Unit III

Windowing and Clipping

Window -

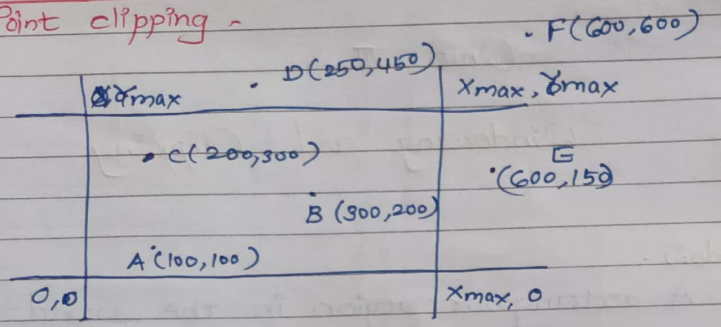
- A rectangular region in the world coordinate system
- Coordinate system used to find an object in the real world.

Viewport -

- The selection of the screen where the photos around the window in the world coordinate system will be drawn.
- A coordinate system / transformation is required to display an image, encompassed by the window in the viewport.

-The process of mapping the part of world coordinate scence to device is referred to as viewing transformation.

Point clipping -



(X_{min}, Y_{min}) and (X_{max}, Y_{max}) represents the lower left and top-right corners of clipping window.

(X, Y) is inside the region only if

$X_{min} \leq X \leq X_{max}$

$Y_{min} \leq Y \leq Y_{max}$.

Steps

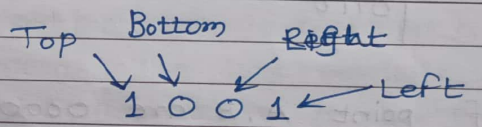
1. Determine min and max coordinates of viewing pane.
2. Read the coordinates for a point in question.
3. Check whether given input point lies in between minimum (X_{min}, Y_{min}) and maximum (X_{max}, Y_{max}) coordinate of viewing pane.
4. If yes display the point which lies inside the region otherwise discard it.

Line clipping

Cohen-Sutherland Line Clipping

- Divides a two dimensional space into 9 regions.

1001	1000	1010
0001	0000	0010
0101	0100	0110



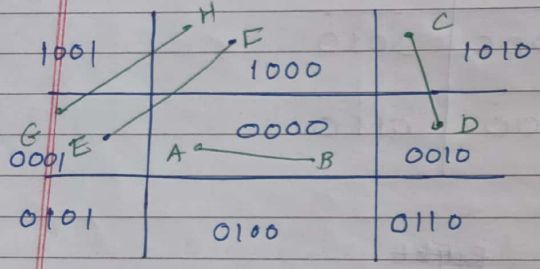
Four bit code calculation -

1. Set first bit 1 if end point of line lies to left of window $(X < X_{Wmin})$
2. Set second bit 1 if end point of lines to right of window. $(X > X_{Wmax})$
3. Set third bit 1 if end point of line lies to bottom of window $(Y < Y_{Wmin})$.
4. Set fourth bit 1 if end point of line lies to top of window $(Y > Y_{Wmax})$

* 3 possibilities :

1. Line is completely inside the window. (should be displayed)
2. Line lies completely outside the window. (should be completely removed).
3. Line can be partially inside the window.

⇒ Case I :



- The outcode of point A, B are 0000, lies completely inside the window.
- Line is visible, should not be clipped.

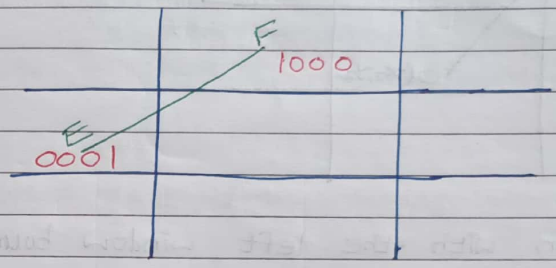
⇒ Case II :

- The outcode of C is 1010 and D is 0010
- Result of logical AND is 0010.

```
C   1 0 1 0
D   0 0 1 0
    -------
    0 0 1 0
```

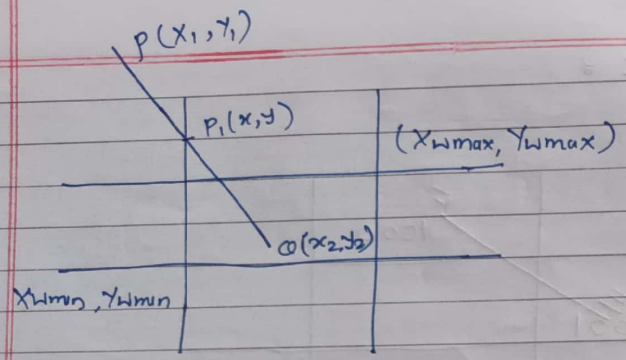
As it is non-zero. Line is outside the window and not to be displayed.

⇒ Case III :



outcode of E ⇒ 0 0 0 1
outcode of F ⇒ 1 0 0 0
0 0 0 0

- The output of logical AND of EF is zero, It means line lies partially inside the window.
- Find the intersection point : EF with the left boundary.
- Consider P_1 as point of intersection. EP1, P_1 has outcode 0000
- Consider line FP1
  Outcode of F is 1000 i.e. Top, is outside the window.
- Find the intersection point with the top boundary. P2.
- P_2 ⇒ 0000
- $P_1 P_2$ is visible.



Intersection with the left window boundary, so x co-ordinate will be x_{wmin}. Find Y value.

For line PQ slope $(m) = \frac{y_2 - y_1}{x_2 - x_1}$.

Slope of $PP_1 = \frac{y - y_1}{x - x_1}$

$$m = \frac{y - y_1}{x - x_1}$$

$$y - y_1 = m(x - x_1) \quad \text{but } x = x_{wmin}$$

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

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

For left boundary

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

For right boundary

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

For top boundary

$$m = \frac{y - y_1}{x - x_1}$$

$$x - x_1 = \frac{y - y_1}{m} \quad \text{as } y = y_{wmax}$$

$$x = x_1 + \frac{y - y_1}{m}$$

$$x = x_1 + \frac{y_{wmax} - y_1}{m}$$

For bottom boundary

$$x = x_1 + \frac{y_{wmin} - y_1}{m}$$

Algorithm :-

1. Read (Xmin, Ymin) and (Xmax, Ymax).
2. Read A(X_1, Y_1) and B(X_2, Y_2) end points of line.
3. Compute outcode

   If $y_1 > y_{max}$ then outcode_A(1) = 1 else 0
   $y_1 < y_{min}$ then outcode_A(2) = 1 else 0
   $x_1 > x_{max}$ then outcode_A(3) = 1 else 0
   $x_1 < x_{min}$ then outcode_A(4) = 1 else 0
   Same for B.

4. If both endpoints have an outcode 0000 then given line is completely inside and display the line.
5. If step 4 fails perform the logical AND operation on both outcodes.

5.1a. If the result of logical AND is not 0000, then line is completely outside.

5.2 Else line is partially visible.

5.2a. Choose an endpoint of line that is outside the given window.

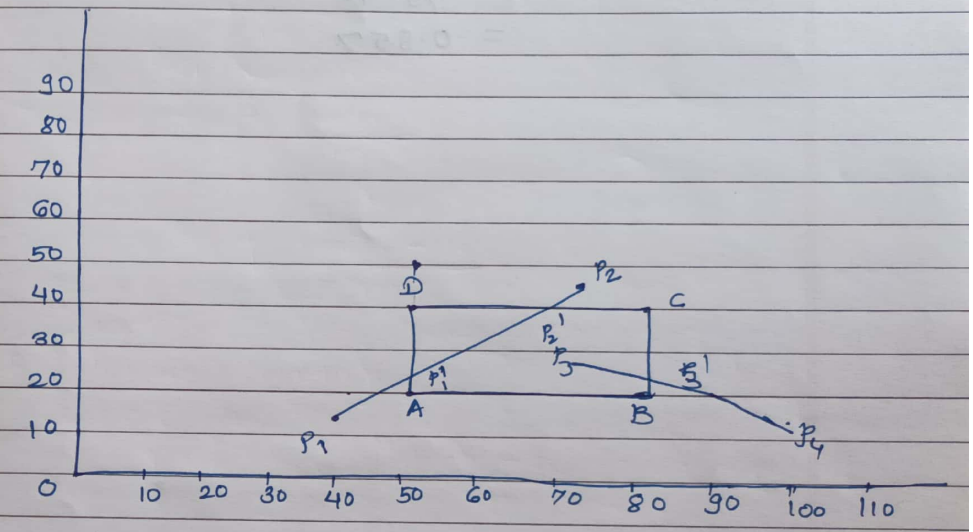
5.2b. Find the intersection point of window

5.2.c. Replace endpoint with intersection point and update the outcode.

5.2.d. Repeat step 2.

6. Repeat step 3 for all lines.

Use Cohen-sutherland algorithm to clip two lines P_1(40,15), P_2(75,45) and P_3(70,20), P_4(100,10) against a window A(50,50), B(80,10), C(80,40) and D(50,40).



Outcode of P1 = 0001
Outcode of P_2 = 1000

P_1 AND P_2 0001
1000
0000

Partially visible.

outcode of P_1 is 0001 i.e. left boundary.

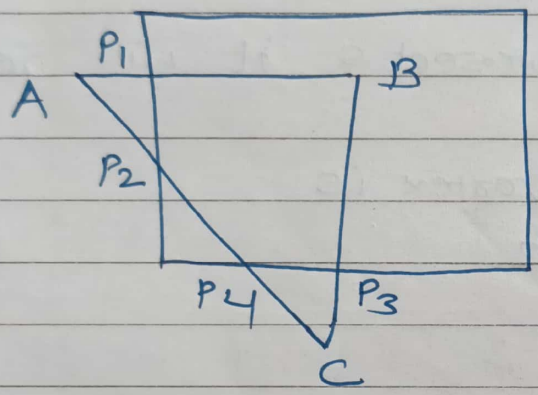
Slope of line = $m = \frac{y_2 - y_1}{x_2 - x_1}$

$= \frac{45 - 15}{75 - 40} = \frac{30}{35}$

$= 0.857$

Polygon Clipping -

Performs a clipping of polygon against each window edge.
Accepts an ordered sequence of vertices V1, V2, V3 .... Vn and set of vertices defining the clipped polygon.



1. Clip Left :
- Consider the polygon ABC to be clipped against the window as shown in figure.
- The edge AB and edge AC intersects the left boundary.
- For edge AB, intersection point is P1 and For edge AC, intersection point is P2. Store it as P1B and P2C.

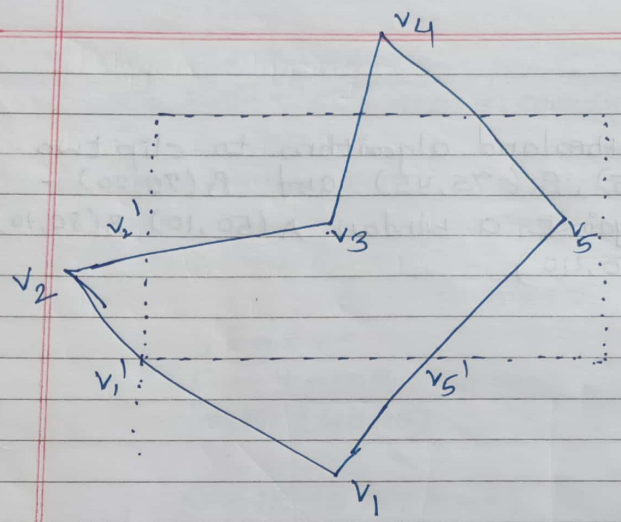
2. Clip Right :
- No edge of the polygon intersects the right side of the window so the output vertex list is not updated remains same as previous step.

3. Clip bottom.
- As the edge BC vertex B is inside the window and C lies outside the window boundary.
- store point intersection of BC i.e. P3 in the output vertex list.
- For edge AC store point P4 as intersection point in the output vertex list.

4. Clip top
- As no edge intersects, it will remain same.

Final Output vertex is
P1 B P3 P4 P2

Algorithm :
1. Read co-ordinates of polygon.
2. Read co-ordinates of clipping window.
3. Consider the left edge of window.
4. Compare the vertices of each edge of polygon, individually with the clipping plane.
5. Save the resultant intersections and vertices in the new list.
6. Repeat the process 4,5 for remaining edges
7. Stop



Input { V_1, V_2, V_3, V_4, V_5 }

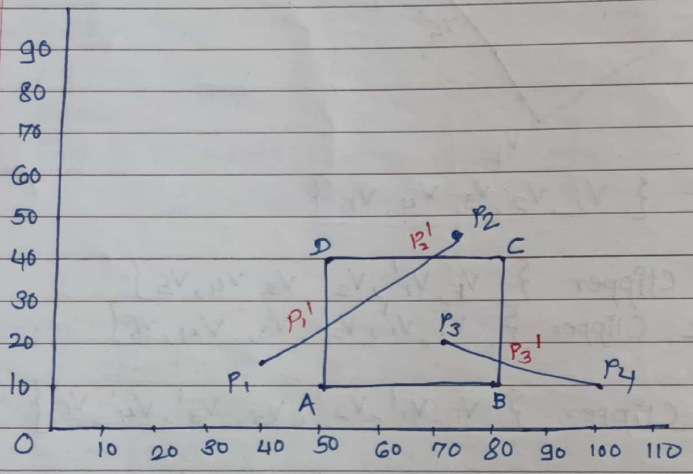
Left clipper { $V_1, V_1', V_2', V_3, V_4, V_5$ }
Right Clipper { $V_1, V_1', V_2', V_3, V_4, V_5$ }

Top Clipper { $V_1, V_1', V_2', V_3, V_3', V_4, V_5$ }

Bottom Clipper { $V_2'', V_2', V_3, V_3', V_4', V_5, V_5'$ }

Example -

Use cohen sutherland algorithm to clip two lines $P_1(40,15)$, $P_2(75,45)$ and $P_3(70,20)$ - $P_4(100,10)$ against a window A(50,10), B(80,10) C(80,40), D(50,40)



Line P_1, P_2

$P_1(40,15)$ & $P_2(75,45)$

P_1 Outcode = 0001

P_2 Outcode = 1000

0000 Logical AND

Line is partially visible.

Outcode of P_1 is 0001 i.e. it intersects the left edge.

Say the intersection point P_1'. P_1' is on line $x = x_{min} = 50$. its x coordinate is 50.

slope of line $= m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{45-15}{75-40} = \frac{30}{35}$

$= 0.857$

$y = mx + c$

$c = y - mx$

for $P_1(40,15)$

$c = 15 - (0.857 \times 40)$

$= 15 - 34.28$

$= -19.28$

$x = x_{min} = 50$.

$y = mx + c$

$= 0.857 \times 50 - 19.28$

$= 42.85 - 19.28$

$= 50, 23.57$

$P_1' = (50, 23.57)$

Outcode of endpoint P_2 is 1000. i.e. it intersects the top edge. Lets say P_2' on line Y.

$y = 40$

$y = mx + c$

$x = \frac{y - c}{m}$

$$x = \frac{40 + 19.28}{0.857} = \frac{59.28}{0.857} = 69.17$$

$P_2' = (69.17, 40)$.

$P_1'(50, 23.57)$, $P_2'(69.17, 40)$ is visible.

Line P_3P_4

$P_3(70,20)$, $P_4(100,10)$

$P_3 \Rightarrow 0000$
$P_4 \Rightarrow 0010$
$\quad\quad 0000$ Logical AND

Line is partially visible.

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{10-20}{100-70} = \frac{-10}{30} = -0.33$$

$y = mx + c$

$c = y - mx$

$P_3 = (70,20)$

$c = 20 - (-0.33) \times 70$

$= 20 + 23.1 = 43.1$

$c = 43.1$

$y = mx + c$

$= (-0.33) \times 80 + 43.1$

$= -26.4 + 43.1$

$y = 16.7$

$P_3'(80, 16.7)$, $P_3(70,20)$ line is visible.

Example

ABCD rectangular window. A(20,20), B(90,20), C(90,70) and D(20,70). Use clipping algorithm to clip lines.

i) P_1P_2 with $P_1(10,80)$, $P_2(80,90)$
ii) P_3P_4 with $P_3(10,10)$, $P_4(70,60)$

1001	1000	1010
0001	0000	0010
0101	0100	0110

0 0 0 0
↓ ↓ ↓ ↓
T B R L

Translation -

- Change the position by adding some integers to each coordinate.
- A translation by (t_x, t_y, t_z) transforms the point (x, y, z) to $(x+t_x, y+t_y, z+t_z)$.

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

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

$$P' = T \cdot P$$

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

$$P' = T \cdot P$$

$$= \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 3 & 8 \\ 2 & 4 & 9 \\ 2 & 7 & 12 \\ 1 & 1 & 1 \end{bmatrix}$$

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

Scaling -

- Used to change the size of an object.

S_x = Scaling in x direction
S_y = Scaling in y direction
S_z = Scaling in z direction

$$P' = S \cdot P$$

$$X' = S_x \cdot x \qquad Y' = S_y \cdot y \qquad Z' = S_z \cdot z$$

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Ex: A cube is defined by 8 vertices: A(0,0,0), B(2,0,0), C(2,2,0), D(0,2,0), E(0,0,2), F(0,2,2), G(2,0,2), H(2,2,2). Perform following 3D transformations on the cube.

- Translation by $T = [t_x, t_y, t_z] = [5\ 3\ 4]$
- Scaling by $S = [S_x, S_y, S_z] = [1\ 2\ 0.5]$

Translation $T = [5\ 3\ 4]$

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

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

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

Scaling by $S = [1 \quad 2 \quad 0.5]$

$$P' = S \cdot P$$

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

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

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

Rotation about x-axis -

$$X \rightarrow Y \rightarrow Z \rightarrow X$$

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

Here, y and z coordinates change whereas the x-coordinate remains the same.

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

Anticlockwise

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

Clockwise

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

Rotation about Y axis :

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

Rotation about Z axis :

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

* Rotation about an arbitrary line -

1. Translate the rotation axis such that it passes through the origin.
2. Rotate the translation axis such that it coincides one of the principal axis.
3. Perform the actual rotation operation.
4. Perform inverse rotation to bring rotation axis to its original location.

Rotation about x-axis by 90° in clockwise direction.

$$P' = R_x(\theta = -90^\circ) \cdot P$$

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

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

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

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

1. $T = \begin{bmatrix} 1 & 0 & 0 & -x_1 \\ 0 & 1 & 0 & -y_1 \\ 0 & 0 & 1 & -z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

$$\begin{bmatrix} d & 0 & a & 0 \\ -ab/d & c/d & b & 0 \\ -ac/d & -b/d & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Sheor transformation

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

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

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

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

Ex: Rotate a triangle ABC with vertices A(2,2,2), B(3,4,7), C(8,9,12) about Y-axis in anti-clockwise direction by angle 90°.

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

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

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

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

	X	Y	Z	
	2	3	8	1
	2	4	9	1
	2	7	12	1
	1	1	1	1

Reflection

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

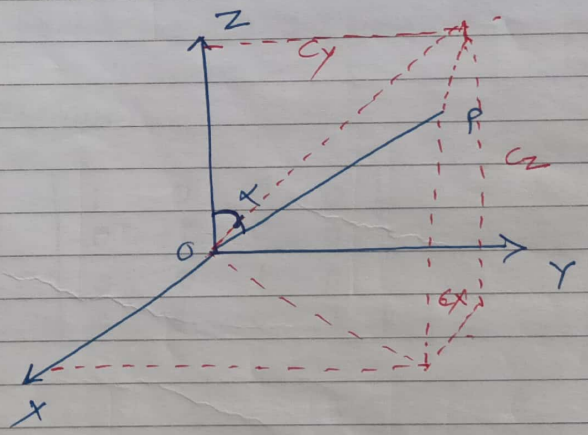
$$\text{Ref}(y=0) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

Rotation about an arbitrary axis in plane -

Assume angle θ degree, passing through the point (x_0, y_0, z_0) and direction (c_x, c_y, c_z)

1. Translate by $T = (x_0, y_0, z_0)^T$ and bring to origin.
2. Next, we rotate the axis into one of the principal axis, say Z-axis ie. rotation with respect to X and Y axis. $Z(|Rx|, |Ry|)$.
3. We rotate by θ degrees $Z(R_z(\theta))$
4. Undo the rotation
5. Undo the translation.



About x-axis rotation
to place the object in xz plane.

About Y-axis rotation
to place the object coincidence with z-axis

Reflection -

1) In XY plane

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

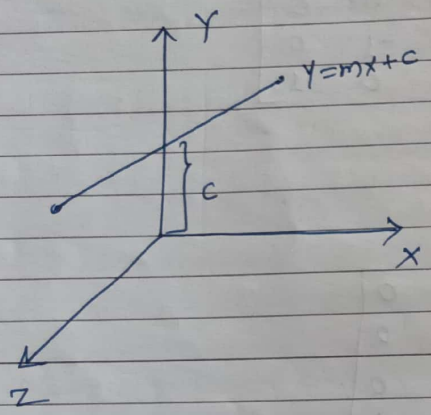
2) YZ plane

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

3) XZ plane

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

Find a sequence of transformations required to rotate a solid ~~line~~ object w.r.t line $y = mx + C$ in anticlockwise direction by angle θ.



Line $Y = mx + C$, Here C is the Y-intercept

translating line by the amount $[0\ -C]$

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

Rotate a line about X-axis by the angle α in anticlockwise direction such that it falls in XZ plane.

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

Rotate about Y-axis by angle β in a clockwise direction such that it gets align with Z.

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

Now line $Y = mx + C$ is aligned with z-axis perform actual rotation

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

Inverse rotation about ~~Y~~ and X-axis

$$R_{y(\beta^{-1})}^{-1} = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

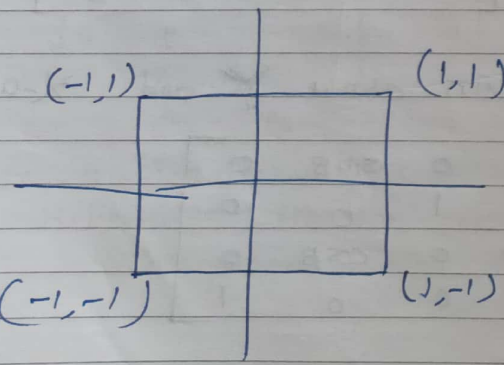
$$R_{x(\alpha)}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Inverse the translation

$$T^{-1} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & c \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M = T^{-1} \cdot R_{x(\alpha)}^{-1} \cdot R_{y(\beta)}^{-1} \cdot R_{z(\theta)} \cdot R_{y(\beta)} \cdot R_{x(\alpha)} \cdot T$$

Rotate a origin centered square with 2 unit length of each side in the clockwise direction with a rotation angle of 90°.



* Find a transformation matrix for rotation about a line parallel to the X-axis and passing through the reference point $P(x_r, y_r, z_r)$

i) Translation by $T(-x_r, -y_r, -z_r)$
ii) Rotation about X-axis by angle θ
iii) Inverse translation.

$$M = T^{-1} \cdot R_x(\theta) \cdot T$$

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

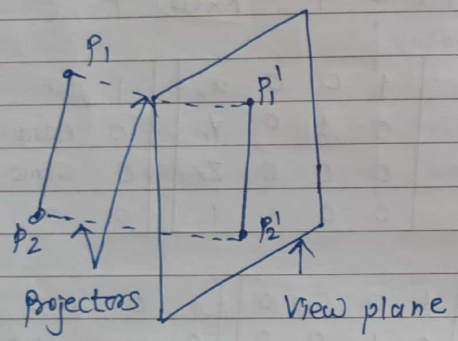
$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_r \\ 0 & \cos\theta & -\sin\theta & -y_r\cos\theta + z_r\sin\theta \\ 0 & \sin\theta & \cos\theta & -y_r\sin\theta - z_r\cos\theta \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

Projection -

- Converting a 3D object into 2D object.
- Mapping of the object in view plane.
- The plane on which the object is projected. Known as projection plane.

1. Parallel projection :



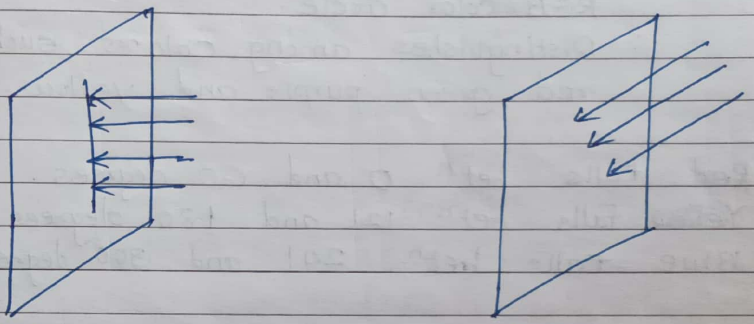
- Coordinate positions of the object are transformed to view plane by parallel rays.
- Discards the z coordinate and parallel lines from vertex are extended until they intersect the view plane.
- Preserves the true shape and size of object on view plane.

Orthographic projection -

- When all the projectors are parallel to each other and perpendicular to view plane.

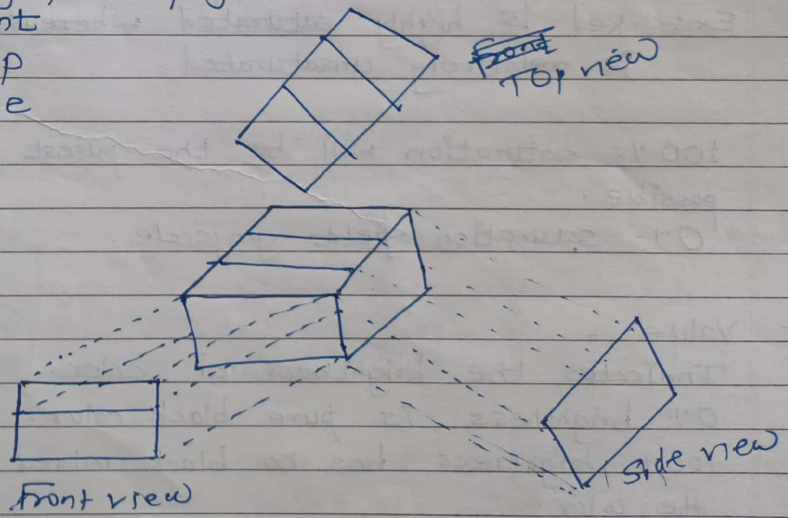
Oblique projection.

IF projectors are parallel to each other but are not perpendicular to the view plane.



Orthographic projection

1) Front
2) Top
3) Side



~~Oblique parallel projection~~ -

HSV color model -

- Remaps the RGB primary colors into primary dimensions that are easier for humans to understand.
- Hue : Specifies the angle of the color on RGB color circle.
  : Distinguishes among colors such as red, green, purple and yellow.

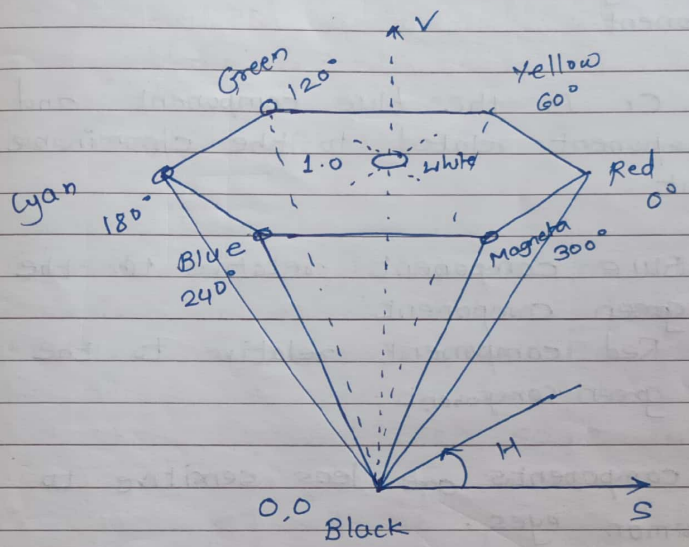
Red Falls bet^{n} 0 and 60 degrees.
Green Yellow Falls bet^{n} 121 and 180 degrees.
Blue Falls bet^{n} 241 and 300 degrees.

- Saturation -
Controls the amount of color used.
Ex: Red is highly saturated whereas pink is relatively unsaturated.

100 %. saturation will be the purest color possible.
0 %. saturation yields grayscale.

- Value -
Indicates the brightness of color.
0 %. brightness is pure black color.
100 %. brightness has no black mixed into the color.

- If the value dimension of color is set to 0 %. amount of hue and saturation does not matter as the color will be black.
- If the saturation of color is set to 0 %, the hue does not matter as there is no color used.



Yellow Falls 61 to 120

H	S	V	Color
0	1.0	1.0	Red
120	1.0	1.0	Green
240	1.0	1.0	Blue

Graphics designer uses the HSV Color model in selection of colors of paint.

YCbCr –

- Used in digital video processing.

i) Luminance Y :
- Brightness of the color.
- Light intensity of the color.
- Human eye is more sensitive to this component.

ii) Cb and Cr is the blue component and red component related to the chrominence component.

Cb ⇒ Blue component relative to the green component.

Cr ⇒ Red component relative to the green component.

These components are less sensitive to the human eyes.

Since the Y component is more sensitive ; it needs to be more correct.

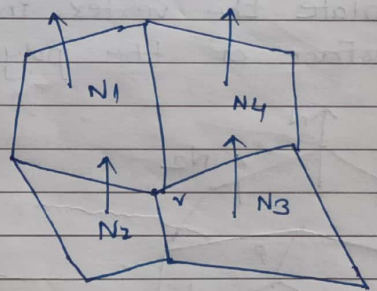
$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Shading –

- Implementation of illumination model at the pixel points of the graphical objects.
- It is a method to create or enhance the illusion of depth in an image by varying
- Used to compute the intensities and colors to display the surface.
- Two primary ingredients :
  i) Properties of the surface.
  ii) Properties of the illumination falling on it.

Constant Intensity shading –

- A fast and straight forward method for rendering an object with polygon surface.
- A single intensity is calculated for each polygon
- All points over the surface of the polygon are then displayed with same intensity.



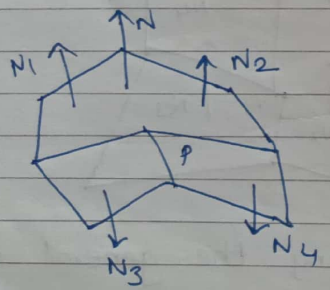
Useful in displaying the general appearances of the curved surfaces.

Gouraud Shading -

- Developed in the 1970s.
- Interpolation technique.
- Intensity levels are calculated at each vertex and interpolated across the surface.
- Intensity values for each polygon are matched with the values of adjacent polygon along the common edges.
- It eliminates the intensity discontinuities.

Process :

i) Determining the average unit normal vector at each vertex of the polygon.
ii) Apply an illumination model at each polygon vertex to obtain the light intensity at that position.
iii) Linear interpolate the vertex intensities over the surface of the polygon.



$$N = \frac{N_1 + N_2 + N_3 + N_4}{|N_1 + N_2 + N_3 + N_4|}$$

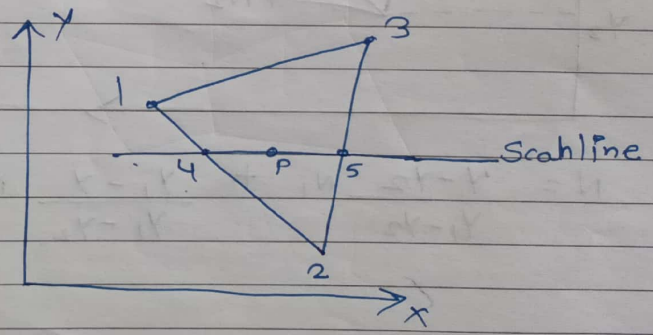
Generalised form -

$$N_V = \frac{\sum_{i=1}^{n} N_i}{\sum_{i=1}^{n} N_i}$$

$$I_4 = \frac{Y_4 - Y_2}{Y_1 - Y_2} I_1 + \frac{Y_1 - Y_4}{Y_1 - Y_2} I_2$$

$$I_5 = \frac{Y_5 - Y_2}{Y_3 - Y_2} I_3 + \frac{Y_3 - Y_5}{Y_3 - Y_2} I_2$$

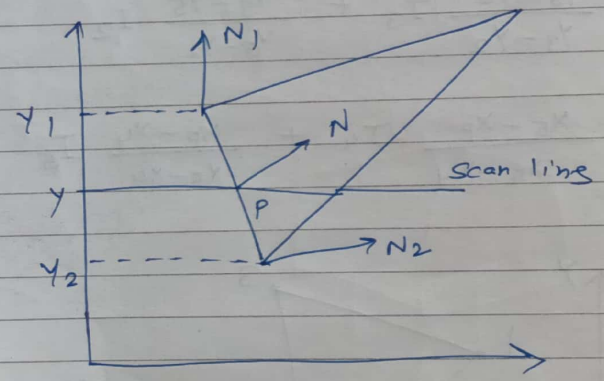
$$I_p = \frac{X_5 - X_p}{X_5 - X_4} I_4 + \frac{X_p - X_4}{X_5 - X_4} I_5$$



Phong Shading -

- A more accurate method.
- i) Interpolate the normal vector.
  ii) Apply the illumination model to each surface point.
- Displays more realistic highlights on surface

i) Determine the average unit normal vector at each vertex of the polygon.
ii) Linearly interpolate the vertex normal over the projected area.
iii) Apply an illumination model.



$$N = \frac{Y - Y_2}{Y_1 - Y_2} N_1 + \frac{Y_1 - Y}{Y_1 - Y_2} N_2$$

A(150,150), B(150,200), C(200,200)
D(200,150)
PQRS P(100,175), Q(170,250), R(250,165)
S(180,106)