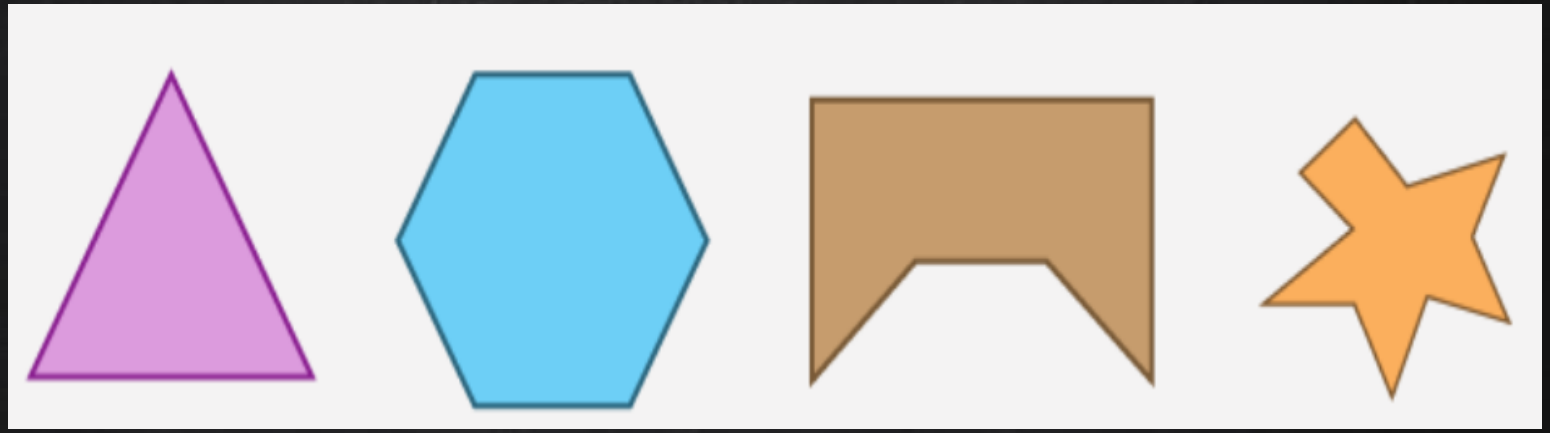
COMPUTER GRAPHICS
&
IMAGE PROCESSING
MODULE 2

Namitha Ramachandran

Module - 2(Filled Area Primitives and transformations) Filled Area Primitives- Scan line polygon filling, Boundary filling and flood filling. Two-dimensional transformations- Translation, Rotation, Scaling, Reflection and Shearing, Composite transformations, Matrix representations and homogeneous coordinates. Basic 3D transformations.

Namitha Ramachandran

In geometry, a polygon can be defined as a flat or plane, two-dimensional closed shape bounded with straight sides. It does not have curved sides.



Namitha Ramachandran

Filled Area Primitives-

Area filling algorithms often need to identify interior regions of objects.

Odd Even Rule (Odd Parity Rule)

1. Construct a line segment from point to be examined to point outside of a polygon.
2. Count the number of intersections of line segment with polygon boundaries.
3. If Odd number of intersection, then Point lies inside of Polygon.
4. Else, Point lies outside of polygon.

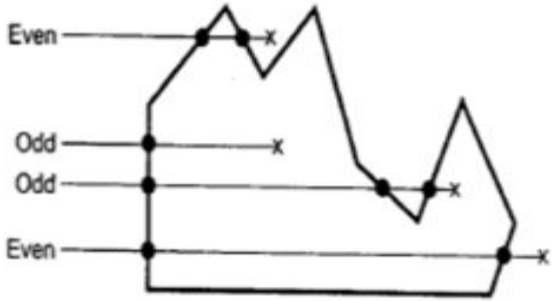


Fig. (a)

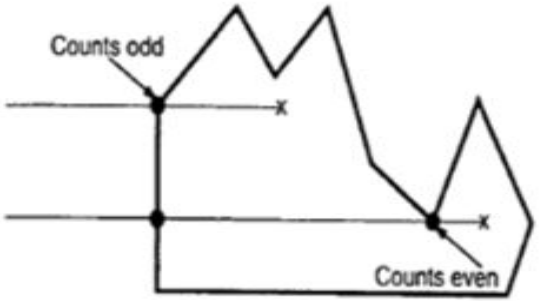
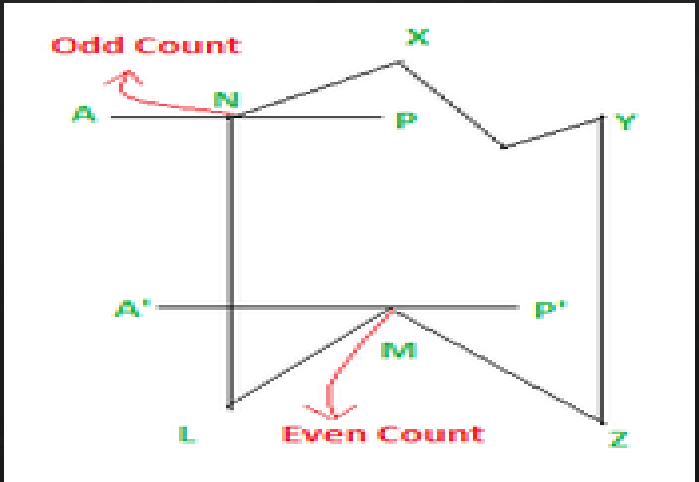


Fig. (b)

Scan line cut a vertex is a special case ;

Consider edges intersect at a vertex, if their other endpoints are on the same side of the scan line count number of intersections as 2(or even).

Consider edges intersect at a vertex,if their other endpoints are on the opposite sides of the scan line count number of intersections as 1(or odd).

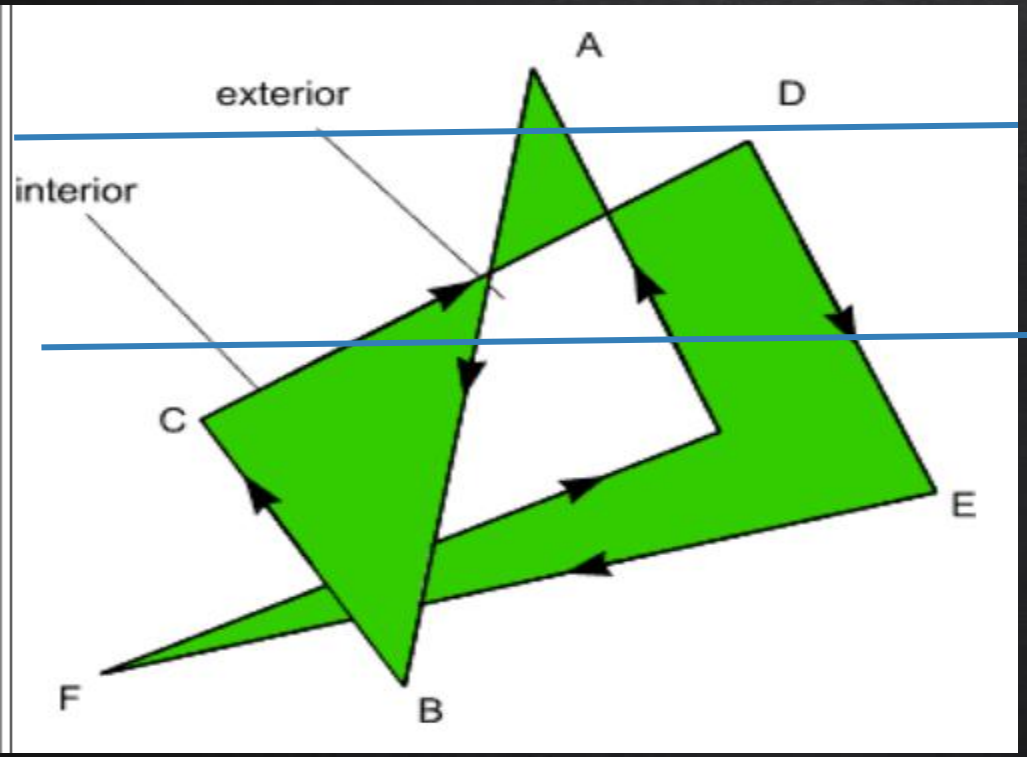


Namitha Ramachandran

Non Zero winding number Rule

1) Initial value of winding number=0.
2) Then draw a line from point (p-point to test) to outside the polygon which does not pass through any vertex.
3) Add 1 to the winding number every time we intersect a polygon edge that crosses the line from right to left and subtract 1 every time we intersect an edge that crosses from left to right.
4) The interior points are that have a non zero value for the winding number .
5) Exterior points are those whose value of the winding number is zero .

Namitha Ramachandran



LEFT TO RIGHT:-1 (CLOCKWISE)

RIGHT TO LEFT: +1 (COUNTER-CLOCKWISE)

INITIAL VALUE = 0

Namitha Ramachandran

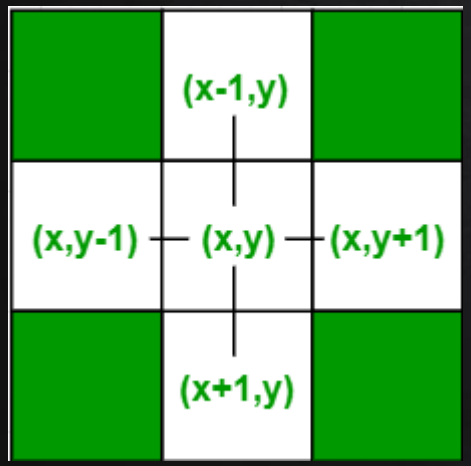
Boundary Fill Algorithm starts at a pixel inside the polygon to be filled and paints the interior proceeding outwards towards the boundary.
This algorithm works **only if** the color with which the region has to be filled and the color of the boundary of the region are different.
If the boundary is of one single color, this approach proceeds outwards pixel by pixel until it hits the boundary of the region.

***Boundary Fill Algorithm is recursive in nature*.** It takes an interior point(x, y), a fill color, and a boundary color as the input. The algorithm starts by checking the color of (x, y). If it's color is not equal to the fill color and the boundary color, then it is painted with the fill color and the function is called for all the neighbors of (x, y).
If a point is found to be of fill color or of boundary color, the function does not call its neighbors and returns.
This process continues until all points up to the boundary color for the region have been tested.

Namitha Ramachandran

The boundary fill algorithm can be implemented by **4-connected pixels or 8-connected** pixels.

4-connected pixels : After painting a pixel, the function is called for four neighboring points.
These are the pixel positions that are right, left, above, and below the current pixel.
Areas filled by this method are called 4-connected.



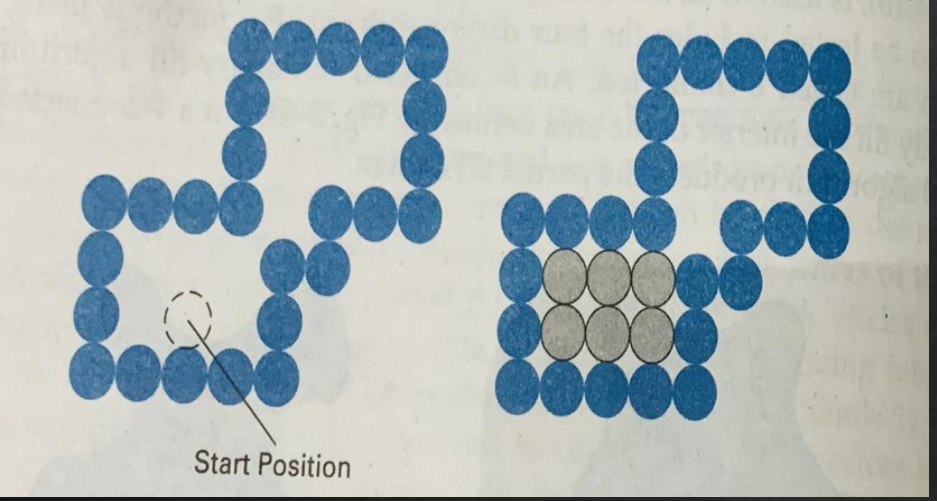
4-connected pixels

Namitha Ramachandran

Boundary Fill Algorithm(4 connected approach)

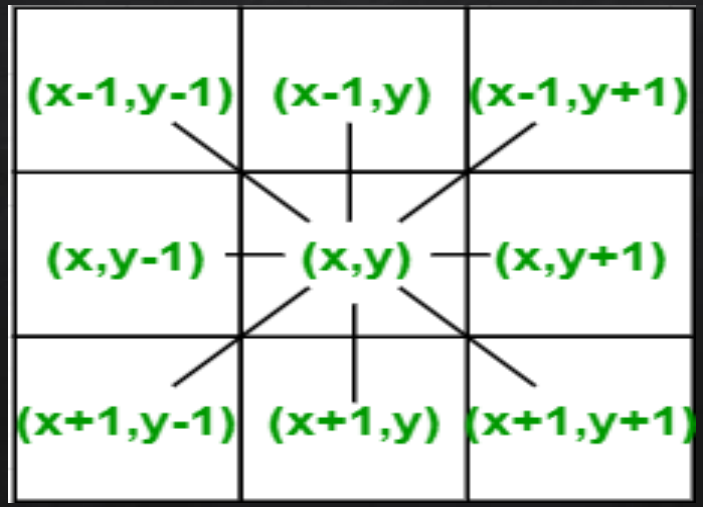
```
void boundaryFill4(int x, int y, int fill_color,int boundary_color)
{
    if(getpixel(x, y) != boundary_color &&
       getpixel(x, y) != fill_color)
    {
        putpixel(x, y, fill_color);
        boundaryFill4(x + 1, y, fill_color, boundary_color);
        boundaryFill4(x, y + 1, fill_color, boundary_color);
        boundaryFill4(x - 1, y, fill_color, boundary_color);
        boundaryFill4(x, y - 1, fill_color, boundary_color);
    }
}
```

Namitha Ramachandran



4 connected approach fails in such a situation
We have 8 connected approach as a solution .

8-connected pixels



Namitha Ramachandran

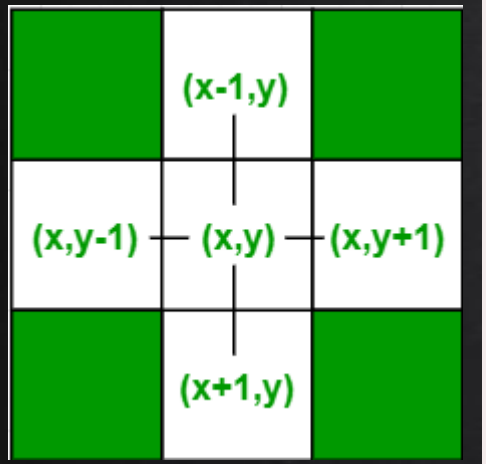
```
void boundaryFill8(int x, int y, int fill_color,int boundary_color)
{
    if(getpixel(x, y) != boundary_color &&
       getpixel(x, y) != fill_color)
    {
        putpixel(x, y, fill_color);
        boundaryFill8(x + 1, y, fill_color, boundary_color);
        boundaryFill8(x, y + 1, fill_color, boundary_color);
        boundaryFill8(x - 1, y, fill_color, boundary_color);
        boundaryFill8(x, y - 1, fill_color, boundary_color);
        boundaryFill8(x - 1, y - 1, fill_color, boundary_color);
        boundaryFill8(x - 1, y + 1, fill_color, boundary_color);
        boundaryFill8(x + 1, y - 1, fill_color, boundary_color);
        boundaryFill8(x + 1, y + 1, fill_color, boundary_color);
    }
}
```

Flood Fill Algorithm

In this method, a point or seed which is inside the region is selected.
This point is called a seed point.
Then four connected approaches or eight connected approaches is used to fill with a specified color .
When boundary is of many colors and interior is to be filled with one color we use this algorithm.

In fill algorithm, we start from a specified interior point (x, y) and reassign all pixel values are currently set to a given interior color with the desired color.
Using either a 4-connected or 8-connected approach, we then step through pixel positions until all interior points have been repainted.

Namitha Ramachandran



4-connected pixels

```
Procedure floodfill (x, y,fill_ color, old_color: integer)
    If (getpixel (x, y)=old_color)
   {
    setpixel (x, y, fill_color);
    fill (x+1, y, fill_color, old_color);
     fill (x-1, y, fill_color, old_color);
    fill (x, y+1, fill_color, old_color);
    fill (x, y-1, fill_color, old_color);
     }
}
```

Disadvantages

Very slow algorithm
May be fail for large polygons
Initial pixel required more knowledge about surrounding pixels.

Namitha Ramachandran

In Flood fill, all the connected pixels of a selected color get replaced by fill color.

In Boundary fill, the program stops when a given color boundary is found.

Disadvantages of Boundary-Fill over Flood-Fill:

In boundary-fill algorithms, each pixel must be compared against both the new colour and the boundary colour. In flood-fill algorithms, each pixel need only be compared against the new colour. Therefore flood-fill algorithms are slightly faster.

Advantages of Boundary-Fill over Flood-Fill:

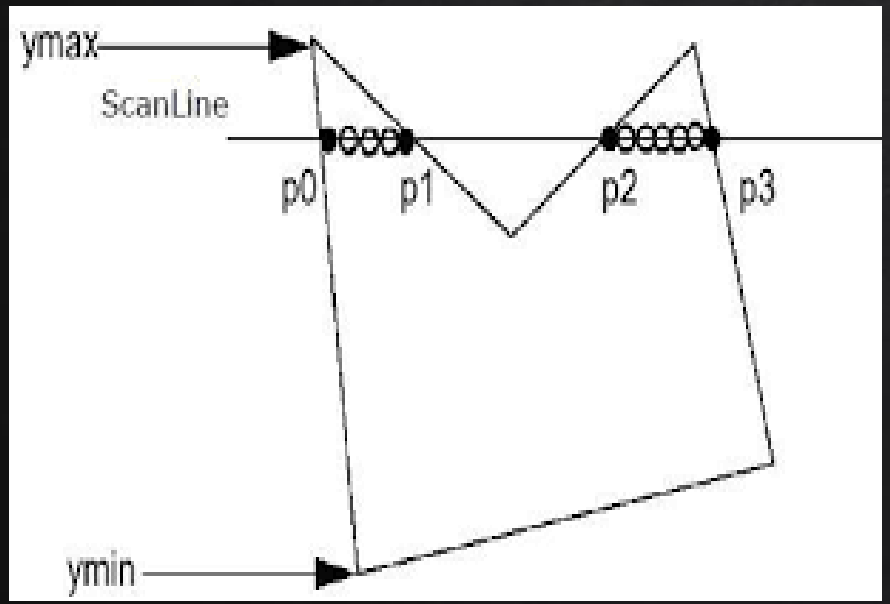
All pixels in the region must be made the same colour when the region is being created.

Namitha Ramachandran

Scanline Polygon filling Algorithm

Scanline filling is basically the filling up of polygons using horizontal lines or scanlines.

This algorithm works by intersecting scanlines with polygon edges and fills the polygon between pairs of intersections.



Special cases of polygon vertices:

If both lines intersecting at the vertex are on the same side of the scanline, consider it as two points.

If lines intersecting at the vertex are at opposite sides of the scanline, consider it as only one point.

Namitha Ramachandran

To effectively perform a polygon fill, we can store the polygon boundary in a **sorted edge table** that contains all the information necessary to process the scan lines efficiently, Each entry in the table for a particular scan line contains the maximum y value for that edge, the x-intercept value (at the lower vertex) for the edge, and the inverse slope of the edge.
For each scan line , edges are in sorted order from left to right.

Next process the scan lines from the bottom of the polygon to its top, and produce an active edge list for each scan line crossing the polygon boundaries.
The active edge list for a scan line contains all edges crossed by that scan line, with iterative coherence calculations used to obtain the edge intersections.

Step 1 − Find out the Ymin and Ymax from the given polygon.

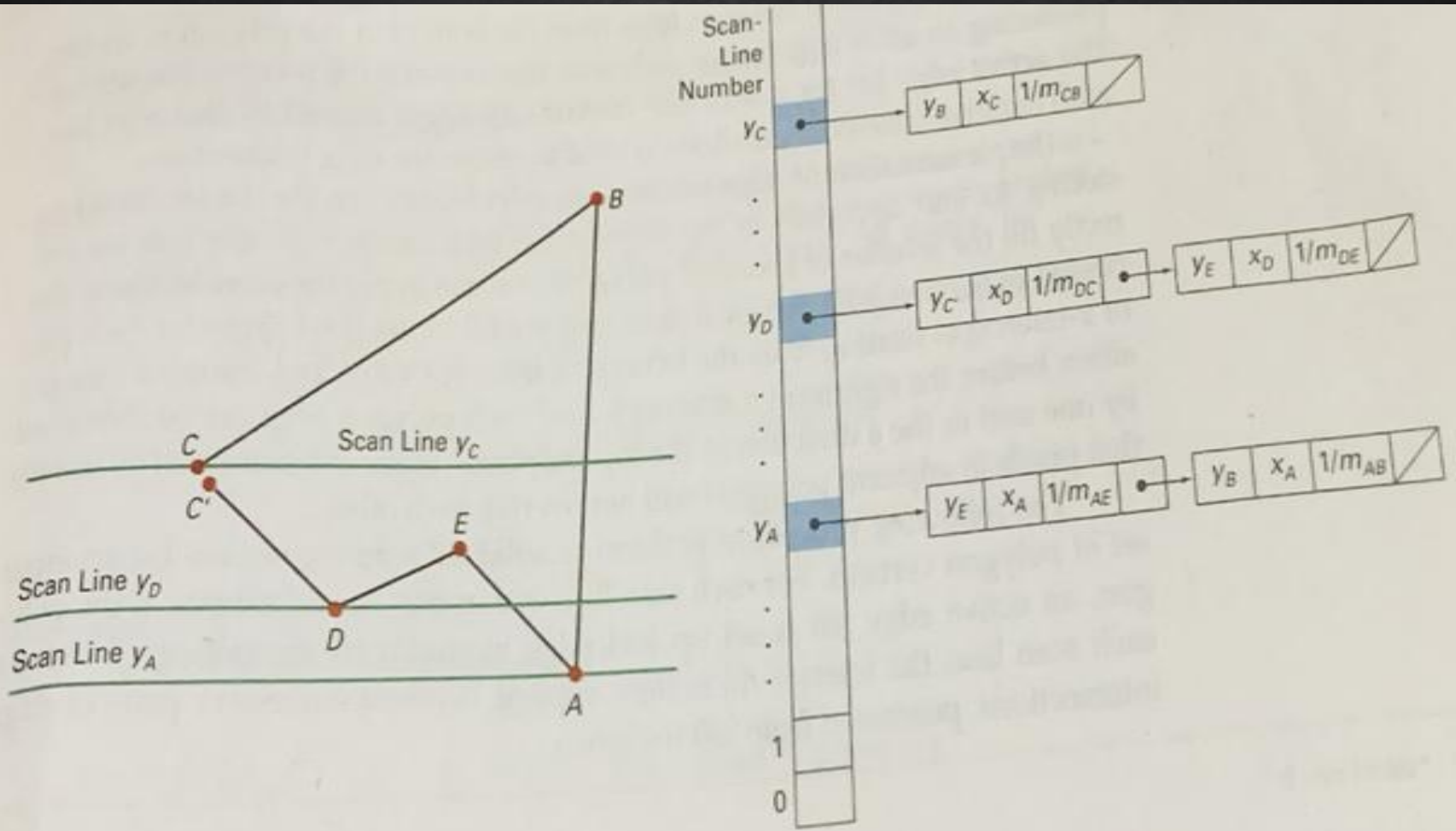
Step 2 − ScanLine intersects with each edge of the polygon from Ymin to Ymax. Name each intersection point of the polygon.

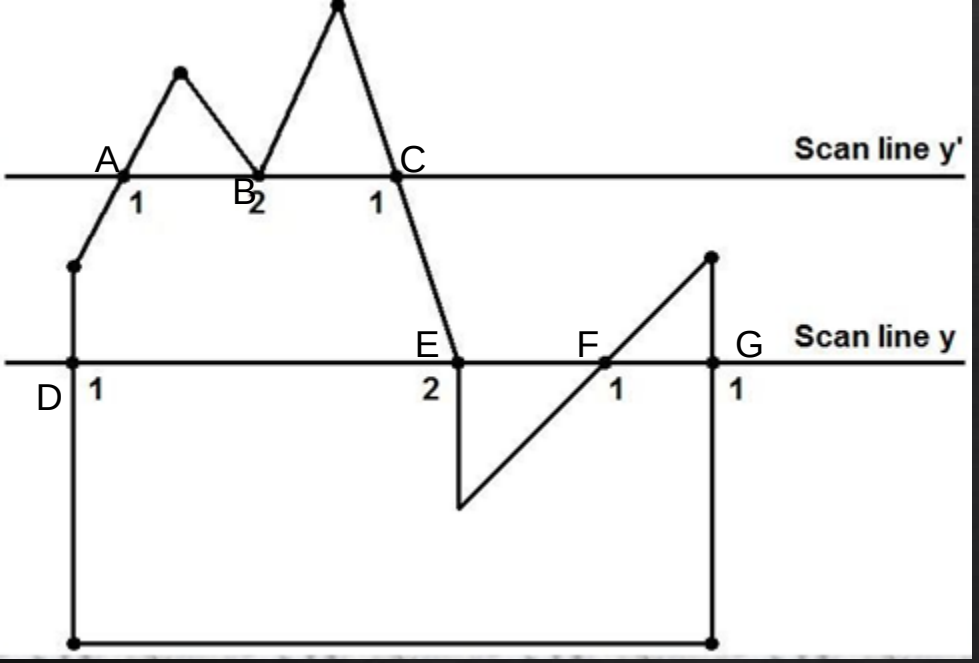
Example : consider previous figure ,intersections are named as p0,p1,p2,p3

Step 3 − Sort the intersection point in the increasing order of X coordinate i.e. (p0,p1) ,(p1,p2) ,(p2,p3).

Step 4 − Fill all those pair of coordinates that are inside polygons and ignore the alternate pairs.

{ ignore (p1,p2) }


Scan-
Line
Number
y_C
y_D
y_A
1
0
y_B x_C $1/m_{CB}$
$y_{C'}$ x_D $1/m_{DC}$
y_E x_D $1/m_{DE}$
y_E x_A $1/m_{AE}$
y_B x_A $1/m_{AB}$
B
C
C'
E
D
A
Scan Line y_C
Scan Line y_D
Scan Line y_A



Scan line Y'

Intersection points :

A,B,C ,Where B is a vertex and other ends of the 2 lines causes vertex B are on the same side of scan line, so take B two times .

Pairs (A,B) ,(B,C)

Scan line Y

Intersection points :

D,E,F,G , Where E is a vertex and other ends of the 2 lines causes vertex E are on opposite side of scan line, so take E once .

Pairs (D,E) ,(F,G)

Namitha Ramachandran

Coherence Properties

Coherence is simply that the properties of one part of a scene are related in some way to other parts of the scene so that the relationship can be used to reduce processing.

Slop of this polygon boundary can be expressed in terms of the scan line intersection coordinates :

$$m = \frac{(yk+1-yk)}{(xk+1-xk)}$$

Where , $yk+1-yk=1$

$m = 1/(xk+1-xk)$

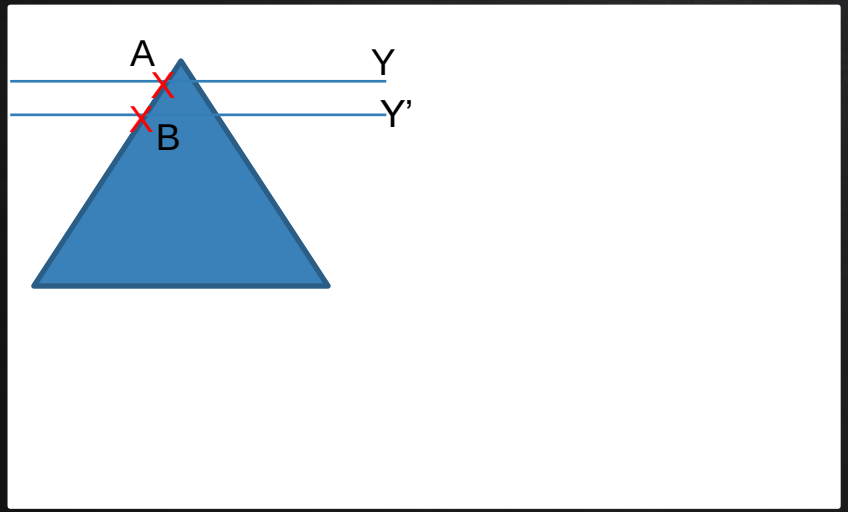
$(xk+1-xk)=1/m$

$Xk+1= 1/m + xk$

Namitha Ramachandran

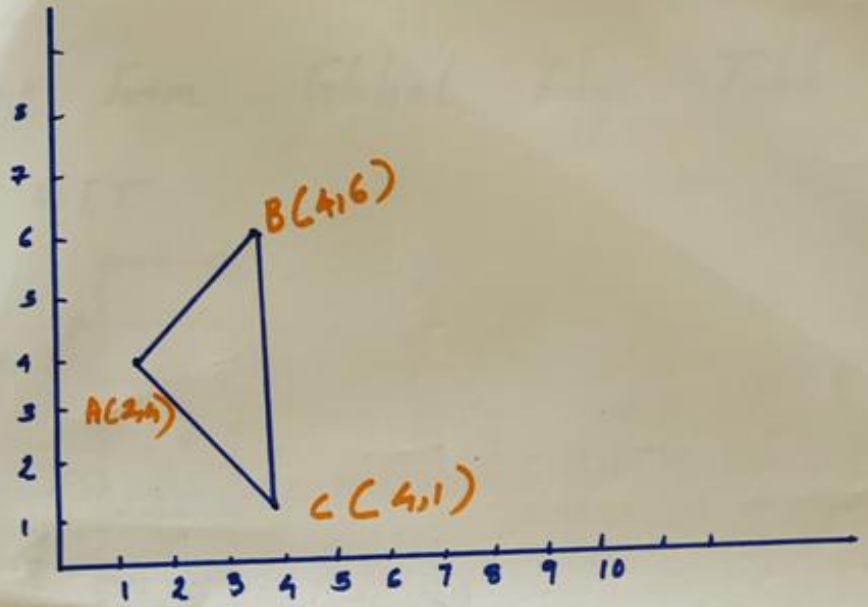
Adjacent scanlines are separated by a distance of 1, which means y difference is unity.
y-y'=1

Using the properties of Coherence, after completing one scan line processing, we don't want to start from the beginning of the next scan line, instead, we can directly calculate the edge intersection point on the next scan line and proceed from that point .



In the given diagram, after processing scan line "y" start with "y' ", but here we don't want to start processing from the beginning of "y' ", instead we can directly find coordinate of intersection "B" using coherence property and start from that point of scan line "y' " ,this will reduce complexity of algorithm .

Namitha Ramachandran



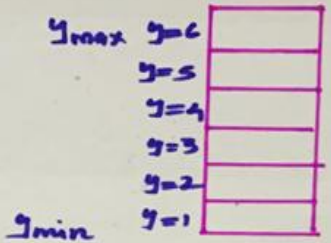
Step 1: Sort the edges from y_{min} to y_{max}

Given A(2,4), B(4,6), C(4,1)

Sorted List

C(4,1), A(2,4), B(4,6)

step 2 : Form Global Edge Table



Start with ymin.

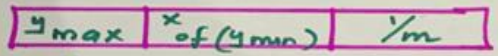
y=1 we have vertex C; and C is connected with other vertices through edges

C connected to A - edge CA

C connected to B - edge CB

Add 'C' and 'A, B' in to GET with the help of linked list.

A and B can be represented using node structure



Consider edge CA

C (4,1) A (2,4)

ymax = 4

x of ymin = 4

slope $m = \frac{4-1}{2-4} = -\frac{3}{2}$

$1/m = -2/3$

Consider edge CB

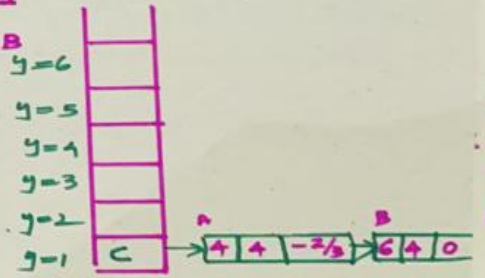
C (4,1) B (4,6)

ymax = 6

x of ymin = 4

slope $m = \frac{6-1}{4-4} = \frac{5}{0}$

$1/m = 0/5 = 0$

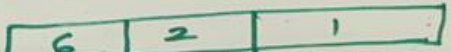


at y=2 No vertex with y=2 so put NULL in GET.

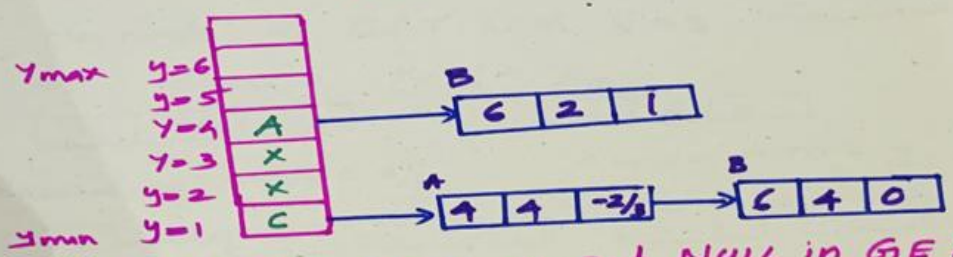
at y=3 No vertex with y=3 so put NULL in GET

at y=4, we have vertex with y=4 i.e A(2,4)
A connected to 'B' and 'C' but C's y value < A's y value means 'C' is already processed; hence consider AB

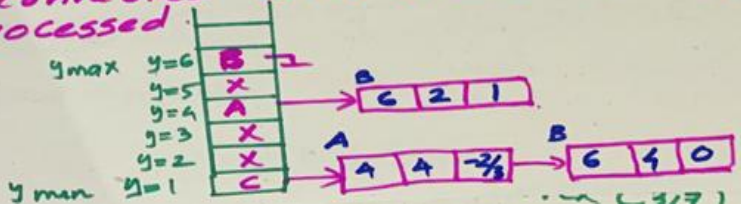
A (2,4) B (4,6)



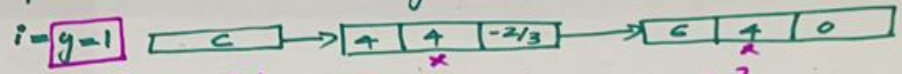
slope $m = \frac{6-4}{4-2} = \frac{2}{2} = 1$ $1/m = 1/1 = 1$



y=5 No vertex with y=5 so put NULL in GET
y=6 we have vertex with y=6, i.e vertex 'C'
But C is connected with 'A' and 'B' they are already processed

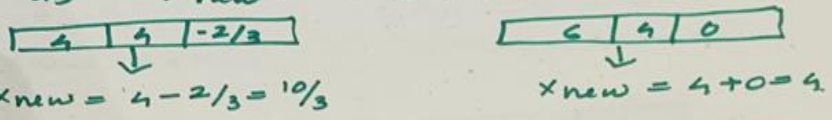


Step 3: Active Edge Table



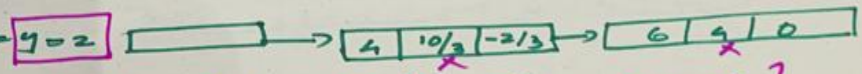
Active Edge { (4,1) , (4,1) }

i=y=2 No node in GET; NULL here; increment x as $x_{new} = x_{old} + 1/m$



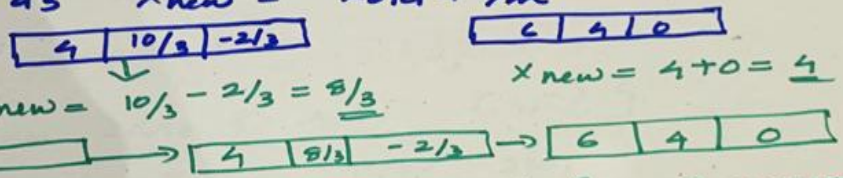
$x_{new} = 4 - 2/3 = 10/3$

$x_{new} = 4 + 0 = 4$



Active Edge { (10/3, 2) , (4,2) }

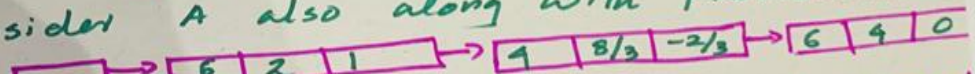
i=y=3 No node in GET with y=3 increment x as $x_{new} = x_{old} + 1/m$



$x_{new} = 10/3 - 2/3 = 8/3$

$x_{new} = 4 + 0 = 4$

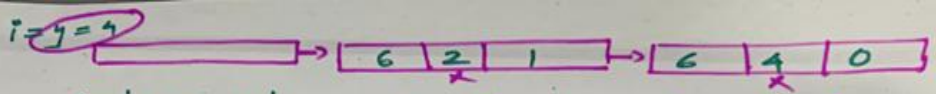
i=y=4 Ain GET, so before x increment Consider A also along with previous nodes



here First eliminate node whose y value = current i=y value; Now i=y=4

So remove

Namitha Ramachandran

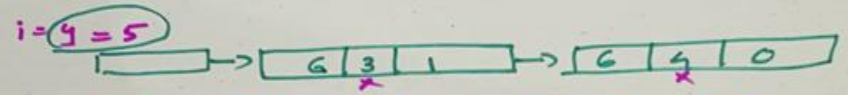


Active edge { (2,4), (4,4) }

i=y=5, No entry in GET with at y=5

increment $x_{new} = x_{old} + 1/m$

$x_{new} = 2+1 = 3$ $x_{new} = 4+0 = 4$



Active Edge { (3,5) (4,5) }

i=y=6 No entry in GET, increment x

$x_{new} = 3+1 = 4$ $x_{new} = 4+0 = 0$

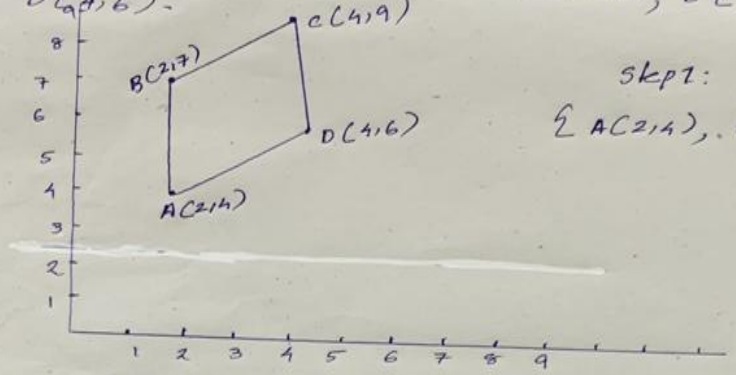
Active edge { (4,6), (4,6) }

y reached y_{max} now stop.

Active edge table

1st pair : { (4,1), (4,1) }
2nd pair : { (10/3, 2), (4,2) }
3rd pair : { (5/3, 3), (4,3) }
4th pair : { (2,4), (4,4) }
5th pair : { (3,5), (4,5) }
6th Pair : { (4,6), (4,6) }

Determine the Content of Active Edge Table to fill the polygon with the vertices A(2,4), B(2,7), C(4,9) and D(4,6).

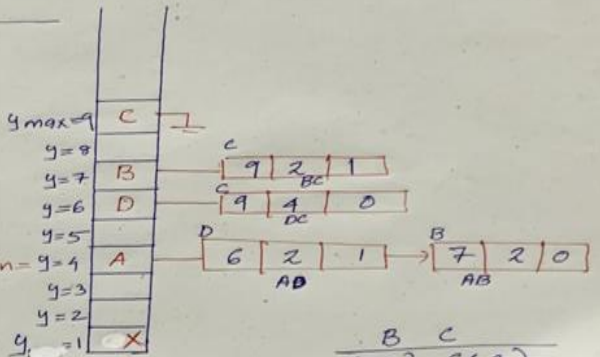


Step 1: Sorted vertices
{ A(2,4), D(4,6), B(2,7), C(4,9) }

Step 2: GET

AD
(2,4) (4,6)
$m = \frac{6-4}{4-2} = \frac{2}{2} = 1$
$y_{max} = 6$
x of $y_{min} = 2$

AB
(2,4) (2,7)
$m = \frac{7-4}{2-2} = \frac{3}{0}$
$1/m = \frac{0}{3} = 0$
$y_{max} = 7$
x of $y_{min} = 2$

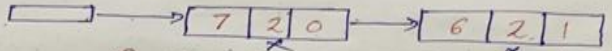


D C
(4,6) (4,9)
$m = \frac{9-6}{4-4} = \frac{3}{0}$
$1/m = 0/3 = 0$
$y_{max} = 9$
x of $y_{min} = 4$

B C
(2,7) (4,9)
$m = \frac{9-7}{4-2} = \frac{2}{2} = 1$
$1/m = 1$
$y_{max} = 9$
x of $y_{min} = 2$

Step 3: Active Edge Table

$y = i = 4$

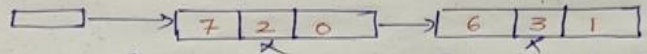


{ (2,4), (2,4) } First Pair

$y = i = 5$ No node in $y = 5$ at GET then increment x

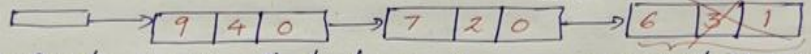
$x_{new} = x_{old} + 1/m$ $x = 2+0 = 2$ $x = 2+1 = 3$

$y = i = 5$



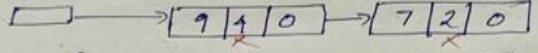
{ (2,5), (3,5) } Second Pair

$y = i = 6$ One node in GET (D) Consider that node also with previous set



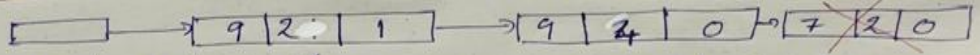
eliminate node that has y_{max} = Current $y = 6$

$y = i = 6$

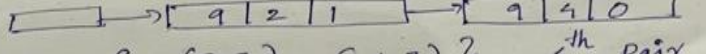


{ (4,6), (2,6) } Third Pair

$y = i = 7$ one node in GET, consider this also

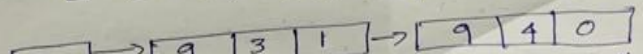


eliminat node whose y_{max} = Current y is 7



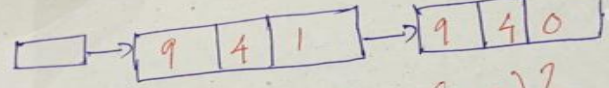
$y = 7$ { (2,7), (4,7) } 4th pair

$i = y = 8$



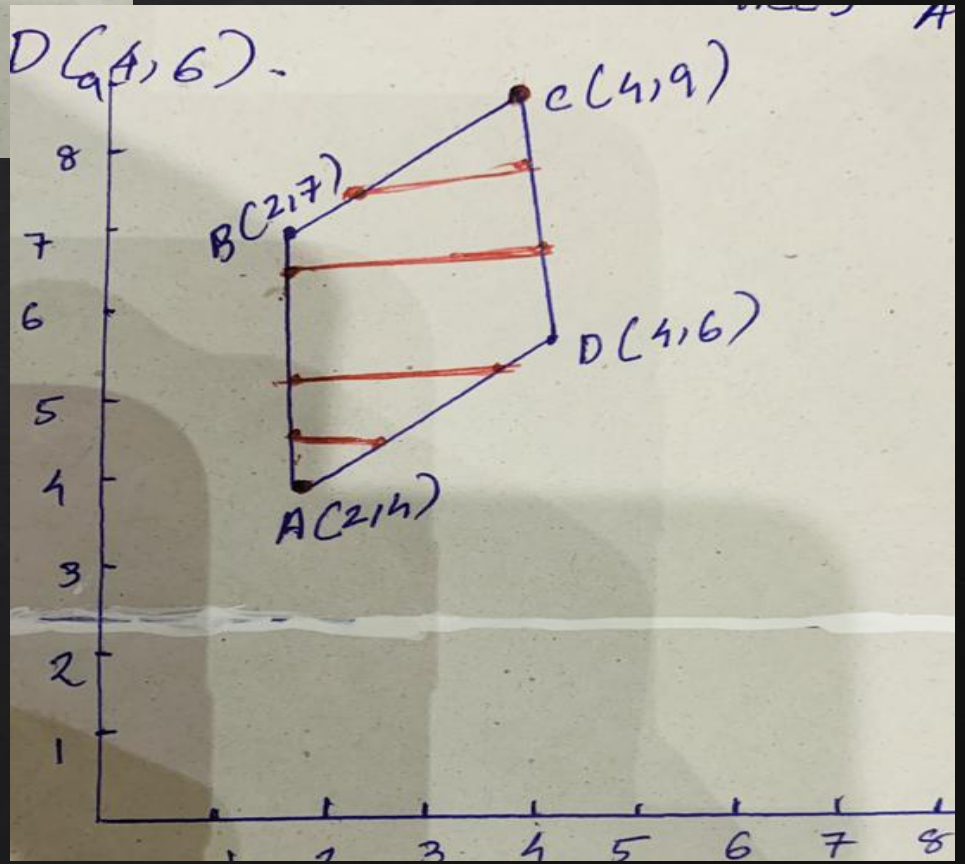
{ (3,8), (4,8) } 5th Pair

i=y=9



{ (4,9) , (4,9) } 6th pair

Reached ymax - stop



2D Transformation

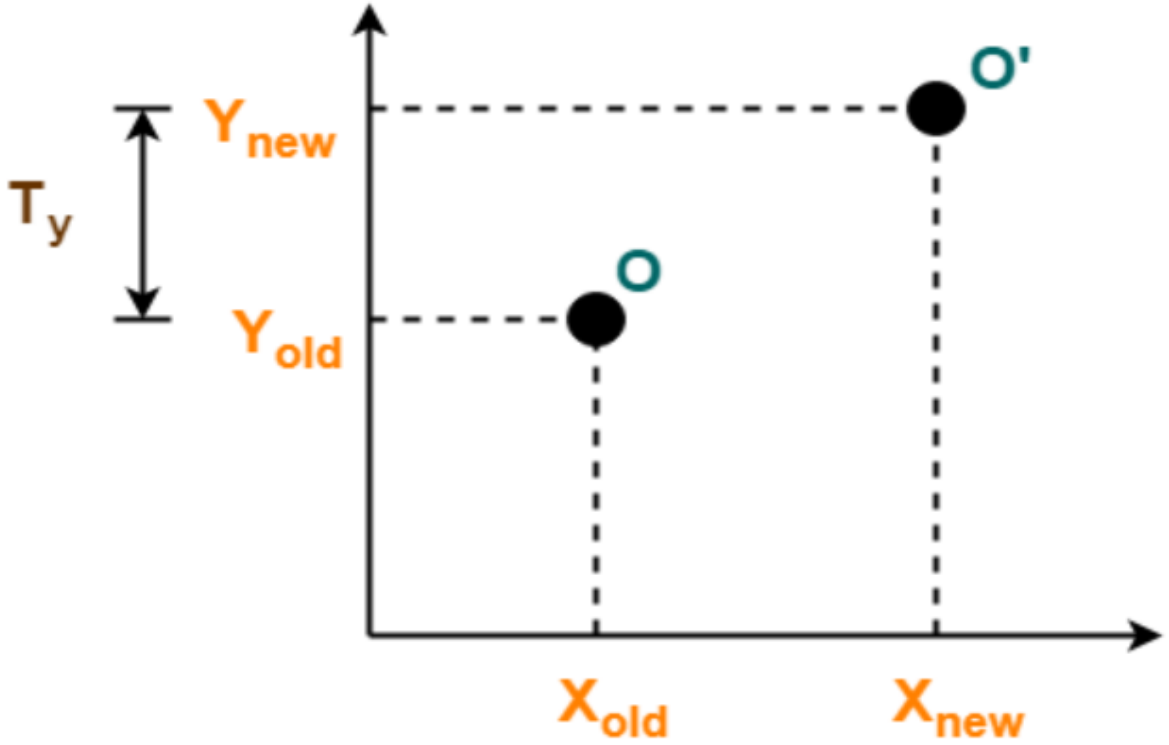
Transformation means changing some graphics into something else by applying rules.

We can have various types of transformations such as **translation,rotation scaling , reflection, shearing,** etc.

When a transformation takes place on a 2D plane, it is called 2D transformation.

Namitha Ramachandran

2D Translation is the process of moving an object from one position to another in a two-dimensional plane.



(X_{new}, Y_{new})

Namitha Ramachandran

This translation is achieved by adding the translation coordinates to the old coordinates of the object as-

$$X_{new} = X_{old} + T_x$$
$$Y_{new} = Y_{old} + T_y$$

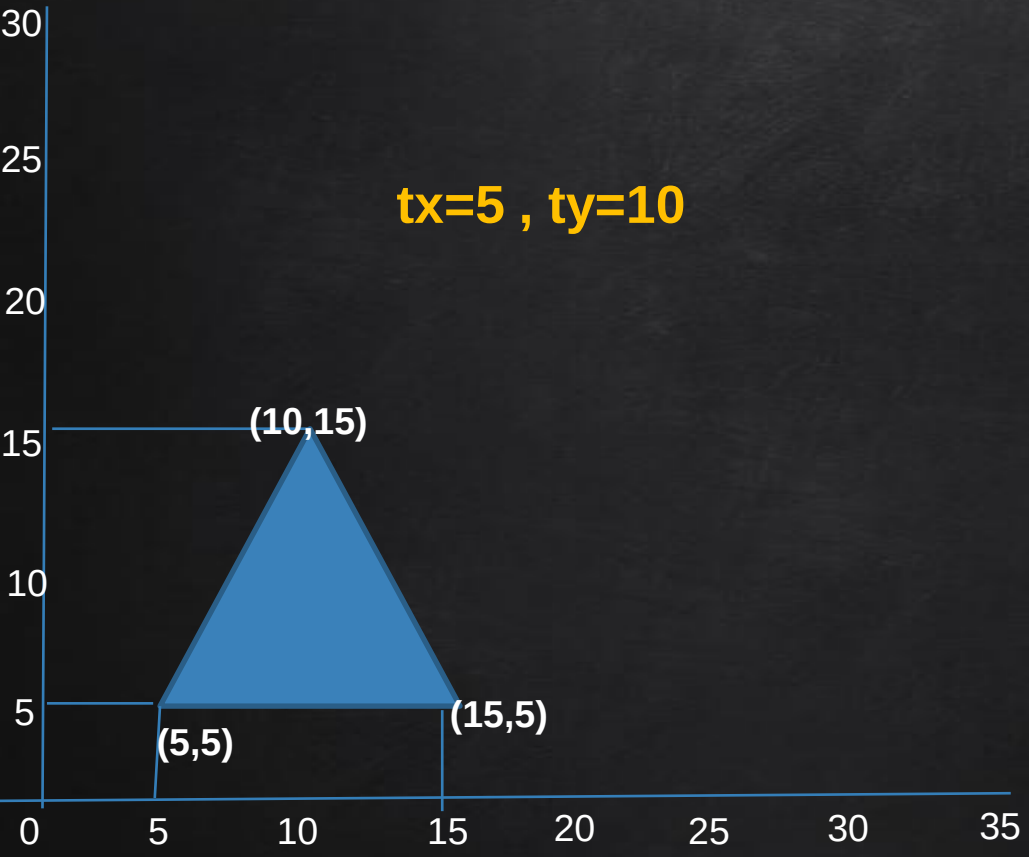
In Matrix form, the above translation equations may be represented as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$

Translation Matrix

$$P' = P + T$$

Namitha Ramachandran



tx=5 , ty=10

(10,15)□ (15,25)

(5,5)□ (10,15)

(15,5)□ (20,15)

Namitha Ramachandran

2D SCALING

In computer graphics, scaling is a process of modifying or altering the size of objects.

Scaling may be used to increase or reduce the size of the object.
Scaling subjects the coordinate points of the original object to change.
Scaling factor determines whether the object size is to be increased or reduced.

If scaling factor > 1, then the object size is increased.
If scaling factor < 1, then the object size is reduced.
Specifying a value of 1 for both s_x and s_y leaves the size of objects unchanged .

Namitha Ramachandran

When Sx and Sy are assigned the same value , **uniform scaling is produced** that maintains relative object proportions.

When Sx and Sy are assigned the value ½,length and distance from origin are reduced by half.

Namitha Ramachandran

- Initial coordinates of the object O = (X_{old}, Y_{old})
- Scaling factor for X-axis = S_x
- Scaling factor for Y-axis = S_y
- New coordinates of the object O after scaling = (X_{new}, Y_{new})

This scaling is achieved by using the following scaling equations-

$$X_{new} = X_{old} \times S_x \qquad Y_{new} = Y_{old} \times S_y$$

Namitha Ramachandran

In Matrix form, the above scaling equations may be represented as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} x \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

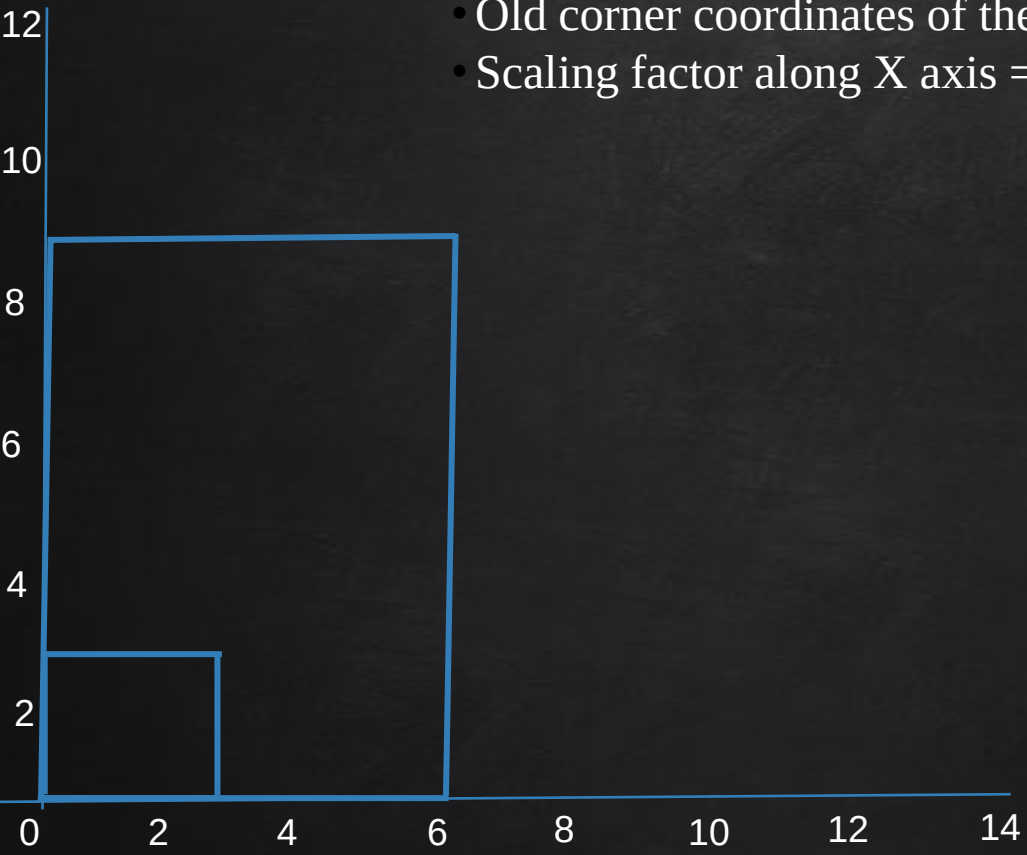
Scaling Matrix

$$X_{new} = X_{old} \times S_x$$

$$Y_{new} = Y_{old} \times S_y$$

P'=S.P

Namitha Ramachandran



- Old corner coordinates of the square = A (0, 3), B(3, 3), C(3, 0), D(0, 0)
- Scaling factor along X axis = 2 Scaling factor along Y axis = 3

(0,3)

X new=0 x 2=0

Ynew =3 x 3=9 (0,9)

(3,3)

X new=3 x 2=6

Ynew =3 x 3=9 (6,9)

(3,0)

X new=3 x 2=6

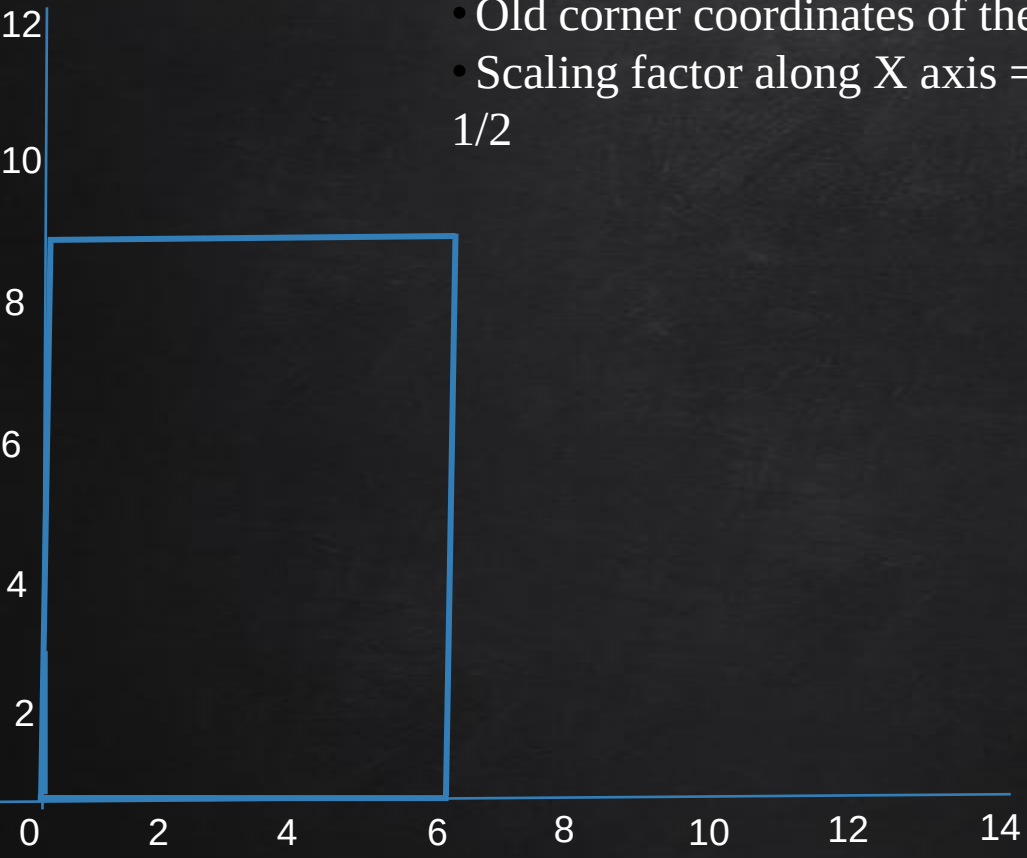
Ynew =0 x 3=0 (6,0)

(0,0)

X new=0 x 2=0

Ynew =0 x 3=0 (0,0)

Namitha Ramachandran



- Old corner coordinates of the square = A (0, 0), B(6, 0), C(6, 9), D(0, 9)
- Scaling factor along X axis = 1/2 Scaling factor along Y axis = 1/2

(0,0)

X new=0 x 0.5=0

Ynew =0 x 0.5=0 **(0,0)**

(6,0)

X new=6 x .5=3

Ynew =0 x .5=0 **(3,0)**

(6,9)

X new=6 x .5=3

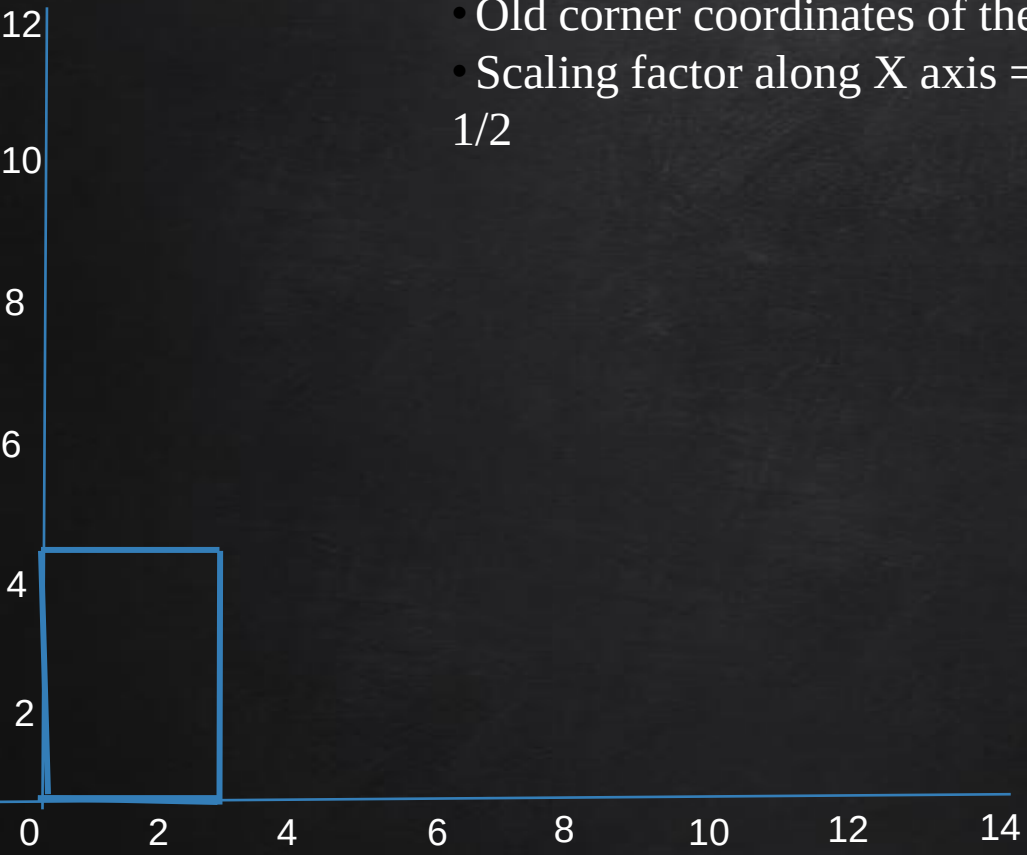
Ynew =9 x .5=4.5 **(3,4.5)**

(0,9)

X new=0 x.5=0

Ynew =9x .5=4.5 **(0,4.5)**

Namitha Ramachandran



- Old corner coordinates of the square = A (0, 0), B(6, 0), C(6, 9), D(0, 9)
- Scaling factor along X axis = 1/2 Scaling factor along Y axis = 1/2

(0,0)

X new=0 x 0.5=0

Ynew =0 x 0.5=0 (0,0)

(6,0)

X new=6 x .5=3

Ynew =0 x .5=0 (3,0)

(6,9)

X new=6 x .5=3

Ynew =9 x .5=4.5 (3,4.5)

(0,9)

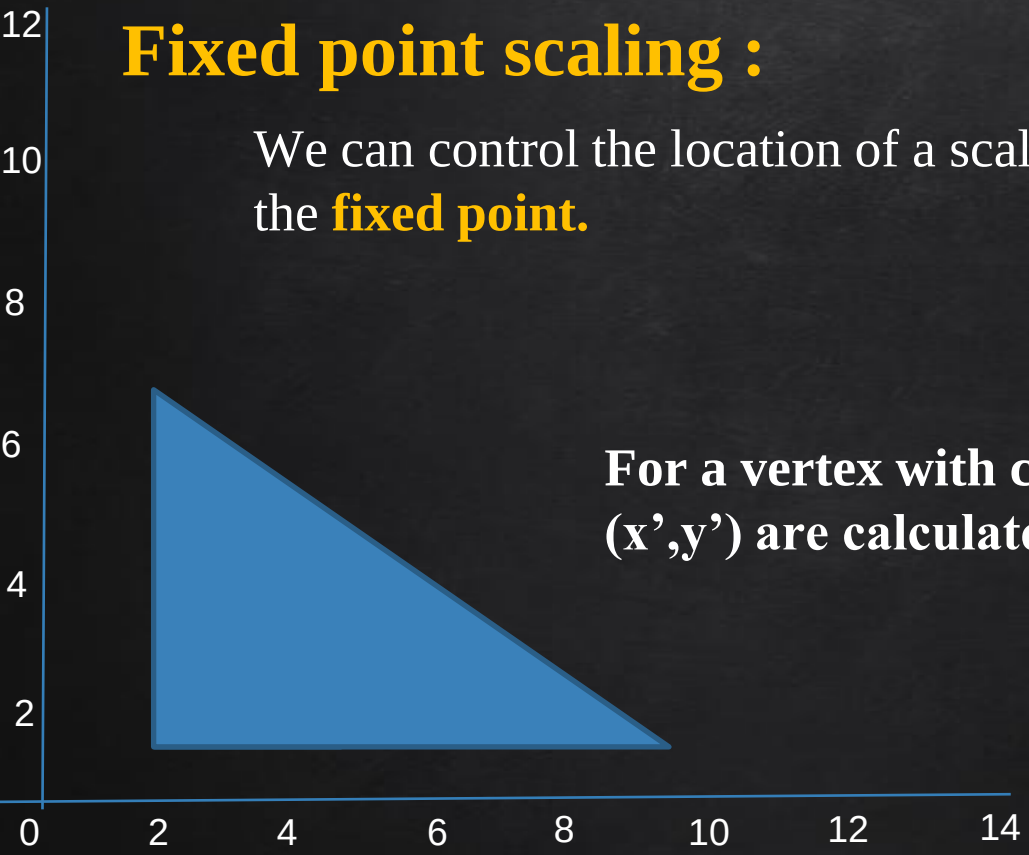
X new=0 x.5=0

Ynew =9x .5=4.5 (0,4.5)

Namitha Ramachandran

Fixed point scaling :

We can control the location of a scaled object by choosing a position, called the **fixed point.**



Example
Scale w r t (2,1)

For a vertex with coordinates (x,y) the scaled coordinates (x',y') are calculated w r t fixed point (xf,yf)

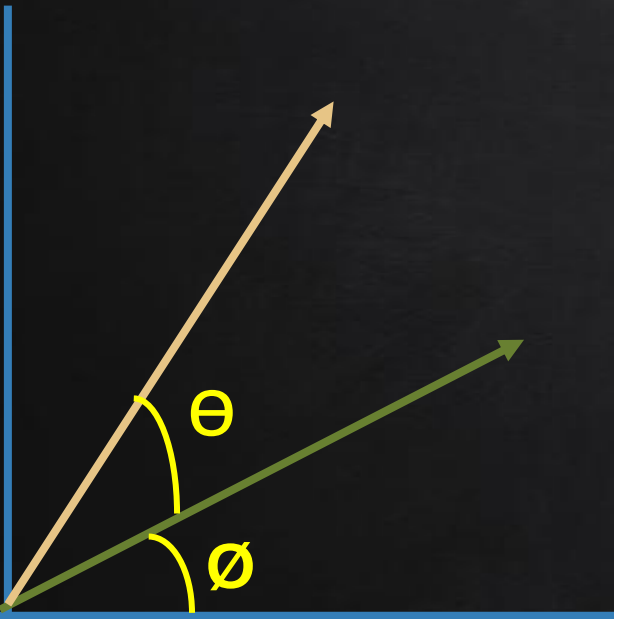
$$x' = x.s_x + x_f(1-s_x)$$

$$y' = y.s_y + y_f(1-s_y)$$

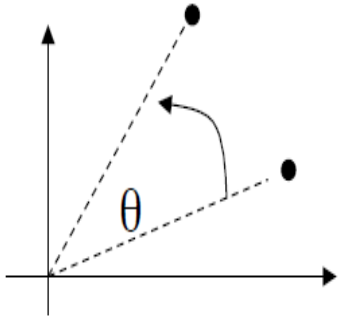
Namitha Ramachandran

2D ROTATION

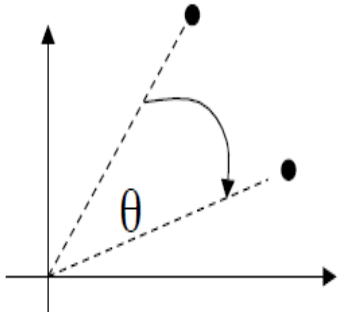
2D Rotation is the proc
in a two-dimensional p



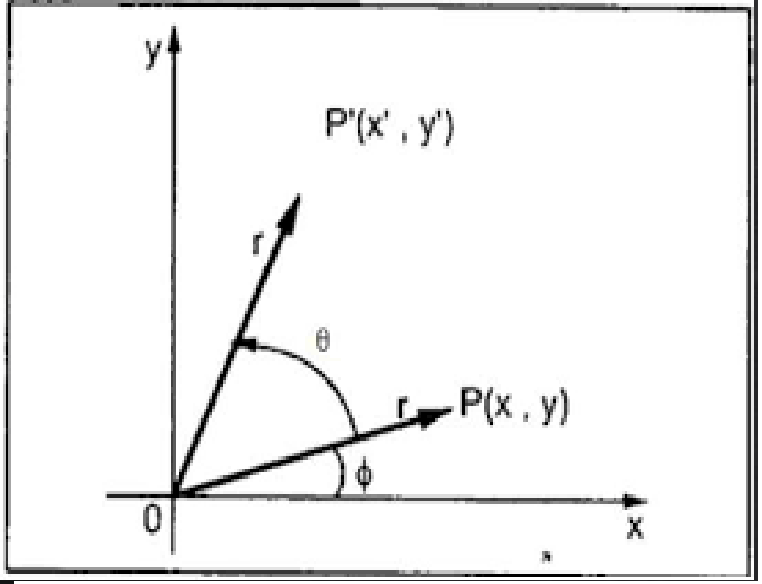
- Default rotation center: Origin (0,0)



θ> 0 : Rotate counter clockwise



θ< 0 : Rotate clockwise



Using standard trigonometric the original coordinate of point P X, Y can be represented as –

$$X = r\cos\phi \ldots\ldots (1)$$

$$Y = r\sin\phi \ldots\ldots (2)$$

Same way we can represent the point P' X', Y' as –

$$x' = r\cos(\phi + \theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta \ldots\ldots (3)$$

$$y' = r\sin(\phi + \theta) = r\cos\phi\sin\theta + r\sin\phi\cos\theta \ldots\ldots (4)$$

Substituting equation 1 & 2 in 3 & 4 respectively, we will get

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

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

$$\begin{vmatrix} x' \\ y' \end{vmatrix} = \begin{vmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{vmatrix} \begin{vmatrix} x \\ y \end{vmatrix}$$

Namitha Ramachandran

This rotation is achieved by using the following rotation equations-

$$X_{new} = X_{old} \times \cos\theta - Y_{old} \times \sin\theta$$

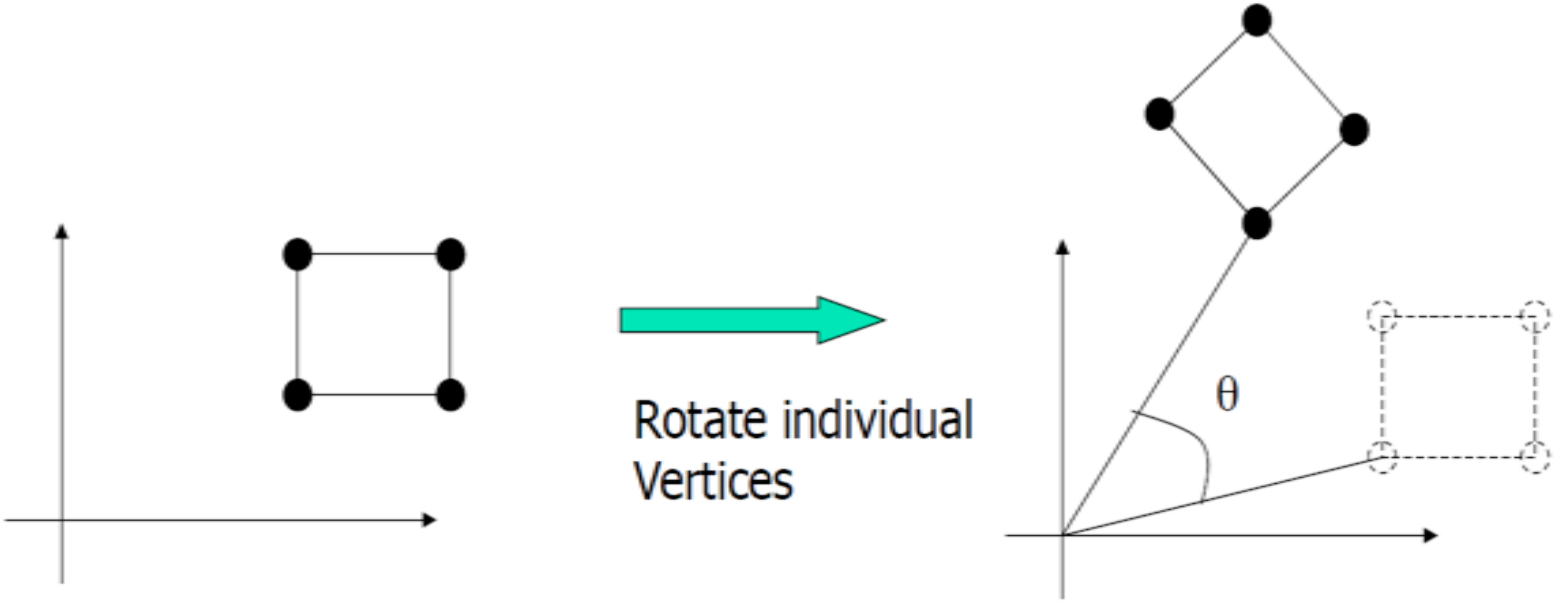
$$Y_{new} = X_{old} \times \sin\theta + Y_{old} \times \cos\theta$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

Rotation Matrix

Namitha Ramachandran

- How to rotate an object with multiple vertices?



Given a triangle with corner coordinates (0, 0), (1, 0) and (1, 1). Rotate the triangle by 90 degree anticlockwise direction and find out the new coordinates.

(0,0) → (0,0) $0\cos 90 - 0\sin 90 = 0$

$0\sin 90 + 0\cos 90 = 0$

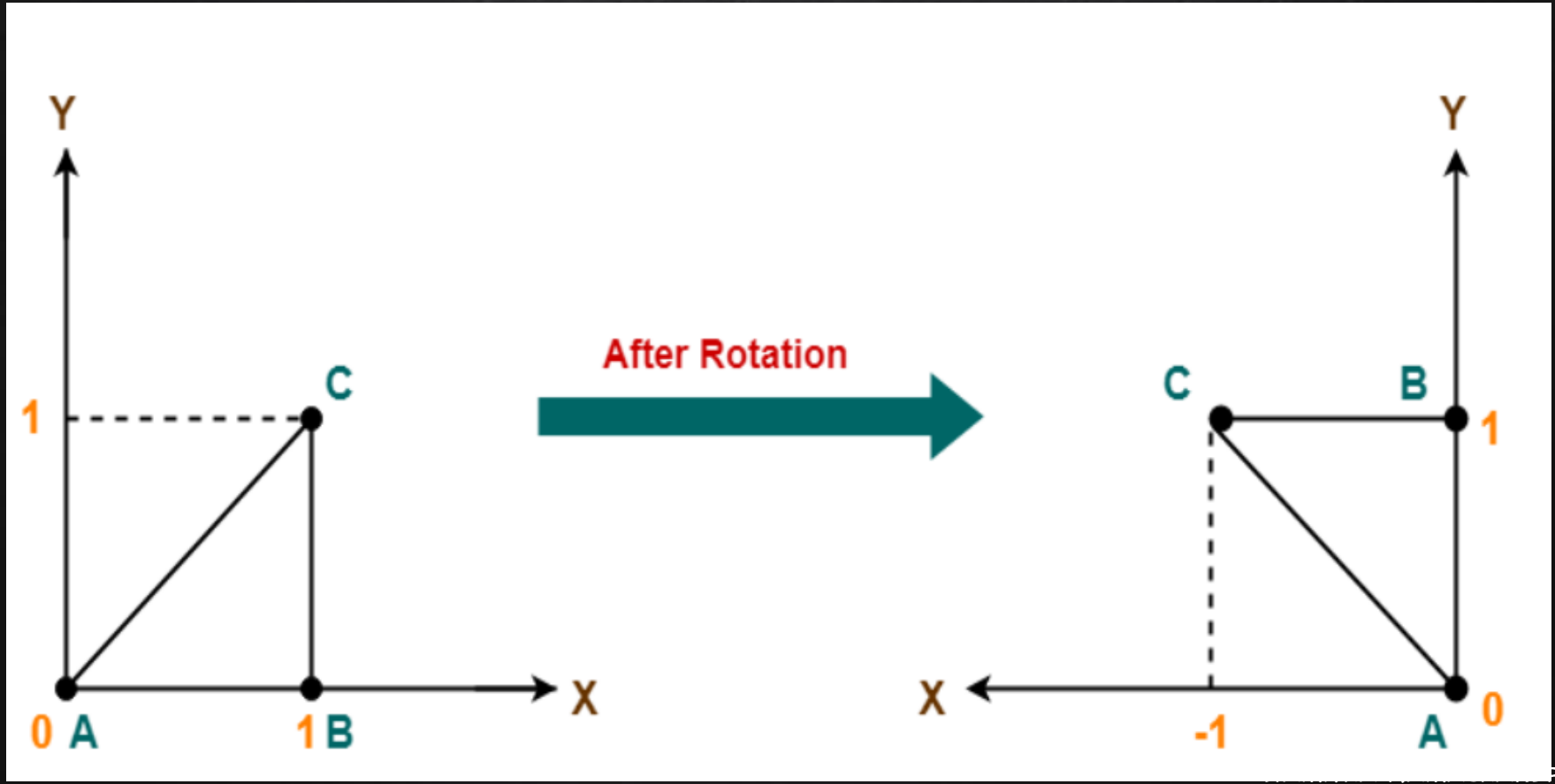
(1,0) → (0,1) $1\cos 90 - 0\sin 90 = 0$

$1\sin 90 + 0\cos 90 = 1$

(1,1) → (-1,1) $1\cos 90 - 1\sin 90 = -1$

$1\sin 90 + 1\cos 90 = 1$

Namitha Ramachandran


Y
1
C
0 A
1 B
X
After Rotation
Y
C
B
1
X
-1
A
0

Homogeneous coordinates

In homogeneous coordinate system, **two-dimensional coordinate positions (x, y) are represented by triple-coordinates(x,y,1)**.

we can perform all transformations using matrix multiplications. This allows us to pre-multiply all the matrices together.

Namitha Ramachandran

- Translation:

$$\begin{vmatrix} x' \\ y' \\ 1 \end{vmatrix} = \begin{vmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

- Rotation:

$$\begin{vmatrix} x' \\ y' \\ 1 \end{vmatrix} = \begin{vmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

- Scaling:

$$\begin{vmatrix} x' \\ y' \\ 1 \end{vmatrix} = \begin{vmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

Composite Transformation

Composing Transformation – the process of applying several transformations in succession to form one overall transformation.

Arbitrary Rotation Center

To rotate about an arbitrary point P (px,py) by

1) Translate the object so that P will coincide with the origin: T(-px, -py)

2)Rotate the object: R(θ)

3)Translate the object back: T(px,py)

Namitha Ramachandran

Matrix multiplication is associative

M3 x M2 x M1 = (M3 x M2) x M1 = M3 x (M2 x M1)

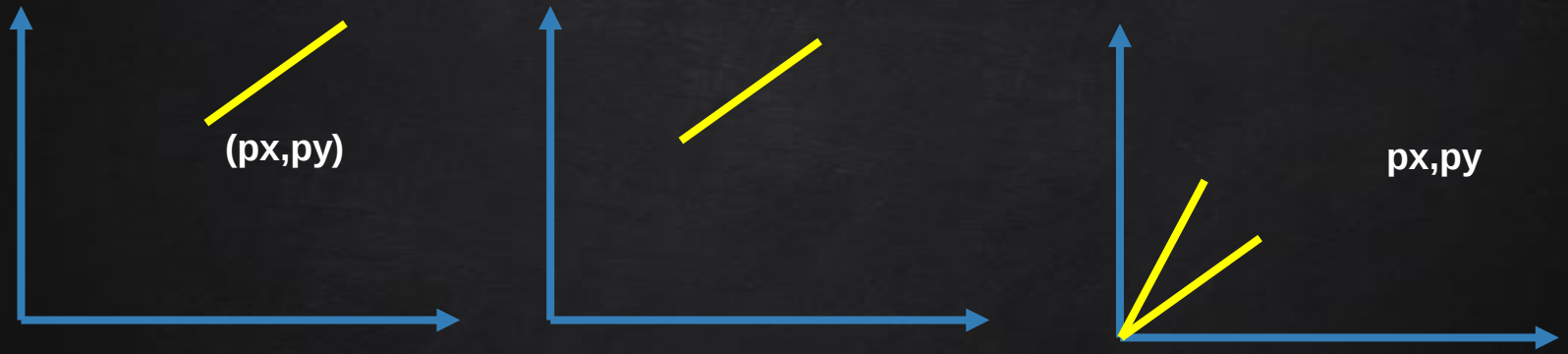
Transformation products may not be commutative A x B != Bx A

Namitha Ramachandran

T(-px,-py)

R(θ)

T(px,py)



Namitha Ramachandran

- Put in matrix form: $T(px,py)\ R(\theta)\ T(-px, -py) * P$

$$\begin{vmatrix} x' \\ y' \\ 1 \end{vmatrix} = \begin{vmatrix} 1 & 0 & px \\ 0 & 1 & py \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & -px \\ 0 & 1 & -py \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

$$\begin{bmatrix} \cos\theta & -\sin\theta & px.(1-\cos\theta) + py.\sin\theta \\ \sin\theta & \cos\theta & py.(1-\cos\theta) - px.\sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

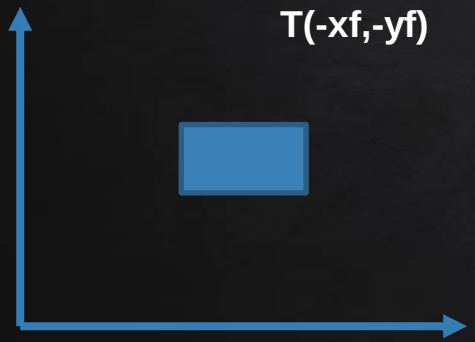
Arbitrary Scaling Pivot

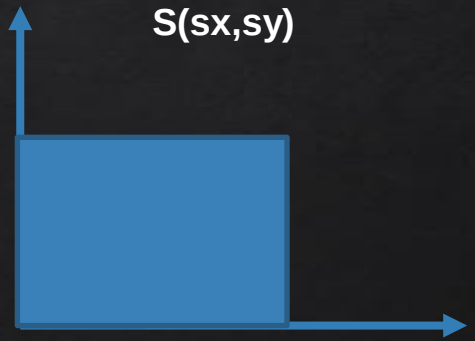
To scale about an arbitrary pivot point P(xf,yf):

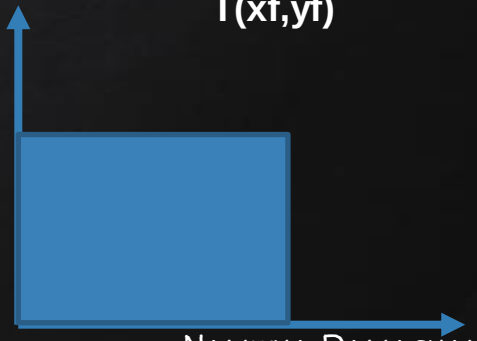
Translate the object so that P will coincide with the origin: T(-xf, -yf)

Scale the object: S(sx, sy)

Translate the object back: T(xf,yf)







$$\begin{bmatrix} 1 & 0 & xf \\ 0 & 1 & yf \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -xf \\ 0 & 1 & -yf \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} sx & 0 & xf \\ 0 & sy & yf \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -xf \\ 0 & 1 & -yf \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} sx & 0 & -sx.xf + xf \\ 0 & sy & -sy.yf + yf \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

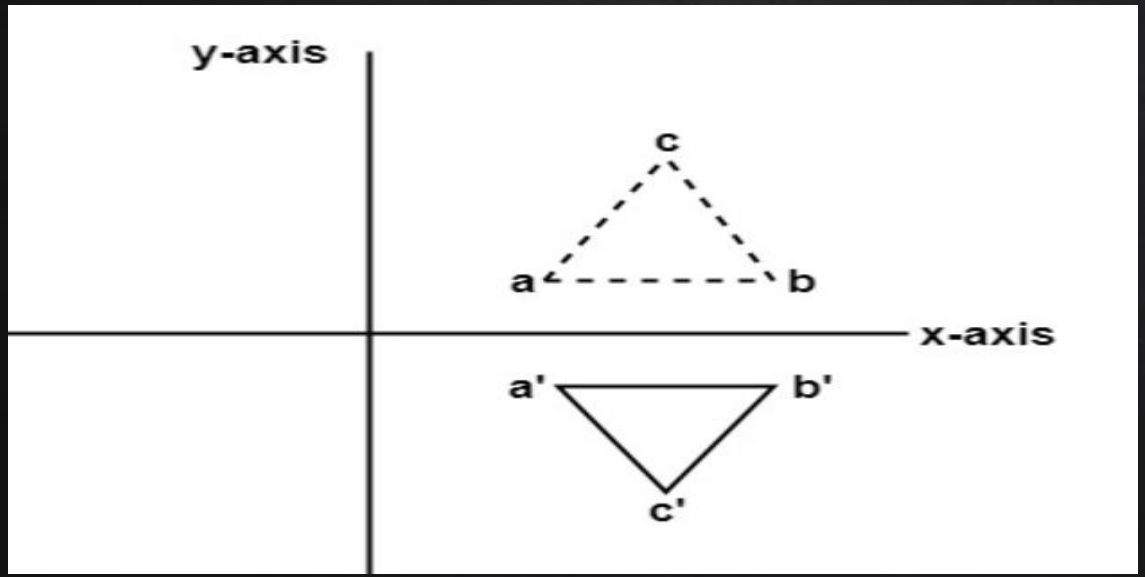
$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} sx & 0 & xf\,(1-sx) \\ 0 & sy & yf(1-\ sy) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Namitha Ramachandran

Reflection

It is a transformation that produces a mirror image of an object. The mirror image can be either about the x-axis or y-axis. The object is rotated by180°.

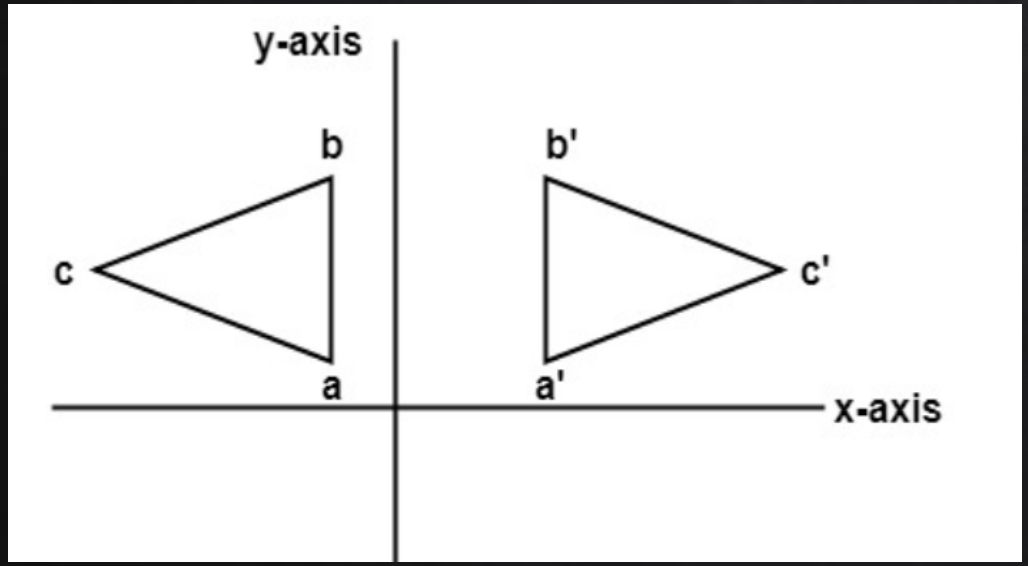
Reflection about x-axis:



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

Namitha Ramachandran

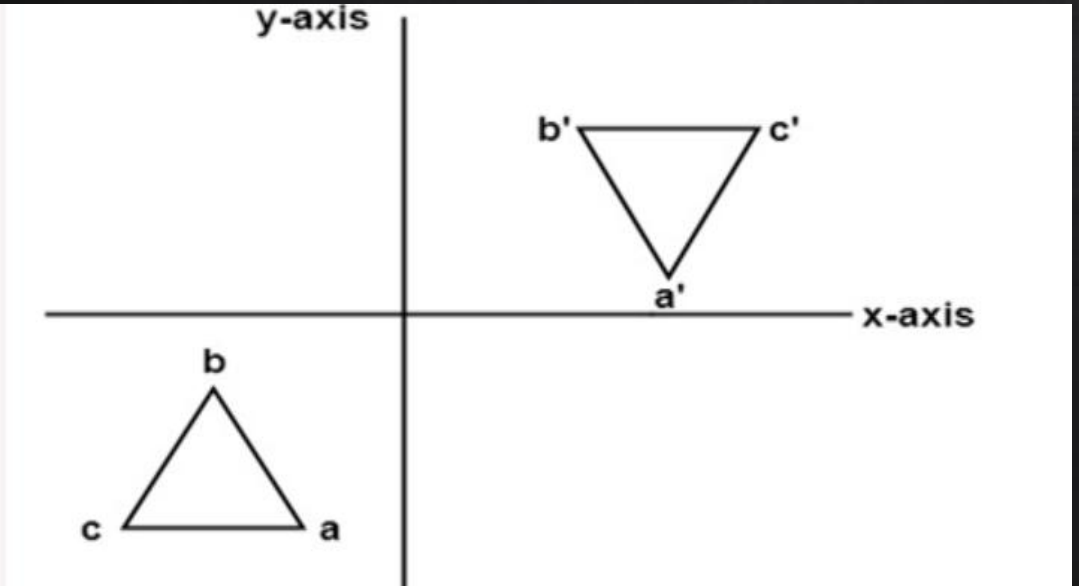
Reflection about y-axis:



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

Namitha Ramachandran

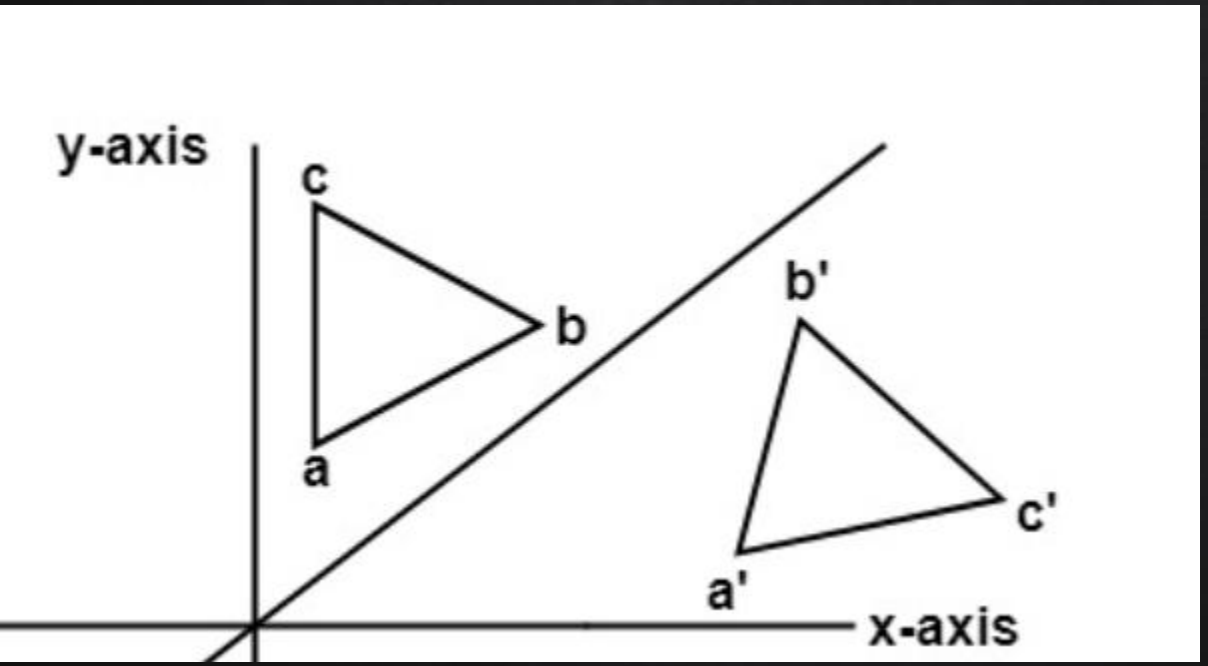
Reflection about an axis perpendicular to xy plane and passing through origin:



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

Namitha Ramachandran

Reflection about line y=x:



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

Namitha Ramachandran

Reflection about line y=-x:

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

Namitha Ramachandran

Shear

A transformation that slants the shape of an object is called the shear transformation. There are two shear transformations **X-Shear** and **Y-Shear**.

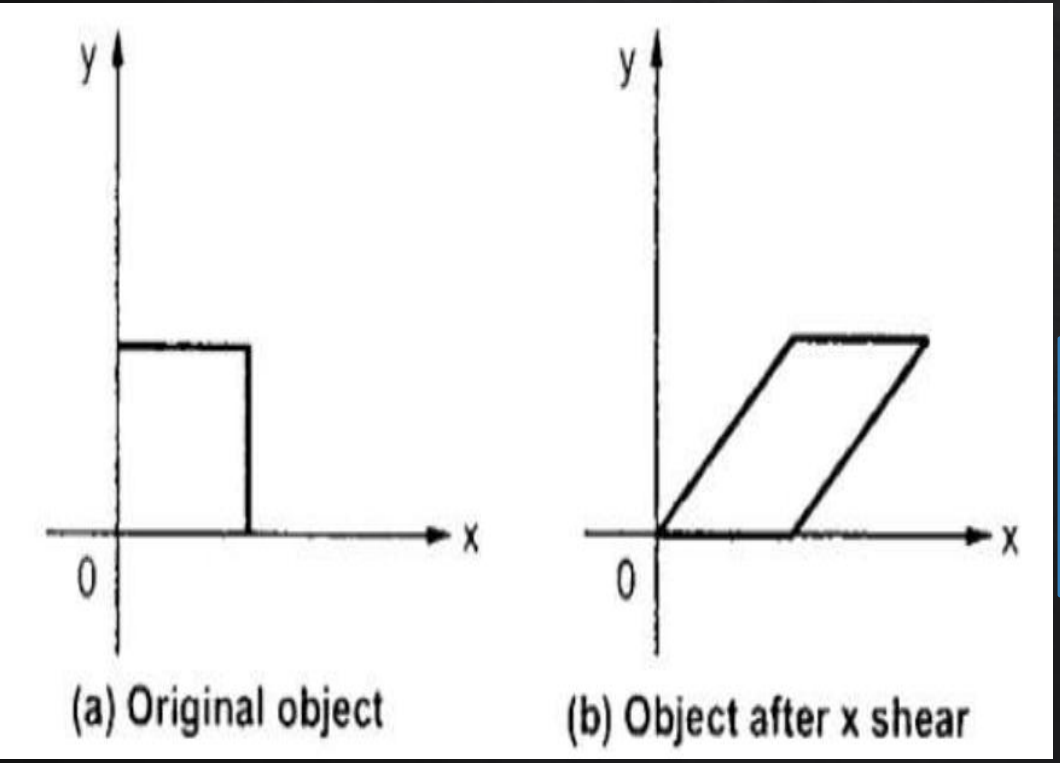
One shifts X coordinates values and other shifts Y coordinate values. However; in both the cases only one coordinate changes its coordinates and other preserves its values.

Shearing is also termed as **Skewing**

Namitha Ramachandran

X-Shear

The X-Shear preserves the Y coordinate and changes are made to X coordinates, which causes the vertical lines to tilt right or left



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

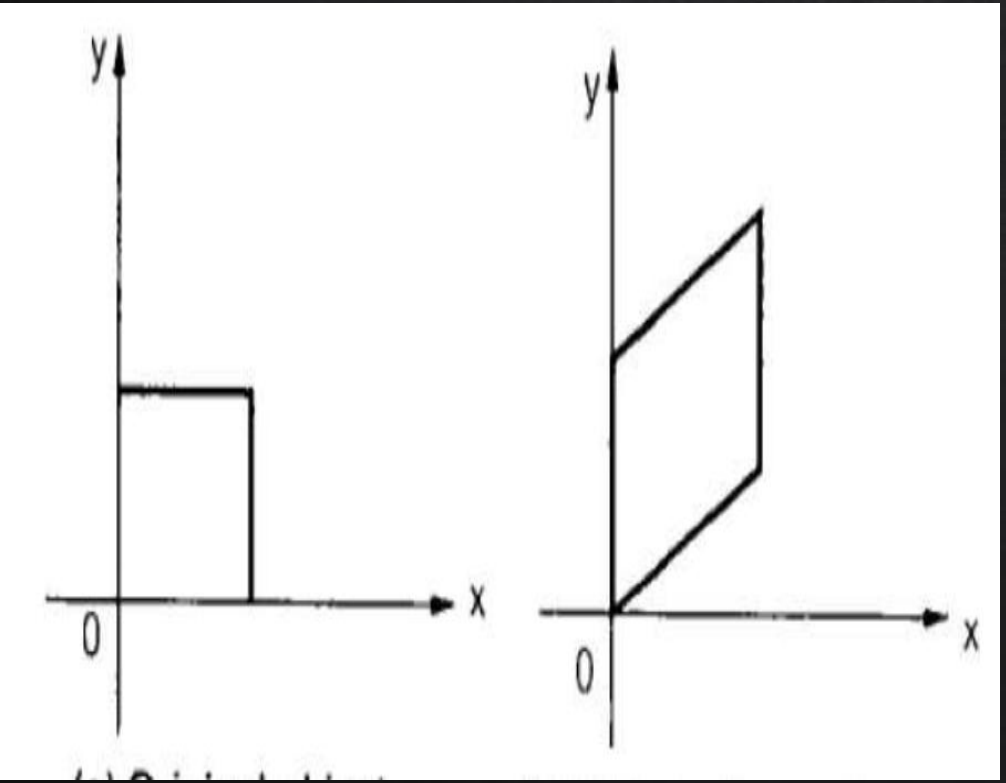
$$y' = y$$

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & shx & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Namitha Ramachandran

Y-Shear

The Y-Shear preserves the X coordinates and changes the Y coordinates



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

$$x' = x$$

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ shy & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Namitha Ramachandran

X Shear wrt a reference point yRef

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & shx & -shx*yref \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

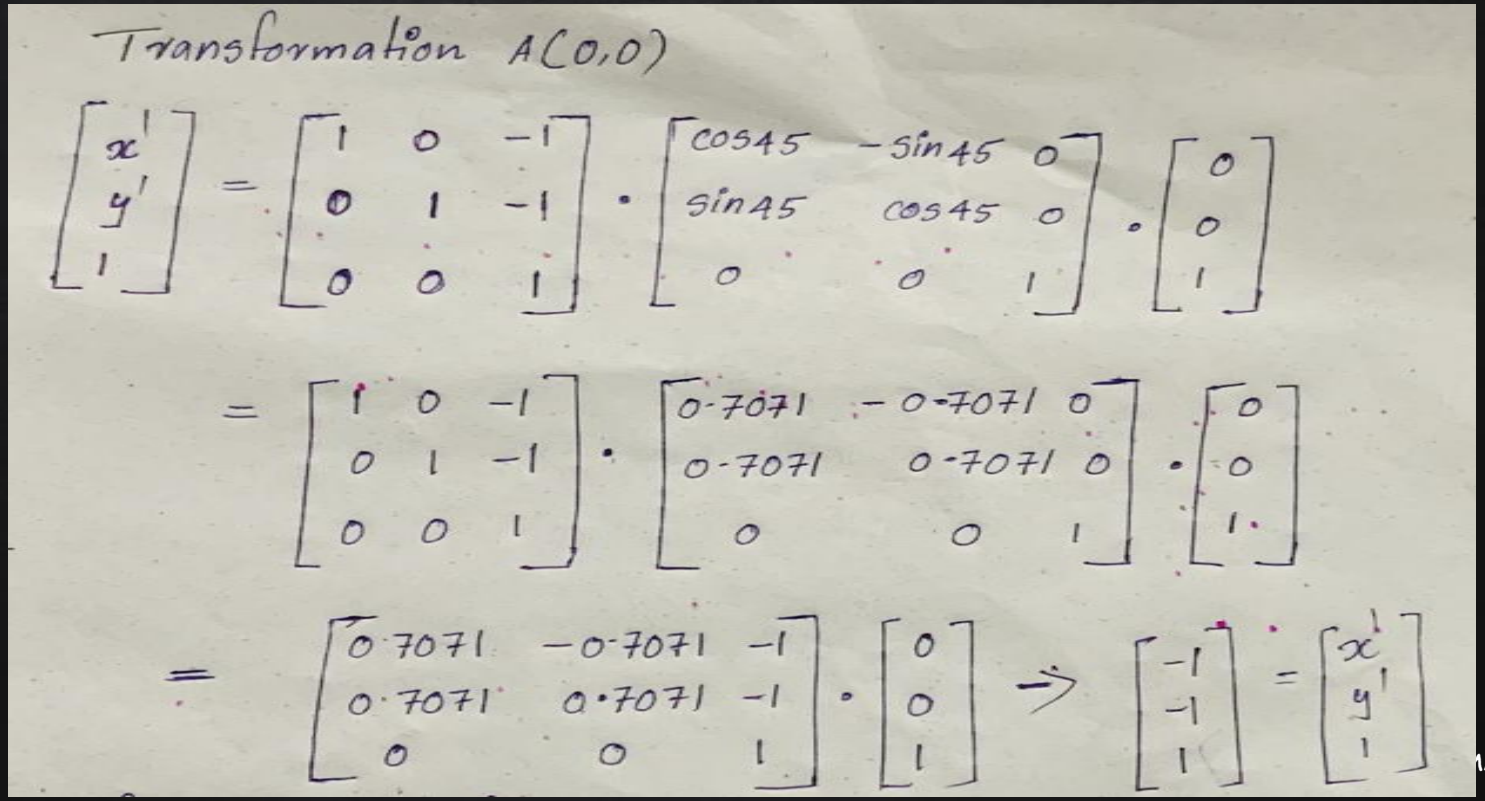
Namitha Ramachandran

Y Shear wrt a reference point xRef

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ shy & 1 & -shy*xref \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Namitha Ramachandran

Perform 45 degree rotation of a triangle A(0,0) ,B(1,1) and C(5,3) about the origin and about the fixed point(-1,-1).



Transformation $B(1,1)$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.7071 & -0.7071 & 0 \\ 0.7071 & 0.7071 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 0.7071 & -0.7071 & -1 \\ 0.7071 & 0.7071 & -1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} -1 \\ 0.4142 \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

Transformation c(5,3)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.7071 & -0.7071 & 0 \\ 0.7071 & 0.7071 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 5 \\ 3 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0.7071 & -0.7071 & -1 \\ 0.7071 & 0.7071 & -1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 5 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.4142 \\ 0.4142 \\ 1 \end{bmatrix}$$

Show that the composition of two successive rotations are additive
R(ɵ1).R(ɵ2)=R(ɵ1+ ɵ2)

we can write rotation matrix $R(\theta_1)$ as

$$R(\theta_1) = \begin{bmatrix} \cos\theta_1 & \sin\theta_1 \\ -\sin\theta_1 & \cos\theta_1 \end{bmatrix} \text{ and } R(\theta_2) = \begin{bmatrix} \cos\theta_2 & \sin\theta_2 \\ -\sin\theta_2 & \cos\theta_2 \end{bmatrix}$$

$$R(\theta_1) \cdot R(\theta_2) = \begin{bmatrix} \cos\theta_1 & \sin\theta_1 \\ -\sin\theta_1 & \cos\theta_1 \end{bmatrix} \times \begin{bmatrix} \cos\theta_2 & \sin\theta_2 \\ -\sin\theta_2 & \cos\theta_2 \end{bmatrix}$$

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

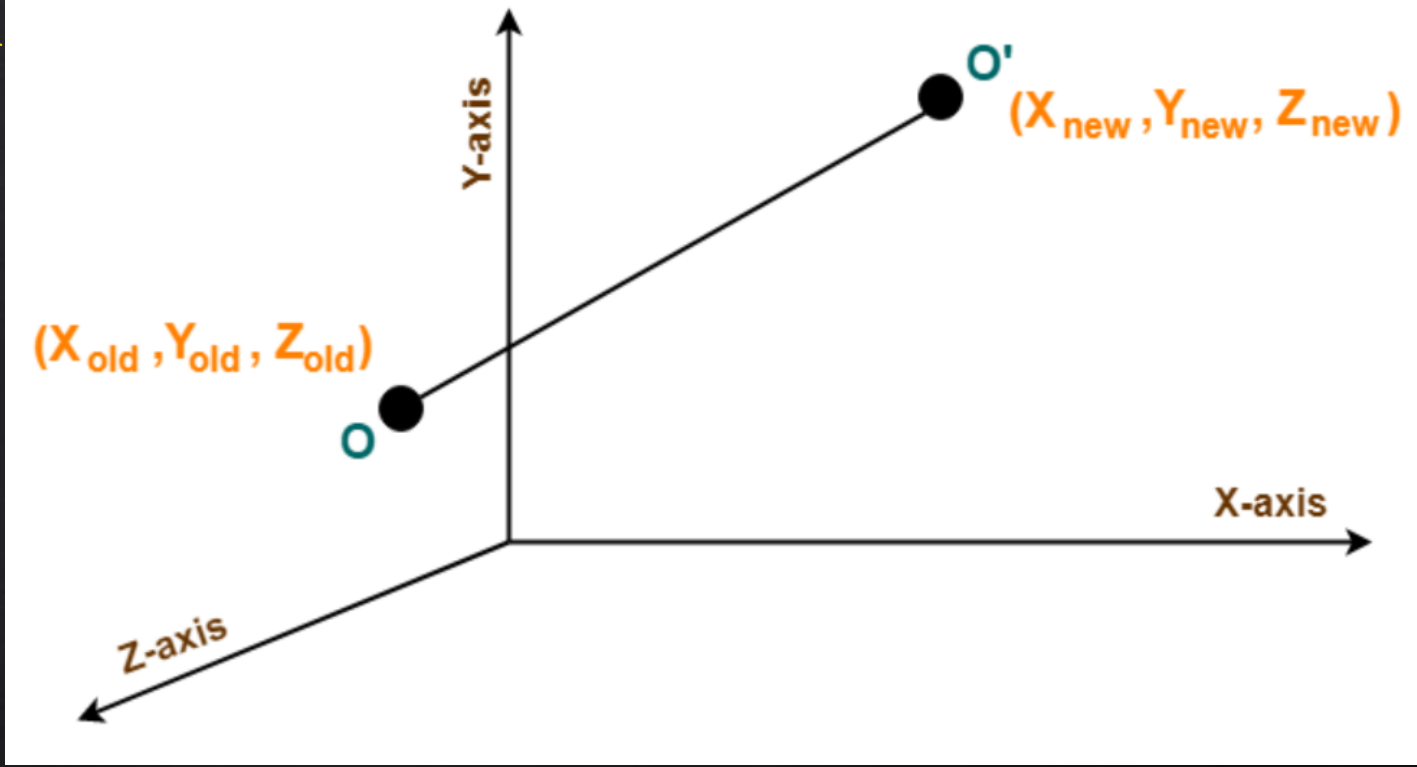
$$= \begin{bmatrix} \cos(\theta_1 + \theta_2) & \sin(\theta_1 + \theta_2) \\ -\sin(\theta_1 + \theta_2) & \cos(\theta_1 + \theta_2) \end{bmatrix}$$

Basic 3D transformations

3-D Transformation is the process of manipulating the view of a three-D object with respect to its original position by modifying its physical attributes through various methods of transformation like **Translation, Scaling, Rotation, Shear , Reflection** etc.

Namitha Ramachandran

3D Translation



Translation vector or Shift vector = (T_x, T_y, T_z)

Namitha Ramachandran

Translation vector or Shift vector = (T_x, T_y, T_z)

$$X_{new} = X_{old} + T_x$$
$$Y_{new} = Y_{old} + T_y$$
$$Z_{new} = Z_{old} + T_z$$

$$P' = T.P$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ Z_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ Z_{old} \\ 1 \end{bmatrix}$$

Namitha Ramachandran

Given a 3D object with coordinate points A(0, 3, 1), B(3, 3, 2), C(3, 0, 0), D(0, 0, 0). Apply the translation with the distance 1 towards X axis, 1 towards Y axis and 2 toward Z axis and obtain the new coordinates of the object.

Translation vector = $(T_x, T_y, T_z) = (1, 1, 2)$

For Coordinates A(0, 3, 1)

- $X_{new} = X_{old} + T_x = 0 + 1 = 1$
- $Y_{new} = Y_{old} + T_y = 3 + 1 = 4$
- $Z_{new} = Z_{old} + T_z = 1 + 2 = 3$
- New coordinates of A = (1, 4, 3)

For Coordinates B(3, 3, 2)

- $X_{new} = X_{old} + T_x = 3 + 1 = 4$
- $Y_{new} = Y_{old} + T_y = 3 + 1 = 4$
- $Z_{new} = Z_{old} + T_z = 2 + 2 = 4$

New coordinates of B = (4,4,4)

Namitha Ramachandran

For Coordinates C(3, 0, 0)

- $X_{new} = X_{old} + T_x = 3 + 1 = 4$
- $Y_{new} = Y_{old} + T_y = 0 + 1 = 1$
- $Z_{new} = Z_{old} + T_z = 0 + 2 = 2$
- New coordinates of C = (4, 1, 2).

For Coordinates D(0, 0, 0)

$X_{new} = X_{old} + T_x = 0 + 1 = 1$

$Y_{new} = Y_{old} + T_y = 0 + 1 = 1$

$Z_{new} = Z_{old} + T_z = 0 + 2 = 2$

New coordinates of D=(1,1,2)

Namitha Ramachandran

3D Rotation

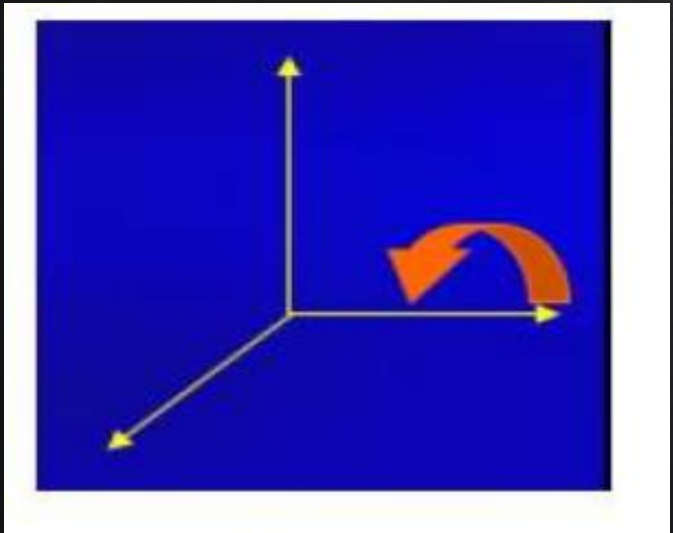
3D Rotation is a process of rotating an object with respect to an angle in a three-dimensional plane .
In 3 dimensions, there are 3 possible types of rotation-

X-axis Rotation
Y-axis Rotation
Z-axis Rotation

Namitha Ramachandran

X-Axis Rotation

$$X_{new} = X_{old}$$
$$Y_{new} = Y_{old} .\cos\theta - Z_{old} .\sin\theta$$
$$Z_{new} = Y_{old} .\sin\theta + Z_{old} .\cos\theta$$

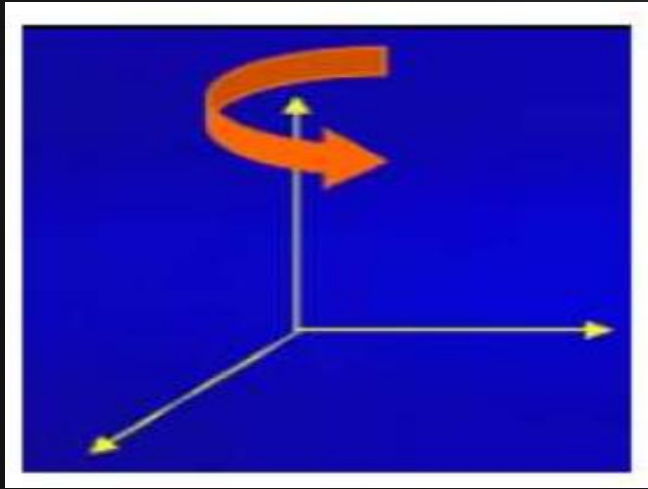


$$\begin{bmatrix} X_{new} \\ Y_{new} \\ Z_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ Z_{old} \\ 1 \end{bmatrix}$$

Namitha Ramachandran

Y-Axis Rotation

$$X_{new} = Z_{old} \cdot \sin\theta + X_{old} \cdot \cos\theta$$
$$Y_{new} = Y_{old}$$
$$Z_{new} = Z_{old} \cdot \cos\theta - X_{old} \cdot \sin\theta$$



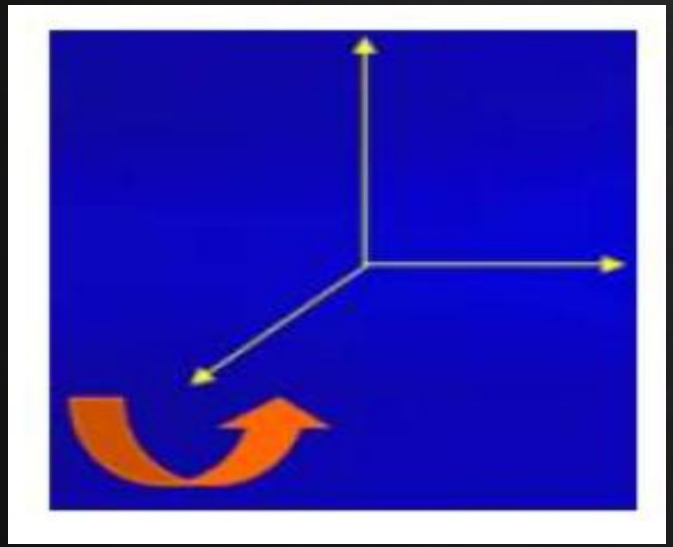
$$\begin{bmatrix} X_{new} \\ Y_{new} \\ Z_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ Z_{old} \\ 1 \end{bmatrix}$$

Namitha Ramachandran

Z-Axis Rotation

$$X_{new} = X_{old} .\cos\theta - Y_{old} .\sin\theta$$
$$Y_{new} = X_{old} .\sin\theta + Y_{old} .\cos\theta$$
$$Z_{new} = Z_{old}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ Z_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ Z_{old} \\ 1 \end{bmatrix}$$



Namitha Ramachandran

Given a homogeneous point (1, 2, 3). Apply rotation 90 degree towards X, Y and Z axis and find out the new coordinate points.

X-Axis Rotation

$$X_{new} = X_{old} = 1$$
$$Y_{new} = Y_{old} \times \cos\theta - Z_{old} \times \sin\theta = 2 \times \cos90° - 3 \times \sin90° = 2 \times 0 - 3 \times 1 = -3$$
$$Z_{new} = Y_{old} \times \sin\theta + Z_{old} \times \cos\theta = 2 \times \sin90° + 3 \times \cos90° = 2 \times 1 + 3 \times 0 = 2$$

New coordinates after rotation = (1,-3,2)

Namitha Ramachandran

Y-Axis Rotation

$X_{new} = Z_{old} \times \sin\theta + X_{old} \times \cos\theta = 3 \times \sin90° + 1 \times \cos90° = 3 \times 1 + 1 \times 0 = 3$
$Y_{new} = Y_{old} = 2$
$Z_{new} = Y_{old} \times \cos\theta - X_{old} \times \sin\theta = 2 \times \cos90° - 1 \times \sin90° = 2 \times 0 - 1 \times 1 = -1$

New coordinates after rotation = (3,2,-1)

Z-Axis Rotation

$X_{new} = X_{old} \times \cos\theta - Y_{old} \times \sin\theta = 1 \times \cos90° - 2 \times \sin90° = 1 \times 0 - 2 \times 1 = -2$
$Y_{new} = X_{old} \times \sin\theta + Y_{old} \times \cos\theta = 1 \times \sin90° + 2 \times \cos90° = 1 \times 1 + 2 \times 0 = 1$
$Z_{new} = Z_{old} = 3$

New coordinates after rotation = (-2,1,3)

Namitha Ramachandran

General 3D Rotation

When an object is to be rotated about an axis that is parallel to one of the coordinate axes,

Translate the object so that the rotation axis coincides with the parallel coordinate axis.

Perform the specified rotation about that axis.

Translate the object so that the rotation axis is moved back to its original position.

$$P' = T^{-1} \cdot R(\theta) \cdot T \cdot P$$

Namitha Ramachandran

When the object is to be rotated about an axis that is not parallel to one the coordinate axes

Translate the object so that the rotation axis passes through the coordinate origin.

Rotate the object so that the axis of rotation coincides with one of the coordinate axes.

Perform the specified rotation about that coordinate axis.
Apply inverse rotation to bring the rotation axis back to its original orientation.
Apply the inverse translation to bring the rotation axis back to its original position.

Namitha Ramachandran

3D Scaling

Scaling is a process of modifying or altering the size of objects.

Scaling may be used to increase or reduce the size of object.
Scaling subjects the coordinate points of the original object to change.
Scaling factor determines whether the object size is to be increased or reduced.
If scaling factor > 1, then the object size is increased.
If scaling factor < 1, then the object size is reduced.

$$X_{new} = X_{old} \times S_x$$
$$Y_{new} = Y_{old} \times S_y$$
$$Z_{new} = Z_{old} \times S_z$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ Z_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ Z_{old} \\ 1 \end{bmatrix}$$

Given a 3D object with coordinate points A(0, 3, 3), B(3, 3, 6), C(3, 0, 1), D(0, 0, 0). Apply the scaling parameter 2 towards X axis, 3 towards Y axis and 3 towards Z axis and obtain the new coordinates of the object.

Scaling factor along X axis = 2
Scaling factor along Y axis = 3
Scaling factor along Z axis = 3

For Coordinates A(0, 3, 3)

$X_{new} = X_{old} \times S_x = 0 \times 2 = 0$
$Y_{new} = Y_{old} \times S_y = 3 \times 3 = 9$
$Z_{new} = Z_{old} \times S_z = 3 \times 3 = 9$

New coordinates of corner A after scaling = (0, 9, 9).

For Coordinates B(3, 3, 6)

$X_{new} = X_{old} \times S_x = 3 \times 2 = 6$
$Y_{new} = Y_{old} \times S_y = 3 \times 3 = 9$
$Z_{new} = Z_{old} \times S_z = 6 \times 3 = 18$

New coordinates of corner B after scaling = (6, 9, 18).

Namitha Ramachandran

For Coordinates C(3, 0, 1)

$$X_{new} = X_{old} \times S_x = 3 \times 2 = 6$$
$$Y_{new} = Y_{old} \times S_y = 0 \times 3 = 0$$
$$Z_{new} = Z_{old} \times S_z = 1 \times 3 = 3$$

New coordinates of corner C after scaling = (6, 0, 3).

Coordinates D(0, 0, 0)

$$X_{new} = X_{old} \times S_x = 0 \times 2 = 0$$
$$Y_{new} = Y_{old} \times S_y = 0 \times 3 = 0$$
$$Z_{new} = Z_{old} \times S_z = 0 \times 3 = 0$$

New coordinates of corner D after scaling = (0, 0, 0).

Namitha Ramachandran

Scaling with respect to a selected fixed point(xf,yf,zf)

Translate the fixed point to the origin.
Scale the object relative to the coordinate origin .
Translate the fixed point back to its original position.

T(xf,yf,zf).S(sx,sy,sz).T(-xf,-yf,-zf)=

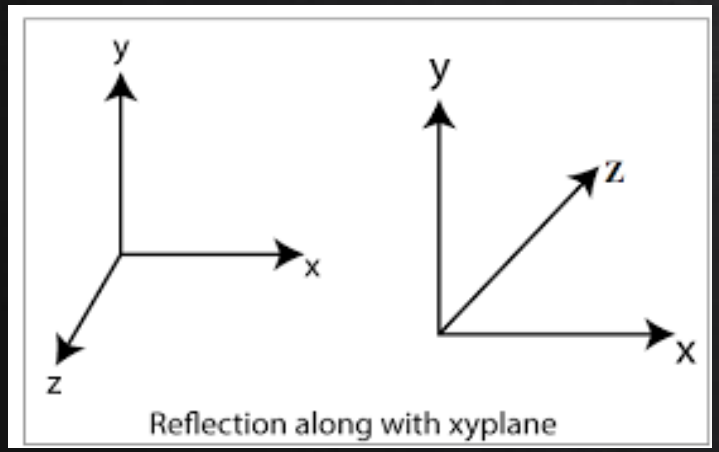
$$\begin{bmatrix} Sx & 0 & 0 & (1-sx)xf \\ 0 & sy & 0 & (1-sy)yf \\ 0 & 0 & sz & (1-sz)zf \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Namitha Ramachandran

3D Reflection

Reflection is a kind of rotation where the angle of rotation is 180 degree.
The reflected object is always formed on the other side of the mirror.
The size of the reflected object is the same as the original object's size.

Reflection Relative to XY Plane:



Reflection along with xyplane

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ Z_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ Z_{old} \\ 1 \end{bmatrix}$$

Reflection Relative to YZ Plane

$$X_{new} = -X_{old}$$
$$Y_{new} = Y_{old}$$
$$Z_{new} = Z_{old}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ Z_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ Z_{old} \\ 1 \end{bmatrix}$$

Namitha Ramachandran

Reflection Relative to XZ Plane

$$X_{new} = X_{old}$$
$$Y_{new} = -Y_{old}$$
$$Z_{new} = Z_{old}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ Z_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ Z_{old} \\ 1 \end{bmatrix}$$

Namitha Ramachandran

Given a 3D triangle with coordinate points A(3, 4, 1), B(6, 4, 2), C(5, 6, 3). Apply the reflection on the XY plane and find out the new coordinates of the object.

For Coordinates A(3, 4, 1)

$X_{new} = X_{old} = 3$

$Y_{new} = Y_{old} = 4$

$Z_{new} = -Z_{old} = -1$

New coordinates of corner A = (3, 4, -1).

For Coordinates B(6, 4, 2)

$X_{new} = X_{old} = 6$

$Y_{new} = Y_{old} = 4$

$Z_{new} = -Z_{old} = -2$

New coordinates of corner B = (6, 4, -2).

Namitha Ramachandran

For Coordinates B(6, 4, 2)

$X_{new} = X_{old} = 6$
$Y_{new} = Y_{old} = 4$
$Z_{new} = -Z_{old} = -2$
New coordinates of corner B after reflection = (6, 4, -2).

For Coordinates C(5, 6, 3)

$X_{new} = X_{old} = 5$
$Y_{new} = Y_{old} = 6$
$Z_{new} = -Z_{old} = -3$
New coordinates of corner C = (5, 6, -3).

Namitha Ramachandran

3D Shear

In a three dimensional plane, the object shape can be changed along the X direction, Y direction as well as Z direction.

So, there are three versions of shearing-

Shearing in X Axis-

$$X_{new} = X_{old}$$
$$Y_{new} = Y_{old} + Sh_y \times X_{old}$$
$$Z_{new} = Z_{old} + Sh_z \times X_{old}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ Z_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ Sh_y & 1 & 0 & 0 \\ Sh_z & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ Z_{old} \\ 1 \end{bmatrix}$$

3D Shearing Matrix

(In X axis)

Namitha Ramachandran

Shearing in Y Axis-

$$X_{new} = X_{old} + Sh_x \times Y_{old}$$
$$Y_{new} = Y_{old}$$
$$Z_{new} = Z_{old} + Sh_z \times Y_{old}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ Z_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & Sh_x & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & Sh_z & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ Z_{old} \\ 1 \end{bmatrix}$$

3D Shearing Matrix

(In Y axis)

Namitha Ramachandran

Shearing in Z Axis-

$$X_{new} = X_{old} + Sh_x \times Z_{old}$$
$$Y_{new} = Y_{old} + Sh_y \times Z_{old}$$
$$Z_{new} = Z_{old}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ Z_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & Sh_x & 0 \\ 0 & 1 & Sh_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ Z_{old} \\ 1 \end{bmatrix}$$

3D Shearing Matrix

(In Z axis)

Namitha Ramachandran