1) (●) Rotation

- A 2D rotation is applied to an object by repositioning it along circular path in the xy plane centred at pivot point.

We can write the components:

$P'x = Px\cos\theta - Py\sin\theta$.

$P'y = Px\sin\theta + Py\cos\theta$.

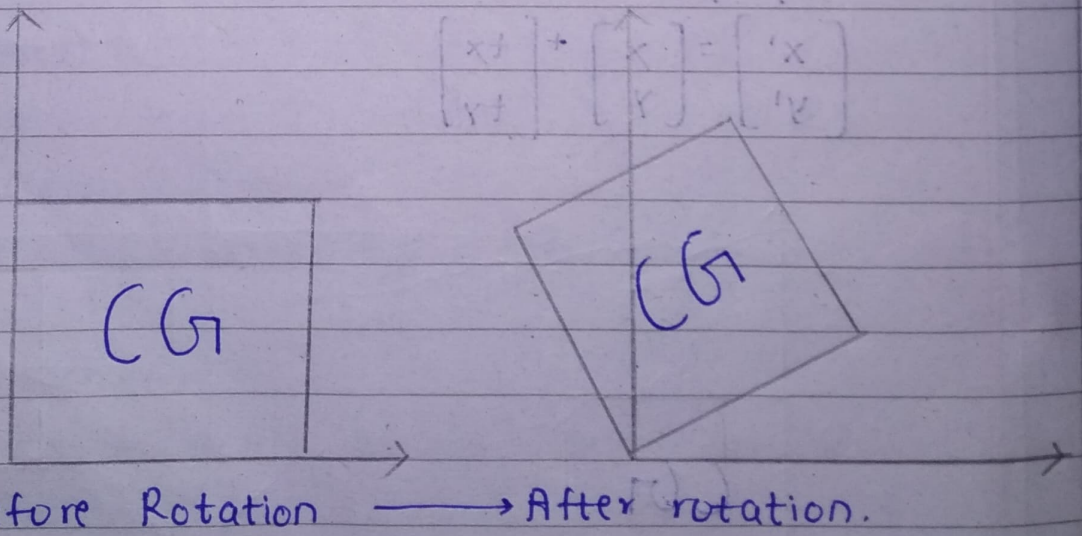
or in matrix form,

$P' = R \cdot P$

θ can be clockwise (-ve) or counterclockwise (+ve)

Rotation Matrix

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



2) (●) Scaling

Scaling changes the size of an object and involves two scale factors, S_x and S_y for the x-axis and y-axis respectively.

$P'x = S_x * P_x$

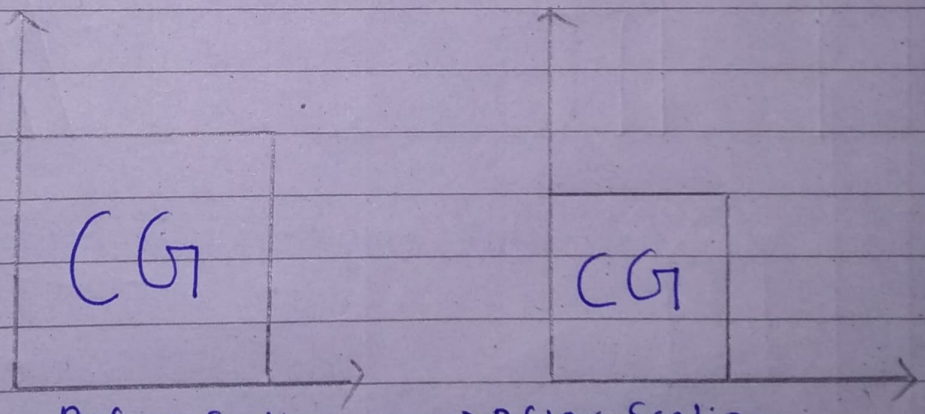
$P'y = S_y * P_y$

or in Matrix form, $P' = S * P$

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

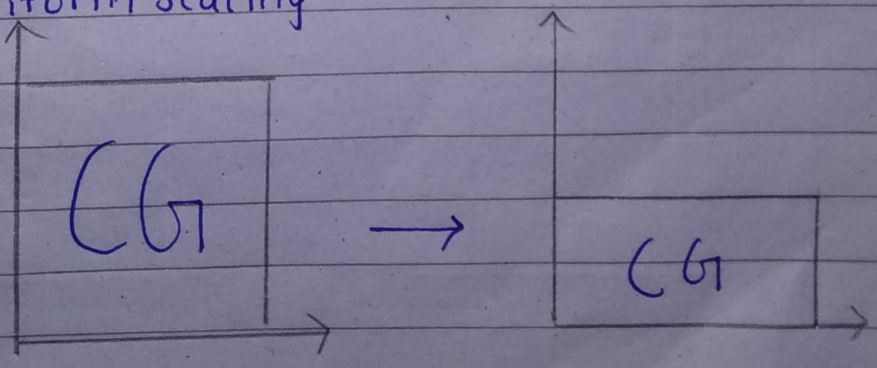
Uniform Scaling:

Scaling where Horizontal & Vertical factors are same is Uniform Scaling.



Before Scaling ⟶ After Scaling.

*Non Uniform Scaling:



3 (a) Three-Dimensional Scaling.

- Scaling means changing the size of an object.
- Scaling in 3D can be represented by a scaling vectors in scaling matrix.

Scaling vector V is defined as $[S_x\ S_y\ S_z]$.

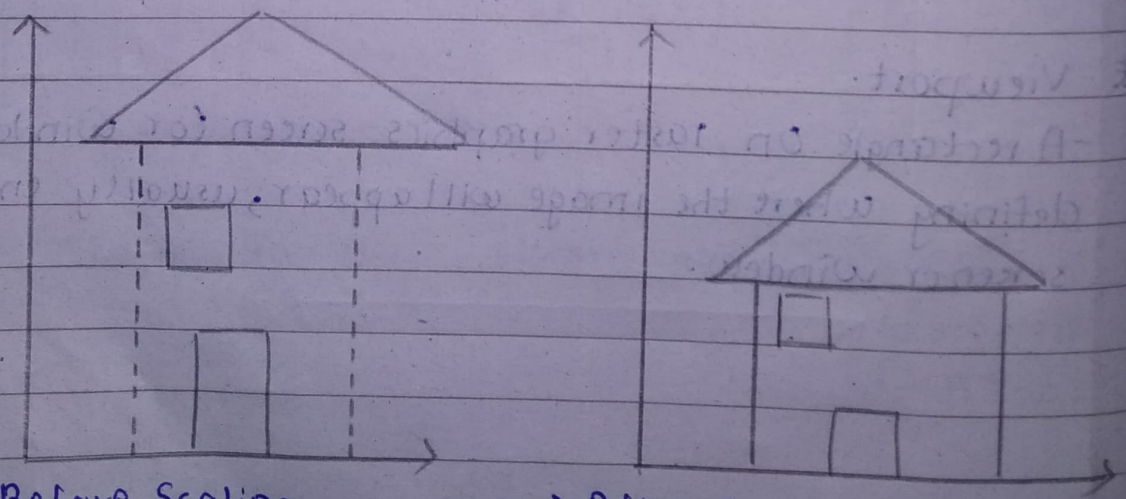
Scaling matrix(sv) in 3D-space with scaling vectors V is given by,

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

$$[P'] = [P][S_v]$$

$$[P'] = [x\ y\ z] \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix}.$$

where P' is transformed.



Before Scaling ⟶ After Scaling.

* 3D Scaling.

Uniform Scaling (scaling relative to co-ordinate origin).

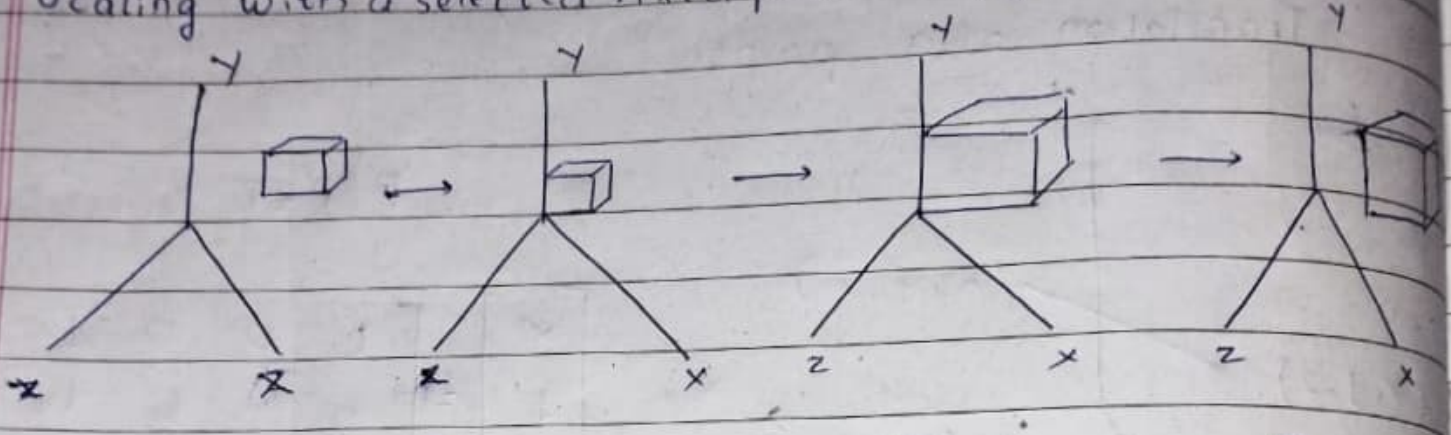
$x' = x * s_x$, $y' = y * s_y$, $z' = z * s_z$

$$P' = P * S$$

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

* Relative Scaling.

Scaling with a selected fixed point (x_f, y_f, z_f)



Original Translate Scaling Inverse Translate.

3067 - Sudesh

* Explain Cohen-Sutherland Line Clipping Algorithm.

A)=

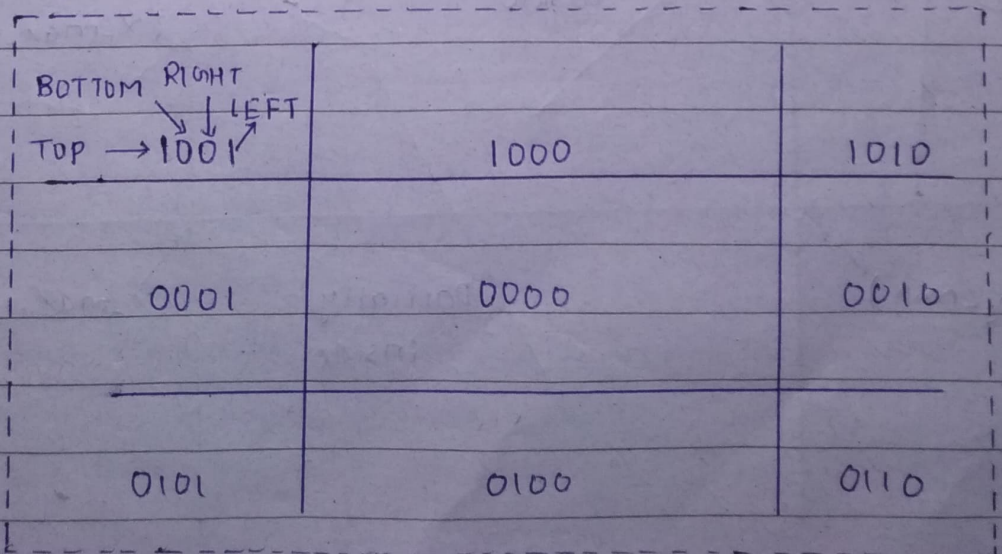
COHEN-SUTHERLAND divides a two-dimensional space into 9 regions and then efficiently determines the lines and portions of lines that are inside the given rectangular area.

The algorithm can be outlines as follows:-

Nine Regions are created, eight "outside regions and one "inside" region.

For a given line extreme point (x, y), we can quickly find it's region's four bit code. Four bit code can be computed by comparing x and y with four values. (x_min, x_max, y_min and y_max).

If x is less than x_min then bit number 1 is set.
If x is greater than x_max then bit number 2 is set.
If y is less than y_min then bit number 3 is set.
If y is greater than y_max then bit number 4 is set.

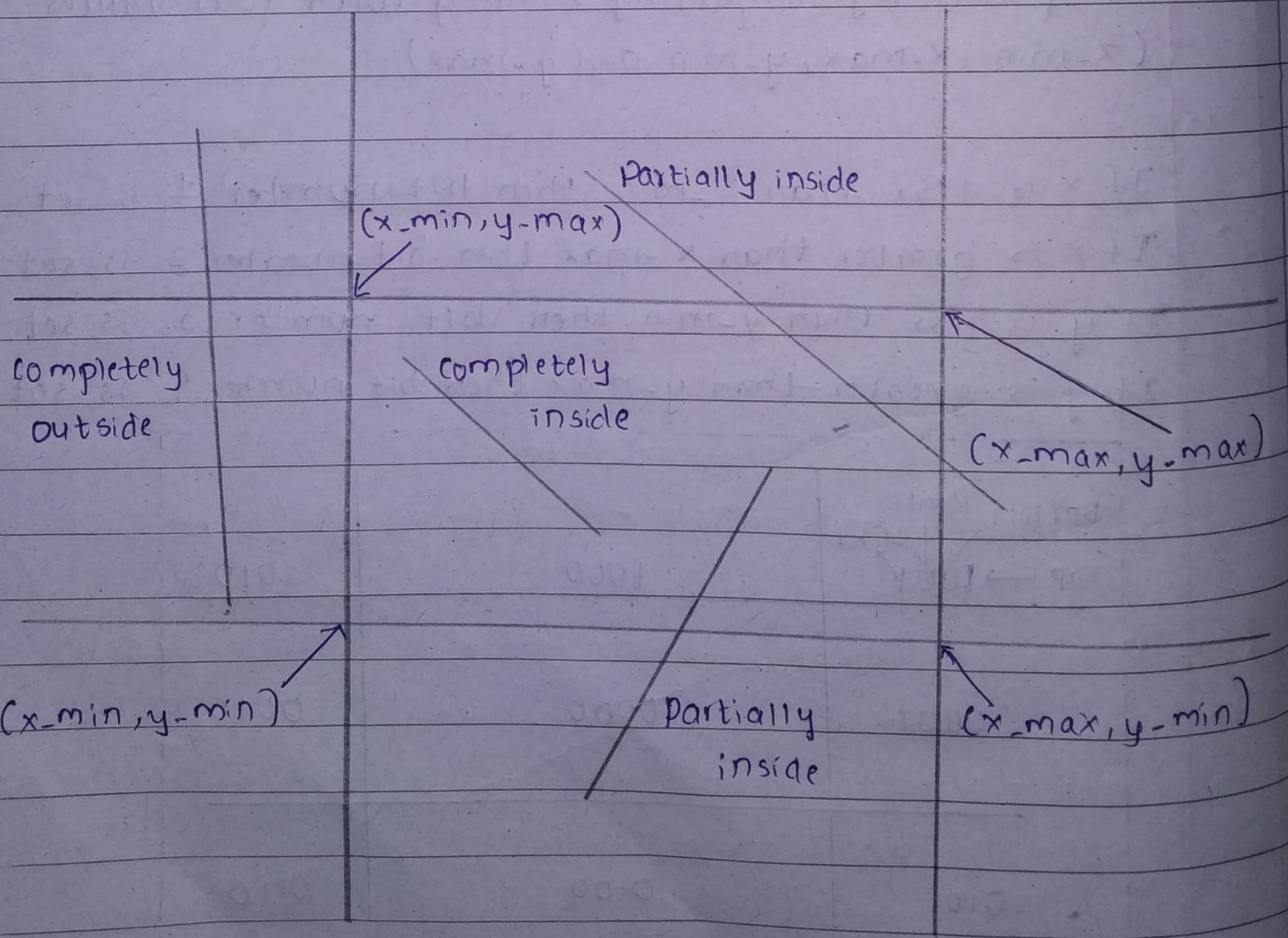


There are three possible cases for any given line:

1. Completely inside given rectangle: Bitwise OR of region of two and points of line is 0 (Both points are inside the rectangle)

2. Completely outside given rectangle : Both endpoints share atleast one outside region which implies that line doesn't cross visible region.

3. Partially inside window: Both endpoints are in different regions. In this case, the algorithm finds one of two points outside rectangular region.



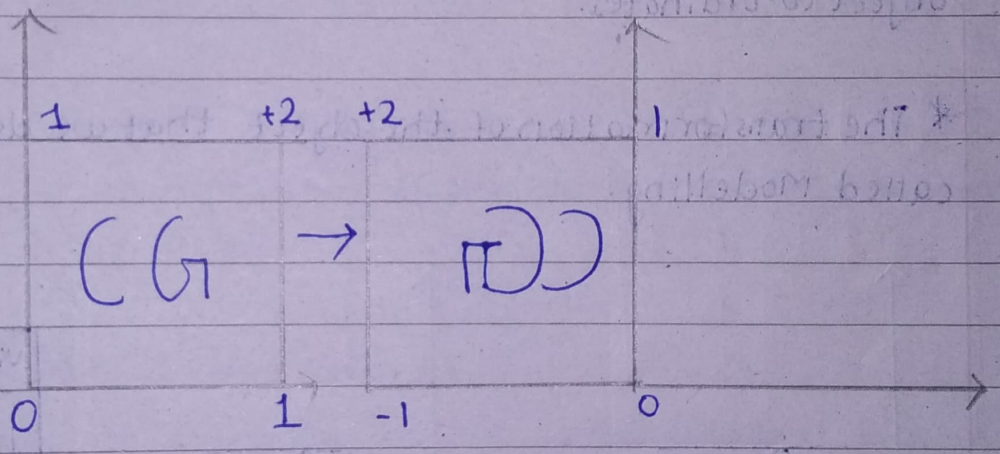
(5) Reflection.

- Reflection is the mirror image of original Object.
- Reflection on both x and y co-ordinates can be represented as,

$[R'] \rightarrow$ Matrix for Reflection.

$$\therefore [R']_{2\times 2} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

Let: Initial co-ordinates of O : (x, y)
Final co-ordinates of O : (x', y')



Before Reflection → After Reflection.