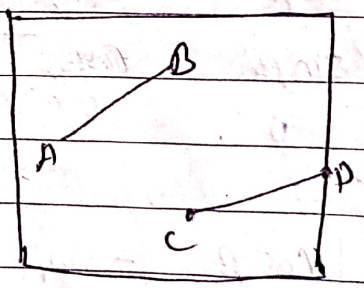
* Application of Clipping

* Point Clipping

* Line Clipping



: Before Clipping



: After Clipping

* Rules for the visibility of the line:

i) If both the end points have bit code 0000 the line is visible.

ii) If atleast one of the end point in non zero and
   a) The logical "AND"ing is 0000 then the line is partially visible.
   b) If the logical "AND"ing is non-zero then line is not visible.

Bit 1	Bit 2	Bit 3	Bit 4

∴ Fig:- Region Code

Formula for calculating region code for point (x, y)

Bit 1 = Sign $(y - y_{max})$

Bit 2 = Sign $(y_{min} - y)$

Bit 3 = Sign $(x - x_{max})$

Bit 4 = Sign $(x_{min} - x)$

Where,

$$\text{sign } a = \begin{cases} 1 & \text{if } a \text{ is positive or zero} \\ 0 & \text{otherwise} \end{cases}$$

* Let R be the Rectangular window whose lower left hand corner is at L(-3,1) and upper right hand corner is at R(2,6). Find the end point codes for the following line using Sutherland - Cohen algorithm and write the clipping category for line AB, CD, EF, GH, IJ:

A(-4,3) B(-1,9)
C(-2,5) D(4,8)
E(-2,3) F(1,2)
G(-1,-2) H(3,3)
I(-4,7) J(-2,10)

Soln

Given $(x_{max}, y_{max}) = (2, 6)$

$(x_{min}, y_{min}) = (-3, 1)$

Bit 1 = Sign $(y - y_{max})$ = Sign $(y - 6)$

Bit 2 = Sign $(y_{min} - y)$ = Sign $(1 - y)$

Bit 3 = Sign $(x - x_{max})$ = Sign $(x - 2)$

Bit 4 = Sign $(x_{min} - x)$ = Sign $(-3, -x)$

Where,

$$\text{Sign } a = \begin{cases} 1 & \text{if } a \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Now,

Region code for point A $(-4, 3)$ = 0001

Region code for point B $(-1, 9)$ = 1000

Region code " " C $(-2, 5)$ = 0000

" " " " D $(4, 8)$ = 1010

" " " " E $(-2, 3)$ = 0000

" " " " F $(1, 2)$ = 0000

" " " " G $(-1, -2)$ = 0101

" " " " H $(3, 3)$ = 0010

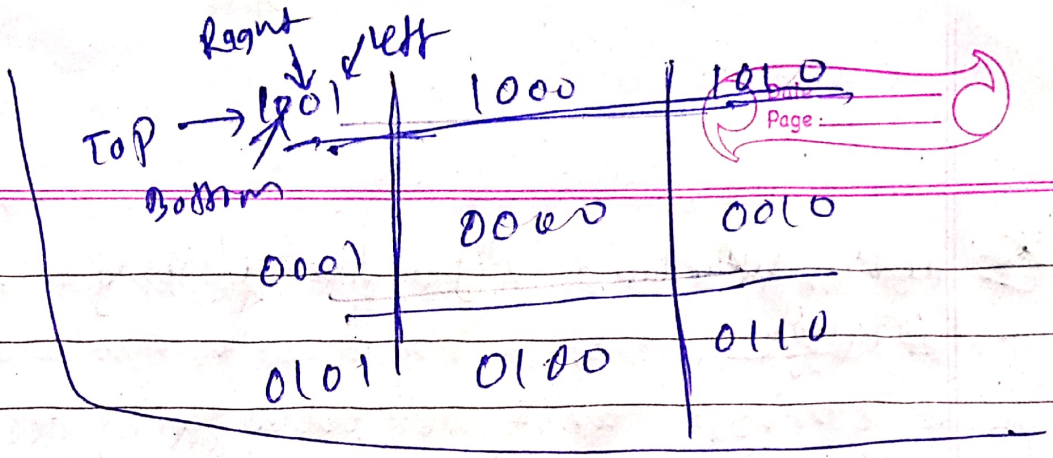
" " " " I $(-4, 7)$ = 1001

" " " " J $(-2, 10)$ = 1000

For AB line :-

$$\begin{array}{r} 0001 \\ \text{AND } 1000 \\ \hline 0000 \end{array}$$

∴ So AB is partially visible line



For CD

$$\begin{array}{r} 0000 \\ \text{AND}\ \ 1010 \\ \hline 0000 \end{array}$$

∴ So, CD is partially visible line

For EF

∴ Since, both ~~reg~~ end points have region code 0000 So EF is visible line.

For GH

$$\begin{array}{r} 0100 \\ \text{AND}\ \ 0010 \\ \hline 0000 \end{array}$$

∴ So, GH is partially visible line

For IJ

$$\begin{array}{r} 1001 \\ \text{AND}\ \ 1000 \\ \hline 1000 \end{array}$$

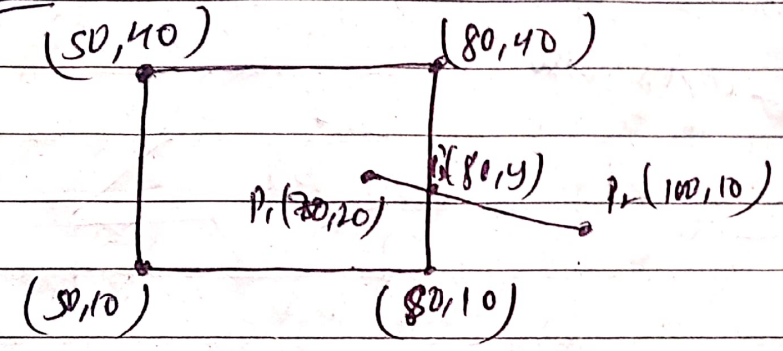
∴ So, IJ is invisible line

* Cohen Sutherland ~~Algorithm~~ line Clipping Algorithm

(1) Assign the region code for each end-points of a line.
(2) If both end-points have region code 0000 then accept that line.
(3) Else perform logic AND operation for both region code.
  (3.1) If the result is not 0000, then reject that line
  (3.2) Else perform clipping operation
    (3.2.1) Choose an endpoint of a line that is outside clipping window.
    (3.2.2) Find the intersection point at the window boundary.
    (3.2.3) Replace end-point with intersection point and update region code.
    (3.2.4) Repeat Step (2) until the line is accepted or rejected.
(4) Repeat from Step (1) for other lines.

* Use the cohen-sutherland algorithm to clip line $P_1(70,20)$ and $P_2(100,10)$ against a window lower lefthand corner (50,10) and upper righthand corner (80,40).

Soln



Given

Region code for $P_1 = 0000$

Region code for $P_2 = 0010$

AND

0000

So, P_1 and P_2 is partial visible line. Hence, we need clipping

We know,

Slope of P_1P_2 = Slope of P_2P' $\left(\text{Slope}(m) = \frac{y_2 - y_1}{x_2 - x_1}\right)$

or, $\frac{10-20}{100-70} = \frac{10-y}{100-80}$

or, $\frac{-10}{30} = \frac{10-y}{20}$

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

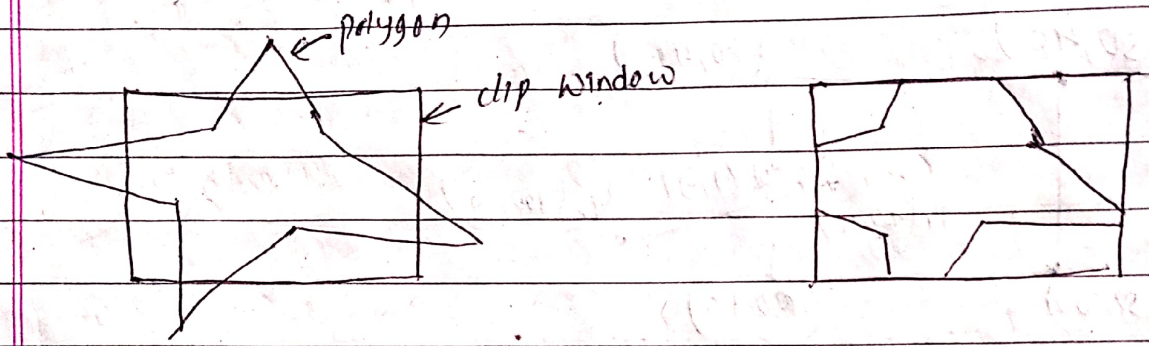
or, $10 + \frac{20}{3} = y$

$\therefore y = 3\cancel{3333}3$ 10.6667

$\therefore$ So, the intersection point of the windows boundary is (80, 16.6~~7~~)

Polygon Clipping

→ A polygon clipping is defined as the process of removing those part of a polygon which lies outside the clipping window.



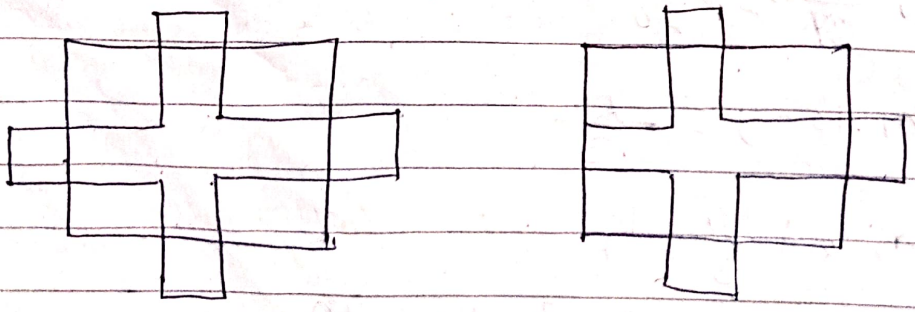
∴ Fig (a): Before clipping ∴ Fig (b): After clipping

Sutherland-Hodgman Polygon Clipping Algorithm

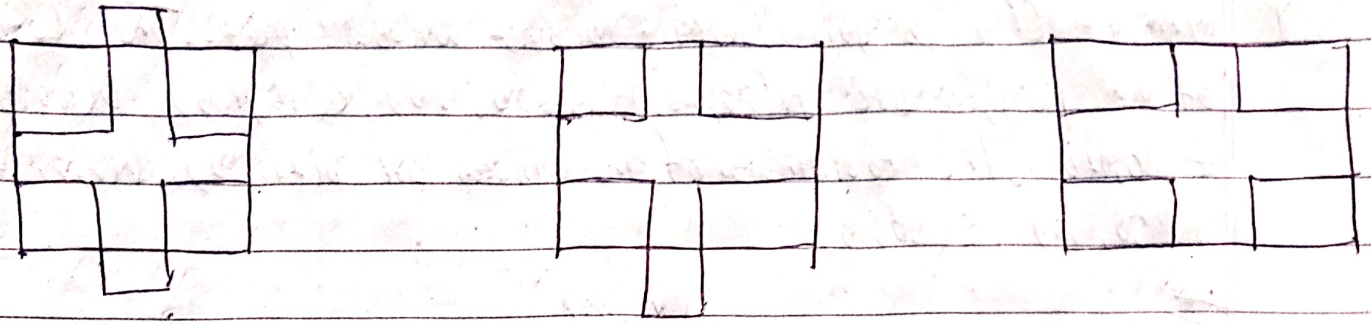
→ Sutherland-Hodgman polygon clipping uses davide and conquer strategy to clip the given polygon against the edges of the clip-window.

→ It starts with the initial set of polygon vertices.

→ First start with the clipping polygon against the left boundary of the window. Then successively against the right boundary, bottom boundary and finally against the top boundary as shown in below figure:



∴ Fig (a): Original figure ∴ Fig (b) Clipping against left boundary



Fig(c) Clipping against right boundary

Fig(d) Clipping Against top boundary

Fig(e) Clipping Against bottom boundary

Algorithm

For each clip window boundary perform the following steps:-

(1) Construct an input vertex list $(v_0, v_1, v_2 \dots v_n)$ where $v_0 = v_n$.

(2) Create empty Output vertex list.

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

a) If out-in (v_i is outside the window boundary and v_{i+1} is inside)
- Add intersection point v_i' and v_{i+1} to the output vertex list.

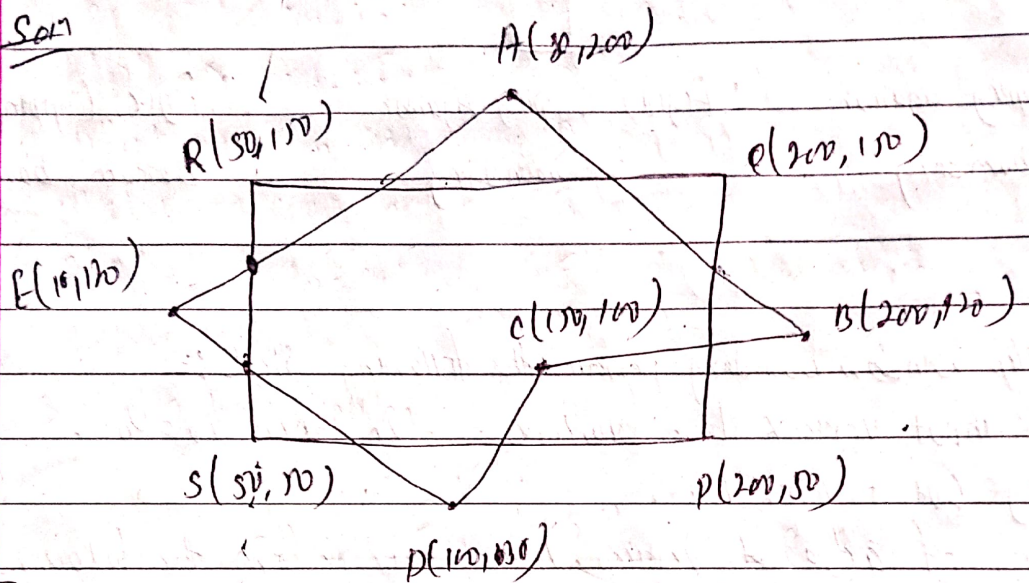
b) If in-in (both v_i, v_{i+1} are inside the window boundary)
- Add v_{i+1} to the Output vertex list

c) If in-out (v_i is inside the window boundary and v_{i+1} is outside)
- Add intersection point v_i' to the Output vertex list.

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

* Example:- Clip polygon ABCDE against window PQRS. The co-ordinates of the polygon are A(80,200), B(220, 120), C(130, 100) D(100,30), E(10,120). The coordinates of the window are P(200,50), Q(200,150), R(50,150), S(50,50)

Soln



Step (1)

Clipping polygon against left edge of the window (left Clipping)

Vertex	Case	Output vertex	Remarks
AB	in - in	B	
BC	in - in	C	
CD	in - in	D	
DE	in - out	D'	New vertex
EA	out - in	E'A	New vertex

Step (2) :- Clipping polygon against right edge of the window (Right Clipping

Vertices	Case	Output vertex	Remarks
AB	in-out	A'	New vertex
B'C	out-in	B'C	New vertex
CD	in-in	D	
DD'	in-in	D'	
D'E	in-in	E'	
E'A	in-in	A	

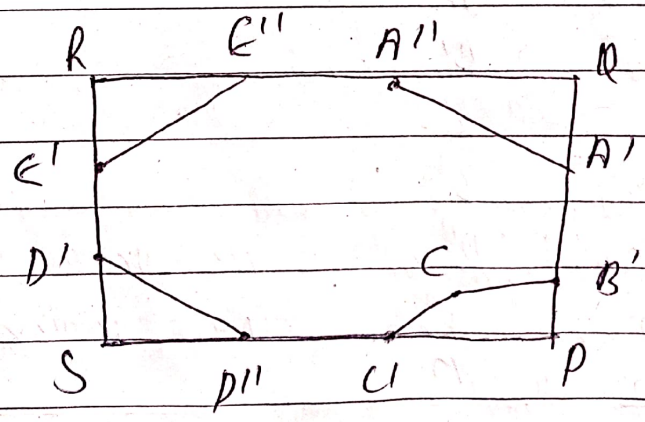
Step (3) Clipping polygon against bottom edge of the window (Bottom Clipping)

Vertices	Case	Output vertex	Remarks
AA'	in-in	A'	
A'B'	in-in	B'	
B'C	in-in	C	
CD	in-out	C'	
DD'	out-in	D''D'	New vertices
D'E'	in-in	E'	New vertex
E'A	in-in	A	

Step (4) Clipping polygon against top edge of the window (Top Clipping)

Vertices	Case	output vertex	Remarks
AA'	out-in	A''A'	New vertex
A'B'	in-in	B'	
B'C	in-in	C	
CC'	in-in	C'	
C'D''	in-in	D''	
D''D'	in-in	D'	
D'E'	in-in	E'	
E'A''	in-out	E''	New vertex

So After clipping



Unit-4 Visible Surface Determination and Computer Graphics Algorithm

→ Image Space and object Space techniques
→ Hidden Surface Removal - Depth Comparison
→ Z-Buffer Algorithm.
→ Back-Face Removal
→ The painter's Algorithm
→ Scan-Line Algorithm
→ Light and Color and different Color Models (RGB, CMY, YIQ)

object

Date
Page

Visible Surface Detection Method

Visible Surface Detection Method or Hidden Surface removal is major concern for realistic graphics for identifying those parts of a scene that are visible from a chosen viewing position.

The Two Approaches are:-

a) Image Space methods:- Visibility is detected point by point at each pixel position on the projection plane.

b) Object-Space Method:- Compares objects and parts of objects to each other within the scene definition to determine which surface as a whole we should label as visible.

Sn	Object space	Sn	Image Space
1.	Object Image space is object based. It concentrates on relations among objects in the scene.	1.	It is a pixel based method. It is concerned with the final image, what is visible within each raster pixel.
2.	Here Surface visibility is determined	2.	Here line visibility or point visibility is determined.
3.	It is performed at the precision with which each object is defined, No resolution is considered.	3.	It is performed using the resolution of the display device.
4.	Calculations are not based on the resolution of the display so change of object can be easily adjusted.	4.	Calculations are resolution base, so the change is difficult to adjust
5.	Object-based algorithms Operate on continuous object data.	5.	These Operate on object data.
6.	These were developed for vector graphics system.	6.	These are developed for raster devices.

7. Vector display used for object method has large address space.	7. Raster Systems used for image space methods have limited address space.
8. Object precision is used for application where speed is required.	8. They are suitable for application where accuracy is required.
9. It requires a lot of calculations if the image is to enlarge.	9. Image can be enlarged without losing accuracy.
10. If the number of objects in the scene increases, computation time also increases.	10. In this method complexity increases with the complexity of visible parts.

Z-Buffer Algorithm

* Algorithm :-

(1) START

(2) Initialize the buffer values
   i) DepthBuffer (x,y) = 0
   ii) FrameBuffer (x,y) = Ibackground

(3) Process each polygon (one at a time)
   i) For each ~~processed~~ projected (x,y) pixel position of a polygon, calculate depth z.

   a) if z > depthbuffer (x,y)
      → Compute Surface Color (i.e. Zsurface (x,y))
      → Set depthBuffer (x,y) = z
      → framebuffer (x,y) = Zsurface (x,y)

(4) STOP

→ After all the surfaces are processed, the depth buffer contains the depth value of the visible surface and refresh buffer contains the corresponding intensity value for that surface.

→ The depth value of the surface position (x,y) is calculated by plane equation of surface.

$$z = \frac{-Ax - By - D}{C}$$

let, depth z' at position $(x+1, y)$

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

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

∴ Here, $-A/C$ is constant for each surface, so corresponding depth value across a scan line are obtained from preceeding values by simple calculation.

* Advantages of Z-Buffer Algorithm

i) It is easy to implement.
ii) It can be implement in hardware to overcome the speed problem.
iii) The Z-value of a polygon can be calculated incrementally.
iv) No object-object comparison is required

* Disadvantages of 2-Buffer Algorithm

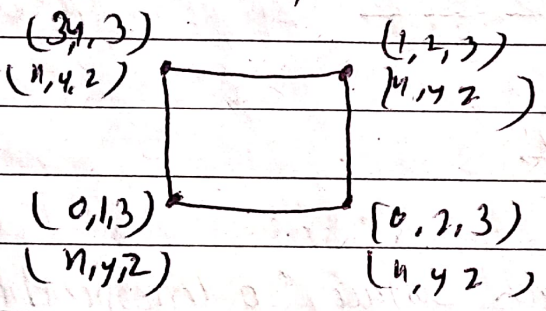
i) This method requires additional buffer (if compared with depth sort method) and the overhead involves in updating the buffer.

ii) It requires large memory

iii) It is time consuming process

Example:-

Assume the polygon is given as follows:-



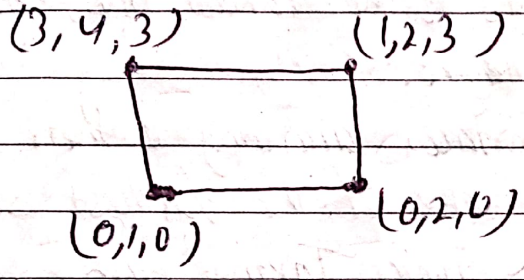
In starting, assume that the depth of each pixel is infinite

∞	∞	∞
∞	∞	∞
∞	∞	∞
∞	∞	∞

The depth value at every place in the given polygon is 3 on applying the algorithm, the result is:

3	3	3
3	3	3
3	3	3
3	3	3

Now, let's change the Z-values, in the given figure. Z-values goes from 0 to 3. i.e.



Now, the Z-value generated on the pixel will be

3	3	3
2	2	2
1	1	1
0	0	0

The Painter's Algorithm

→ The painters algorithm uses both object space and image space method for visible surface detection.

→ The painter's algorithm is also called depth Sort algorithm or priority fill algorithm.

→ The painter algorithm simulates how a painter typically produces his/her painting by starting with the background and then progressively adding new (Nearer) objects to the canvas. Thus, each layer of paint covers up the previous layer.

→ Similarly, We first Sort Surfaces according to their distance from the view plane. The intensity value of the farthest surface are then entered into the refresh buffer. Taking each succeeding surface in turn (in decreasing depth order), we paint the surfaces intensities onto the frame buffer over the intensities of the previously processed surfaces.

→ The Conceptual steps that are performed in depth-Sort algorithm are:-

i) Sort all polygons according to the decreasing depth or largest value of z.

ii) Now, Scanning to convert the various surfaces which is in the order starting with the surface which has greatest depth.

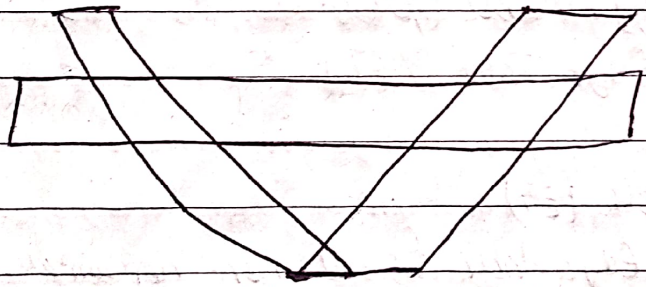
iii) Comparing is to be done on the basis of various overlapping surfaces so that the ~~user~~ user will determine which surface is to be kept visible.

iv) In the refresh buffer, enter the intensity value for the determined surface i.e. the surface which is determined to be visible.

v) The above process is going to be repeat for all the available surfaces.

Problem

One of the major problem in this algorithm is intersecting polygon surfaces as shown in the below figure



→ Different polygons may have same depth.

→ The nearest polygon could also be farthest.

We cannot use simple depth-sorting algorithm to remove the hidden surfaces in the images.

Solutions

For intersecting polygons, We can split one polygon into two or more polygons which can then be painted from back to front. This needs more time to compute intersection between polygons. So, it becomes complex algorithm for ~~such~~ such surface existence.

Scan-line Algorithm

→ Scanline method is an image space method for visible surface detection.

→ This method computes and compares depth values to create an image one scan line at a time.

→ This algorithm is similar to Scan line Algorithm for polygon filling but it deals with multiple surfaces.

→ In Scan line method, three important tables are maintained:-

a) The Edge Table (ET)
b) The Polygon Table (PT)
c) Active Edge Table (AEL)

a) Edge Table (ET)

→ Created an Edge table for all non-horizontal edges of all polygons projected on the viewplane.

→ Edge Table process:-

- Entries in the ET are stored into bucket based on each edge's smaller y-coordinates and within bucket are ordered by increasing X-coordinate.

→ Edge Table fields:-

- The X-Coordinate of the end with the smaller y-coordinate.
- The Y-Coordinate of the edge's other end.
- The X-increment ΔX used in stepping from one Scanline to the next.
- Polygon identification number, indicating the polygon to which the edge belongs.

- Pointers into the Surface face table to connect edges to surfaces.

x	y_{max}	Δx	ID	→

b) Active Edge Table (AEL)

Fields

→ Scanline

→ Edge

→ Surface

Scanline	Edge	Surface

c) Polygon Table (PT)

→ Polygon ID

→ The coefficients of the plane equation

→ Shading or color information for the polygon

→ An "In-Out" Boolean Flag, initialized to false and used during Scan-line processing

ID	Plane Equation	Shading Information	In-Out

→ The Scan-line method works as follows:

i) To facilitate the search for surfaces crossing a given scan-line an active list of edges is formed for each scan-line as it is processed.

ii) The active list stores ~~edges~~ only those edges that cross the scan-line in order of increasing x.

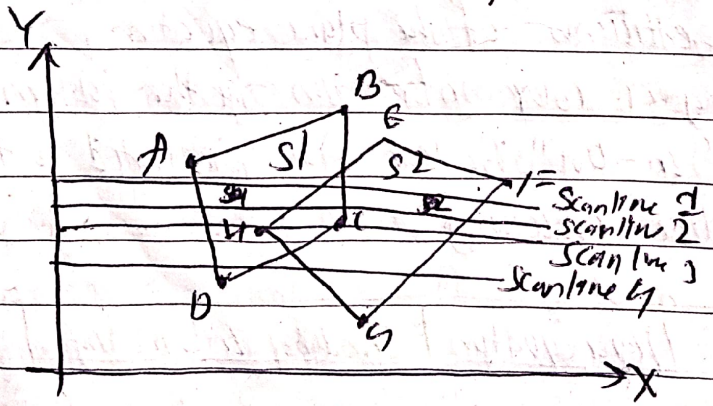
iii) Also a flag is set for each surface to indicate whether a position along a scan-line is either inside or outside the surface.

iv) Each scan-line are processed from left to right for pixel positions.

v) The surface flag is turned on (true) when the scan line intersect left edge of the surface and the flag is turned off (false) when cross from right edge of the surface.

vi) When Multiple surfaces flags are turned on for a scanline position, then need to perform depth calculations.

Scanline Method: Example



Sorted edges:- AD, DC, GH, FG, BC, HE, AB and EF

Polygon ~~has~~ table has entries for ABCD and EFGH

Prepare an Active Edge Table (AET):-

Active Edge Table	
Scanline	Edges
1	AB, BC, HE, FG
2	AD, HE, BC, FG
3	AD, HE, BC, FG
4	AD, DC, GH, FG

← Sorting from left to right

Advantages :-

→ Any number of overlapping surfaces are processed.
→ Takes advantages of coherence principle (there are any regularities in the scene)
→ Can be applied to non-polygonal object
→ Deals with any surface (Transparent, transiucent and opaque)

Disadvantages :-

→ It doesnot work for surfaces that cut through or cyclically overlap each other.
→ Depth calculations needs to performed for overlapping polygons.
→ Additional memory buffer is required.
→ Complex.

Back-Face Algorithm (Plane Equation Method)

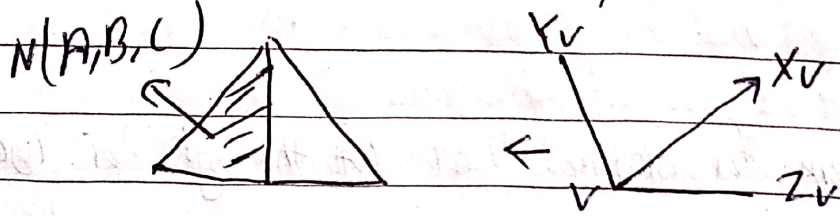
A fastest and simple object space method used to remove hidden surface from a 3D object drawing is known as "plane Equation" and applied to each side after any rotation of the object take place. It is commonly known as back-face detection of a polyhedron is based on the "inside-outside" tests.

A point (x, y, z) is inside a polygon surface of

$$Ax + By + Cz + D < 0$$

We can simplify this test by considering the normal vector N to a polygon surface which can cartesian components (A, B, C)

If V is the vector in viewing direction from the eye position then this polygon is a backface if, $V \cdot N > 0$



In the equation $Ax + By + Cz + D = 0$, if A, B, C remains constant, then varying value of D results in a wholy family of parallel planes. One of which $(D=0)$ contains the origin of the co-ordinates system an.

If $D > 0$, plane is behind the origin (Away from observer)

If $D < 0$, plane is in front of origin (towards the observer)

If we clearly defined an object to have centred at origin, then all those surface that are viewable will have negative D and unviewable surface have positive D.
So, simply our hidden surface removal routine defines the plane corresponding to one of 3D surface from the co-ordinate of 3 point on it and computing D, visible surface are detected.

Light and Colors

Color Model:-

i) RGB → RED, GREEN, BLUE
→ Widely used in computer graphics and Image processing

ii) YIQ
→ Chrominance (I and Q), Luminance (Y)
→ Used for US TV Broadcast
→ Y channel containing luminance information and I and Q channel carried the color Information

iii) CMY and CMYK
CMY – Cyan – Magenta – Yellow
→ ~~Pen~~ printing

Unit:-5 Animation and Virtual Reality

→ Basic principles of Animation and Types of Animation
→ Introduction to the flash interface
→ Setting Stage dimensions, Working with panels, Panel layouts
→ Layers and Views
→ Shaping Objects - Overview of Shapes, Drawing and Modifying Shapes
→ Bitmap Images and Sounds
→ Animation - Principles, Frame by frame animation, Tweening, Masks
→ Introduction to Virtual reality

Animation

→ Animation is the process of displaying still images in a rapid sequence to create the illusion of movement.

→ Animation is espically useful for illustrating concepts that involve movement. Concept such as playing guitar or hitting a cricket ball is difficult to explain using a text or a single photograph. Animation makes it easier to portray these aspects of multimedia application.

Steps of Animation Sequence Designing

i) Animation Story layout
ii) Animation Object Definition
iii) Frame Specification
iv) Generation of in-between frames

Basic Principles of Animation

i) Squash and Stretch
ii) Anticipation
iii) Arcs
iv) Slow in - Slow out
v) Appeal
vi) Timing and Spacing
vii) Exaggeration
viii) Staging
ix) Secondary Action
x) Follow Through
xi) Overlap
xii) 3D Effect

Types of Animation

i) Traditional Animation
ii) 2-D Animation
iii) 3D Animation
iv) Motion Graphics
v) Stop Animation

Introduction to Virtual Reality (VR)

→ Virtual reality is an artificial reality that makes users to feel in a virtual environment by using the computer hardware and software.

→ Virtual reality is a computer generated, immersive (or wide field), multisensory information program which tracks a users in a real time.

→ It is used in application areas like aircraft pilot training, training for surgical procedures, engineering and scientific visualization, computer games etc.

Advantages:-

i) Interaction with the environment
ii) User Interface
iii) User can see and even feel the shaped surface under his/her fingertips.
iv) Flight Simulators and games.
v) CAD/CAE
vi) Biomedical Engineering the projects mentioned are use of virtual reality for viewing of X-RAYs and MRI's.

viii) Rendering and 3-D lighting, Modeling for resource management

Disadvantages:-

i) New technologies have also revealed new problems.
ii) VR in medical treatment is going through some growing pains.
iii) There are limitations with VR devices as well in regards to usability.
iv) Lack of standardization of hardware and protocols.
v) Most troublesome are the side effects it can induce, like disorientation, dizziness and nausea.
vi) People often find navigating in 3-D ~~Star~~ Spaces and performing actions in free space extremely difficult.
vii) Practical problems in Spatial Cognition research.

Components

i) Input processor
ii) Simulation processor
iii) Rendering processor
iv) World Database

Types of virtual reality

i) Non - Immersive
ii) Immersive
iii) Semi - Immersive

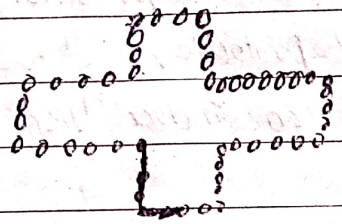
Application

i) Industrial and Manufacturing
ii) Healthcare
iii) Education
iv) Military
v) Engineering
vi) Retail
vii) Marketing and Advertising
viii) Emergency Response

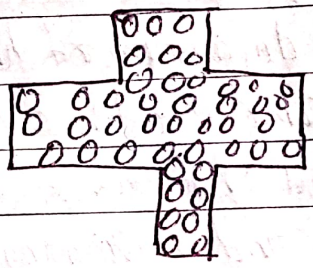
Questions

What do you mean by area filling? Explain boundary fill algorithm.

→ Area filling or Region filling is the process of filling image or region. Filling can be of boundary or interior region as shown in fig. Boundary Fill Algorithms are used to fill the boundary and flood-fill algorithm are used to fill the interior.



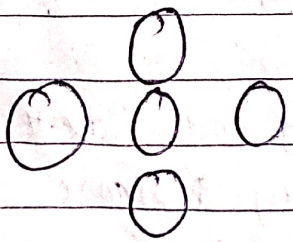
Boundary Fill region



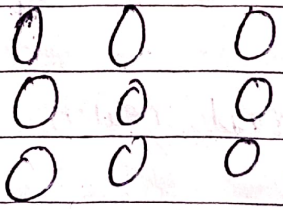
Interior Flood Filled Region

* Boundary Filled Algorithm:

This algorithm uses the recursive method. First of all, a starting pixel called as the seed is considered. The algorithm checks boundary pixel or adjacent pixels are colored or not. If the adjacent pixel is already filled or colored then leave it, otherwise fill it. The filling is done using four connected or eight connected approaches.



Four Connected



Eight Connected

Four connected approaches is more suitable than the eight connected approaches.

i) Four Connected Approaches :- In this approach, left, right, above, below are tested.

ii) Eight Connected Approaches :- In this approach, left, right, above, below and four diagonals are selected.

Boundary can be checked by seeing pixels from left and right first. Then pixels are checked by seeing pixels from top to bottom. The algorithm takes time and memory because some recursive calls are needed.

* Problem With Recursive Boundary Fill Algorithm

It may not fill regions sometimes correctly when some interior pixel is already filled with color. The algorithm will check this boundary pixel for filling and will found already filled so recursive process will terminate. This may vary because of another interior pixel unfilled.

Define horizontal and vertical retrace of electro bean. Explain architecture of random scan display system.

Define refresh rate and persistance of monitor. Explain architecture of random scan display system with block diagram.

Horizontal Retrace :- On raster-scan display, at the end of each scan line, the electro bean returns to the left side of the screen to begin displaying the next scan line, it is called the horizontal retrace of the electron beam.

Vertical Retrace :- On raster-scan display, at the end of each frame (displayed in 1/80th to 1/60th of a second), the electron bean returns to the top of the left corner of the screen to begin the next frame. This is called vertical retrace of the electro beam.

Refresh rate :- It can be defined as the number of times the screen is refreshed per second. The term refresh rate refers to how many times per second your monitor can display a new image which is measured in hertz (Hz).

Persistence :- It is defined as the time it takes the emitted light from the screen to decay to one tenth of its original intensity. It basically used to describe something that lasts for a long time.

* Random Scan Display / ~~Raster~~ Vector Display

In Random-Scan display electron beam is directed only to the areas of screen where a picture has to be drawn. It is also called vector display as it draws picture one line at time. It can draw and refresh component lines of a picture in any specified sequence. A number of lines regulates refresh rate in random-scan graphics. ~~An~~ area of memory called refresh ~~buffer~~ display files stores picture definition as a set of line drawing commands. The system returns back to first line command in the list, after all the drawing commands have been processed.

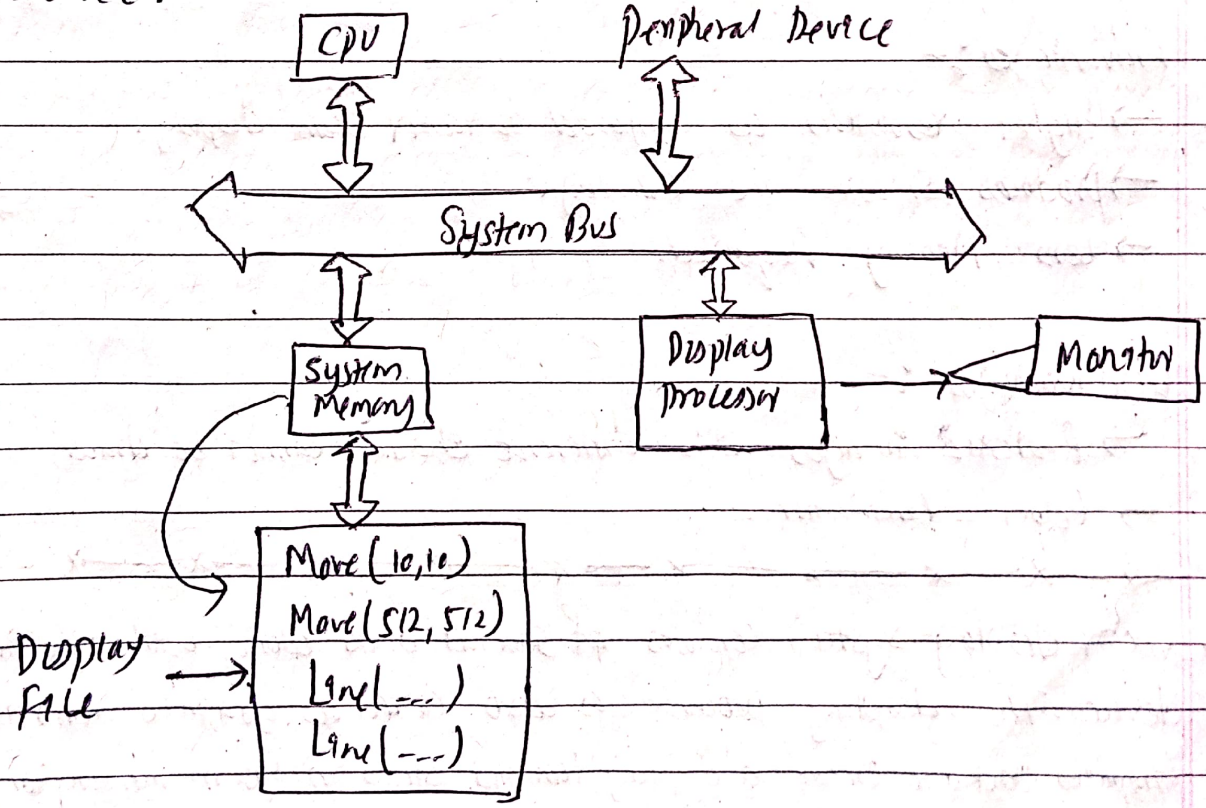


Fig:- Architecture of vector Display System

Random-Scan Display processor :- Input in the form of an application program is stored in the system memory along with graphics package. Graphics package translates the graphic commands in application program into a display file stored in System Memory. This display file is then accessed by the display processor to refresh the screen. The display processor cycles through each command in the display file program. Sometimes the display processor in a random-scan is referred as Display processing chip / Graphic Controller.

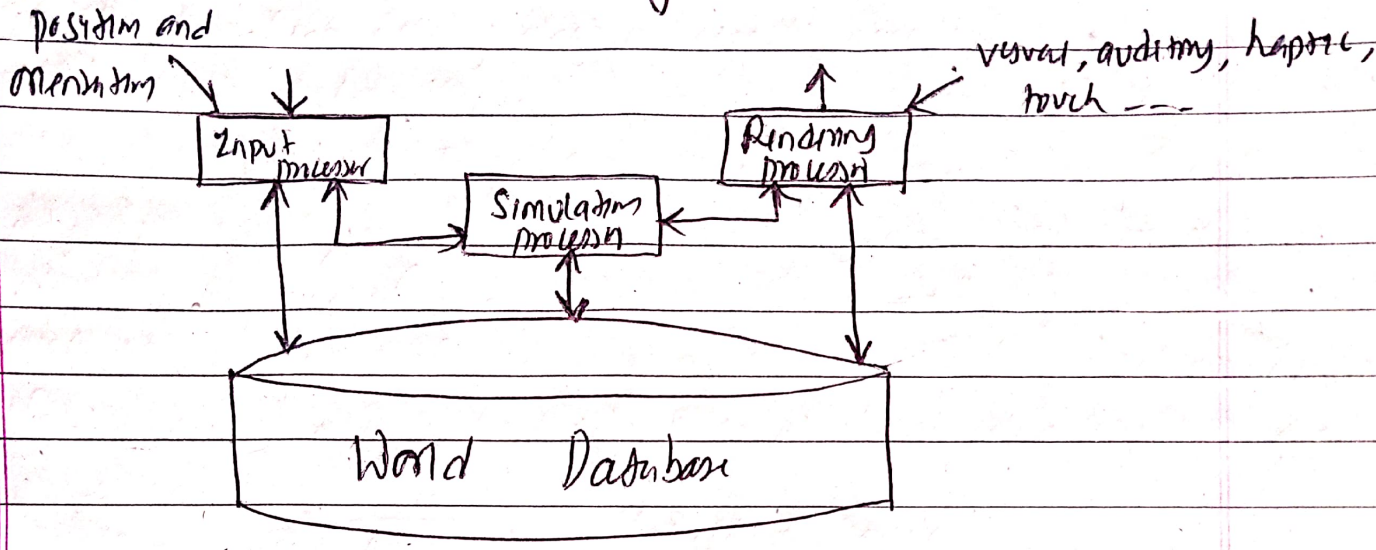
Advantages :-
→ Higher resolution as compared to raster scan display.
→ Produces smooth line drawing.
→ less Memory Required.

Disadvantages :-
→ Realistic images with different shades cannot be drawn
→ Colour limitation.

→ Vector display System consists of several units along with the peripheral devices. The display processor is also called as graphic controller.
→ Graphics package creates a display list and stores in System memory (consist of points and line drawing commands) called display list or display file.
→ Refresh time around 30 cycle per second
→ Vector display technology is used in monochromatic or beam-penetration color CRT.
→ Graphics are drawn on a vector display System by directing the electron beam along component line.

x - y

Explain the architecture of VR System with necessary components.



∴ Fig:- Architecture of a VR System

As we can see in the image above, the input processor controls the device used to input information to the computer and send the coordinate data to the rest of the system (mouse, trackers and the voice recognition system) in a reduced time-frame.

The Simulation processor represents the code of the VR System. It takes the user input along with any tasks programmed and determines the actions that will take place in the virtual world.

The rendering processor creates the sensations, the output to the user. Different rendering processes are used for haptic, visual or auditory sensations.

A VR System also has a world database, which stores the objects from the virtual World.