Introduction to Computer Graphics -

→ Computer graphics involves display, manipulation & storage of pictures and experimental data for proper visualization.

→ Computer graphic systems called as active or passive.

→ In both the cases input to the system is a scene description & output is a static or animated scene to be displayed. The end product of computer graphics is a picture.

→ Picture → In broad sense it is a collection of lines, points and text to be displayed on a graphics device.

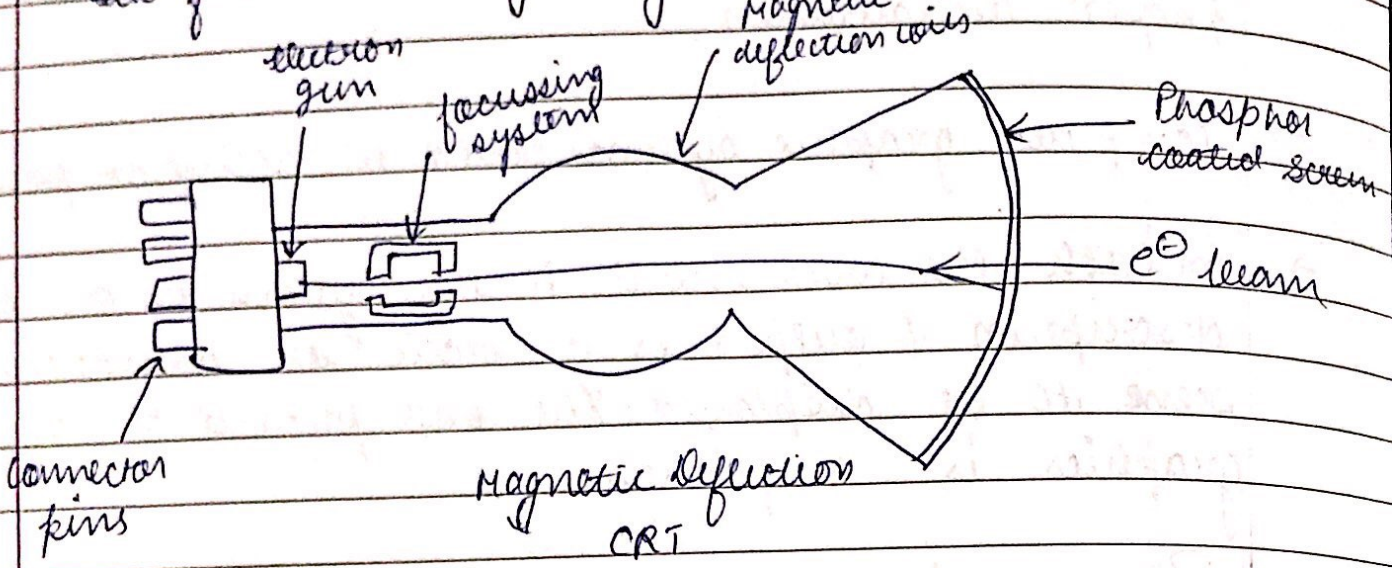
Q.) write the name of the OS which first introduced concept of graphics?

Ans. windows 73.

→ Computer graphics aims to duplicate the manner in which humans perceive & understands objects (3D). By representing them in various ways.

→ Computer graphics system comprises of a host computer with support of a fast processor (graphics card), large memory, frame buffer & other hardware devices such as display devices, I/p devices.

CRT - In earlier times operations of most display devices depends on the CRT design. They are used for Black n white home television sets for a very long time.



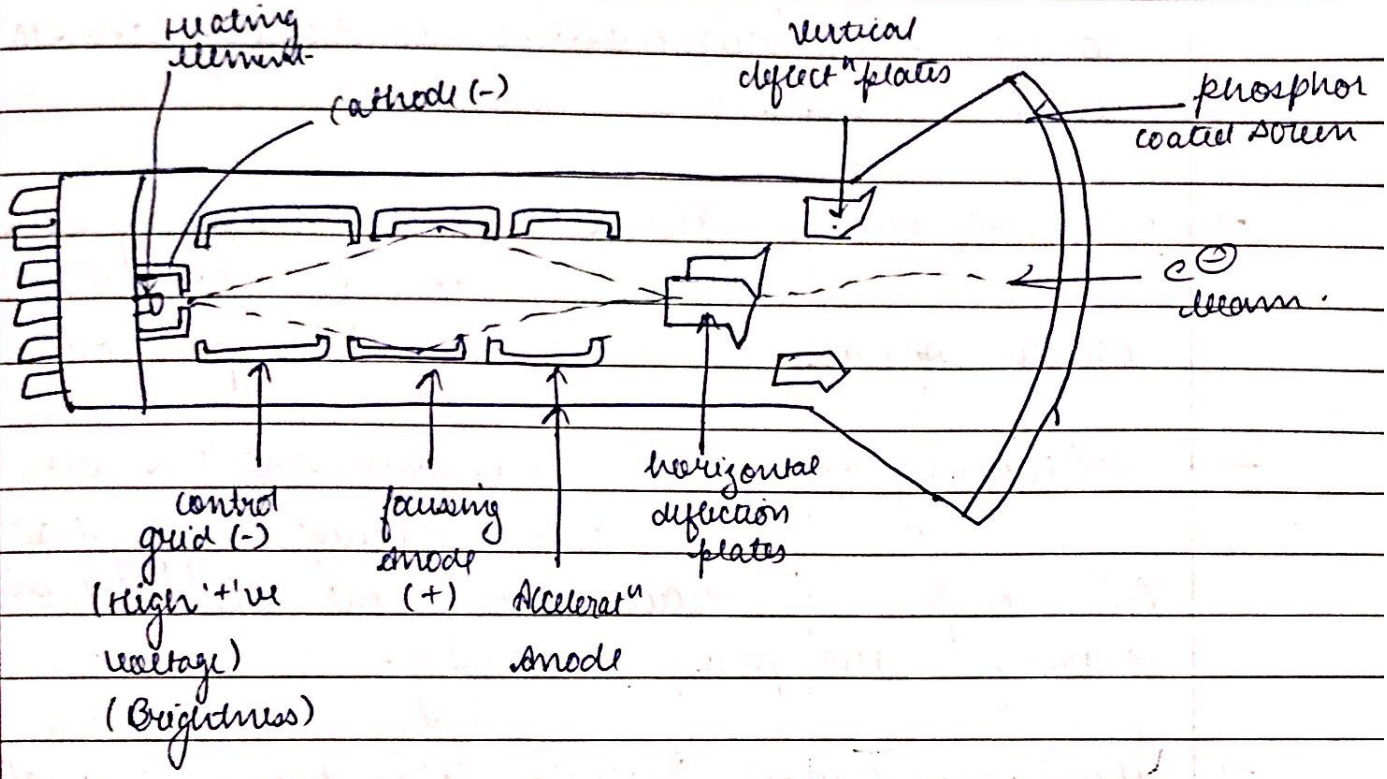
Magnetic Deflection CRT

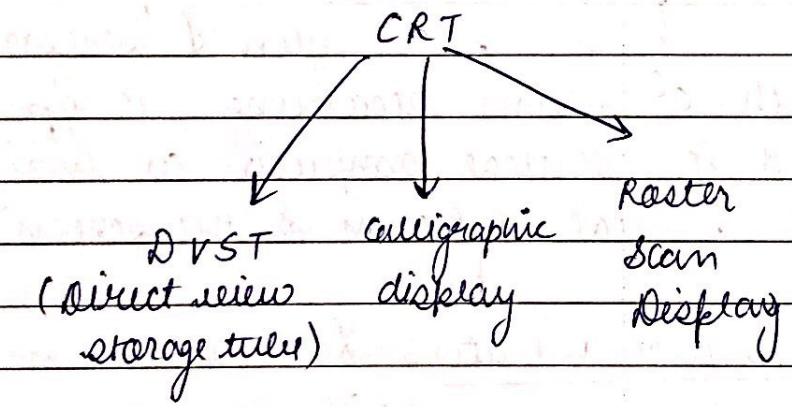
Basic working :-

The $e^{\ominus}$ gun emits a beam of $e^{\ominus}$ or cathode rays which passes through the focusing system & magnetic deflection coils to finally hit the phosphor coated screen. Phosphor emits small spots of light which fades away very rapidly. The sol^{n} to this problem is repeatedly striking the beam on the same point to make the phosphor glow regularly. This type of display is called refreshed CRT.

Page No.:
Date:

Detailed CRT Diagram:-





The e⊖ gun assembly consists of indirectly heated cathode, a control grid surrounding the cathode, a focussing anode & accelerating anode. The sole function of e⊖ gun is to provide the focused e⊖ beam which is accelerated towards the phosphor screen.

Material - The cathode is a Ni cylinder coated with Ba & Sr (Barium & strontium).

The emitting surface of cathode should be very small

→ Control Grid : It is a metal cylinder (iron permalloy steel) with a small hole in the cover. It is kept at high '-ve' potential in order to control the brightness by controlling the voltage w.r.t. cathode.

→ Focussing anode:- The purpose of focussing anode is to concentrate the e⊖ beam which spreads due to inter e⊖ repulsion. (500 V)

→ Accelerating Anode :- The accelerating anode is connected to high '+' ve voltage supply (1500 V). This helps in accelerating the speed of e⊖ for striking the phosphor-coating.

→ Horizontal Deflection Plates :- It is placed b/w e⊖ gun & screen. It can detect the e⊖ beam according to i/p signal. It is connected to vertical amplifier as horizontal plates provide vertical deflection on the screen.

→ Vertical Deflection Plates :- It works in synchronization with the horizontal plates to provide horizontal deflection to the e⊖ beam.

→ Phosphor :- Phosphor screen emits photons if accelerated e⊖ hit the material. Typical conversion factor of the phosphor screen are b/w 20 - 200 photons/e⊖. Two mostly used are P43 & P46.

Page No.:
Date:
YOUVA

Terms related to CRT-

1.) Dot Pitch :- It represents the distance which seperates two phosphor dots of the same color. Lower the dot pitch higher the image quality.

2.) Persistence :- It is defined as the time it takes the emitted light from the phosphor screen to decay by 90% of the original light o/p. Phosphors with low persistence requires high refresh rate and are useful for animation purpose. On the other hand high persistance phosphors are useful for static, complex pictures.

3.) Refreshment :- It represents number of images which are displayed per second which is Minimum 60 Hz/sec.

4.) Resolution :- It represents no. of pixels per unit surface area also called as dots per inch (DPI) A resolution of 300 DPI means 300 columns and rows of pixels per sq. inch.

5.) Aspect Ratio :- It is the ratio of vertical points to horizontal points necessary to produce length lines on both directions on the screen.

eg - Aspect ratio 3/4 means vertical line plotted with 3 points has the same length as a horizontal line plotted with 4 points.

Q.) A TV screen has 525 scan lines and aspect ratio of 3:4 if each pixel contains 12 bits words of intensity

information. How many bits per second are required to 30 frames/second.

Scan lines = 525

aspect ratio = 3/4

$$\frac{525}{x} = \frac{3}{4}$$

$$x = \frac{525 \times 4}{3}$$

$$x = 175 \times 4 = 700$$

Resolution = 525 × 700

Information carried in 1 frame = 525 × 700 × 12

total info = 525 × 700 × 12 × 30.

8) Calculate size of frame for a display supporting ~~two~~ true colors & has 1024 × 1024 px. on the screen. If the screen is 9 inches by 12 inches also calculate ~~asking~~ aspect ratio.

Resolution = 1024 × 1024

Size = 9" × 12"

V H

1024 × h = 9"

1024 × w = 12"

h = 9/1024

w = 12/1024

Aspect Ratio = $\frac{9/1024}{12/1024}$ = 9/12 = 3/4

R → 8 bit

G → 8

B → 8

24

frame size = 1024 × 1024 × 24.

Page No.:
Date:
YOUVA

Q.) A TV screen has 1024 scan lines with aspect ratio 4:3 & bit depth 24 what is the resolution of the screen & the frame size.

Scan lines = 1024

aspect ratio = 4/3.

$$\frac{1024}{x} = \frac{4}{3}.$$

$$x = \frac{3 \times 1024}{4} = 768$$

Resolution = $1024 \times \frac{1024 \times 3}{4.} \rightarrow 768.$

$= 1024 \times 768 \times 24$

Q.) How long would it take to load a frame size 640 x 480 with 12 bits/px. If 10^5 bits can be transfered per second.

$$= \frac{640 \times 480 \times 12}{10^5} =$$

Q.) What is the difference b/w HD, FHD, UHD, 2K & 4K display devices in terms of resolution, aspect ratio, frame size, processor speed & RAM size.

CRT

1) Direct view storage (DVST)
2) Calligraphic Display (Random scan)
3) Raster Scan Display

DVST :-

1.) Simplest type of CRT with 2 guns < primary / Flood

2.) It is a line drawing display with line can be drawn from any addressable point to other.

3.) The display is erased by flooding the entire line with a specific voltage.

4.) uses long persistence phosphor.

5.) Modifying any part of image requires redrawing the entire image.

6.) Cannot display color.

7.) Dynamic animation is not possible.

* Random Scan Display :-

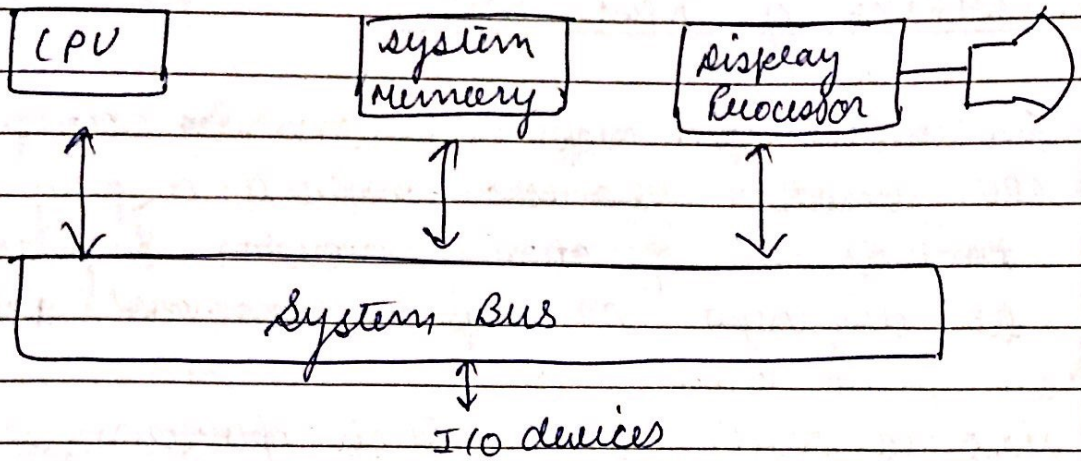
1.) e⊙ beam directly strikes those parts of screen where the picture is to be displayed.

2.) Draws one line at a time.

3.) Phosphor used is of short persistence.

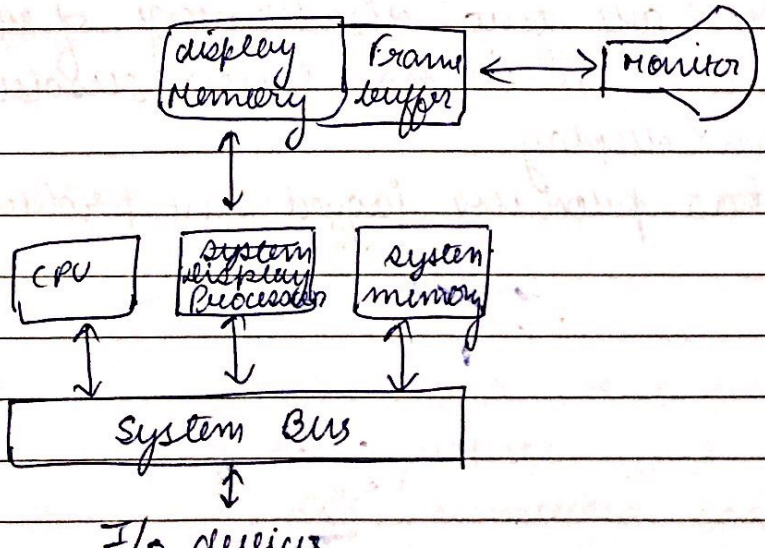
4.) The display commands are stored in refresh buffer.

5.) animation is possible with segmentation.



* Raster Scan Display :-

1.) e$^{\ominus}$ beam scans rows of the screen one by one from top to bottom.
2.) Each screen point has intensity value either 0 or 1
3.) The intensity value is stored in refresh buffer or frame buffer.
4.) Frame buffer for black & white systems is called bitmap and for systems with multiple bits per pixel called as pixmap.
5.) Refresh rate is 60 - 80 frames per second.

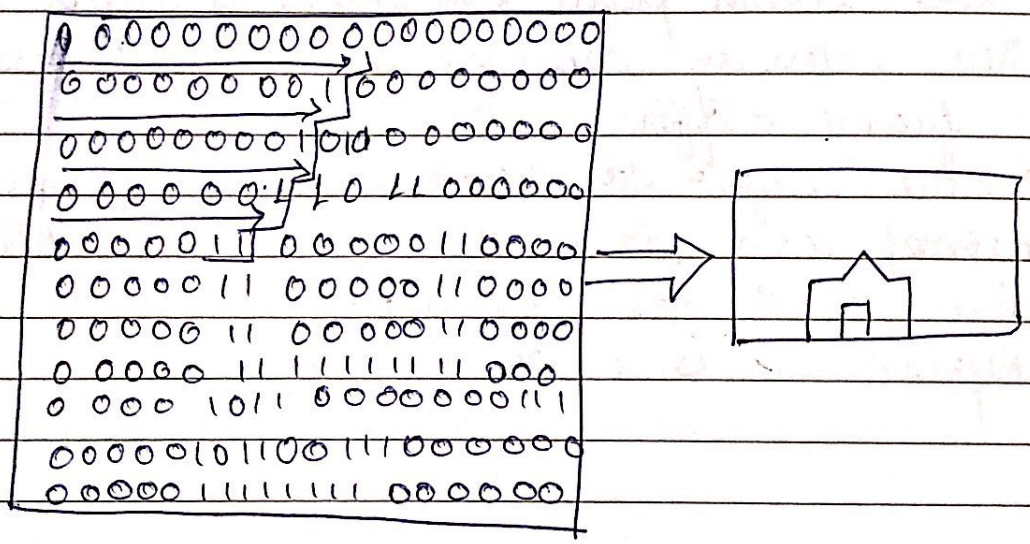


Architecture of Raster Scan Display

working of Raster Scan:-

In raster scan display system in addition to CPU another processor called as display controller controls the operation of monitor. Information to be displayed or graphics command are stored in frame buffer.
Video controller starts refresh operation at the top left corner of the screen.
Intensity value at this position is fetched from the frame buffer & plotted on the screen.

Frame Buffer



There are two disadvantage of raster scan display is ① it has lower resolution than random scan display.
② It has produces jagged-line problem.

Page No.:
Date:
YOUVA

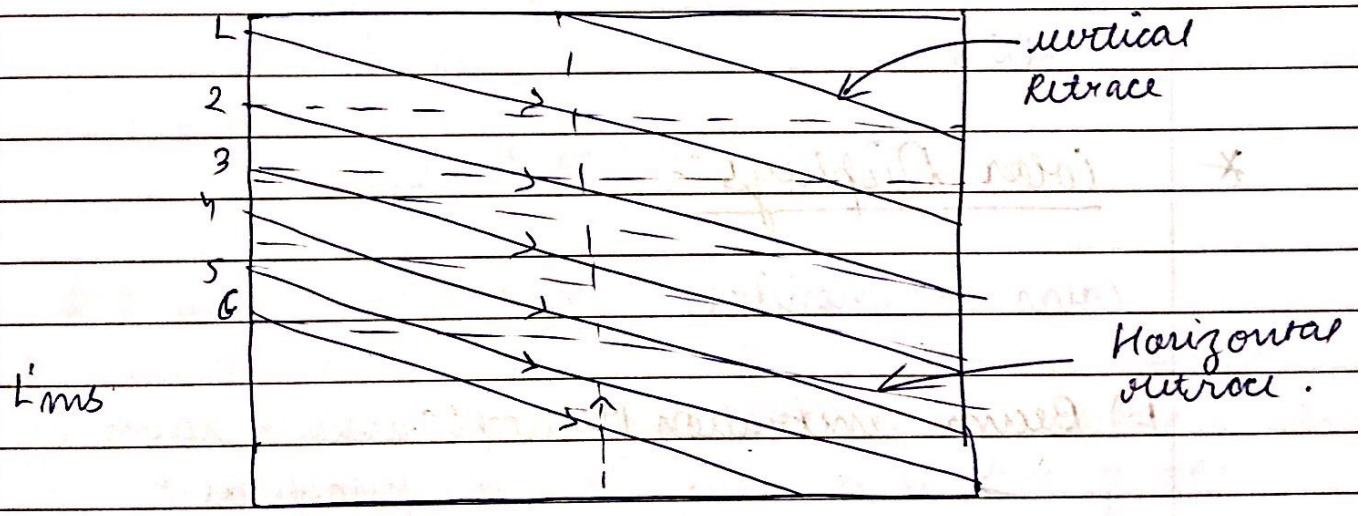
Interlacing :-

Horizontal retrace :- while refreshing on raster scan displays, after refreshing each scan line beam returns to the left of the screen which is called horizontal retrace.

→ It is a started line to shift the beam to the ~~ft~~ left for second horizontal line.

Vertical Retrace - After reaching the last scanline and upto the right bottom corner, the beam returns to the top left corner by vertical retrace.

→ The time required is 21 times of H.R.



Working :-

Each frame is divided into two fields each containing half of the picture. one field contains all the odd numbered scan lines

and the other field has only even numbered scan lines. The whole frame is displayed into two passes. In the first pass it scans only odd scan lines from top to bottom after completing the odd field the beam moves to the top scans remaining even lines. This sequence is repeated again & again. Interlacing of all the even and odd scan lines allow is to see a picture in half the time.

* Flat Pannel Display :-

1.) Plasma
2.) LCD
3.) LED's

* Color Displays :-

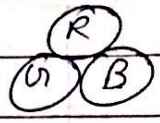
Color CRT monitor.

1.) Beam Penetration Method :- uses 2 layer of phosphor.
→ can produce only upto 4 color (R, G, Y, O)

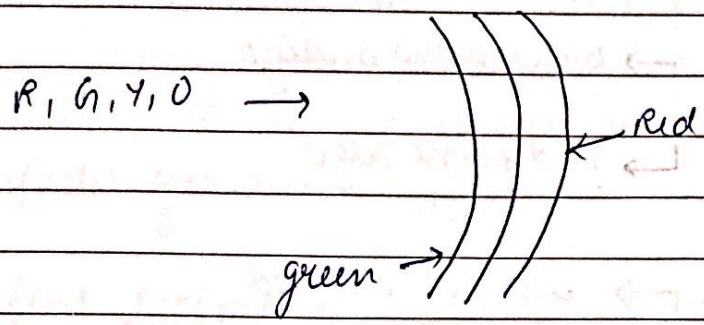
2.) Shadow Mask Method :- uses 3 $e^{\ominus}$ guns arranged in triangular pattern.
→ Perforated metal sheet is used between $e^{\ominus}$ gun & face of CRT.

Page No.:
Date:
YOUVA

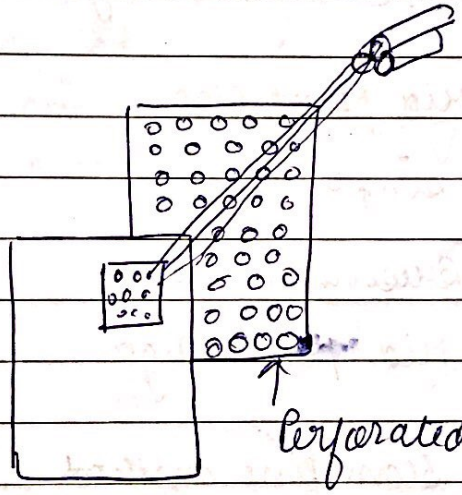
Beam Penetration Method -



Low resolution



High Resolution.



Shadow Mask Method -

To ensure that each $e^{\ominus}$ gun strikes correct phosphor dot; a perforated metal sheet (shadow mask) is arranged b/w $e^{\ominus}$ gun and face of CRT.

Perforations in the metal sheet are also arranged in the same pattern as the dot on the phosphor. By varying the strength, intensity of different color & range of different color can be generated. The only disadvantage of this method is reduces brightness of the CRT.

Scan Conversion :-

- Line
  - Simple method
  - DDA
  - Bresenham's Algo
  - Mid point algo
- circle
  - simple method
  - Bresenham's Algo
  - Mid point Algo
- ellipse
  - simple method
  - Bresenham's Algo
  - Mid point Algo
- Polygon → Scan line method

Scan conversion refers to the technique in which we find out the co-ordinates on the screen for an output premitive to be displayed. Before going for complex figures we start with basic regular objects such as line, circle, ellipse & polygon.

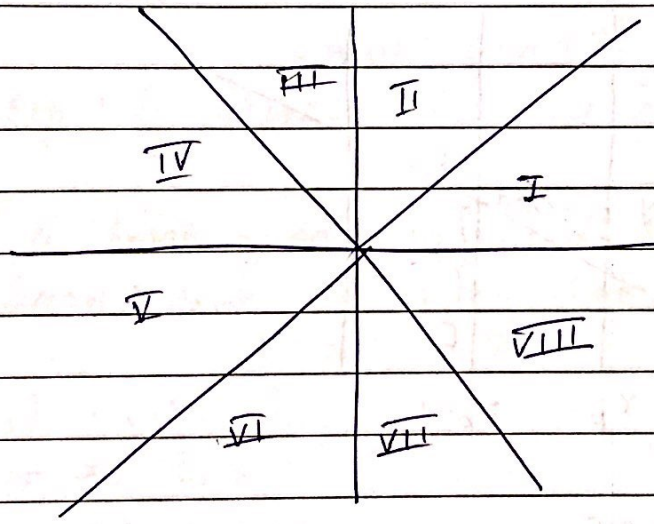
Line - A line on the screen can be horizontal, vertical and slanted with a given slope. We need to therefore calculate the exact co-ordinates for the line to be displayed on the screen. There are 4 methods - simple method, DDA, Bresenham's algo & mid pt. algo.

(i) simple method -

In this method we make use of mathematical eqⁿ $y = mx + c$ for drawing any line on the screen. The major disadvantage of this method is the use of multiplication operation which is costly for early time display processors hence this method is not suitable for deployment to any computer graphic system.

(ii) Digital Differential Analyzer (DDA) :-

This method is also known as incremental algorithm. It removes the drawback of multiplicat^{n} in the simple method by dividing the whole screen into 8 octants.



For I octant

$m = \frac{dy}{dx}$

$\frac{dy}{dx} < 1$

$dy < dx$

$x = x + 1$

$y = y + m$.

For II octant

$\frac{dy}{dx} > 1$

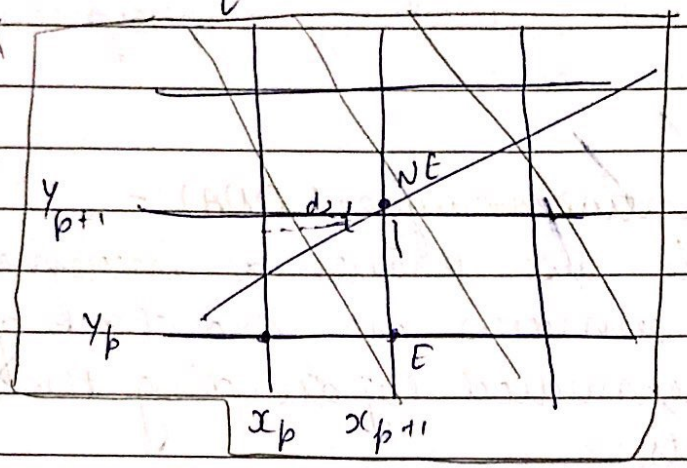
$dy > dx$

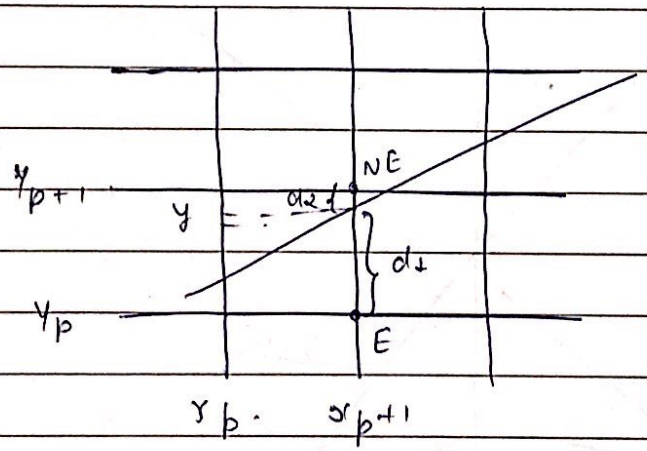
$y = y + 1$

$x = x + \frac{1}{m}$

(iii) Bresenham's Line Drawing Algorithm:-

→ very efficient and fast
→ uses only incremental integer calculations
→ can be used for circle and similar curves.





if $d_1 > d_2 \rightarrow NE$.

$$d_1 - d_2 = 2m(x_p + 1) - 2y_p + 2c - 1$$

At each successive calculation of x-values we calculate successive decision variables using incremental integer calculations.

case 1: when m < 1

$$d_{new} = 2\Delta y(x_p + 1) - 2\Delta x y_p + c$$

$$(\Delta d)_E = d_{new} - d_{old}$$

$$= 2\Delta y$$

case 2: Chosen NE.

$$d_{new} = 2\Delta y(x_p+1) - 2\Delta x(y_p+1) + c$$
$$= 2\Delta y - 2\Delta x.$$

Algorithm:-

1.) Input both the line end points and store initial end point as (x_0, y_0).

2.) Plot (x_0, y_0)

3.) Calculate $\Delta x, \Delta y, 2\Delta y - 2\Delta x, 2\Delta y$.

$$d_{start} = 2\Delta y - \Delta x.$$

4.) At each step we test decision variable

if $d < 0 \rightarrow$ choose E

$$d_{new} = d_{old} + 2\Delta y.$$

else $\rightarrow$ choose NE.

$$d_{new} = d_{old} + 2\Delta y - 2\Delta x$$

5.) This step 4 is repeated till the end point is reached.

Q.) Draw a line from (10,12) to (20,18) using Bresenham's line drawing algorithm.

$(x_0, y_0) \rightarrow (10, 12)$

$\Delta x = x_2 - x_1$

$\Delta x = 20 - 10 = 10$

$\Delta y = y_2 - y_1$

$\Delta y = 18 - 12 = 6$

$2\Delta y - 2\Delta x = -8$

$2\Delta y = 12.$

$d_{start} = 2\Delta y - \Delta x$

$= 12 - 10 = 2$

$d > 0 \rightarrow$ we chose NE (11,13)

$d_{new} = 2 + 12 - 20$

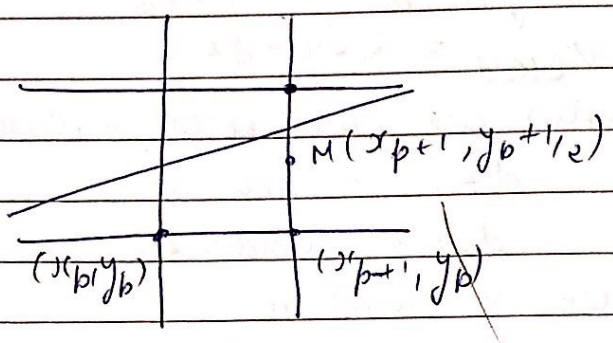
$= -6$

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

* Mid Point Line Drawing Algorithm :-

→ Here we find out the mid-point b/w NE & E and test if this mid-pt. lies above or below this line path.

→ This approach generates the same pixels as Bresenham's but it can be easily applied to other conics



$$y = \left(\frac{dy}{dx}\right)x + B$$

$$dy = y_1 - y_0 \quad , dx = x_1 - x_0$$

$$f(x,y) = ax + by + c = 0$$

$$m = -\frac{a}{b}$$

$$f(x,y) = dy * x + (-dx) * y + B * dx = 0$$

$$a = dy, \quad b = -dx, \quad c = B * dx.$$

if $f(x,y) > 0$ → M lies below the line.

if $f(x,y) < 0$ → M lies above the line.

Algorithm:

1) Input both the line end pts. and store initial end pt. as (x_0, y_0)

2) Plot (x_0, y_0)

3.) Calculate

$$d_{start} = dy - \frac{dx}{2}$$

4.) If $d > 0$ choose NE

$$(\Delta d)_{NE} = dy - dx$$

if $d \leq 0$ choose E

$$(\Delta d)_E = dy$$

5) Repeat step 4 till the end pt.

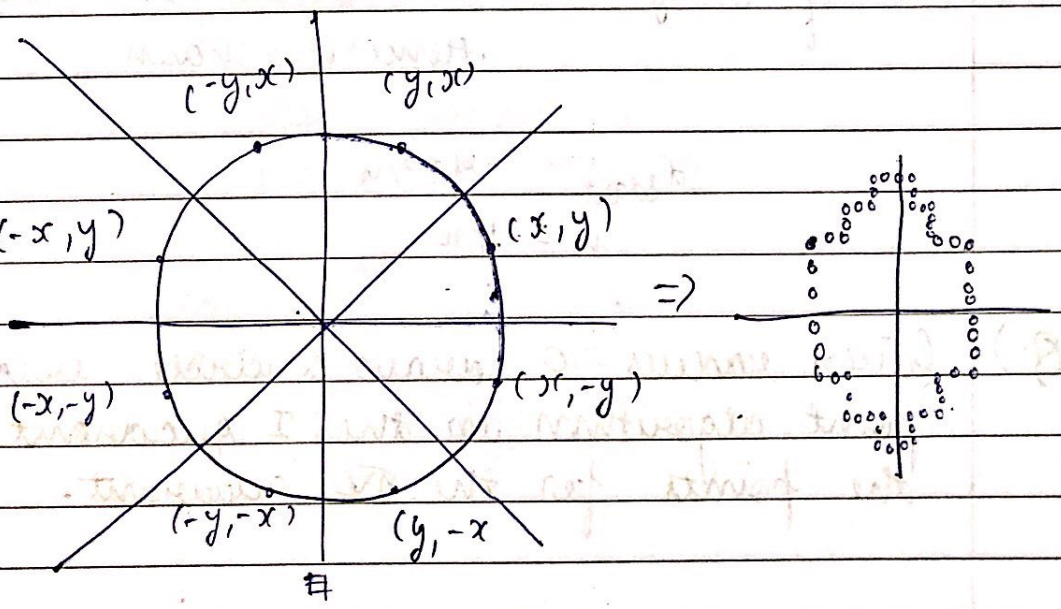
Scan Conversion Method:-

Simple method:-

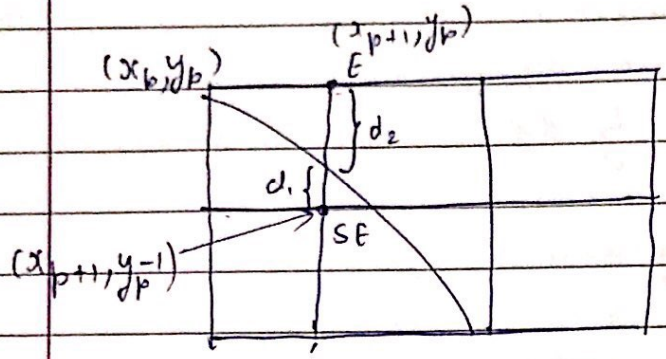
$$(x - x_c)^2 + (y - y_c)^2 = r^2$$

for a circle centered at (x_c, y_c)

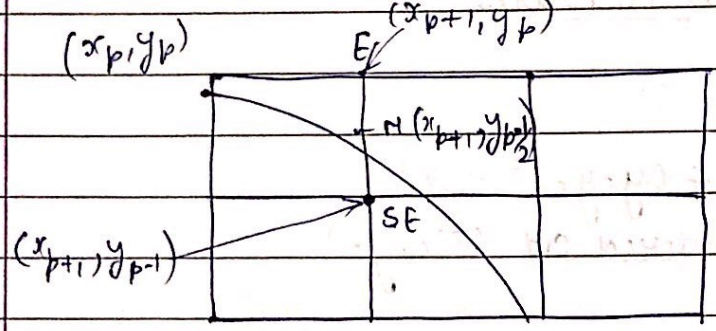
* Since there is symmetry in the eight octants of the circle, we need to calculate the points only for one octant, rest 7 octants can be calculated the same way.



→ Bresenham's circle algorithm -



Mid - point circle algorithm :-



If $f(x,y) \geq 0 \rightarrow$ M lies outside the circle, Hence SE is chosen.

If $f(x,y) < 0 \rightarrow$ M lies inside the circle. Hence E is chosen.

$$d_{start} = -r + 5/4$$
$$h = 1 - r$$

Q.) Given radius = 10 draw a circle using mid point algorithm in the I quadrant also list the points for the IV quadrant.

given, r=10

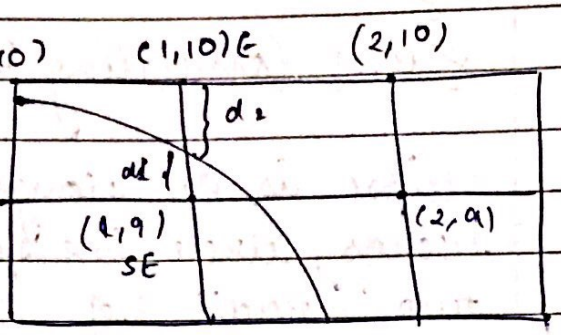
(0,10)

$x = 0$

$y = 10$

$h = 1 - r = 1 - 10 = -9$

$h < 0 \rightarrow E \rightarrow (1,10)$



$h_{new} = h_{old} + 2x + 3$

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

$= -6$

$h < 0 \rightarrow E \rightarrow (2,10)$

$h_{new} = h_{old} + 2x + 3$

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

$= -1 \rightarrow E \rightarrow (3,10)$

$h_{new} = -1 + 2(2) + 3 = 16 \rightarrow (3,9)$

$d_1 - d_2 = d$

$d > 0 \rightarrow E$

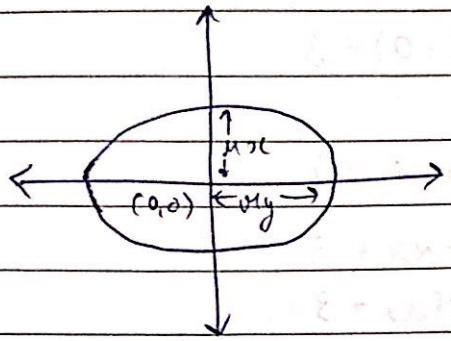
$d < 0 \rightarrow SE$

* **Pseudocode for mid point circle algo:-**

```
x = 0, y = r, h = 1 - r   // initialization pt.
Drawcircle(x, y);
while(y > x)
    if (h > 0) // select T E.
        h = h + 2x + 3
    else // select SE.
        h = h + 2x + (-2y) + 5
        y = y - 1
    end if.
    x = x + 1;
    Drawcircle(x, y);
end while;
```

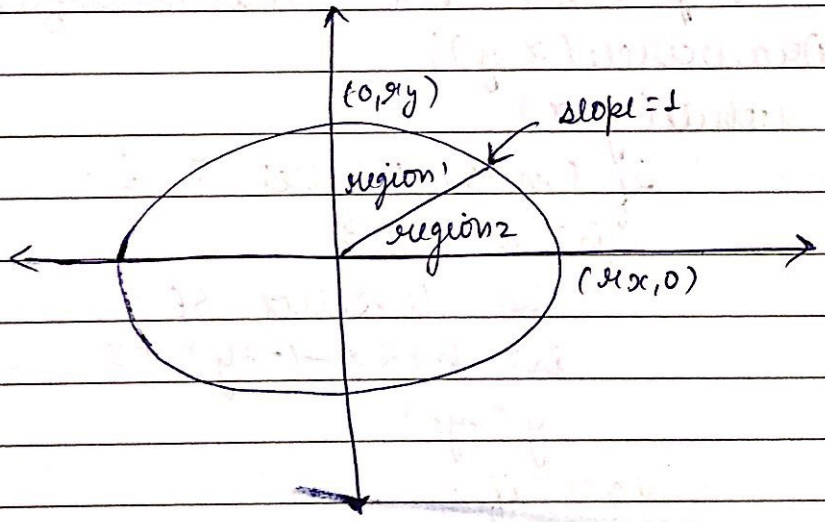
Scan Converting Ellipse :-

Ellipse is an elongated form of circle or in other words, circle is a special case of ellipse where the two radii of circle are equal.



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

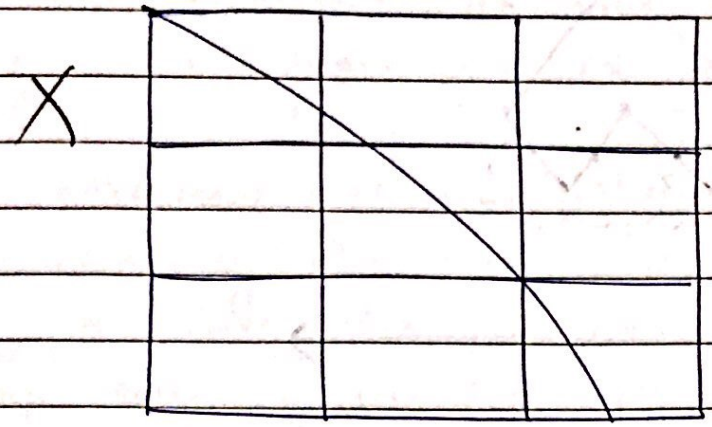
→ Mid point ellipse algorithm :-

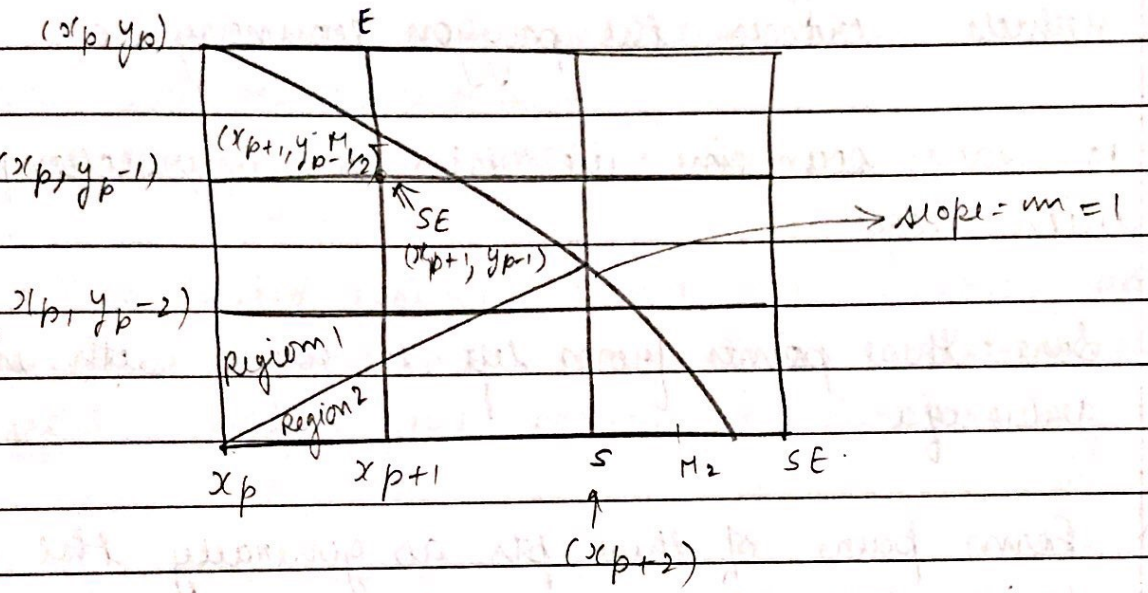


→ suppose we are calculating points in the I quadrant.

→ we divide it into two regions.

Date:





For region 1

$$d = r^2 y (x_p + 1)^2 + r^2 x (y_p + 1/2)^2$$

if $d > 0 \rightarrow E$

$d < 0 \rightarrow SE$

for region 2

$$d = r^2 y (x_p + 1/2)^2 + r^2 x (y_p - 1)^2 - r^2 x - r^2 y$$

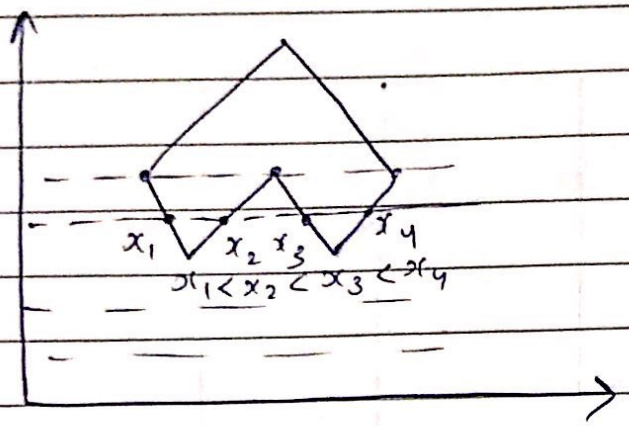
if $d > 0 \rightarrow SE$

$d < 0 \rightarrow S$

→ Polygon Scanning Algorithm-

There are two approaches to fill the area inside the o/p primitive.

1.) Scan line polygon fill algorithm.
2.) seed fill algorithm.



1.) In this algo we work on each scan line which crosses the polygon boundary.

2.) For each scan line we obtain the intersection points with edges.

3.) Sort these points from left to right with increasing value of x.

4.) Form pairs of these pts. as generally the no. of points are even. For vertices we require special handling through active edge table.

5.) Between each pair fill the pixels with desired color & update the intersection pts. for each line.

6.) This process ends when scan ~~of~~ line reach maximum value on the Y axis.

There are two important features of scan line based polygon filling algorithm -

1.) Scan-line coherence
2.) Polygon edge coherence

Page No.:
Date:
YOUVA

Scan line coherence - it means that values do not change much from one scan line to another.

Edge coherence - it means that slope of the edge intersected by the scan line (i). is almost the same as intersected by scan line i+1. It will change only if we pass through a vertex.

For coping up this situation we use two data structure: ~~the not~~

(i) <u>sorted-edge table (SET)</u> - used bucket sort.

SET is build using bucket sort with as many bucket as there are scan lines within each bucket edges are sorted by increasing value of x.

NOTE* :- Only non-horizontal edges are stored.

(ii) <u>Active edge list</u>:-
It contains all the edges crossed by scan line at the current stage of iteration. This is a list of edges that are active for this current scan line sorted by x intersections. It also called AET.

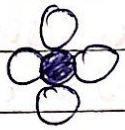
<u>Seed Fill Algorithm</u>:-

→ It is the second approach for filling a polygon.
→ It assumes the knowledge of at least one pixel interior to a polygon or region.

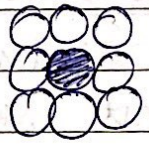
regions
- interior defined: interior region has one colour while exterior region has another (flood fill).
- Boundary defined: all pixels on the boundary have a unique color & pixels exterior to boundary may also have boundary color.

* Boundaryfill Algorithm:

→ Takes an i/p an interior pt. (x, y) fill color and of course the boundary color.
→ Starting with (x, y) we test for each neighbouring pixel point of (x, y) if the pixel is not filled with boundary color, it is filled with fill color.
→ Testing of neighbouring pixels can be done using two methods :- 4 connected
8 connected.



4 connected



8-connected.

Algorithm :-
Boundary fill.

1.) begin
2.) initialize fill color and boundary color.
3.) get current pixel value at (x, y);
if (current value ≠ fillcolor and current value ≠ boundary value)
begin
set current with pixelcolor;
recursively fill neighbouring pixel;

```
end if;
end boundary fill.
```

Stack based approach is used.

(i) Push the seed pixel on to the stack.
(ii) while the stack is not empty pop a pixel from the stack and set the color.
(iii) For each adjacent pixel to the current pixel check whether it is a boundary pixel or if it is already filled. If it is so ignore the pixel else push the pixel on the stack.

Flood fill algorithm:-

Here we do not compare in the interior region pixels to have the same colour as that of boundry color. This means starting from a specified interior point (seed pixel). Re-assign all pixel values with new filled colour.

```
flood fill.
begin
initialize fill color;
get color value at current pixel position.
    if (current value == old color)
        begin
            set current pixel with fill color;
            recursively fill neighbouring pixels;
        end if;
        end flood fill;
```

Date: YOUVA

Antialiasing -

The problem of aliasing arises when the o/p primitive on the display device are not smooth and proper as can be drawn with pen & paper. To remove this, is called a process known as anti-aliasing. There are 3 techniques under it -

1.) Increasing resolution
2.) Point ~~sampling~~ sampling.
3.) Area sampling.

1.) Increasing Resolution :-
Increasing in the no. of pixels per unit area decreases the size of the pixels to half. So the jaggies also get reduced (half in size) so the resulting picture looks better but the cost of memory, bandwidth and scan conversion time is quadrupled and this not approach~~es~~ is not

2.) Point sampling - objects to be displayed are 2-dimensional hence in graphic system the frame buffer will be a discrete two dimensional signal. We select a finite set of samples from the signal for the display. This is called as point sampling. In this approach we may miss some important feature of the object.

3.) Area Sampling :- Here we assume that any scan converted o/p primitive occupies some finite area on the display screen. Even the

Page No.:
Date:
YOUVA

thinnest horizontal or vertical line is atleast 1 px thick. line contributes some amount of intensity to each pixel in the columns. Here we sample the areas which contributes either equally or unequally to the display of the o/p primitive.

Transformations :-

→ Transformation is the process of changing the position, size or orientation or combination of these for an object.

Transformations
- co-ordinate
- Geometric.

Basic 2-D transformation :-

an object to be transformed can be represented in the form of matrix $\begin{bmatrix} x \\ y \end{bmatrix}$.

General eqn of Transformation :-

$$[B] = [T][A]$$

[B] = New points after transformation.

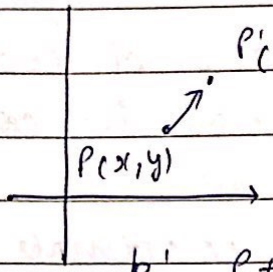
[T] = geometric transformation matrix.

[A] = co-ordinate of points to be transformed.

2- D transformations :-

1.) Transforme Translation : Process of changing the position of an object from one co-ordinate location to another.

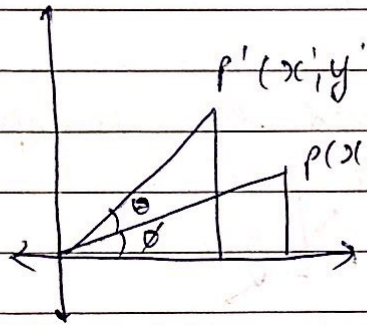
$$x' = x + tx$$
$$y' = y + ty.$$



$$P' = P + T$$

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

2.) Rotation :- an object can be rotated by an angle θ with respect to origin.



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

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

Page No.:
Date:
YOUVA

Note *:-

Rotation matrices are orthogonal. Thus it satisfies the corresponding properties:-

→ Determinant of ~~Rotation~~ rotation matrix is 1. $|T| = 1$.

→ Transpose of the matrix gives inverse matrix.

$$[T]^T = [T]^{-1}$$

→ Sum of squares of individual rows or column is 1.

Rotation → Anticlockwise → +ve
→ clockwise → '-' ve.

special cases of Rotation:-

θ (in degrees)	Matrix
90	$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$
180	$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$
270 or -90°	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$
360° or 0	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

3.) Scaling:- Process of expanding or reducing the dimension or size of an object. Thus, scaling alters the size of an object.

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

If $S_x = S_y$ → uniform scaling.
else:
non uniform scaling.

* Homogeneous co-ordinate system:-
→ The transformation for translation cannot be represented in matrix form.
→ Therefore, we move to another co-ordinate system from $(x, y) \rightarrow (x, y, H)$ where $H \neq 0$.
→ The points can be transformed as.

$$(x, y, H) \rightarrow (x/H, y/H)$$

Hence, we keep $H = 1$ in general.
Also, $(1, 2, 3)$ & $(3, 6, 9)$ represent the same point in homogeneous co-ordinate system.

Q.) Translate a square ABCD with co-ordinates A(0,0) B(5,0), C(5,5) D(0,5) by 2 units in x direction & 3 units in y direction.

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

Object co-ordinates matrix $A = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ 1 & 1 & 1 & 1 \end{bmatrix}$

After transformation.

$$R = T \cdot A \qquad 3 \times 3 \times 3 \times 4$$

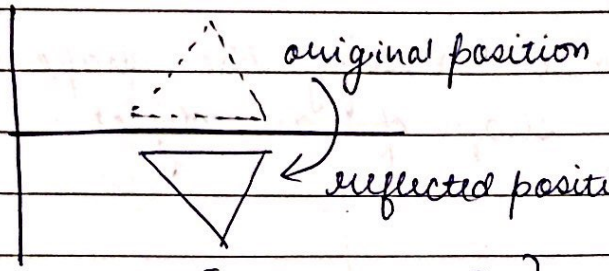
$$= \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 5 & 5 & 0 \\ 0 & 0 & 5 & 5 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

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

Reflection :-

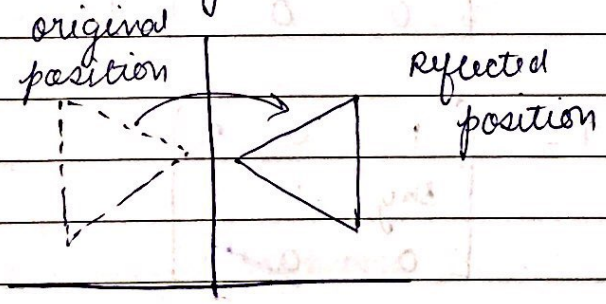
→ Generates the mirror image of an object.
→ It is generated relative to an axis of reflection.
→ Reflection means rotating the object 180° to the axis of reflection.

Case: 1 Reflection about x-axis :-



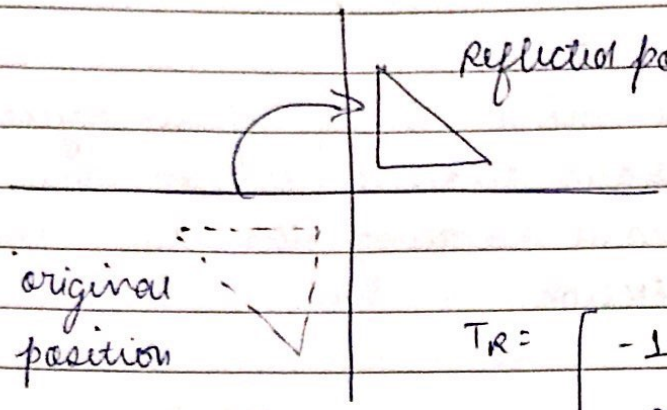
$$T_R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Case 2 : Reflection about y-axis :



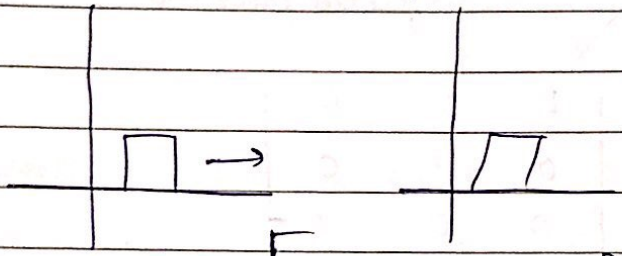
$$T_R = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Case 3: Reflection about axis ⊥ to xy-plane passing through co-ordinate origin.



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

Shear :- used to modify the shape of object. It distorts the shape of an object.



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

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

Q.) A square ABCD given with vertices A(0,0) B(1,0) C(1,1) D(0,1). what would be the effect of Y-shear & XY-shear. when $Shx = 2$ & $Shy = 3$.

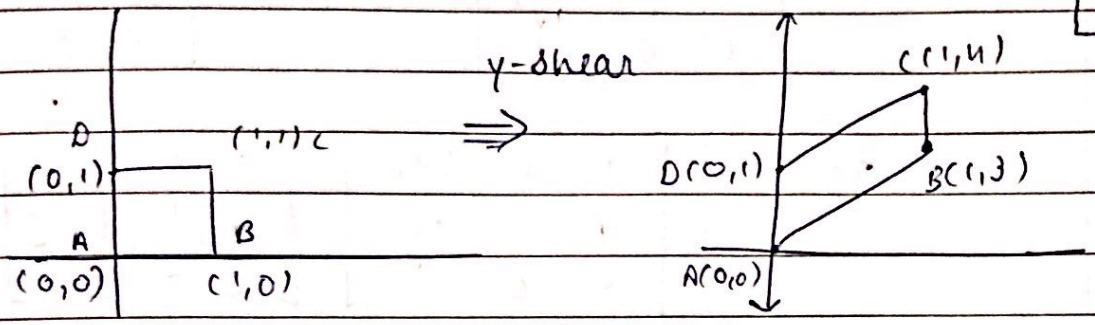
$$O = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

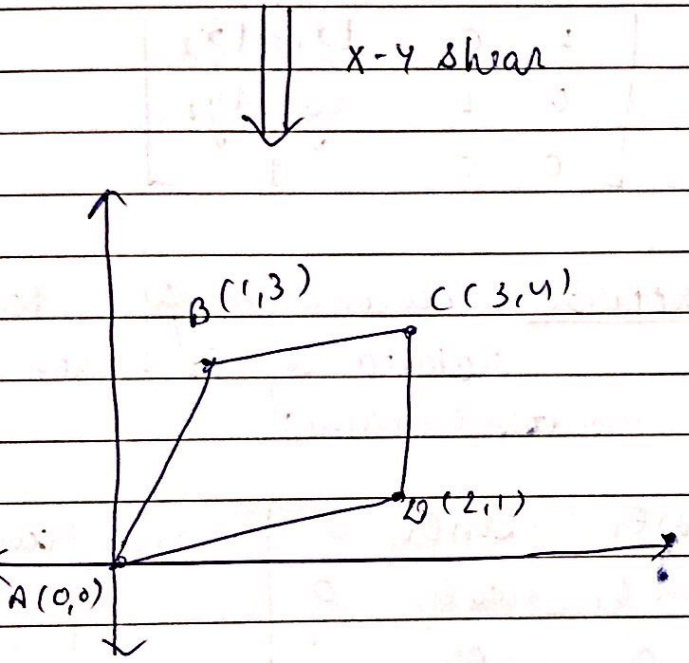
$$P = T \cdot O$$

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

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

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





* Composite Transformations :-

We want to apply a series of transformations to a set of points if done step by step it consumes large amount of time so the second approach is to compose matrices into one final transformation matrix & then apply it to the points.

1.) **Two successive translation :-** Assume a given point x, y to be translated by a factor (t_{x_1}, t_{y_1}) & re-translated

by (tx_2, ty_2)

$$T_1 = \begin{bmatrix} 1 & 0 & tx_1 \\ 0 & 1 & ty_1 \\ 0 & 0 & 1 \end{bmatrix} \qquad T_2 = \begin{bmatrix} 1 & 0 & tx_2 \\ 0 & 1 & ty_2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T = T_1 \cdot T_2 = \begin{bmatrix} 1 & 0 & tx_1 \\ 0 & 1 & ty_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & tx_2 \\ 0 & 1 & ty_2 \\ 0 & 0 & 1 \end{bmatrix}$$

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

2.) Two, Successive rotation - assume a pt to be rotated by θ_1 & then by θ_2 we calculate the resultant matrix

$$T_1 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad T_2 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

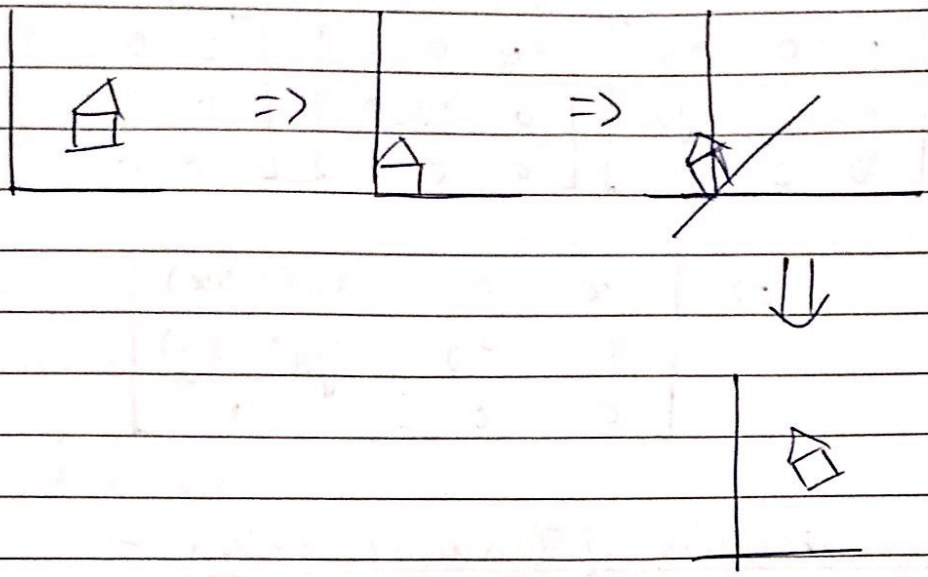
$$T_1 \cdot T_2 = \begin{bmatrix} \cos\theta_1 \cdot \cos\theta_2 - \sin\theta_1 \sin\theta_2 & -\cos\theta_1 \sin\theta_2 - \sin\theta_1 \cos\theta_2 & 0 \\ \sin\theta_1 \cdot \cos\theta_2 + \cos\theta_1 \cdot \sin\theta_2 & -\sin\theta_1 \sin\theta_2 + \cos\theta_1 \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

3.) Rotation about arbitrary point in space -

Consider an object in xy-plane to rotate it about a point we translate it to origin, rotate it

about origin & then translate back to original position.

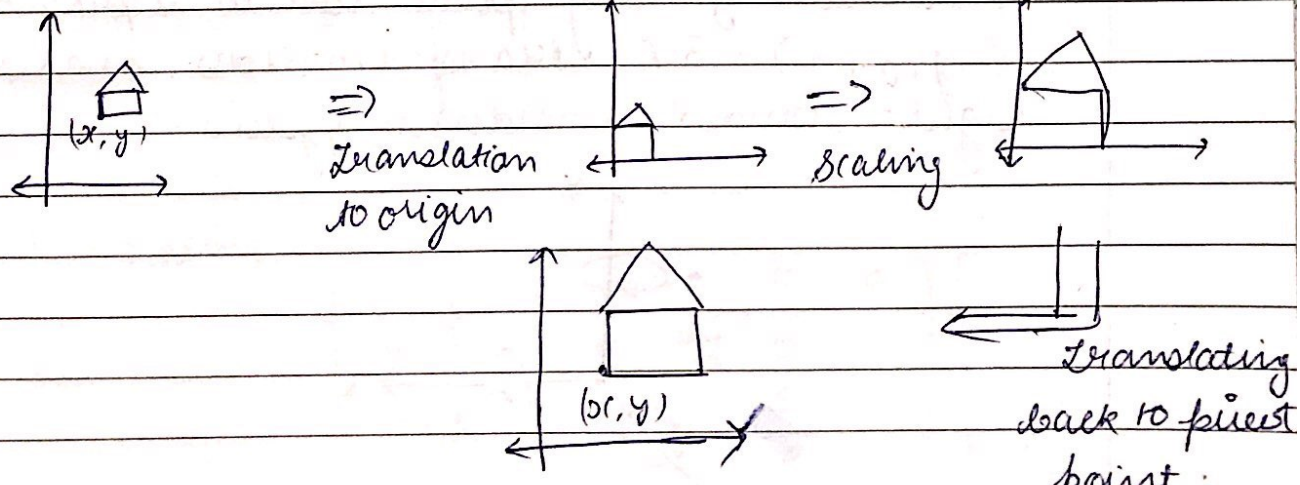


* 4) Two Successive Scaling :

Assume an object to be scaled by (Sx_1, Sy_1) and again by (Sx_2, Sy_2)

$$\begin{bmatrix} Sx_1 Sx_2 & 0 & 0 \\ 0 & Sy_1 Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Scaling about an arbitrary point (x_n, y_n) first, translate it to origin then scale it & then translate it back to original position.



matrix :

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

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

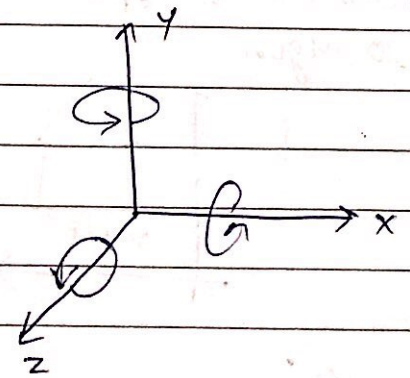
* **Commutativity of Transformations :-**

There are only few cases for which transformation is commutative.

(i) Translation & translation
(ii) Scaling & scaling.
(iii) Rotation & rotation.
(iv) Uniform scaling & rotation.

* **3-Dimensional Transformation :-**

In 2-D graphics there was no concept of depth with respect to the viewer. For 3-D graphics 2-D graphics is extended to include 3rd co-ordinate 'z'. Any point can be represented as (x, y, z, w). 3-D transformations makes use of right-handed co-ordinate system.



In right handed system anti-clockwise rotation is considered positive rotation.

3-D translation

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

3-D rotation

1.) About z-axis

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

2.) About x-axis

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

3.) About y-axis

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

3-D scaling :-

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

* 3-D Reflection :-

1.) Reflection in xy-plane

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

2.) Reflection in yz plane.

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

3.) Reflection in xz plane

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

* 3-D Shear :-

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

for x shear, $c = d = 0$, $a = b = 0$
for y shear, $a = b = 0$, $e = f = 0$
for z shear, $c = d = 0$, $e = f = 0$

Viewing Transformation :

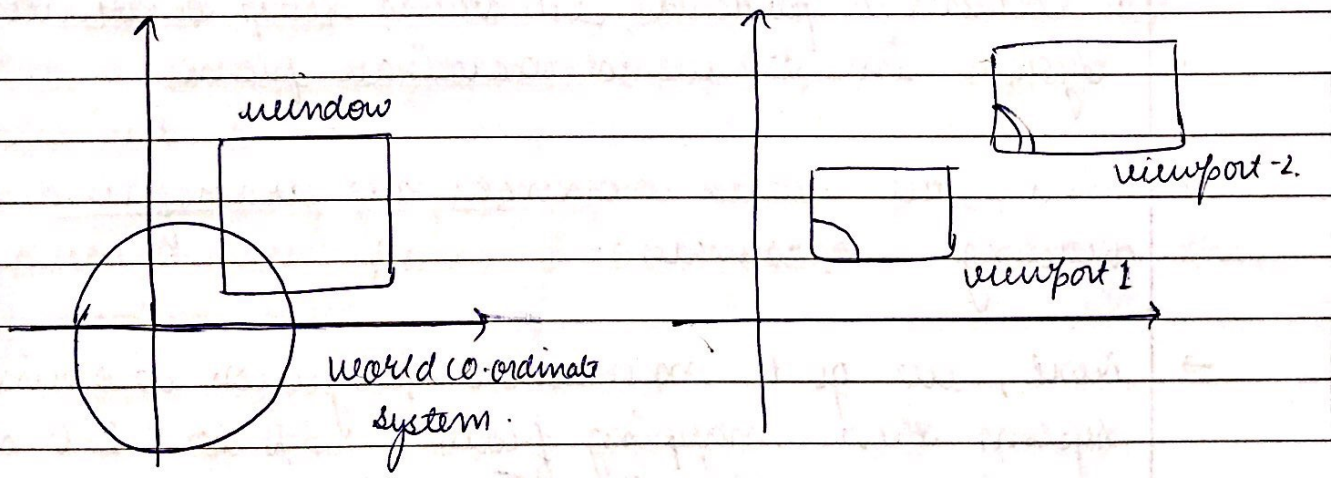
→ To display the view of a picture on to the display device, we make use of viewing transformation.

→ Our picture has to be ~~moved~~ mapped to the device or screen co-ordinates.

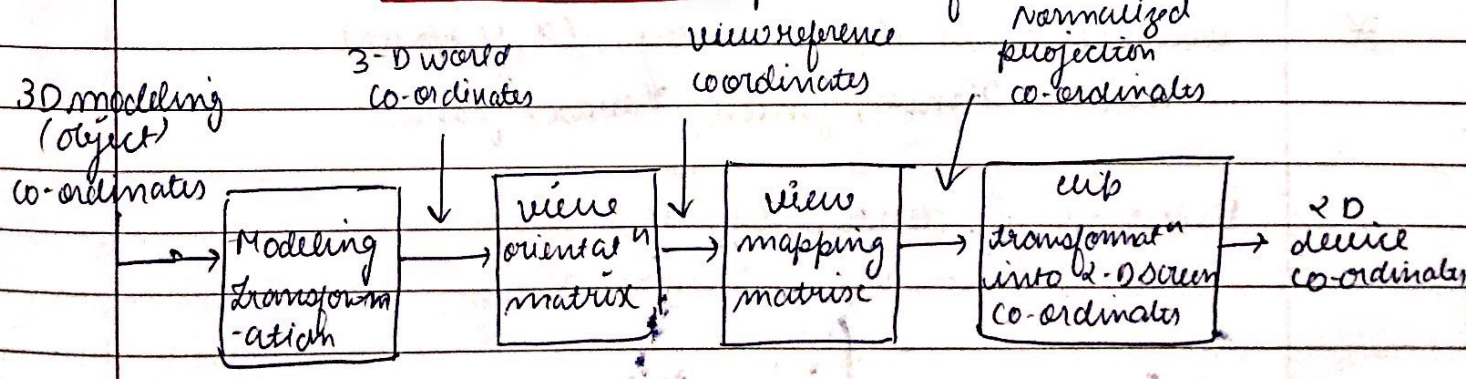
co-ordinates → screen/device coordinates
co-ordinates → world co-ordinates.

* **window** : Rectangular region of the world that is visible.

* **viewport** : The rectangle region of screen space that is used to display the window.



window to viewport transformation :-



We can define viewport anywhere on the screen hence changing the position of the viewport we can view objects at different positions on the screen. Ex- CCTV. We can also change the size of viewport, therefore changing the size of object & proportions of the object on the display screen.

Note* Window & viewport generally have standard rectangular shapes but can have any orientation. A real world object is to be displayed on the screen through the viewport.

Window to viewport transformation process:-

→ The object is described in object co-ordinate system also called modeling co-ordinate system.

→ To obtain a particular orientation setup a viewing system in 3-D world co-ordinate plane.

→ Now, the world co-ordinates are transformed in viewing coordinates.

→ Next, we go to normalised projection co-ordinate system then mapping from 3-D to 2-D device co-ordinates which includes clipping.

* Window is denoted by $(x, y$ space) with $x_{min}, y_{min}, x_{max}, y_{max}$.

Viewport is denoted by $(u, v$ space) with $u_{min}, v_{min}, u_{max}, v_{max}$.

The steps are:

1.) Translate windows to origin
2.) Scale it to view port size.
3.) Translate it to viewport location

$$M_{wv} = T(u_{min}, v_{min}) \cdot S(s_x, s_y) \cdot T(-x_{min}, -y_{min})$$

where

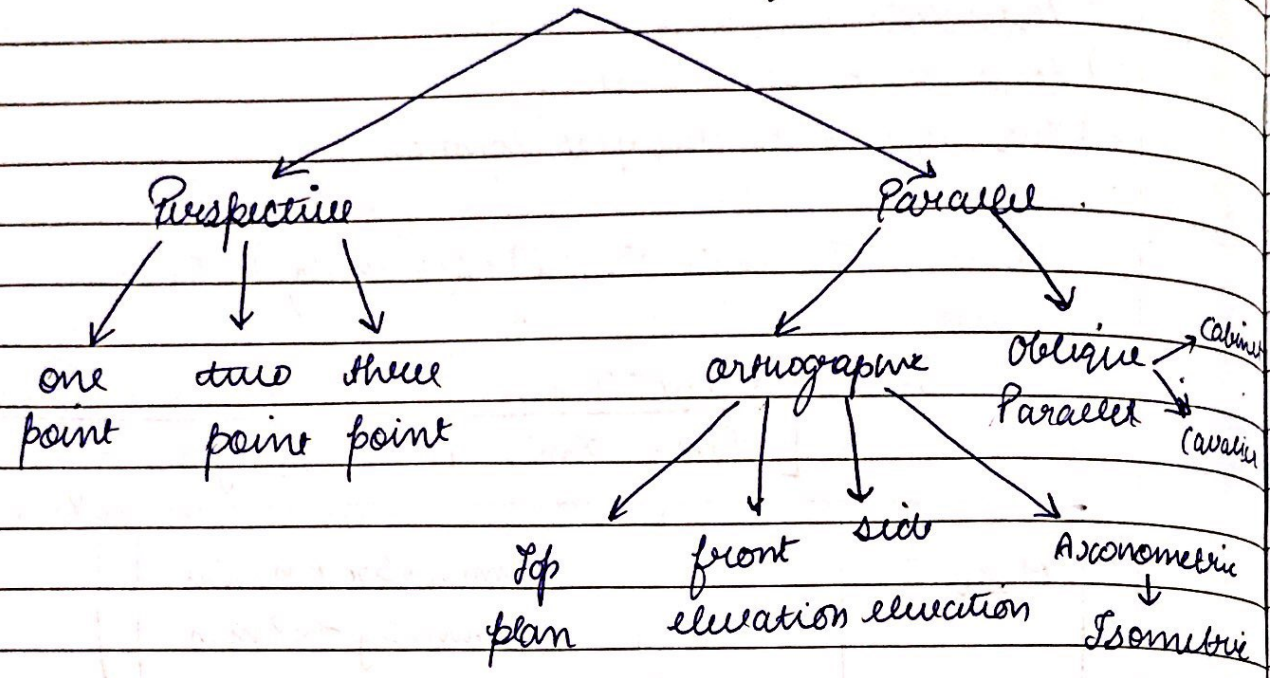
$$s_x = \left[\frac{u_{max} - u_{min}}{x_{max} - x_{min}}\right], \quad s_y = \left[\frac{v_{max} - v_{min}}{y_{max} - y_{min}}\right]$$

$$M_{wv} = \begin{bmatrix} s_x & 0 & -x_{min} \cdot s_x + u_{min} \\ 0 & s_y & -y_{min} \cdot s_y + v_{min} \\ 0 & 0 & 1 \end{bmatrix}$$

* Projections :-

It is the process of transformation of a point in 3-D co-ordinate system to the points in 2-D co-ordinate system. Thus, projection of a 3-D object is defined by straight projection rays (projectors) emanating from the centre of projection (COP) passing through each point of the object & intersecting the projection plane.

Planar Geometric Projections



(1) Perspective Project^{n} :- This projection is normal to human eye.

→ Distance from the centre of projection to the projection plane is finite

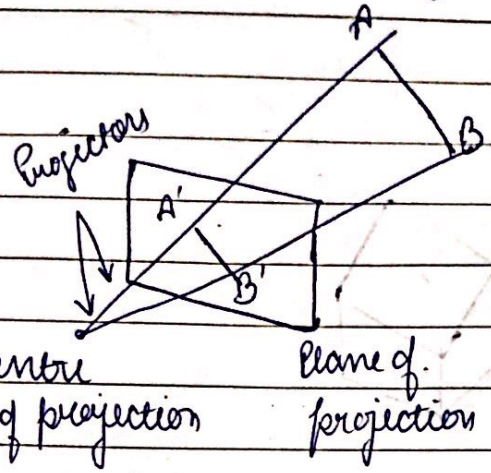
→ objects which are farther appears smaller than the objects of same size which are nearer. This is because of perspective foreshortening.

→ Perspective foreshortening explains that the size of the perspective projection of the object varies inversely with the distance of the object from the centre of projection (COP). Hence, we don't get the real size of the object. However, they appear realistic to our eyes.

→ The perspective projection of any set of parallel lines that are not parallel to the projection plane converge to a point called vanishing point.

→ A perspective projection can have three principle

vanishing points according to three principle axis.

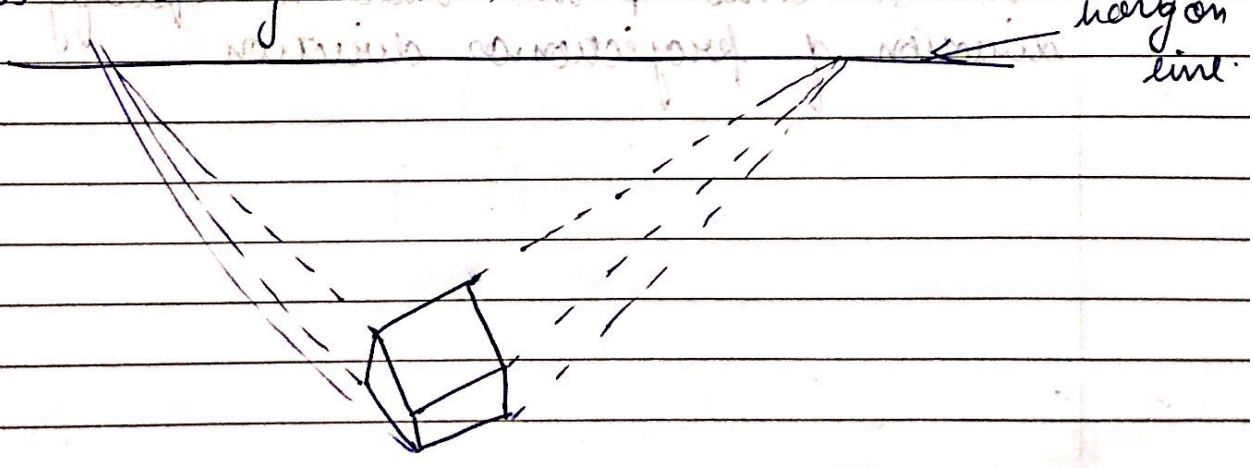


One-point perspective : A graphic has one-point perspective when it contains only one vanishing point on the horizon.

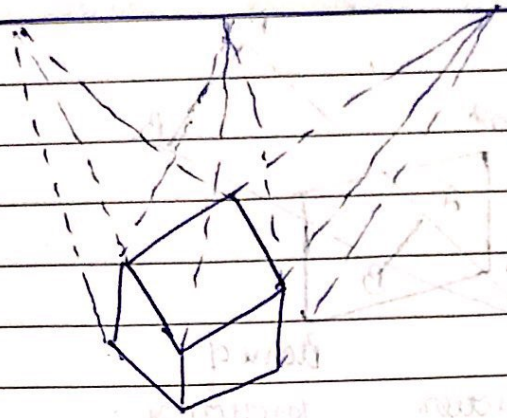
→ Any object that is made up of lines either directly $\perp^{lr}$ with the viewer's view of sight or directly $\perp^{r}$ can be represented with one-point perspective.

eg :- Images of road, railway tracks & building facing the viewer.

Two point perspective : A graphic has two point perspective when it contains two vanishing points on the horizon. This projection exist when the plane of projection is parallel to a cartesian scene in one axis (usually z-axis).

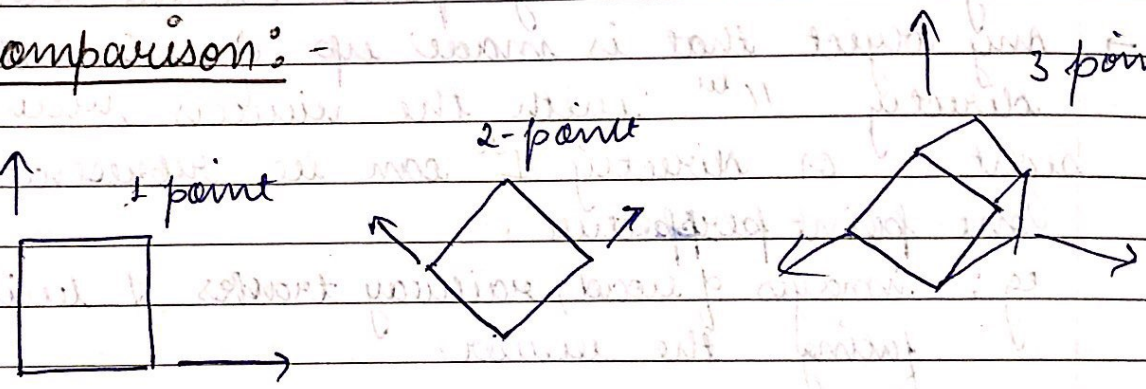


Three point perspective :- In addition of two vanishing point there is now one for the vertical lines of the wall facade.



It exists when perspective is the view of cartesian scene where the picture plane is not parallel to any of the three axis.

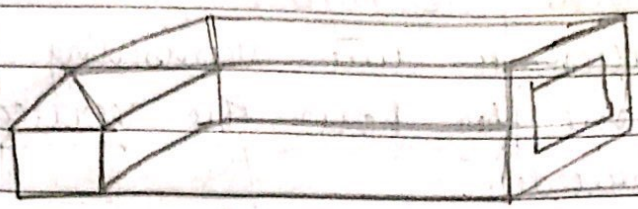
* Comparison :-



② Parallel Projection :-

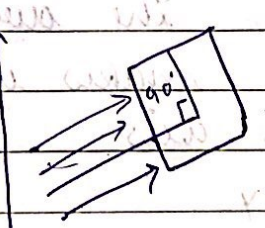
In this rays coming from object converge at infinity.

→ The distance from the centre of projection to the projection plane is infinity. Therefore, projectors are ||el lines & we need to specify a direction of projection or direction



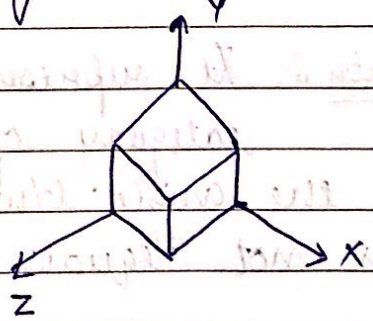
* **Orthographic Projection** :- Projectors in a orthographic projection apart from being parallel are also perpendicular to the view plane. orthographic projection occurs when the direction of projection is $\perp^r$ to the plane of projection.

→ **Axonometric projection** :- It is a type of orthographic projection in which the direction of projection is not parallel to any of the three perpendicular axis.



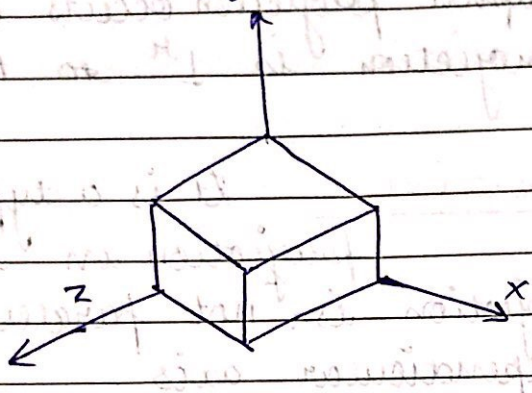
Axonometric projections are further divided into 3 classes :-

(i) **Isometric** :- The direction of projection make equal angles with all three principle axis.

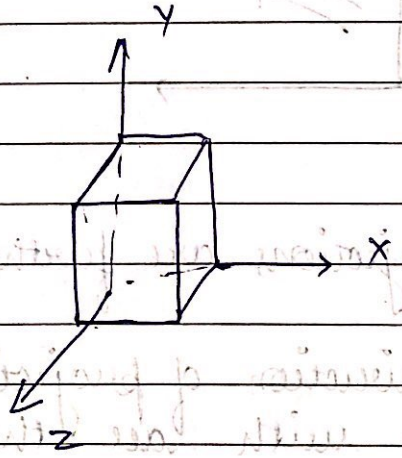


Three-point perspective:
In addition to two vanishing point there is now one for how the vertical lines of the wall recede.

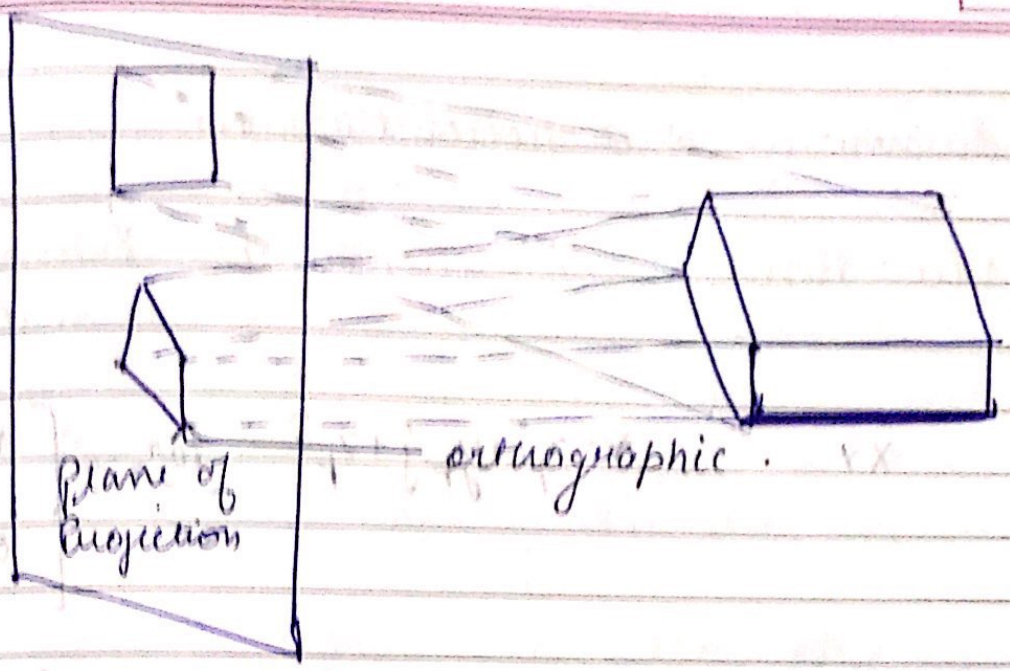
(ii) Dimetric Projection - The direction of projection makes equal angles with exactly two of the principle axis.



(iii) Trimetric Projection - The direction of projection makes unequal angles with the 3 principle axis.



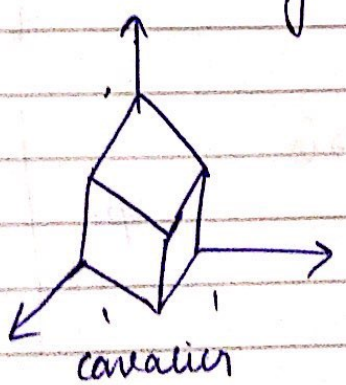
* Oblique Projection :- It represents the second category of parallel projection. It occurs when the angle b/w projectors & plane of projection is not equals to 90°.

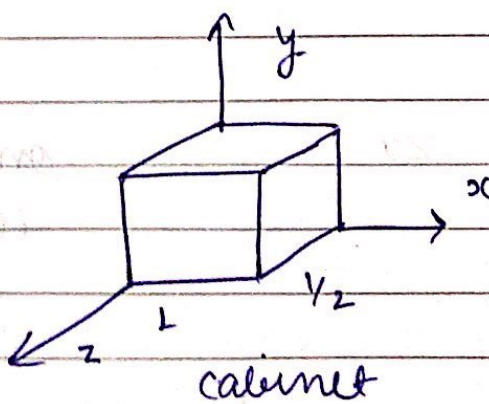


Oblique projections are further define as cavalier & cabinet projections.

Cavalier Projection :- The direction of projection so chosen that makes an angle with the projection plane so that lines ⊥r to projection plane have the same length as the line itself. (There is no foreshortning.)

Cabinet Projection- The direction of projection ~~is~~ so chosen that makes an angle with the plane of projection so that lines ⊥r to the projection plane project at one half of their actual length.





Summary of Projection Matrices

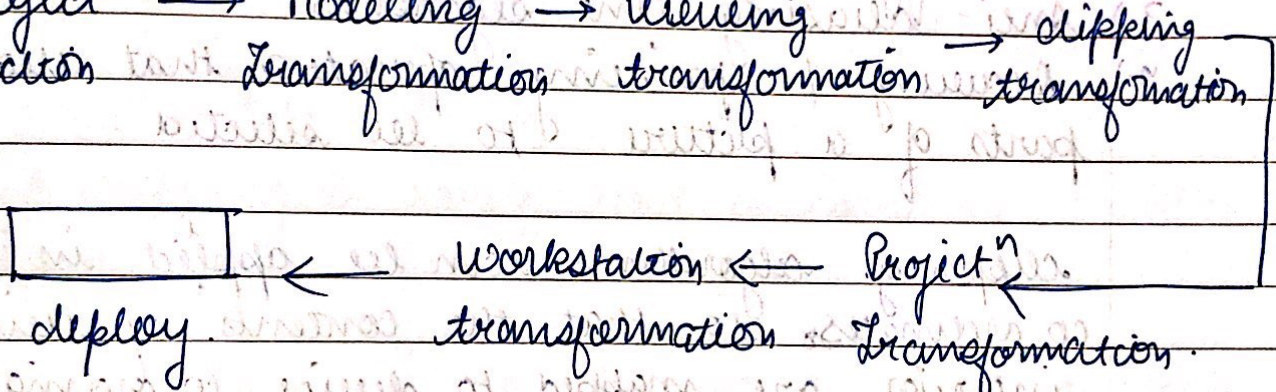
Type of Projection	View Plane	Direction vector	Transformation matrix
Parallel	XY	$\vec{V} = x_p\hat{i} + y_p\hat{j} + z_p\hat{k}$	$Par_{z=0} = \begin{bmatrix} 1 & 0 & -\frac{x_p}{z_p} \\ 0 & 1 & -\frac{y_p}{z_p} \end{bmatrix}$
	YZ	$V = x_p\hat{i} + y_p\hat{j} + z_p\hat{k}$	$Par_{x=0} = \begin{bmatrix} 1 & 0 & -\frac{y_p}{x_p} \\ 0 & 1 & -\frac{z_p}{x_p} \end{bmatrix}$
	ZX	$V = x_p\hat{i} + y_p\hat{j} + z_p\hat{k}$	$Par_{y=0} = \begin{bmatrix} 1 & 0 & -z_p/y_p \\ 0 & 1 & -x_p/y_p \end{bmatrix}$

Type of Projection	View Plane	Look of Projection	Matrix
Perspective	xy	on -ve z-axis $(0, 0, -d)$	$P = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
	yz	on -ve x-axis $(-d, 0, 0)$	$P = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
	zx	on -ve y-axis $(0, -d, 0)$	$P = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

* 3-D Viewing:-

Unlike 2-D, 3-D objects are clipped against view volume & projected on projection plane, then mapped to the view port for the display, only those objects within the view volume will appear on the display device.

Note*: Projection operation can take place before or after view volume clipping.



Date: YOUVA

* 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. The region against which an object is to be clipped is a clip window.

Applications of clipping :-

1.) Extracting part of a defined scene for viewing
2.) Identifying visible surface in 3-D view.
3.) Anti-aliasing line segments
4.) Drawing & painting operation that allows parts of a picture to be selected.

clipping algorithms can be applied in world co-ordinates so that the contents of window interior are mapped to device co-ordinate. Alternatively, the complete world co-ordinate picture can be mapped to device co-ordinates, then clipped against viewport boundaries.

Types of clipping :-

1.) Point clipping.
2.) Line clipping
3.) Area clipping (Polygons)
4.) curve clipping
5.) Text clipping.

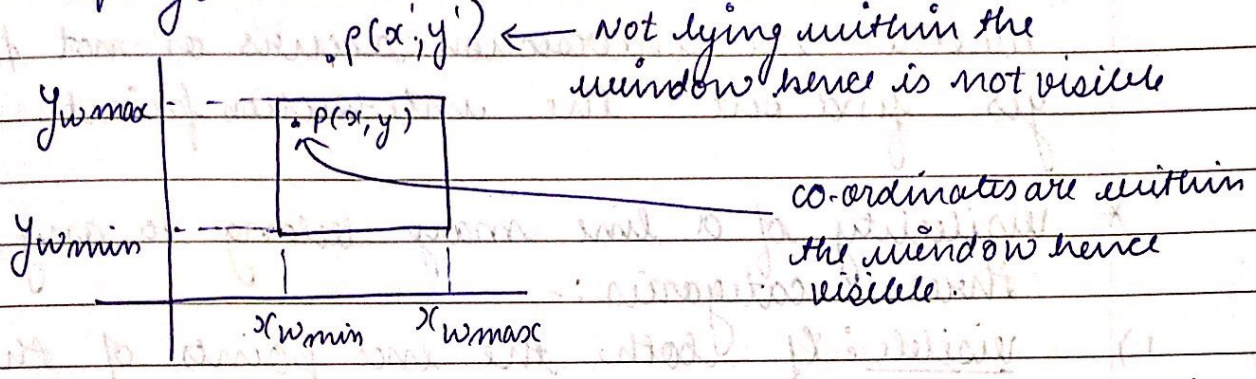
* Point clipping :-

→ Assume a point (x, y) is to be displayed on the screen. We have to decide if this point lies within the clip window or not.

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

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

Then the point (x, y) lies within the window & can be displayed.

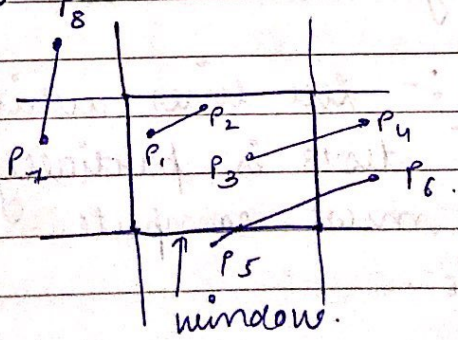


* Point clipping is rarely used as compared to other clipping.

* It can be used in situations which involve particle movements such as explosion, dust etc. for science experiments such as compton scattering etc.

* Line clipping :-

A line consist of series of number of points from the starting point to the ending point to clip a line we need to consider only its endpoints.



In this diagram we observe the line P_1, P_2 is completely inside the window the line P_7, P_8 has both end points outside the window. Line P_3, P_4 has one end point outside the window so, we need to calculate the intersection point with the window.
Line P_5, P_6 has both the end points outside the window, but it intersects the window, hence we need more calculations to check whether the intersection occurs or not & if yes find out the intersection point.

* visibility of a line may belong to any of the three categories :-

1.) Visible : If both the end points of the line lie inside the window is called as visible ex : $P_1 P_2$.

2.) Non-visible : If both the end points are outside the window with no intersection & it qualifies any one of the four inequalities :-

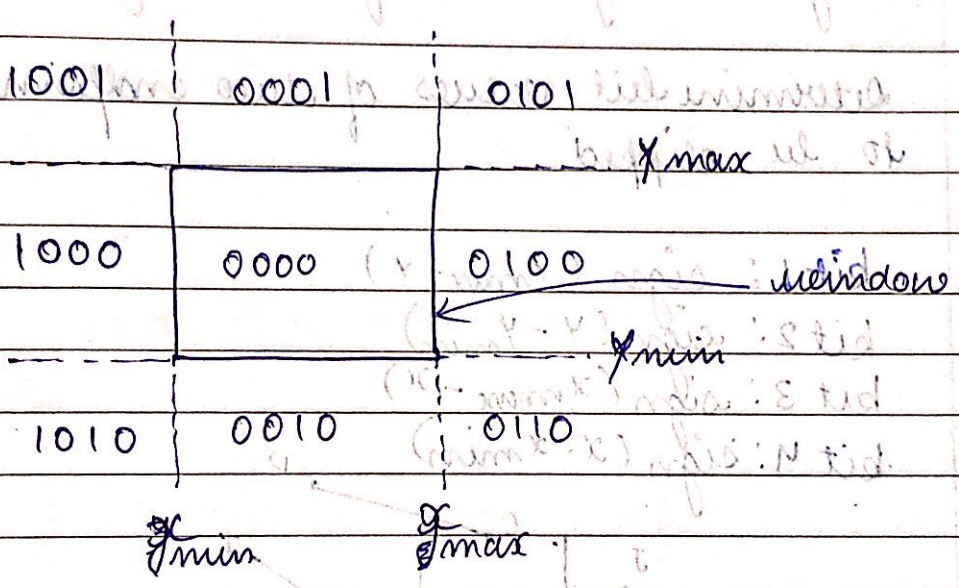
$x_1, x_2 > x_{w,min}$ → Right
$x_1, x_2 < x_{w,max}$ → left
$y_1, y_2 > y_{w,min}$ → above
$y_1, y_2 < y_{w,max}$ → below.

3.) Clipping candidate :- The lines which fail above two tests is partially visible in this case, we must compute intersection points.
ex : $P_3 P_4$, $P_5 P_6$.

* Cohen-Sutherland line clipping algorithm :-

This algorithm quickly common & trivial cases, visible & non-visible cases. This algorithm also reduces calculations by identifying lines which can be accepted or rejected by performing initial tests on a line to determine the intersection.

If the line cannot be trivially accepted or rejected an intersection of the line with a window edge is determined. This algorithm uses divide & conquer strategy.



The algorithm divides the process into 3-phases.

Phase 1: Identify the totally visible lines & draw them.

Phase 2: Identify the totally non-visible & ignore.

Phase 3: Identify those lines which intersect the window & compute intersection point.

Page No.:
Date:

Phase 1 — Draw visible lines

↓

Phase 2 — Ignore non-visible lines

↓

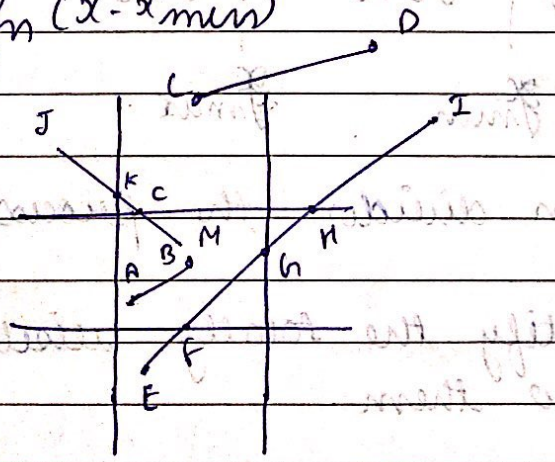
Phase 3 — Line is clipping candidate compute intersection points

Step 1: Input x_{wmin}, x_{wmax}, y_{wmin}, y_{wmax}, $A(x_1, y_1)$, $B(x_2, y_2)$

Step 2: Assign a region code for each of the line endpoints.

Step 3: Determine bit values of the endpoints of the line to be clipped

bit 1: sign $(Y_{max} - Y)$
bit 2: sign $(Y - Y_{min})$
bit 3: sign $(x_{max} - x)$
bit 4: sign $(x - x_{min})$



1.) Line AB has region code 0000 for both endpoints so it can be trivially accepted.

2.) Line CD has 1000 for C and 1010 for D.
On performing AND operation 1000.

AND
1010
1000

we dont get 0000 so it can be trivially rejected.

3.) Line EI has code 0100 for E and 1010 for I. Thus can't be trivially accepted or rejected starting from E at the bottom it is clipped to FI. outcode of F is 0000 so no further clipping from this side.
Now from I, it is ~~clipping~~ clipped to H. H has code of 0010. So clipped with respect to right edge. G has code of 0000. So, no further clipping.

Intersection point calculation: (x_0, y_0) (x_1, y_1)
start end

1.) Top edge:
$$x = x_0 + (x_1 - x_0) \cdot \left(\frac{y_{max} - y_0}{y_1 - y_0} \right)$$

2.) Bottom edge:
$$x = x_0 + (x_1 - x_0) \cdot \left(\frac{y_{min} - y_0}{y_1 - y_0} \right)$$

3.) Left edge:
$$y = y_0 + (y_1 - y_0) \left(\frac{x_{min} - x_0}{x_1 - x_0} \right)$$

4.) Right edge:
$$y = y_0 + (y_1 - y_0) \left(\frac{x_{max} - x_0}{x_1 - x_0} \right)$$

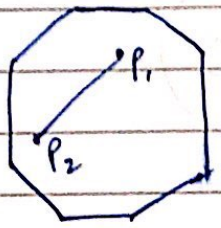
* Polygon clipping :-

Polygon :- An object with more than two vertices is called polygon.

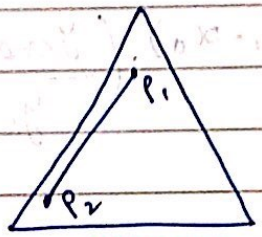
Polygon:-

Convex Polygon:- A polygon is called a convex polygon if a line obtained after joining two interior points of the polygon lies completely inside the polygon.

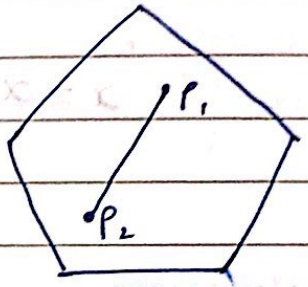
Concave Polygon: A polygon is called a concave polygon if at least one pair of points such that the line joining the points belong to polygon that does not lie completely inside the polygon.



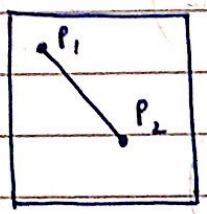
convex



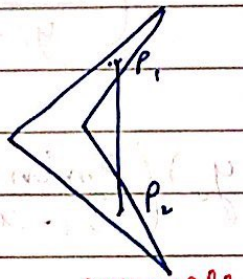
convex



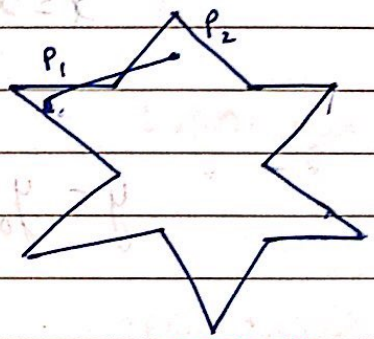
convex.



convex



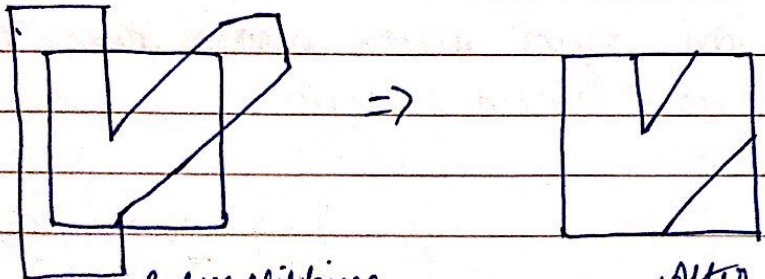
concave



concave.

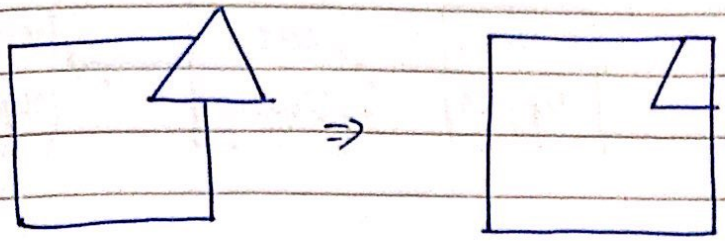
* Sutherland-Hodgman Polygon clipping algo:-

This algorithm deals with clipping of a polygon.



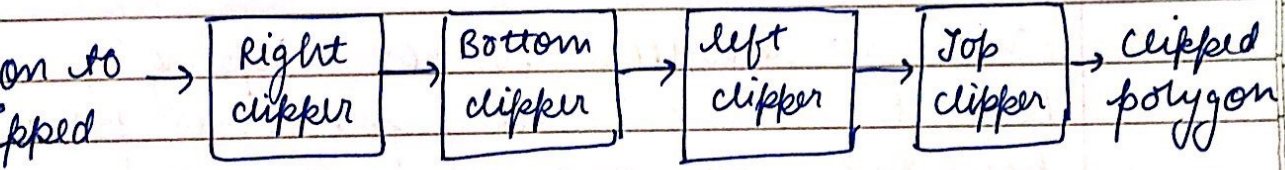
Before clipping

After clipping

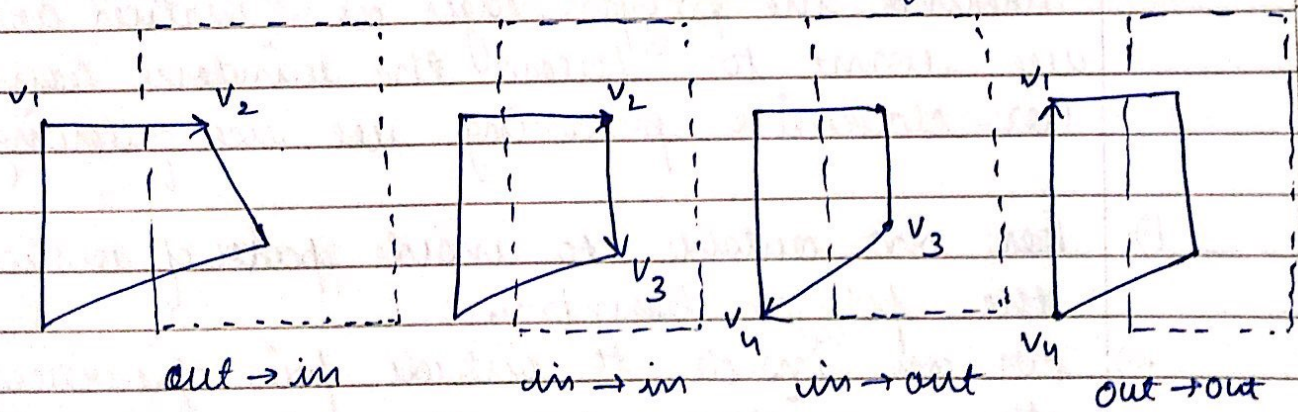


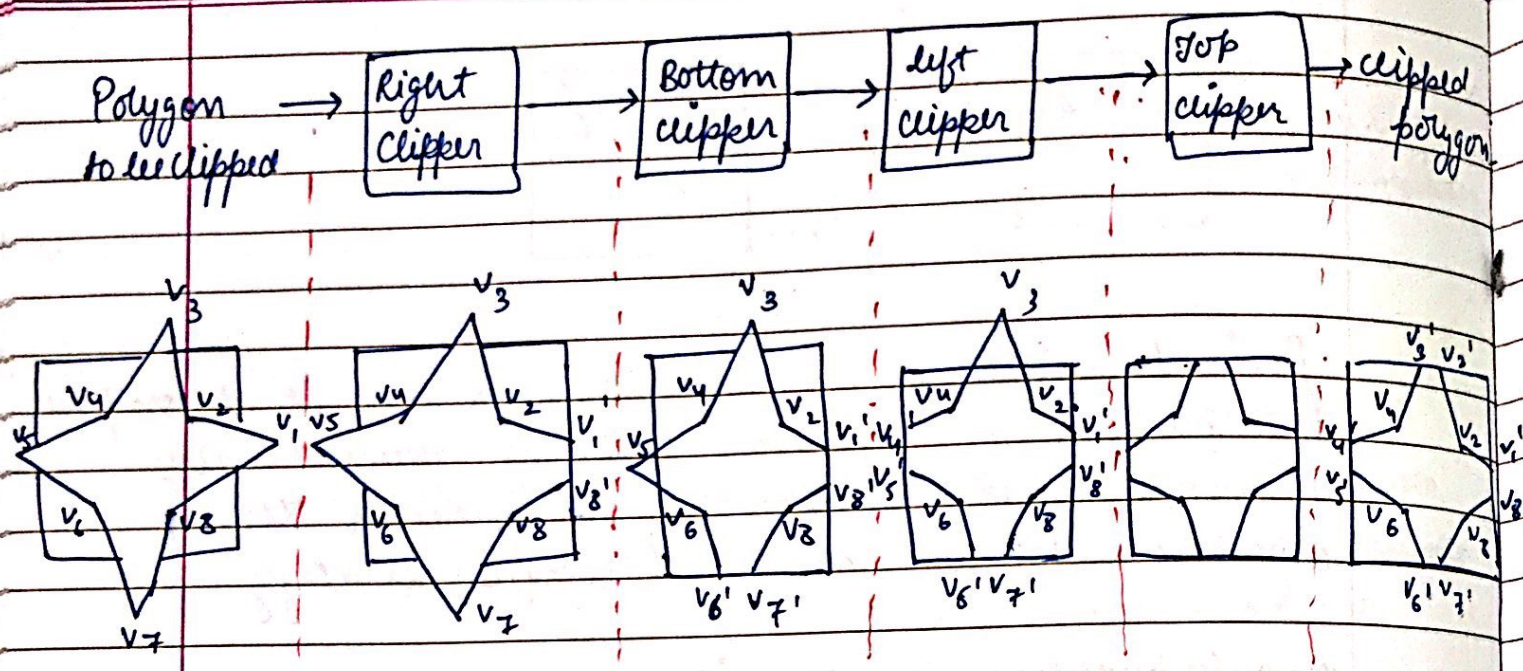
Basic principle of this algorithm is that it decomposes the problem of polygon clipping against a clip window into identical sub-problem. This algorithm follows divide & conquer strategy.

A sub problem is to clip all polygon edges successively against a single clip edge. The outputs of the sub-problem are considered input to the next sub problem of clipping against the other window page.



There are 4 possible cases when processing vertices in sequence aroound the perimeter of a polygon.





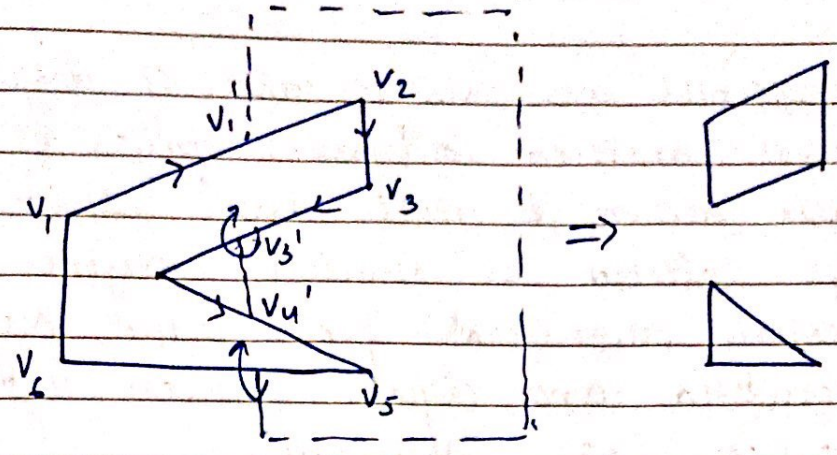
Disadvantages:-

→ convex polygon are correctly clipped but concave polygon may be displayed with extra line. This occurs when the clipped polygon has two or more sections which are seprate.

* Weiler - Atherton Polygon clipping :-

Here, the vertex processing procedures for window boundries are modified so that concave polygons are displayed correctly. The basic idea in this algorithm is that instead of always proceeding around the polygon edges as vertices are processed we want to follow the window boundaries. For clockwise processing we use following rules:-

1.) For an outside to inside pair of vertices follow the polygon boundary.
2.) For an inside to outside pair of vertices follow the window boundary in a clockwise direction.



* Hidden lines and Hidden surfaces:-

whenever 3-d objects are projected on 2-d planes lines & surfaces that are closer to the viewer block the lines and surfaces that are far. The hidden surfaces must be removed to in order to render a realistic image.

The solution involves determination of depth & visibility for all the points / lines / surfaces in the picture. This process is called visible line / visible surface determination or hidden line / hidden surface elimination. Based upon whether to deal with objects directly or with there projected image. visible surface algorithms are broadly classified as:-

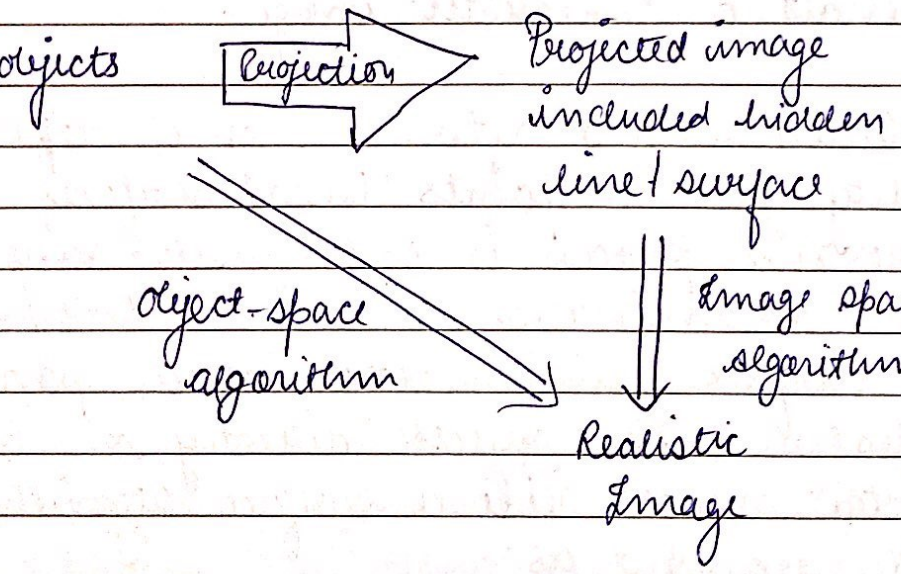
visible surface algo
- object - space method.
- Image - space method.

Object space method.

These methods are implemented on physical co-ordinate system in which the object is defined. These

methods are independent of display device. These method compares parts of object with each other to determine which surface should be labeled as visible. objects which are totally obscured from other objects are removed and objects which are partially obscured are clipped

Image space method:- Are partially implemented of screen co-ordination system. These depends on the resolution of display device. Image space method decides the visibility of an object at each pixel position



which ever method we follow we use sorting & coherence methods to improve performance.

Coherence properties:-

→ Object coherence:- If one object is entirely separate from another object do not compare them.

→ face coherence :- There are smooth variations in properties across the face. So we used the incremental modification in the depth values of the adjacent parts of the face.

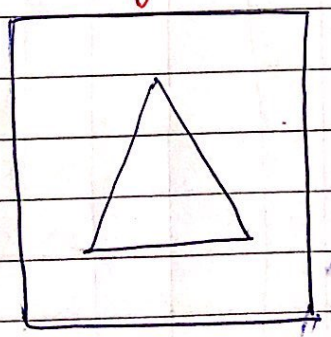
→ Edge coherence :- visibility changes if edges cross behind a visible edge or penetrates a visible face.

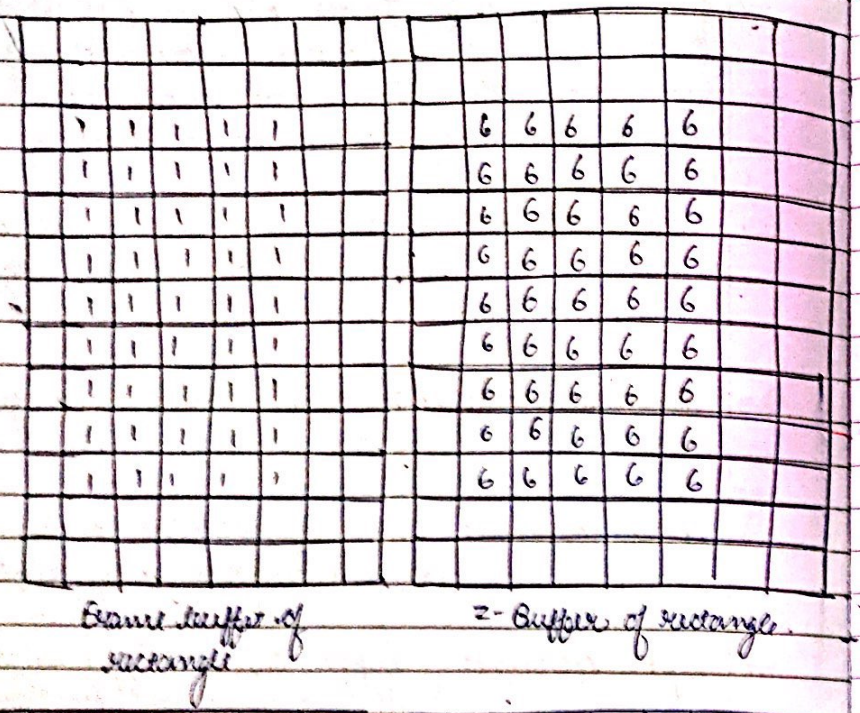
→ scanline coherence :- values don't change much from one scanline to the next.

→ Area coherence : Span of adjacent group of pixels is covered by some visible face.

→ Depth coherence :- Depth values differ slightly in adjacent parts of the same surface.

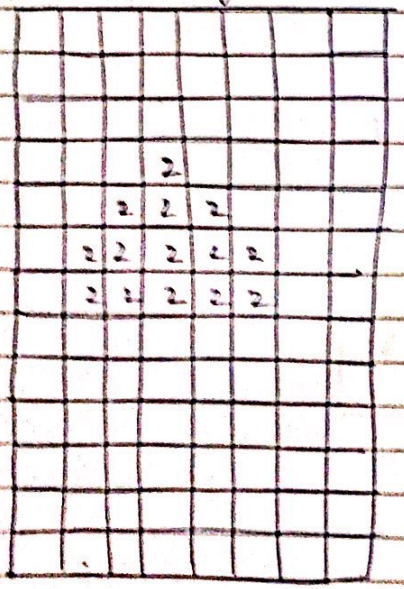
✱ z-Buffer Algorithm :-



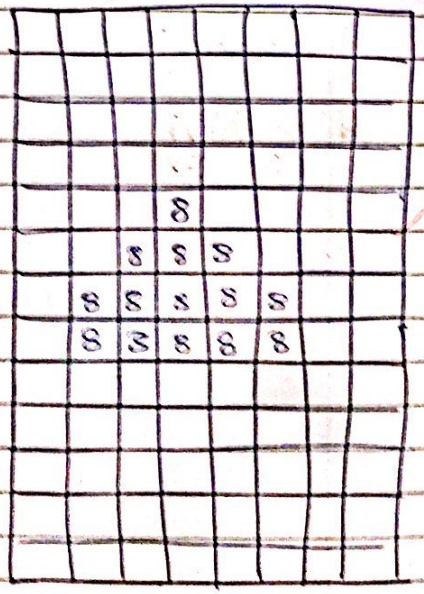


Frame buffer of rectangle

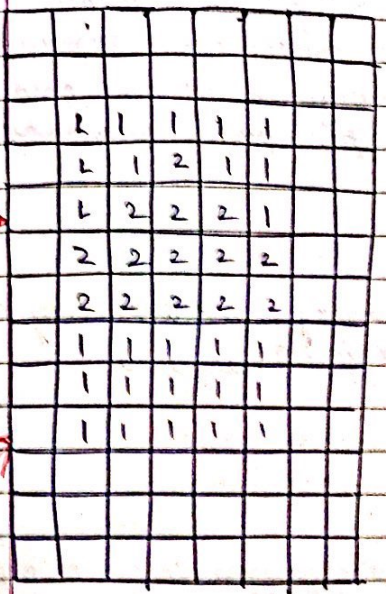
z-Buffer of rectangle



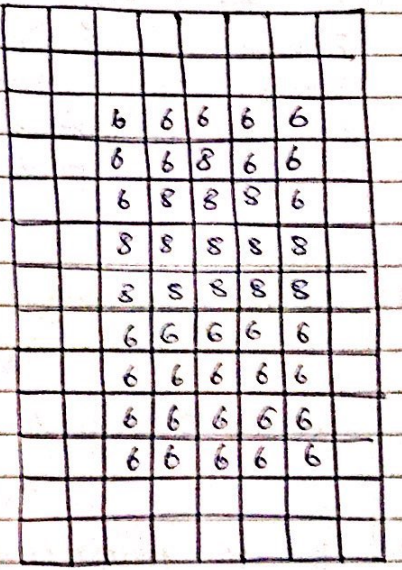
Frame buffer of triangle



z-Buffer of triangle



Frame Buffer



z-Buffer

It is one of the simplest algorithm for the hidden surface removal. It is also known as depth buffer algorithm and is an image space method. The z-value of a new pixel to be written is compared to z-value of pixel already stored in the z-buffer, if the comparison indicates z-value of new pixel is greater than z-value already stored then z-buffer is updated and new intensity is written in the frame buffer, otherwise no action is taken.

Algo:-

Step 1:- Set the frame buffer to background intensity color.

$$\text{frame buffer}(x,y) = I_{background}.$$

Step 2: Set the z-buffer to minimum value.

$$z_{Buffer}(x,y) = 0.$$

Step 3: for each pixel (x,y) in the polygon, calculate depth $z(x,y)$ at that pixel

Step 4: Compare the depth $z(x,y)$ with the value stored in z-buffer at the location.

If $z(x,y) > z\text{-buffer}(x,y)$ then write the polygon attributes to the frame buffer. at the pixel location & replace z-buffer (x,y) with $z(x,y)$ otherwise no action taken.

$$z_{Buffer}(x,y) = z(x,y)$$

$$\text{framebuffer}(x,y) = I_{poly}(x,y)$$

step 5:- Repeat steps 2 to 4 for all polygons.

step 6:- Finally when all the polygons have been processed the depth buffer contains depth value for all visible surfaces & the frame buffer contains the corresponding intensity values.

Disadvantages:-

1.) Double memory requirement.
2.) Spends time while rendering polygons that are not possible.
3.) Device dependent.
4.) It can process only opaque surface.

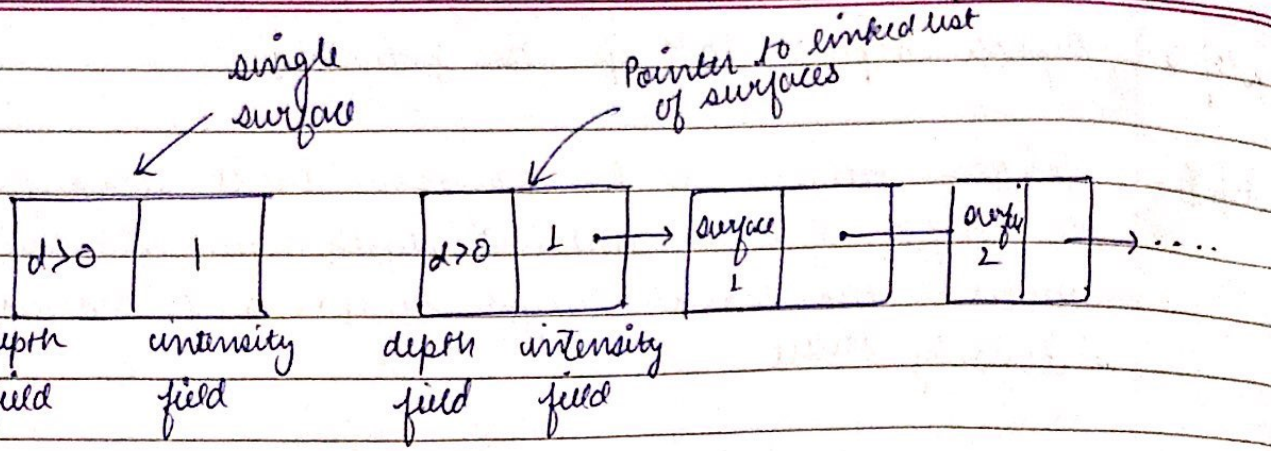
* A-Buffer method:-

The depth buffer method can detect only one visible surface and can store the intensity value for one surface only at each pixel location. Accumulation buffer method overcomes this situation by expanding its buffer so that it can store more than one intensity values for each pixel position.

A Buffer method has two fields:-

(i) Depth field :- It stores a positive or negative real no.

(ii) Intensity field :- It stores either intensity value or pointer to a linked list of surfaces.



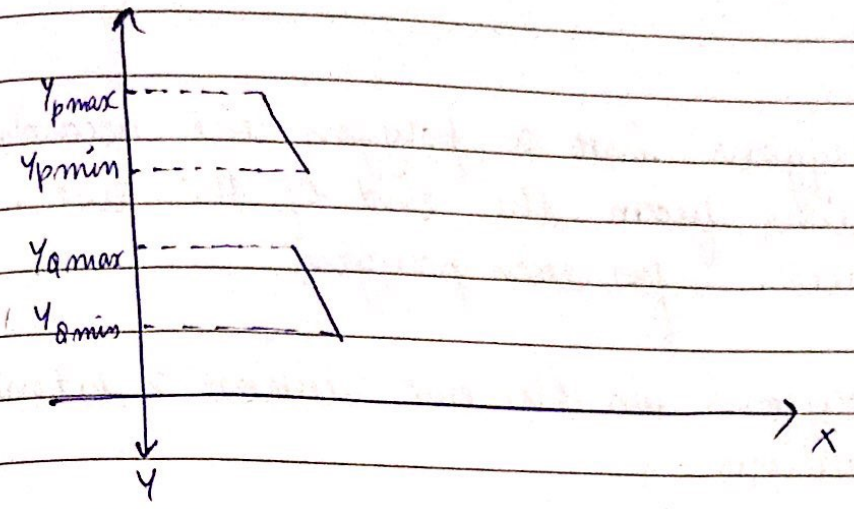
operation of A-buffer is similar to z-buffer. Each scan line is processed one by one but not for a single surface but for determining surface overlaps at each pixel position. opacity factor is determined which tells about the % of transparency of the surface.

* Painter's Algorithm :-

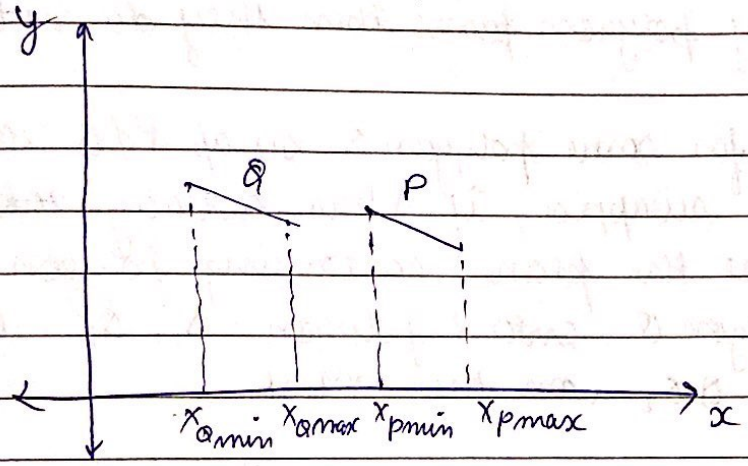
It is based upon the principle that more distance polygons are painted first and over it the next farther polygon totally or partially obscuring them from the view.
Here comes the problem to find priority ordering of the polygons in order to determine which polygons are to be seen converted first.

→ List 0 → consider two polygons p & q.
Polygon p does not obscure polygon q if $z_{q\,max}$ is smaller than $z_{p\,min}$.

List 1 → consider two polygons p & q. Polygon p does not obscure polygon q if $y_{q\,max}$ is smaller than $y_{p\,min}$.
The y extent of p & q does not overlap if $y_{q\,max} < y_{p\,min}$.



Test 2 → The x extent of p & q does not overlap if $x_{qmax} < x_{pmin}$



Test 3 → All the vertices of p lie on that side of the plane containing q which is farthest from the view point

Test 4 → All the vertices of q lie on that side of the plane containing p which ~~ties~~ is closest from the view point.

Test 5 → The projections of polygons p and q on the xy screen do not overlap.

Page No.:
Date:

Algorithm :-

step 1: sort all polygons into a polygon list according to Z_{max}. Starting from the end of the list, assign priorities for each polygon.

step 2: Find all polygons in the list whose Z extents over that of previous.

step 3: for each polygon perform test 1 to test 5.

A (a) If every polygon passes then they do not obscure.

(b) If false for some polygon Q swap P & Q on the list tag Q as swapped, if Q has already ~~taken~~ been tagged, use the plane containing polygon P to divide polygon Q into 2 polygon Q_1, Q_2. Remove Q and place $Q_1 Q_2$ on the list.