COMPUTER GRAPHIC

Computer graphic references:

1. M.Berger," Computer Graphic with Pascal ", B/C Publishing Company, 1984.
2. J.D.Foley & A.Dametal," Introduction to Computer Graphic ", Addison – wesly, 1993.
3. D.Hearn & M.p.Baker," Computer Graphics ", 2nd Ed., Prentice – Hall, 1994.

Computer Graphics Contents :

1. Scan Conversion

   Line, Bresenham's line algorithm, Bresenham's circle algorithm, Ellipses, Area and Sectors, Rectangle, Region filling.
2. Two dimensional graphics transformations

   Transition, Rotation, Reflection, Scaling, Coordinates transformation.
3. Tow dimensional viewing transformation and clipping windows and view part, Rectangular clipping windows, Points, Line segment, Convex polygons clipping windows.
4. Three dimensional graphics

   Three dimensional transformation, translation, Scaling, Rotation, Reflection coordinates transformation. (Viewing transformation as 4×4 matrix with homogeneous coordinates).

INTRODUCTION TO COMPUTER GRAPHIC

Overview :

Computer graphic can be defined as the creation and manipulation of graphic image by means of computer.

Computer graphic started as a technique to enhance the display of information generated by a computer. This ability to interpret and represent numerical data in pictures has significantly increased the computer's ability to represent information to the user in a clear and understandable form.

Application of Graphics :

1. Management may be displayed as charts and diagrams, i.e. large amounts of data are rapidly converted into bar charts, pie charts and graphs.
2. Scientific theories and models may be described in pictorial form.
3. Maps can be created for all kinds of geographic information.
4. Simulation.
5. Video games provides a new form of entertainment.

Remarks :

Mode:

divided to:

1. **Text mode:** deal with characters, numbers and symbols.
2. **Graphics mode:** deal with pixels.

Picture Elements:

1. **Pixel:** is the smallest addressable screen element. It is the smallest piece of the display screen which we can control.
2. **Line :** has patron or type.

   Specification: color, lighting, type, width.

   Types of line: ——————

   – – – – – – – –

   ....
3. **Curve:** depends on start point and end point and angle.

a) Raster – scan display :

This video display screen used by most microcomputer is divided into small rectangles or dots. These dots are referred to as picture elements or pixels.

We can consider CRT (Cathode Ray Tube) screen to consist of number of vertical and horizontal lines, where each horizontal line is made up of pixels figure (1). These horizontal lines are called Raster – scan lines, and the video display referred to a raster – scan display.

b) Resolution :

Means the number of scan lines and the number of pixels on a line or dots per unit area.

- Low resolution 300 scan lines & 400 pixels (lines).
- High resolution more than 1000 scan lines & 1000 pixels (lines).

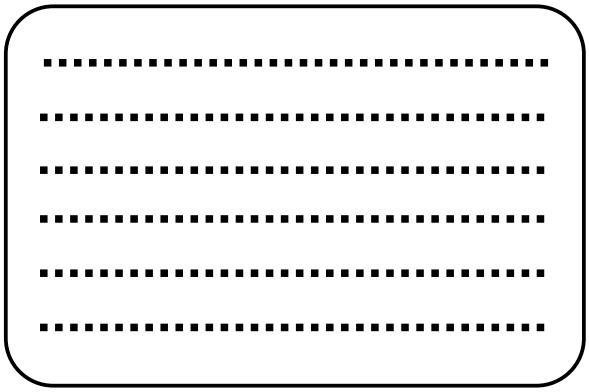


Figure (1) A raster display image

The graphic display :

The graphic display consists of three components :-

a) Frame buffer.

b) Display controller.

c) Scan conversion algorithms.

a) Frame buffer :

Each screen pixel corresponds to a particular entry in a two – dimensional array residing in memory. This memory is called a **frame buffer** or **bit map**.

The number of rows in the frame buffer array equal the number of raster lines on the display screen, and the number of columns in this array equals the number of pixels on each raster line.

The current tends is to have the frame buffer accessible to central processing unit (CPU) of the main memory, Thus allowing rapid of the storage image.

Whenever we wish to display a pixel on the screen, a specific value is placed into the corresponding memory location in the frame buffer array. In figure (2), a value of 1 placed in a location in the frame buffer results in the corresponding (black) pixel being displayed on the screen.

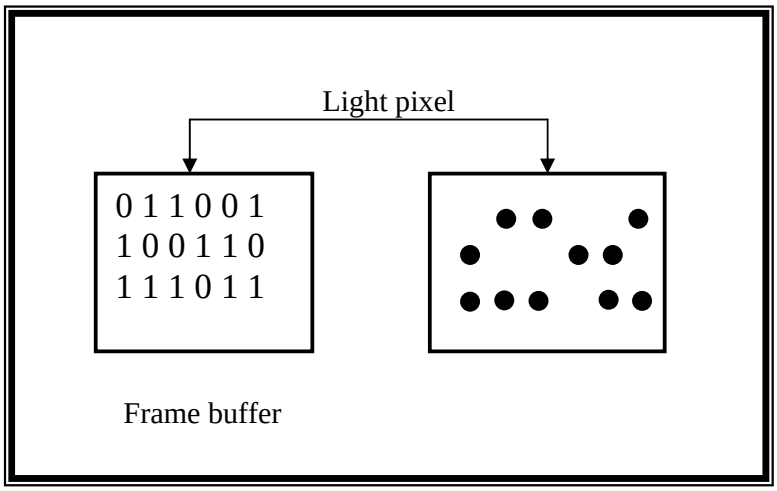


Figure (2) Bit plane of frame buffer

Number of pixels depending on screen type, so that we have many types likes :-

1) Monochrome 2 color (black, green).
2) CGA (Color Graphic Adapter) 16 color 320×200 .
3) EGA (Enhanced Graphic Adapter) 16 color 640×480 .
4) VGA (Video Graphic Adapter) 256 color 640×480 .
5) SVGA (Super Video Graphic Adapter) 256 color 1024×768 .

Each screen location pixel and corresponding memory location in the frame buffer is accessed by an (X,Y) integer coordinate pair. The X value refers to the columns, the Y value refers to the row position.

b) Display controller :

The hardware device reads the contents of frame buffer into a video buffer which then converts the digital representation of a string of pixel values into analog voltage signals that are sent serially to the video screen.

Whenever the display controller encounters a value of 1 in a single – bit – plane frame buffer a high – voltage. Signal is sent to the CRT which truns on the corresponding screen pixel.

c) Scan conversion :

Image are usually defined in terms of equation for example C + D = 5 or graphic descriptions, such as " draw a line from A to B ". Scan conversion is the process of converting this abstract representation of an image into the appropriate pixel values in the frame buffer.

" DRAWING ELEMENTARY FIGURE "

1 . PLOTTING POINT :

Each pixel in Pascal programming language is accessed by a positive integer (x,y) coordinate pair, the value x start at origin 0, and increase from left to right, the y value start at 0 increase from top to bottom, as in figure (3).

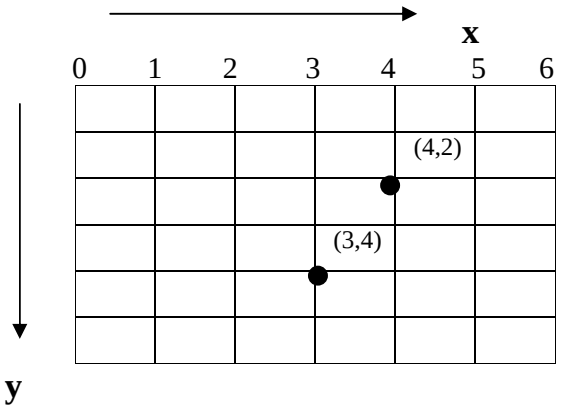


Figure (3) Screen in Pascal programming language

2 . LINE DRAWING ALGORITHMS :

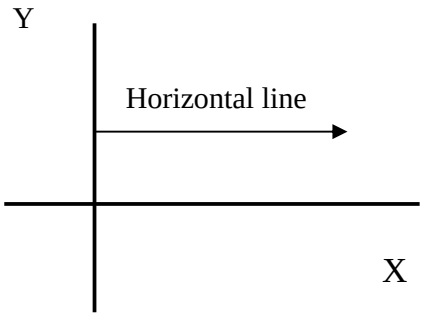
A line segment is displayed by turning on a string of adjacent pixels. In order to draw a line, it is necessary to determine which pixels lie nearest the line and provide the best approximation to the desired line. The accuracy and quality of the displayed line depends on the resolution of the display device. High resolution displays draw lines that look straight and continues and start and end accurately. Lower resolution displays may draw lines with gaps.

2.1 Draw Horizontal Line :

To draw horizontal line, the y value is fixed and the value x varies. The following Pascal codes draw horizontal line from (xstart , y) to (xend , y).

```
For x := xstart to xend do
   plotpoint ( x , y , white ) ;
```

If xstart > xend then (to) in the (for) loop must be replaced by down to.

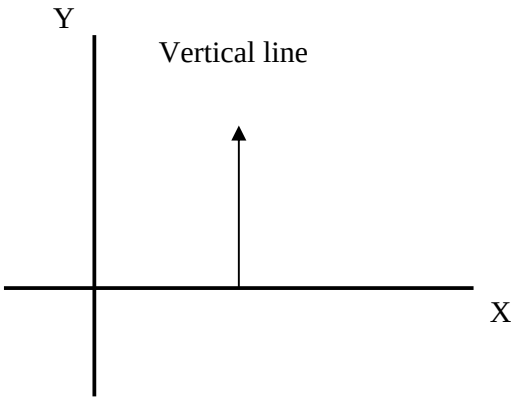


2.2 Draw Vertical Line :

To draw vertical line, the x value is fixed and y value varies. The following Pascal codes draw a vertical line from (x , ystart) to (x , yend).

```
For y := ystart to yend do
   plotpoint ( x , y , white ) ;
```

If ystart > yend then (to) in the (for) loop must be replaced by down to.



2.3 Draw Diagonal Line :

To draw a diagonal line with a slope equal to +1 , we need only repeatedly increment by one unit both the x and y values from the starting to the ending pixels. The following Pascal codes draw a diagonal line:

$$Slope = \frac{y_2 - y_1}{x_2 - x_1} = \frac{\Delta y}{\Delta x}$$

```
x := xstart ;
y := ystart ;
i := 0 ;
while ( x + i ) <= xend do
   begin
     plotpoint  ( x + i , y + i , white ) ;
     i := i + 1 ;
   end ;
```

To draw a line with a slope -1 , replace y+i by y-i in the code.

Example :

Show the tracing to draw a diagonal line from (0 , 0) to (3 , 3).

Solution:

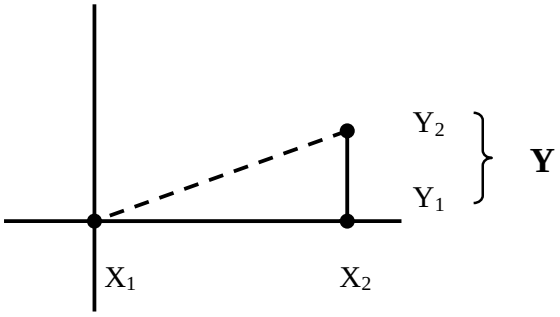
Point 1 (0 , 0) , point 2 (3 , 3).

I	(x+i , y+i)
0	(0,0)
1	(1,1)
2	(2,2)
3	(3,3)
4	stop

2.4 Simple DDA (Digital Differential Analyzer) :

It is a vector generation algorithms (and curve generation) which step along the line (or curve) to determine the pixels which should be turned on by using the numerical method for solving differential equations.

$$y_{i+1} = y_i + \frac{y_2 - y_1}{x_2 - x_1}\Delta x \qquad \ldots\ldots\ldots(1)$$



{ approximate the line length }

if $| x_2 - x_1 | >= | y_2 - y_1 |$ then length = $| x_2 - x_1 |$

else length = $| y_2 - y_1 |$

end { if }

{ select the larger of Δy or Δx to be raster unit }

$$\Delta x = \frac{(x_2 - x_1)}{length}$$

$$\Delta y = \frac{(y_2 - y_1)}{length}$$

{ round the value rather truncate using the sign function makes the algorithm work in all quadrates }

$$x = x_1 + 0.5 \times sign(\Delta x)$$
$$y = y_1 + 0.5 \times sign(\Delta y)$$

```
begin  { main loop }
 for i := 1 to length do
   begin
    plotpoint ( integer ( x ) , integer ( y ) )
```

$$x = x + \Delta x$$
$$y = y + \Delta y$$

```
  end  { for loop }
```

Example :

Draw a line between a points (0 , 0) and (4, 8).

Solution:

$\Delta x = 0.5$ $\Delta y = 1$ $length = 8$ $x = 0.5$ $y = 0.5$

I	Plot	x	y
		0.5	0.5
1	0,0	1	1.5
2	1,1	1.5	2.5
3	1,2	2	3.5
4	2,3	2.5	4.5
5	2,4	3	5.5
6	3,5	3.5	6.5
7	3,6	4	7.5
8	4,7	4.5	8.5
9	stop		

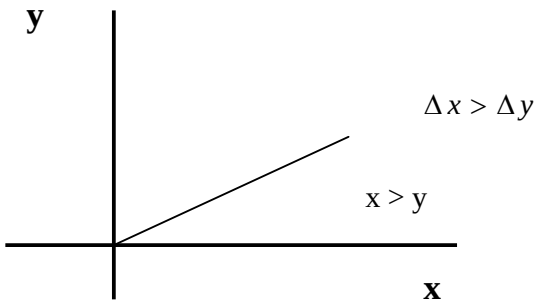
H . W (1) :

1 . Write program to draw a line between points (x_1 , y_1) , (x_2 , y_2) using **DDA** algorithm

2 . Find the points of a line where the first point (5 , 4) and the second point (8 , 6) by using the **DDA** algorithm.

2.5 Bresenham's algorithm :

Bresenham's develop this algorithm that is attractive because it uses only integer arithmetic, and Bresenham's line specially algorithm for the first octant, the line end points (x_1 , y_1) and (x_2 , y_2) assumed not equal.



$x = x_1$

$y = y_1$

$\Delta x = x_2 - x_1$

$\Delta y = y_2 - y_1$

$e = \frac{\Delta y}{\Delta x} - \frac{1}{2}$

begin { the main loop }

for i := 1 to Δx

plot (x , y)

while (e >= 0)

y = y + 1

e = e – 1

end { while }

x = x + 1

e = e + $\Delta y / \Delta x$

end { for }

finish

Example :

By using Bresenham algorithm draw a line by used the following points (0 , 0) and (5 , 3).

Solution:

$x=0$ $y=0$ $\Delta x=5$ $\Delta y=3$

i	plot	X	y	e
		0	0	0.1
1	(0 , 0)		1	-0.9
		1		-0.3
2	(1 , 1)	2		0.3
3	(2 , 1)		2	-0.7
		3		-0.1
4	(3 , 2)	4		0.5
5	(4 , 2)		3	-0.5
		5		0.1
6	stop			

H . W (2) :

1 . Find a point of a line between point (4 , 6) and (10 , 9) using Bresenham's algorithm.

2 . Write a program to draw a line between points (x_1 , y_1) , (x_2 , y_2) using Bresenham's procedure.

2.6 Integer Bresentham algorithm:

Bresenham's algorithm as presented above requires the use of :

1) Floating point arithmetic.
2) Division.

To calculate the slop of a line and to evaluate the error, term the speed of algorithm can be increased by using, integer arithmetic and eliminating the division.

$x = x_1$

$y = y_1$

$\Delta x = x_2 - x_1$

$\Delta y = y_2 - y_1$

$e' = 2\Delta y - \Delta x$

begin { the main loop }

for i = 1 to Δx

plot (x , y)

while (e` >= 0)

y = y + 1

$e' = e' - 2\Delta x$

end { while }

x = x + 1

$e' = e' + 2\Delta y$

end { for }

Finish

H . W (3) :

1 . Find points of a line between (0 , 0) and (5 , 3) using Bresenham's algorithm, without division and without floating point.

2.7 General Bresenham's algorithm :

A full implementation of Bresenham's algorithm requires modification for lines lying the other octant. These can be easily developed by considering the quadrant in which the lines lies and its slope.

$x = x_1$

$y = y_1$

$\Delta x = | x_2 - x_1 |$

$\Delta y = | y_2 - y_1 |$

s_1 = sign ($x_2 - x_1$)

s_2 = sign ($y_2 - y_1$)

{ Interchange Δx and Δy depending on the slope of the line. }

if $\Delta y > \Delta x$ then

temp = Δx

Δx = Δy

Δy = temp

interchange = 1

else

interchange = 0

endif

$e' = 2\Delta y - \Delta x$

{ Main loop }

for i = 1 to Δx

plot (x , y)

while (e' >= 0)

if interchange = 1 then

x = x + s_1

else

y = y + s_2

endif

$e' = e' - 2\Delta x$

end { while }

if interchange = 1 then

y = y + s_2

else

x = x + s_1

endif

$e' = e' + 2\Delta y$

end { for }

Finish

H . W (4) :

1 . Find points of a line between points (0 , 0) , (-2 , -4) using General Bresenham's algorithm.

2 . Write a program to draw a line between points (x_1 , y_1) , (x_2 , y_2) using General Bresenham's procedure.

3 . Write a program to draw a rectangle in any location in the screen without using a line instruction.

4 . Write a program to draw a triangle in any location in the screen without using a line instruction.

5 . Write procedure to draw a polygon in any location in the screen without using a line instruction.

3. CIRCLE DRAWING:

Circles are probably the most used curves to elementary graphics. They often serve as building blocks to generate artistic images. This sheet describes circle drawing algorithm.

3.1 Circle generation Algorithm

For any given point on the circle in clockwise to generation of it there are only three possible selections for the next pixel which best represents the circle horizontally to the right, diagonally downward to the right and vertically downward. These are labeled: m_H, m_D, m_V, respectively in **Figure 4.** The algorithm chooses the pixel which minimizes the square of the distance between one of these pixels and the true circle, i.e. the minimum of:

$$m_H = \left|(x_i + 1)^2 + (y_i)^2 - R^2\right|$$

$$m_D = \left|(x_i + 1)^2 + (y_i - 1)^2 - R^2\right|$$

$$m_V = \left|(x_i)^2 + (y_i - 1)^2 - R^2\right|$$

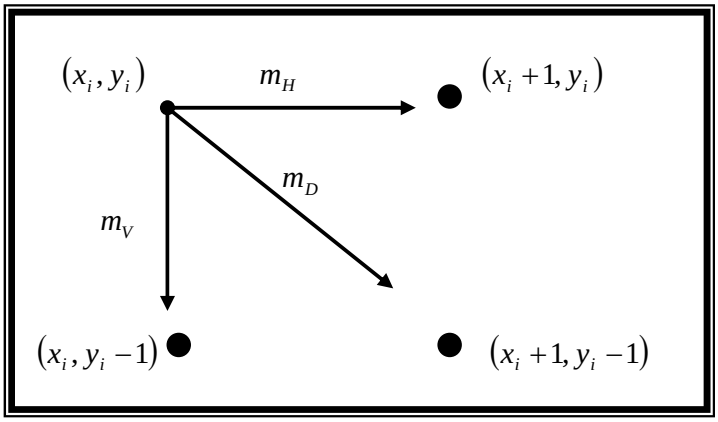


Figure 4: First quadrant pixel selections.

The algorithm

X=xC

Y=YC + R

While (y >= 0)

Plot (x,y)

Eh = |(x+1)²+ y²- R²|

Ed = |(x+1)² + (y-1)² - R²|

Ev = |x² +(y-1)²-R²|

Min = minimum (Eh, Ed, Ev)

If min = Eh then X=X+1: go to 1

If min = Ed then X=X+1:Y=Y-1: go to 1

If min = Ev then y = y - 1

1 Endwhile

Example : Draw a circle by using circle equation, when C=0, R= 8.

Solution:

X	Y	Plot	Eh	Ed	Ev	MIN
0	**8**	**0,8**	**1**	**14**	**15**	**Eh=1**
1		**1,8**	**4**	**11**	**14**	**Eh=4**
2		**2,8**	**9**	**6**	**11**	**Ed=6**
3	**7**	**3,7**	**1**	**12**	**19**	**Eh=1**
4		**4,7**	**10**	**3**	**12**	**Ed=3**
5	**6**	**5,6**	**8**	**3**	**14**	**Ed=3**
6	**5**	**6,5**	**10**	**1**	**12**	**Ed=1**
7	**4**	**7,4**	**16**	**9**	**6**	**Ev=6**
7	**3**	**7,3**	**6**	**24**	**11**	**Ev=6**
8	**3**	**8,3**	**26**	**21**	**4**	**Ev=4**
8	**2**	**8,2**	**21**	**18**	**1**	**Ev=1**
8	**1**	**8,1**	**18**	**17**	**0**	**Ev=0**
8	**0**	**8,0**				

3.2 Circle Generation – Bresenham`s Algorithm

One of the most efficient and easiest to drive of the circle algorithms is due to Bresenham. To begin, note that only one octant of the circle need be generated. The other parts can be obtained by successive reflections. This is illustrated in **Fig. 5.** If the first octant (0 to 45° ccw) is generated, the second octant can be obtained by reflection through the line y=x to yield the first quadrant. The results in the first quadrant are reflected through the line x=0 to obtain those in the second quadrant.

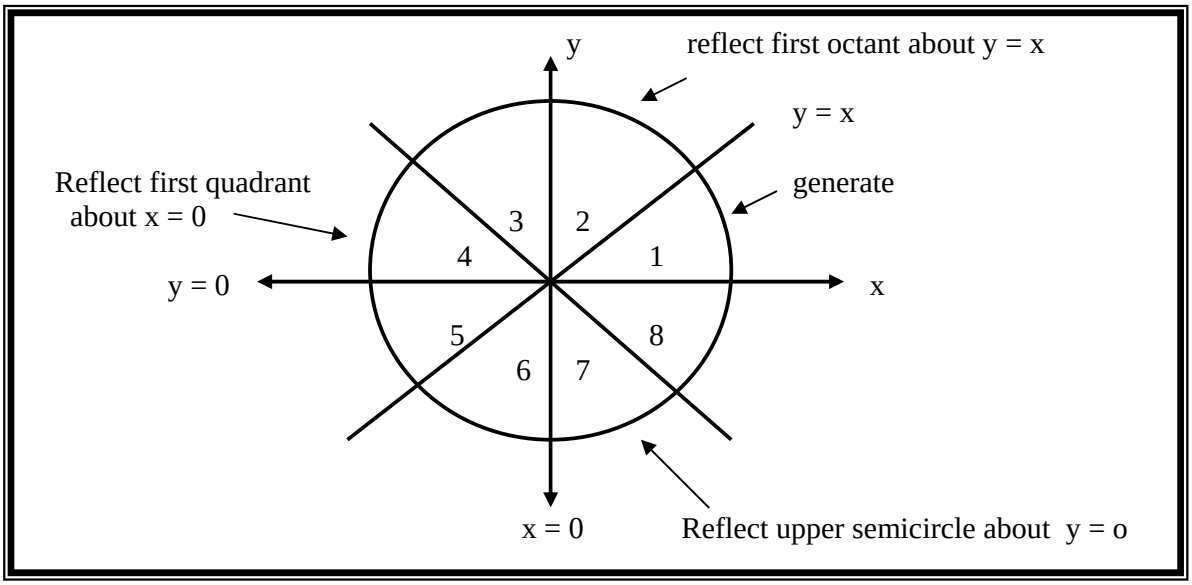


Figure 5: Generation of a complete circle from the fist octant

The combined result in the upper semicircle are reflected through the line y=0 to complete the circle. Bresenham`s Algorithm is consider the first quadrant of an origin- centered circle. If the algorithm begins at x=0 , y=R, then for clockwise generation of the circle y is a monotonically decreasing function of x in the first quadrant. Here the clockwise generation starting at x=0, y=R is chosen. The center of the circle is (0,0). See **Figure (6).**

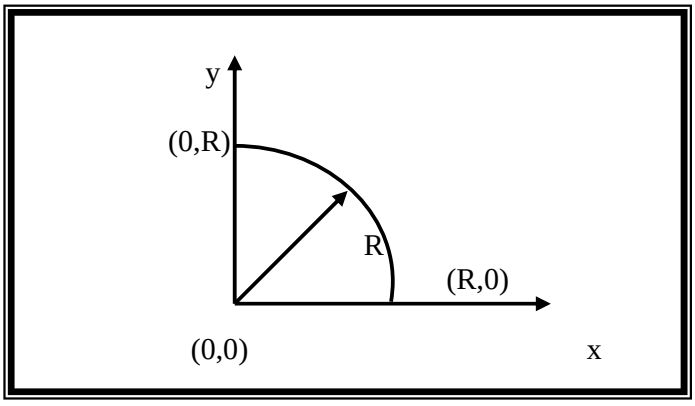


Figure 6: First quadrant of a circle.

The circle equation, when the center (a,b), and the radius R, is:

$$(x-a)^2+(y-b)^2=R^2 \qquad \cdots\cdots\cdots\cdots(1)$$

And when the center of this circle is the origin (0, 0), then the equation:

$x^2+y^2=R^2$ Because of a, b = 0.

The different between the square of the distance from the center of the circle to the diagonal pixel at (x_i+1, y_i-1) and the distance to a point on the circle R^2 is:

$$\Delta_i=(x_i+1)^2+(y_i-1)^2-R^2 \qquad \cdots\cdots\cdots\cdots(2)$$

If $\Delta_i<0$ then find $\delta=m_H-m_D$ $\cdots\cdots\cdots\cdots(3)$

If $\Delta_i=0$ then find $\delta=m_D$ $\cdots\cdots\cdots\cdots(4)$

If $\Delta_i>0$ then find $\delta=m_D-m_V$ $\cdots\cdots\cdots\cdots(5)$

Bresenham`s suggestions that we can find Δ_n depending on Δ_O :

1- Horizontal Case:

$$\begin{aligned}\Delta_n &= (x_n+1)^2+(y_n-1)^2-R^2\\ &= (x_n)^2+(y_n-1)^2-R^2+2x_n+1\\ &= (x_\circ+1)^2+(y_\circ-1)^2-R^2+2x_n+1\\ \Delta_n &= \Delta_\circ+2x_n+1\end{aligned}$$

$$x_n=x_\circ+1, \quad y_n=y_\circ$$

2- Diagonal Case:

$$\begin{aligned}\Delta_n &= (x_n+1)^2+(y_n-1)^2-R^2\\ &= (x_n)^2+(y_n)^2-R^2+2x_n-2y_n+2\\ &= (x_\circ+1)^2+(y_\circ-1)^2-R^2+2x_n-2y_n+2\\ \Delta_n &= \Delta_\circ+2x_n-2y_n+2\end{aligned}$$

$$x_n=x_\circ+1, \quad y_n=y_\circ-1$$

3- Vertical Case:

$$\Delta_n = (x_n + 1)^2 + (y_n - 1)^2 - R^2$$
$$= (x_n + 1)^2 + (y_n)^2 - R^2 - 2y_n + 1$$
$$= (x_\circ + 1)^2 + (y_\circ - 1)^2 - R^2 - 2y_n + 1$$
$$\Delta_n = \Delta_\circ - 2y_n + 1$$

$$x_n = x_\circ \, , y_n = y_\circ - 1$$

By simplified the equation (3), (5) we get:

$$\delta = 2\Delta + 2y - 1$$
$$\bar{\delta} = 2\Delta - 2x - 1$$

When x=0, y=R, and use these values in equation (2):

$$\Delta = 1 + (R-1)^2 - R^2 \quad \Rightarrow \Delta = 2(1-R)$$

Bresenham`s incremental circle algorithm for the first quadrant, all variables are assumed integer:

{ initialize the variables }

$x_i = 0$
$y_i = R$
$\Delta_i = 2(1 - R)$

Limit = 0

1 plot (x_i, y_i)

If $y_i \leq$ limit then 4

If $\Delta_i \langle 0$ then 2

If $\Delta_i \rangle 0$ then 3

If $\Delta_i = 0$ then 20

2 $\delta = 2\Delta_i + 2y_i - 1$

If $\delta \leq 0$ then 10

If $\delta \rangle 0$ then 20

3 $\bar{\delta} = 2\Delta_i - 2x_i - 1$

If $\bar{\delta} \leq 0$ then 20

If $\bar{\delta} \rangle 0$ then 30

Perform the moves

Move m_H

10 $x_i = x_i + 1$
$\Delta_i = \Delta_i + 2x_i + 1$

Goto 1

Move m_D

$x_i = x_i + 1$
20 $y_i = y_i - 1$
$\Delta_i = \Delta_i + 2x_i - 2y_i + 2$

Goto 1

Move m_V

30 $y_i = y_i - 1$
$\Delta_i = \Delta_i - 2y_i + 1$

Goto 1

4 finish.

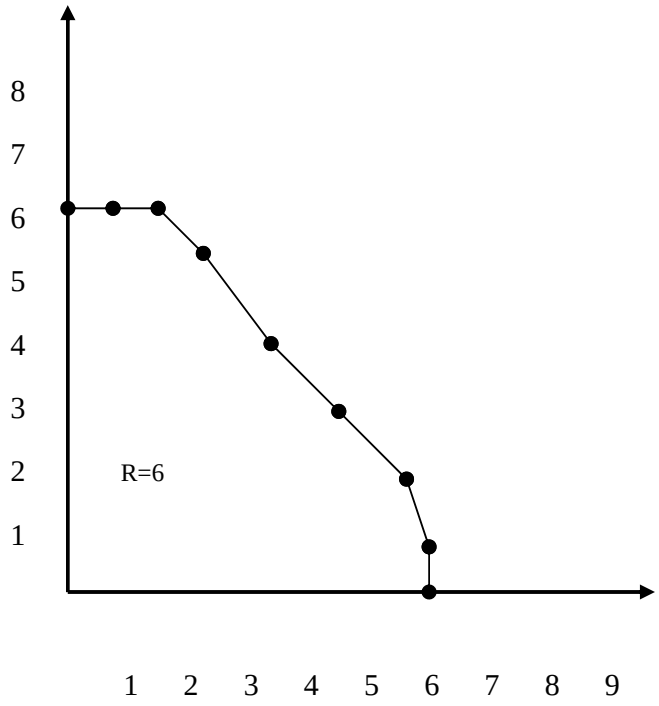
Example : find the pixels in first quadrant of circle where R=6, center =(0,0) by using Bresenham`s algorithm, and draw them.

Solution:

x = 0 , y = r = 6 , $\Delta_i = -10$, limit = 0

plot	Δ_i	δ	$\bar{\delta}$	**x**	**y**
(0,6)	-10	-	-	0	6

(1,6)	-7	-9	-	1	6
(2,6)	-2	-3	-	2	6
(3,5)	-4	7	-	3	5
(4,4)	-2	1	-	4	4
(5,3)	4	3	-	5	3
(6,2)	14	-	-3	6	2
(6,1)	13	-	15	6	1
(6,0)	14	-	13	6	0



3 . 3 Circle Drawing By Using Circle Equation

$(\chi-\chi c)^2 + (y-yc)^2 = R^2$

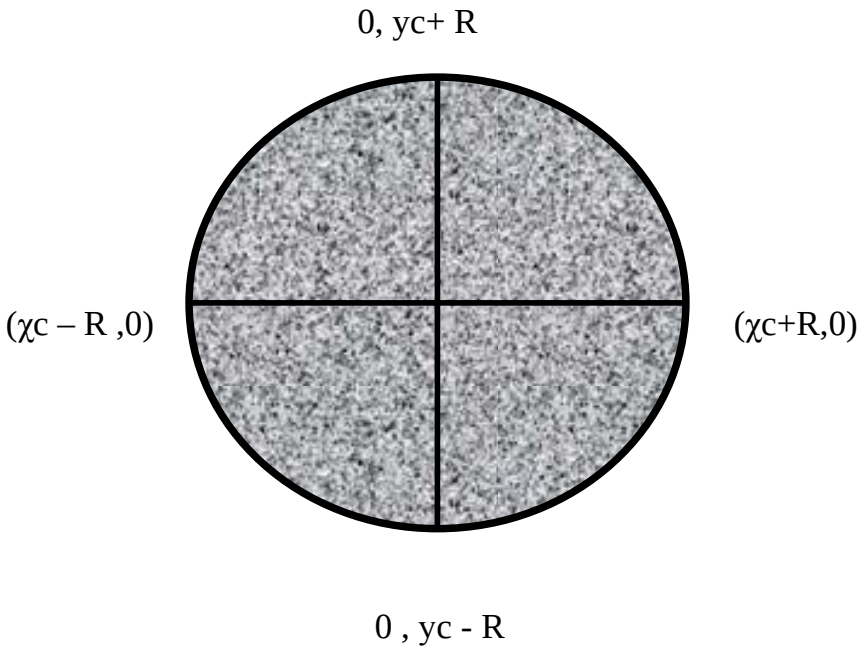
χ , y :- point on the boundary

χc , yc :- the center of the circle

R :- ½ Radios

From equation above we can find that

$$У = yc \pm \sqrt{R^2 - (\chi - \chi c)^2}$$



- لكل نقطة في X لها قيمتين في Y (+ ، -)
- بعبارة أخرى نستطع استخدام for بالشكل التالي :

For X : = (Xc – R) to (Xc + R)

$$У := Yc \pm \sqrt{R^2 - (X - Xc)^2}$$

Example : Draw a circle by using circle equation, when R=5, Xc=0 , Yc=0.

Solution:

For X: = -R To + R

$$У = \sqrt{R^2 - X^2}$$

For x:= -5 to +5

$$Y = \sqrt{5^2 - X^2}$$

X	Y	Plot
-5	0	(-5,0)
-4	-3,+3	(-4,-3),(-4,3)
-3	-4,+4	(-3,-4),(-3,4)
-2		
-1		
0	-5,+5	(0,-5),(0,5)
1		
2		
3	-4,+4	(3,-4),(3,4)
4	-3,+3	(4,-3),(4,3)
5	0	(5,0)

3 . 4 Ellipse

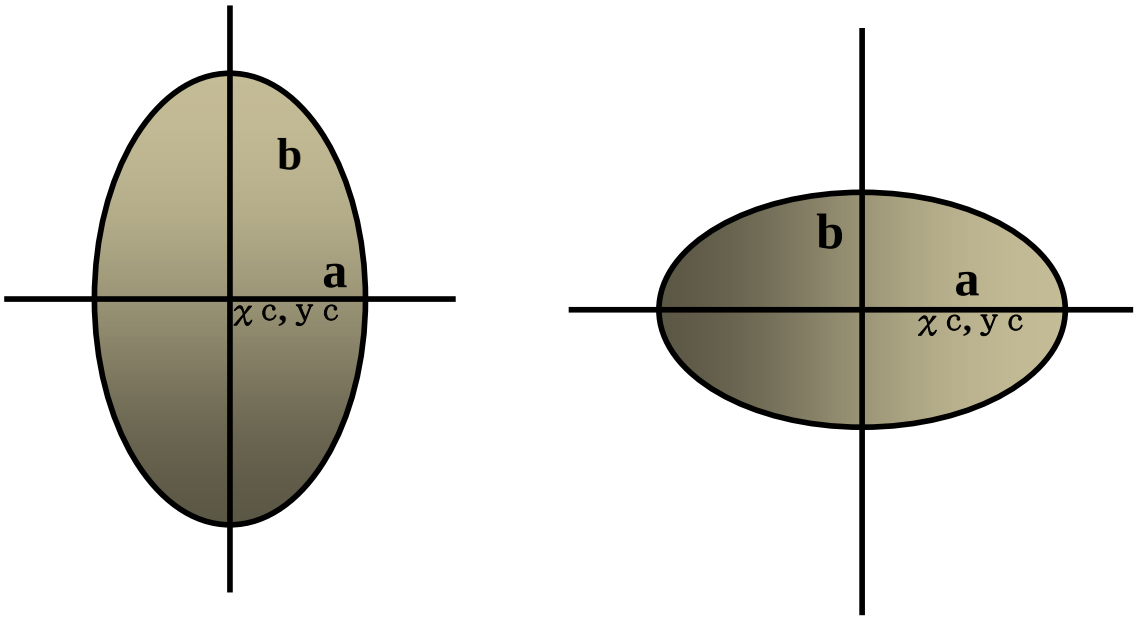
- An ellipse is a variation of a circle.
- Stretching a circle in one direction produce an ellipse
- We shall examine only ellipse that are stretched in the **x** or **y** direction.
- The **polar equation** for this type of ellipse centered at (χ_c, y_c) are:-

$$\left.\begin{array}{l} \chi = \chi_c + a * \cos(\varphi) \end{array}\right\}$$

$$y = y_c + b * \sin(\varphi) \longrightarrow (1)$$

Where: the angle φ assumes values between 0 to 2π

Radius have major axis **a** , minor axis **b**.



The values of (a) and (b) effects the shape of the ellipse :

- If b>a the ellipse is longer in the y -direction.
- If a>b the ellipse is longer in the χ - direction.

The ellipse can be drawn using four-points summitry:

- If (c,d) lies on the ellipse, so do the points (-c,d),(c,-d) and (-c,-d).

The following incremental equations for an ellipse are derived from equation (1)

$X_2 = X_1 * \cos(d\varphi) - (a/b)\, Y_1 * \sin(d\varphi)$

$Y_2 = Y_1 * \cos(d\varphi) + (b/a)\, X_1 * \sin(d\varphi)$

H.W (5):

1 . Draw a circle by using Circle generation Algorithm , when C=0, R= 8.

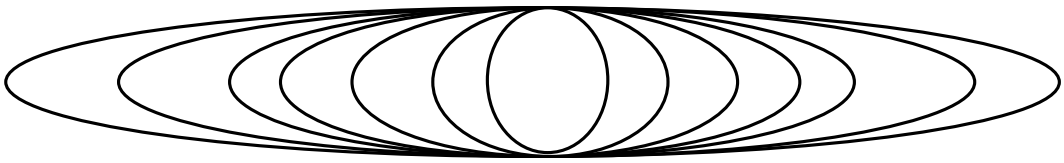
2. write a program to draw a quadrant of circle where radius =R, center=(0,0), by using Bresenham`s incremental circle algorithm.

3. Find the pixels of quadrant circle where center=(0,0) and Radius=10, by using Bresenham`s incremental circle algorithm. Then draw these pixels.

4 . Draw a circle by using circle equation when R=8, Xc=0 , Yc=0.

5. write program to draw ellipse when Xc=320 , Yc=240, a=80, b=30.

6. write program to draw the following figure (without using ellipses instruction) when xc=300, yc=200, a=130, b=30.



TWO DIMENSIONAL

GEOMETRIC TRANSFORMATIONS

Introduction:

Geometric transformations provide a mean by which an image can be constructed or modified. The transformations we examine in this sheet are translation, scaling, rotations, reflection and sharing.

The advantage of used the translation: -

Details appear more clearly.

Reduces a picture more of if is visible.

Change the scale of a symbol.

Rotate it through some angle.

1. Translation:

a point (x , y) is translated to a new position (x` , y`) by move it H units in the horizontal direction and V units in the vertical direction (figure (7)).

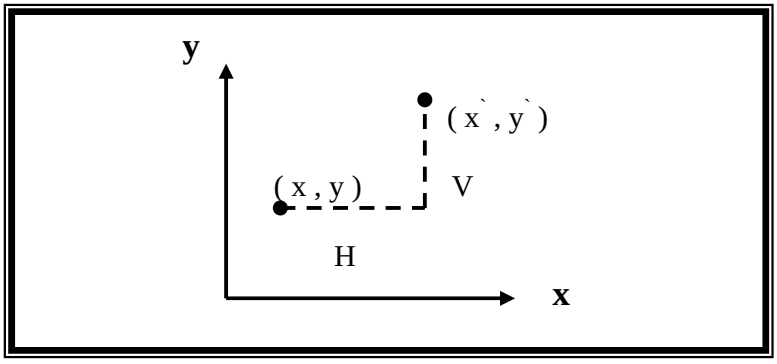


Figure (7) horizontal and vertical displacement

Mathematically this can be represented as:

$$X` = X + H$$

$$Y` = Y + V$$

The H and V represent the horizontal and vertical displacement or distance that the point has moved. If H is positive, the point moves to the right, if H is negative the point moves to the left, similarly, a point V moves the point up, a

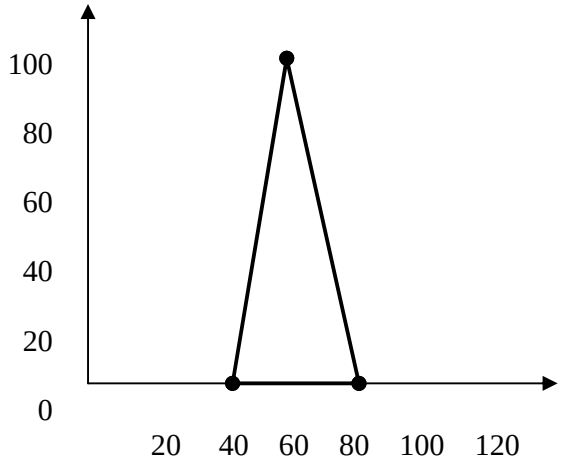
negative V moves it down. Remember that to move object we must translate every point describing the object.

To translate an object in an image we must translate every point defining the object. All point, are displaced the same distance and the object is draw using these transformed points.

Example :

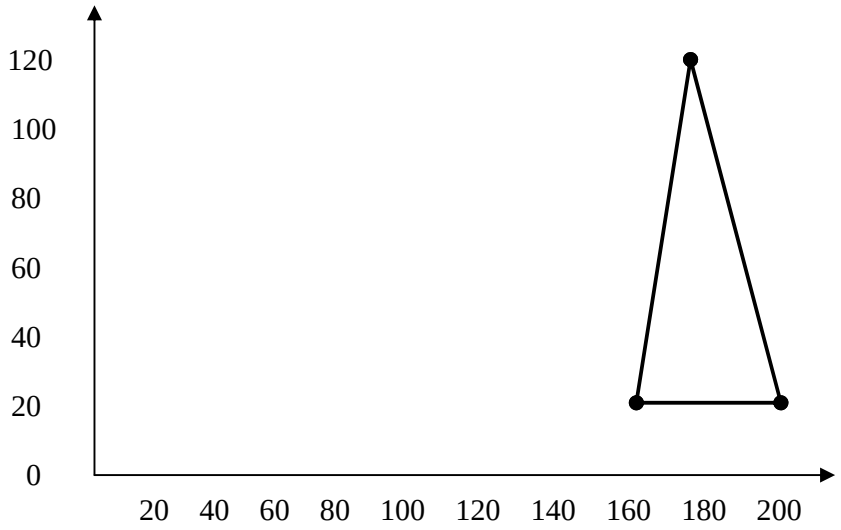
Consider a triangle defined by it three vertices (40 , 0), (80 , 0), (60 , 100) be translated 120 units to the right and 20 units up.

Solution:



H = 120 , V = 20 . The new vertices are:

(160 , 20) , (200 , 20) , (180 , 120)



$$\begin{bmatrix} 40 & 0 \\ 80 & 0 \\ 60 & 100 \end{bmatrix} \xrightarrow[(120,20)]{rightup} \begin{bmatrix} 160 & 20 \\ 200 & 20 \\ 180 & 120 \end{bmatrix}$$

before after

H . W (6) :

1 . Write procedure to translated any picture (up , down , right , left).

2 . Write program to draw a polygon and using translated procedure to translate it's in any direction.

3 . Consider a triangle defined by it three vertices (20 , 0) , (60 , 0) , (40 , 100) be translated 20 units to the left.

4 . Consider a triangle defined by it three vertices (40 , 0) , (100 , 0) , (60 , 100) be translated 40 units to the left and 40 units down.

2. Rotation:

Another useful transformation is the rotation of an object about a specified pivot point. After the object has been rotated, it is still the same distance away from the pivot point, however its orientation has been changed. It is possible to rotate one or clockwise (negative angle) or counterclockwise (positive angle) direction.

Any point (x , y) can be represented by its radial distance, **r**, from the origin and its angle, ϕ , off the x – axis as show in figure (8).

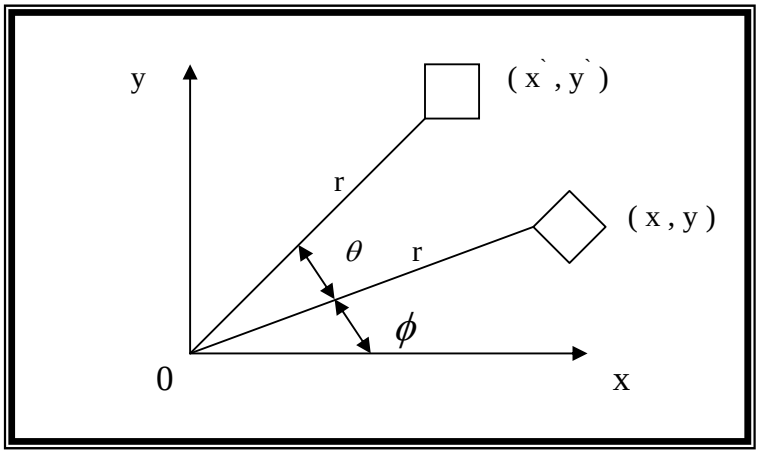


Figure (8) rotation about origin

$$\left.\begin{aligned} x &= r * \cos(\phi) \\ y &= r * \sin(\phi) \end{aligned}\right\} \quad \text{----------(1)}$$

If (x , y) is rotated an angle θ in the counterclockwise direction. The transformed point (x` , y`) is represented as :

$$\left.\begin{aligned} x &= r * \cos(\theta + \phi) \\ y &= r * \sin(\theta + \phi) \end{aligned}\right\} \quad (2)$$

Using the laws of sines and cosines from trigonometry, the equation (2) become:

$$\left.\begin{aligned} x` &= r * \cos(\phi) * \cos(\theta) - r * \sin(\phi) * \sin(\theta) \\ y` &= r * \sin(\phi) * \cos(\theta) + r * \cos(\phi) * \sin(\theta) \end{aligned}\right\} \quad \text{-------.(3)}$$

From the definition of x and y, the equation (3) reduce to:

$$\left.\begin{aligned} x` &= x * \cos(\theta) - y * \sin(\theta) \\ y` &= y * \cos(\theta) + x * \sin(\theta) \end{aligned}\right\} \quad \text{----------(4)}$$

To rotate an object an angle (θ) about a pivot point (x_p , y_p) other than the origin, we perform the following three steps:

Step 1: Translate

Translate the pivot point (x_p , y_p) to the origin. Every point (x , y) defining the object is translated to a new point (x` , y`) where:

$x` = x - x_p$

$y` = y - y_p$

Step 2: Rotate

Use these translated points (x` , y`), θ degree about the origin to obtain the new point (x`` , y``) where:

$x`` = x` * \cos(\theta) - y` * \sin(\theta)$

$y`` = y` * \cos(\theta) + x` * \sin(\theta)$

By substituting for x` and y` :

$x`` = (x - x_p) * \cos(\theta) - (y - y_p) * \sin(\theta)$

$y`` = (y - y_p) * \cos(\theta) + (x - x_p) * \sin(\theta)$

Step 3: Translate

Translate the center of rotation back to the pivot point (x_p , y_p).

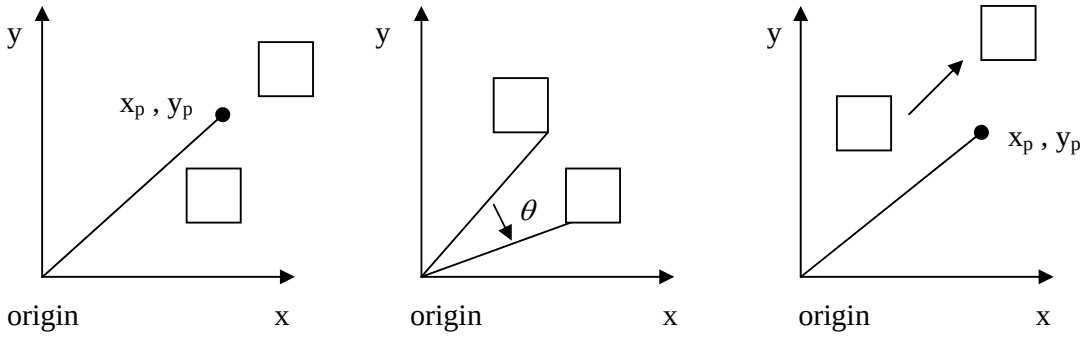
$x''' = x'' + x_p$
$y''' = y'' + y_p$

By substituting for x'' and y'' :

$x''' = (x - x_p) * \cos (\theta) - (y - y_p) * \sin(\theta) + x_p$
$y''' = (y - y_p) * \cos (\theta) + (x - x_p) * \sin(\theta) + y_p$

So to rotate a point (x , y) through a clockwise angle θ about the origin of the coordinate system we write:-

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



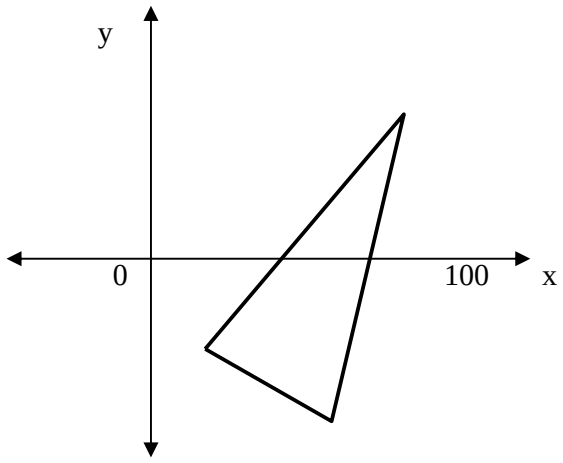
Example :

The triangle (10 , 0) , (30 , 0) , (50 , 80) rotate 45^o clockwise about the origin.

Solution:

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

(7.07 , - 7.07) , (21.21 , - 21.21) , (91.93 , 21.12)



Rotate about the origin

H . w (7) :

1 . Rotate the triangle (10 , 0) , (30 , 0) , (50 , 80) 45^o counterclockwise about the origin.

2 . Rotate the triangle (7 , 8) , (4 , 4) , (10 , 5) 90^o counter clockwise about the point (7 , 8).

3 . Rotate the above triangle 90^o clockwise about the point (4 , 4).

4 . Write program which rotate a polygon.

a) Counter clockwise about the origin.
b) Counter clockwise about the pivot point.
c) Clockwise about the origin.
d) Clockwise about the pivot point.

5 . Write an equation to rotate any picture clockwise about the pivot point.

3. Scaling :

An object can be made by larger by increasing the distance between the points describing the object. In general, we can change the size of an object, or the entire image, by multiplying the distance between points by an enlargement or reduction factor. This factor is called the **" scaling factor "**, and the operation that changes the size is called **scaling**. If the scaling is greater than 1, the object is enlarge, if the factor is less than 1, the object is made smaller, a factor of 1 has no effect on the object. Whenever scaling is performed, there is one point that remains at the same location. This is called the **fixed point** of the scaling transformation.

$x` = x * S_x$

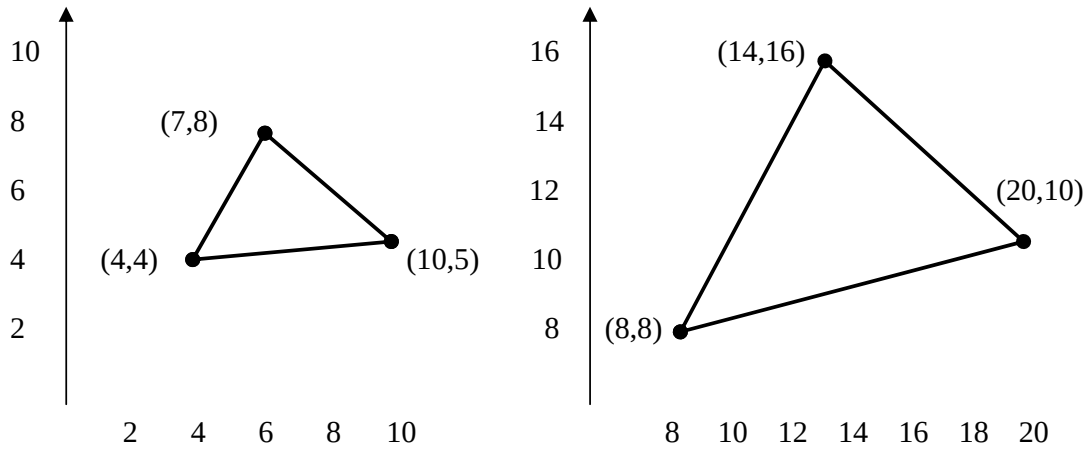
$y` = y * S_y$

Example :

Scale the triangle (4 , 4) , (7 , 8) , (10 , 5) by $S_x = 2$ and $S_y = 2$, about the origin point.

Solution:

The new points are:
(8 , 8) , (14 , 16) , (20 , 10)



H . w (8) :

1 . Magnify the triangle (0 , 0) , (8 , 10) , (12 , 4), 4 times its size, about the origin point.

2 . Magnify the above triangle 1 / 2 its size.

3 . Magnify the triangle (0 , - 3) , (- 6 , - 7) , (6 , - 7), 3 times its size, about the point (0,-3).

4 . Write scale procedure to magnify any image.

5 . Write an equation to magnify any picture without translation.

Matrix representation of transformations :

We are going to use 3×3 matrices, it is necessary to convert the two – dimensional **homogeneous vector**. This is accomplished by associative (x , y) with the homogeneous row vector [x y 1]. After multiplying this vector by a 3×3 matrix, we obtain another homogeneous row vector, one having three components with the last component equal to 1 : [x_1 y_1 1]. The first two

term in this vector are the coordinate pair (x_1 , y_1) which is the transform of (x , y).

We are now ready to give the matrix representation of the transformation: Translation, Scaling, Rotation.

Translation:

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

Scaling:

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

Rotation:

(a) Counterclockwise direction :

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

(b) Clockwise direction :

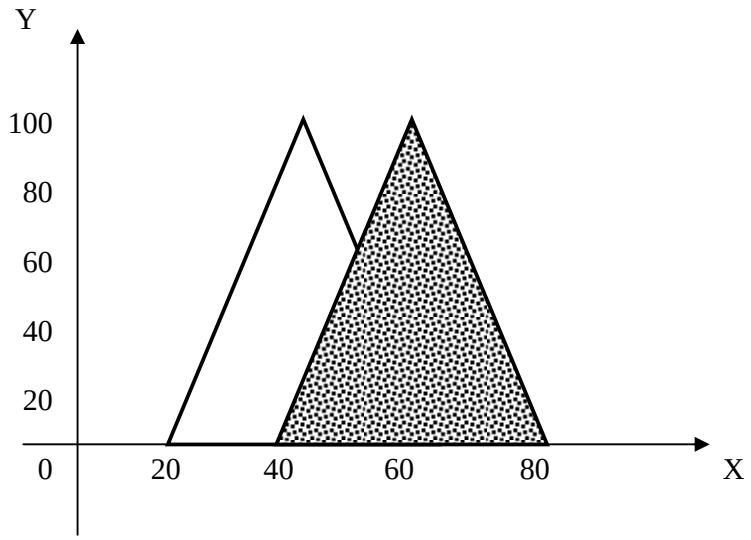
$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \times \begin{bmatrix} \cos(x) & -\sin(x) & 0 \\ \sin(x) & \cos(x) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Example :

Consider a triangle defined by it three vertices (40 , 100), (20 , 0), (60 , 0) be translated 20 units to the right, using matrix representation.

Solution:

$$\begin{bmatrix} 40 & 100 & 1 \\ 20 & 0 & 1 \\ 60 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 20 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 60 & 100 & 1 \\ 40 & 0 & 1 \\ 80 & 0 & 1 \end{bmatrix}$$



Example :

Rotate the triangle (7 , 8) , (4 , 4) , (10 , 5) 90^o counterclockwise about the point (7 , 8), using matrix representation.

Solution:

1) Translate:-

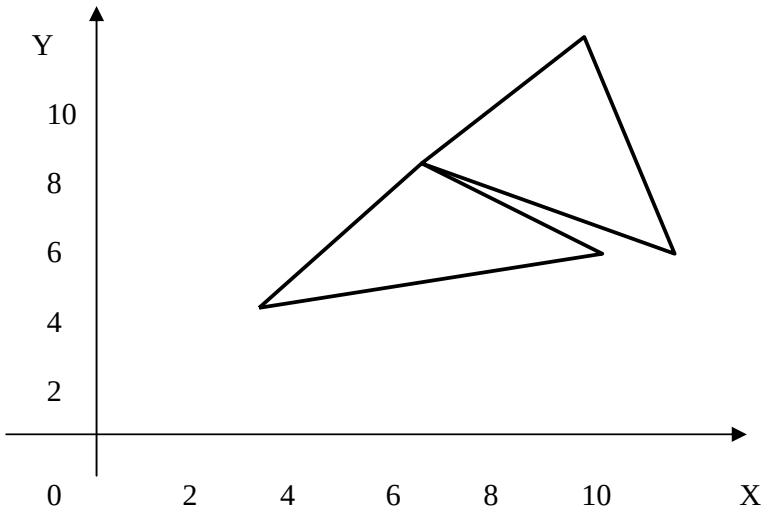
$$\begin{bmatrix} 7 & 8 & 1 \\ 4 & 4 & 1 \\ 10 & 5 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -7 & -8 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ -3 & -4 & 1 \\ 3 & -3 & 1 \end{bmatrix}$$

2) Rotate: -

$$\begin{bmatrix} 0 & 0 & 1 \\ -3 & -4 & 1 \\ 3 & -3 & 1 \end{bmatrix} \times \begin{bmatrix} \cos(90) & \sin(90) & 0 \\ -\sin(90) & \cos(90) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 4 & -3 & 1 \\ 3 & 3 & 1 \end{bmatrix}$$

3) Translate: -

$$\begin{bmatrix} 0 & 0 & 1 \\ 4 & -3 & 1 \\ 3 & 3 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 7 & 8 & 1 \end{bmatrix} = \begin{bmatrix} 7 & 8 & 1 \\ 11 & 5 & 1 \\ 10 & 11 & 1 \end{bmatrix}$$

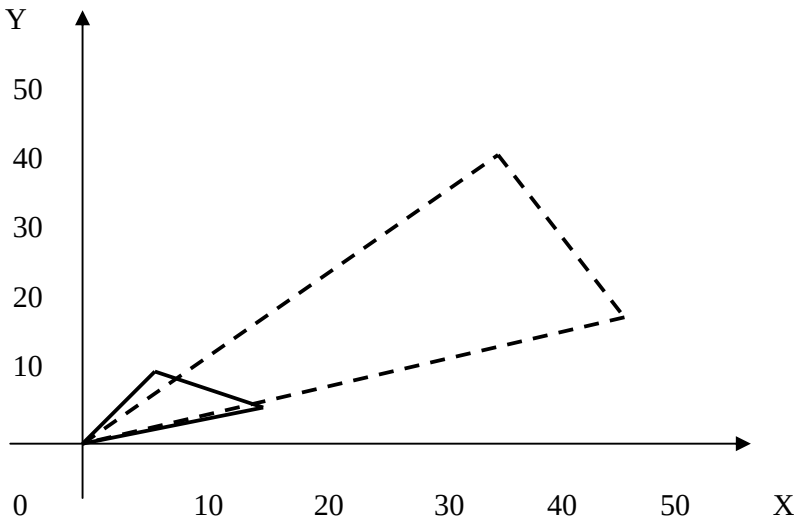


Example :

Magnify the triangle (0 , 0) , (8 , 10) , (12 , 4), 4 times its size, using matrix representation.

Solution:

$$\begin{bmatrix} 0 & 0 & 1 \\ 8 & 10 & 1 \\ 12 & 4 & 1 \end{bmatrix} \times \begin{bmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 32 & 40 & 1 \\ 48 & 16 & 1 \end{bmatrix}$$



4. Reflection :

It is a transformation that produced a mirror image of an object; the mirror image is generated relative to an axis of reflection.

There are different types of reflection:

1 - Reflection about X – axis:

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

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

2 - Reflection about Y – axis:

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

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

3 - Reflection about the origin (0, 0):

$$x = -x$$
$$y = -y$$

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

4 - Reflection about the line y = x :

$x = y$

$y = x$

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

5 - Reflection about the line y = - x :

$x = -y$

$y = -x$

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

Example :

Reflect the shape (20, 70), (40, 50), (60, 70), (40, 90), about:

1- X – axis 2- Y- axis

3- origin (0,0) 4- y = x 5- y = -x , by used matrix representation, and draw the result.

Solution:

1- X – axis:

$x' = x$

$y' = -y$

$$\begin{bmatrix} 20 & 70 & 1 \\ 40 & 50 & 1 \\ 60 & 70 & 1 \\ 40 & 90 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 20 & -70 & 1 \\ 40 & -50 & 1 \\ 60 & -70 & 1 \\ 40 & -90 & 1 \end{bmatrix}$$

2- Y- axis:

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

$$\begin{bmatrix} 20 & 70 & 1 \\ 40 & 50 & 1 \\ 60 & 70 & 1 \\ 40 & 90 & 1 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -20 & 70 & 1 \\ -40 & 50 & 1 \\ -60 & 70 & 1 \\ -40 & 90 & 1 \end{bmatrix}$$

3- origin (0,0):

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

$$\begin{bmatrix} 20 & 70 & 1 \\ 40 & 50 & 1 \\ 60 & 70 & 1 \\ 40 & 90 & 1 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -20 & -70 & 1 \\ -40 & -50 & 1 \\ -60 & -70 & 1 \\ -40 & -90 & 1 \end{bmatrix}$$

4- y = x:

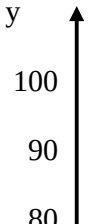
$$x' = y$$
$$y' = x$$

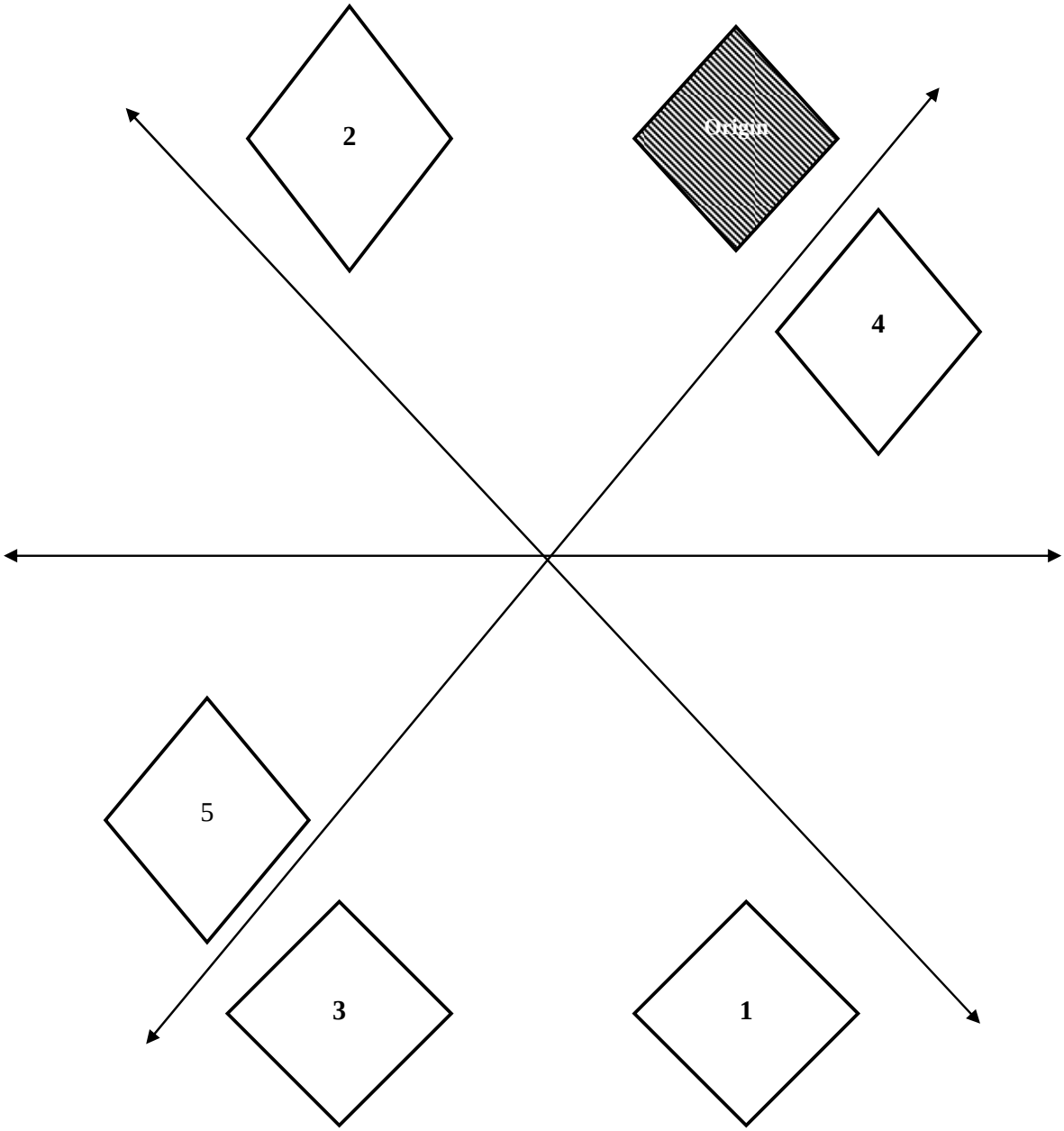
$$\begin{bmatrix} 20 & 70 & 1 \\ 40 & 50 & 1 \\ 60 & 70 & 1 \\ 40 & 90 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 70 & 20 & 1 \\ 50 & 40 & 1 \\ 70 & 60 & 1 \\ 90 & 40 & 1 \end{bmatrix}$$

5- y = -x:

$$x = -y$$
$$y = -x$$

$$\begin{bmatrix} 20 & 70 & 1 \\ 40 & 50 & 1 \\ 60 & 70 & 1 \\ 40 & 90 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -70 & -20 & 1 \\ -50 & -40 & 1 \\ -70 & -60 & 1 \\ -90 & -40 & 1 \end{bmatrix}$$





5. Shearing :

A **shearing** transformation produces a distortion of an object or the entire image. There are two types of shearing (Y-shear & X-shear).

1) Y-shear transforms the point (x,y) to the point ($x^{'},y^{'}$) , where :-

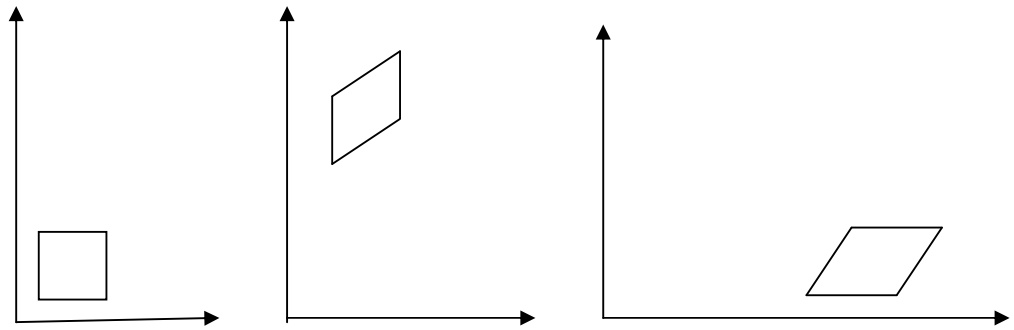
$$X' = X$$
$$Y' = Shy * X + Y \qquad \text{Shy} \neq 0$$

A Y-shear moves a vertical line up or down, depending on the sign of the shear factor Shy. A horizontal line is distorted into a slanted line with slop Shy.

2) X-shear has the opposite effect. That is, the point ($x^{'},y^{'}$) is transformed to the point (x,y) , where :-

$$X' = X + Shx * Y \qquad \text{Shx} \neq 0$$
$$Y' = Y$$

A vertical line becomes slanted line with slop Shx and the horizontal lines are shifted to the right or left, depending on the sign of Shx.



Before Y-shear X-shear

Matrix Representation of Shearing

1:- Y-shear

$$[\,x'\ y'\,] = [\,x \quad y\,]\begin{bmatrix} 1 & shy \\ 0 & 1 \end{bmatrix}$$

2x2

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

3x3

2:- X-shear

$$[\,x'\ y'\,] = [\,x \quad y\,]\begin{bmatrix} 1 & 0 \\ shx & 1 \end{bmatrix}$$

2x2

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

3x3

Example :

Shear the object (1,1), (3,1),(1,3), (3,3) with

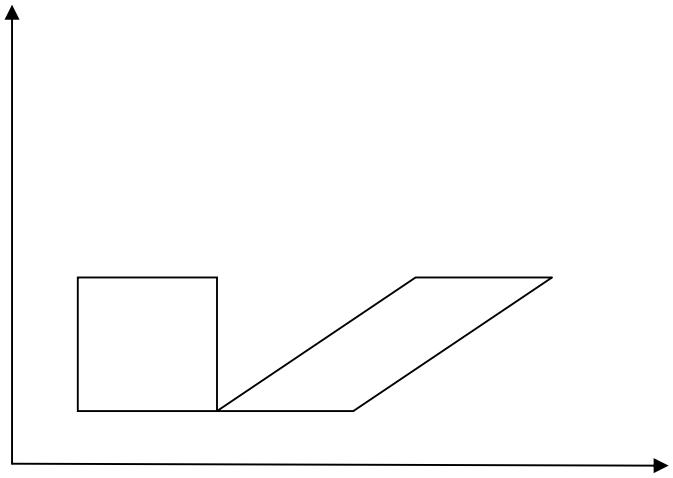
a: shx = 2

b: shy =2

solution:

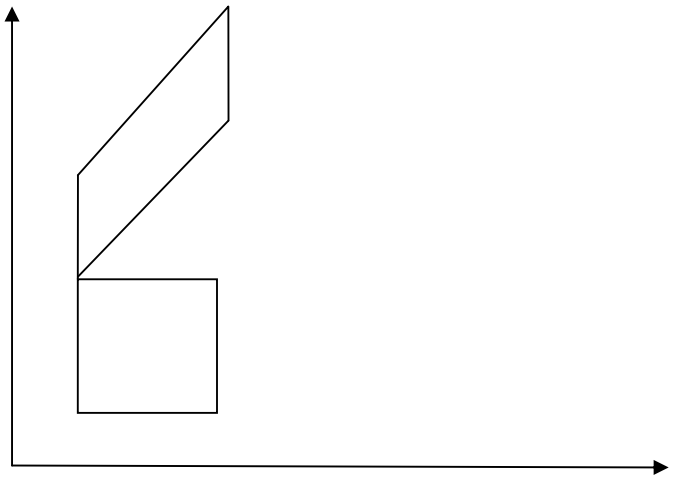
a: shx

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



b: shy

$$\begin{bmatrix} 1 & 1 & 1 \\ 3 & 1 & 1 \\ 1 & 3 & 1 \\ 3 & 3 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 3 & 1 \\ 3 & 7 & 1 \\ 1 & 5 & 1 \\ 3 & 9 & 1 \end{bmatrix}$$



H . W (9):

1 . Magnify the triangle (0 , 0) , (10 , 12) , (14 , 6), 3 times its size, using matrix representation.

2 . Rotate the above triangle 90^{o} clockwise about the point (10 , 12), using matrix representation.

3 . Translated the shape above 20 units to the left, and 10 units up. Using matrix representation.

4 . Reflect the shape (2, 2), (4, 4), (6, 2), about:

a- X – axis b- Y- axis

c- origin (0,0) d- y = x e- y = -x , then draw the result.

5. Shear the shape above with: a- shx = 4

b- shy = 4.

TWO – DIMENSIONAL VIEWING TRANSACTION AND CLIPPING WINDOWS AND VIEW PORTS:

The method for selecting and enlarging portions of a drawing enclosed in a rectangular region is called ***Widowing***. The rectangular region is called a ***window***.

The technique for not showing that part of the drawing which one is not interested in is called ***clipping***.

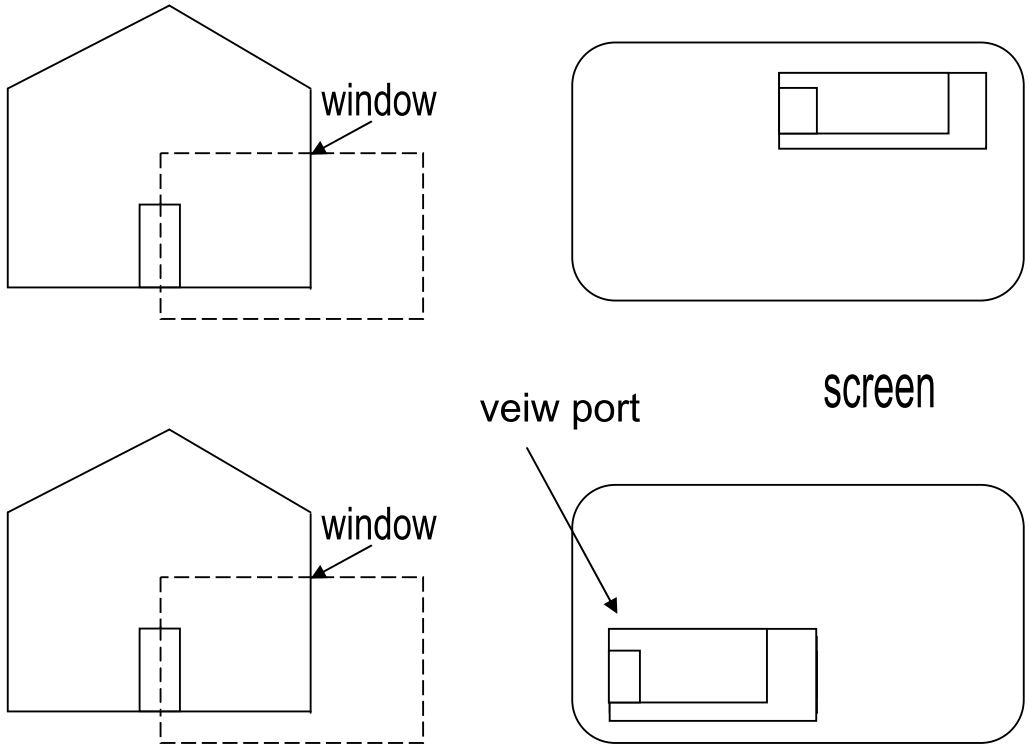
Clipping is a process which divided each element of the picture into its risible and invisible portions allowing the invisible portion to be discarded.

- If we imagine a box about a portion of the object so we could display what is enclosed in the box such a box called a ***Window***.

- If we do not wish to use the entire screen for display, we can imagine a box on the screen and have the image confined to that box such a box in the screen space is called a ***view port***.

- When the window is hanged a different part of the object at the same positron is displayed.

- If we change the view port, we see the same part of the object drawn at a different place on the display.



1- Viewing Transformation :

★ Mapping from object space to image space,

1. Change the window size to become the size of the view port (Scaling).

2. Position the window at the desired location on the screen (Translate) by moving the Lower-left corner of the window to the view port Lower-left corner location. To do this we need 2 step:

Step 1:
Move the corner to the origin (to perform the necessary scaling without disturbing the corner the corner's position).

Step 2:
Move it to the view port corner location.

Example :

A window has left and right boundaries of 3 of 5 and lower and upper boundaries of 0 and 4. The view port is the upper-right quadrant of the screen with boundaries at 0.5 and 1.0 for both X and Y direction, find the viewing transformation?

1- Translate :

The first translation matrix would be

$$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -3 & 0 & 1 \end{vmatrix}$$ To origon

2- Scaling :

The length of the window is

5 – 3 = 2 in the X direction
4 – 0 = 4 in the Y direction

The length of the view port is 1.0 – 0.5 = 0.5 in the X direction

The X scale factor is 0.5/2 = 0.25

In the Y direction is 0.5/4 = 0.125

The scaling transformation matrix is

$$\begin{vmatrix} 0.25 & 0 & 0 \\ 0 & 0.125 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

3- Translate :

Finally to position the view port requires a translation of :

$$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0.5 & 0.5 & 1 \end{vmatrix}$$

The viewing transformation is then:

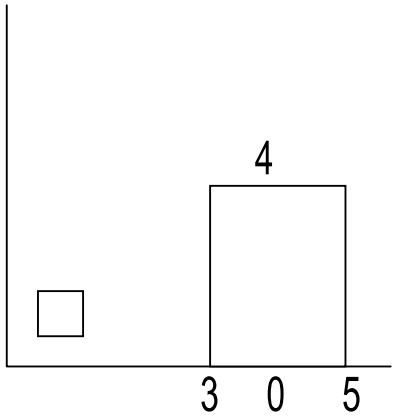
$$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -3 & 0 & 1 \end{vmatrix} \begin{vmatrix} 0.25 & 0 & 0 \\ 0 & 0.125 & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0.5 & 0.5 & 1 \end{vmatrix} = \begin{vmatrix} 0.25 & 0 & 0 \\ 0 & 0.125 & 0 \\ -0.25 & 0.5 & 1 \end{vmatrix}$$

In general the viewing transformation is

$$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -WXL & -WYL & 1 \end{vmatrix} \begin{vmatrix} \frac{(VXH - VXL)}{(WXH - WXL)} & 0 & 0 \\ 0 & \frac{(VYH - VYL)}{(WYH - WYL)} & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ VXL & VYL & 1 \end{vmatrix}$$

$$\begin{vmatrix} \frac{(VXH - VXL)}{(WXH - WXL)} & 0 & 0 \\ 0 & \frac{(VYH - VYL)}{(WYH - WYL)} & 0 \\ VXL - WXL\frac{(VXH - VXL)}{(WXH - WXL)} & VYL - WYL\frac{(VYH - VYL)}{(WYH - WYL)} & 1 \end{vmatrix}$$

V: View port
W: Window
X: Position of a vertical boundary
Y: Horizontal boundary
H: High boundary
L: Low boundary



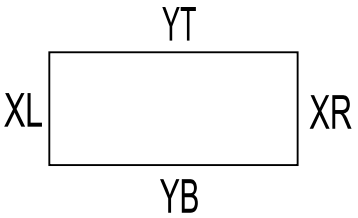
2 - CLIPPING:

If we wish to display only a portion of the total picture, we use a window to select that portion of the picture which is to be viewed (like clipping or cutting out a picture from a magazine). This is known as clipping.

The process of clipping determines which elements of the picture lie inside the window and so are visible.

2 – 1 RECTANGULAR CLIPPING WINDOWS:

The clipping window assumes to be rectangles whose sides are aligned with the coordinate area. The X extent is measured from X min to X max and the Y extent is measured from Y min to Y max.



There are three types of clipping:-

1 – Point 2 – Line 3 – Polygon

a) POINT CLIPPING:

A point p(x,y) is inside the rectangular window (visible) if all the following inequalities are true.

X_{min} X X_{max} Y_{min} Y Y_{max}

If any of these inequalities is false point P is outside the window and is not displayed (invisible).

Point Clipping algorithm

P(x , y) نقطة إحداثياتها , XL , XR , YT , YB معلومة

```
Procedure       out code ( p ; VAR  OP: integer ) ;
Begin
   Op := 0 ;
   If P.x < xl     then    op : = op   OR   1 ;
   If P.x > xr     then    op : = op   OR   2 ;
   If P.y < yb     then    op : = op   OR   4 ;
   If P.y > yt     then    op : = op   OR   8 ;
End;
```

b) Line Clipping :

1001	1000	1010
0001	0000	0010
0101	0100	0110

f

First bit on if x < xl to the left of window.

Second bit on if x > xr to the right of window.

Third bit on if y < yb to the below of window.

Fourth bit on if y > yt to the top of window.

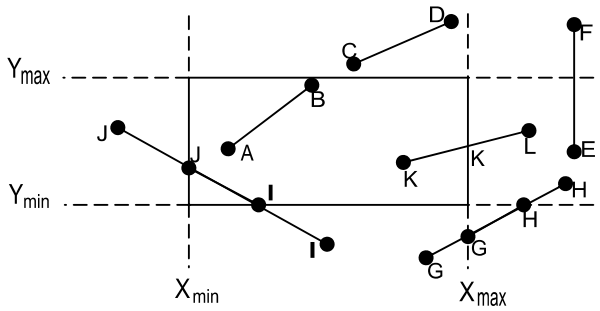
Line Segment Clipping:

Line clipping process is into two phases:

1: Identify those lines which intersect the window and so need to be clipped.

2: Perform the clipping.

All line segments fall into one of the following clipping categories:



1 – Visible:

Both endpoints of the line segment lie within the window (Line AB).

2 – Not visible:

The line segment definitely lies outside the window (Line CD and EF). This will occur if the line segment from (X1, Y1) to (X2, Y2) satisfies any one of the following four in qualities

X1, X2 > X max Y1, Y2 > Y max X1, X2 < X min Y1, Y2 < Y min

3 – Clipping candidate:

The line is in neither category 1 nor 2 (Line GH, IJ, and KL)

1- Simple visibility algorithm.

Check for totally visible lines.

If ((xb < xL) OR (xb > xR)) then 1 .

If ((xe < xL) OR (xe > xR)) then 1 .

If ((yb < yB) OR (yb > yT)) then 1 .

If ((ye < yB) OR (ye > yT)) then 1 .

Draw line

Go to 3

1 - Check for totally invisible lines

if ((xb < xL) AND (xe < xL)) then 2

if ((xb > xL) AND (xe > xL)) then 2

if ((yb < yL) AND (ye < yB)) then 2

if ((yb > yT) AND (ye > yT)) then 2

The line is partially visibly or diagonally crosses the corner; determine the intersections go to 3

2 line is invisible

3 next line

2 - Cohen clipping Algorithm

{w (4) = (xl , xr ,yb , yt) }

Subroutine end point (p , w , pc , s)

If p (1) < w (1) then pc (4) = 1 else pc (4) = 0

If p (1) > w (2) then pc (3) = 1 else pc (3) = 0

If p (2) < w (3) then pc (2) = 1 else pc (2) = 0

If p (2) > w (4) then pc (1) = 1 else pc (1) = 0

S = 0

For I = 1 to 4 s = s + pc (i) Next i

Return

Subroutine logical (pc1 ,pc2 , inter)

Inter = 0

For I = 1 to 4 Inter = inter = pc1 (i) * pc2(i) Next i

Return

Subroutine Cohen (p1 , p2 , w , visible)

Call end point (p1 , w . pc . s1)

Call end point (p2 , w .pc2 , s2)

If S1 = 0 & S2 = 0 then visible = yes

Call logical (pc1 , pc2 , inter)

If inter < > 0 then visible = NO

else visible = partial

If S1 = 0 then 1

else p1 ⟷ p2 end if

1 Return

THE ALGONITHM

FLAG = 0

IF (P2 (1) – P1 (1) = 0) THEN FLAG = - 1

ELSE slop = (p2 (2) – p1 (2)) /(p2 (1) – (p1(1))

End if

```
For  i  = 1  to 4
    Call cohen ( p1 , p2 . w1 visible )
  If visible  = yes then   2
  If  visible = No then    3
      If  flag = -1  and  i ≤  2  then  1
          If  i ≤ 2 then
               Inter = slop * ( w ( i ) – p1 ( 1 ) ) +  p1 ( 2 )
               P2 ( 1 ) = w ( I )
               P2 ( 2 ) = inter
          else   if flag = -1 then   p2 ( 2 ) = w ( i )
                                else inter = (1 /slop ) * ( w ( i ) – p1 ( 2) ) + p1 ( 1 )
                                     p2 ( 1 ) = inter
                                     p2 ( 2 ) = w ( i )
                 end  if
           end if

next  i

draw  p1 p2

end
```

Find Intersection Points

1 - Midpoint Subdivision :

The line segment is divided at its midpoint into two smaller line segments. The clipping categories of the two new line segments are then determined. Each segment in category 3 is divided again into smaller segment and categorized. The bisection and categorization process continues until all segments are in category 1 (visible) or category 2 (invisible).

The midpoint coordination (Xm , Ym) of a line segment joining P1 (X1 , Y1) to P2 (X2 , Y2) are given by

$$X_m = \frac{X + X_2}{2} \qquad Y_m = \frac{Y_1 + Y_2}{2}$$

2 - Line Intersections and Clipping :

We determine the intersection points of the lines in category (3) with the boundaries of the window. The intersection points subdivided the line segment into several smaller line segments which can belong only to category 1 (visible) or category 2 (not visible). The segment in category 1 will be the clipped line segment.

INTERSECTION POINT:

If M is the slope of the line segment between points (X1, Y1) and (X2, Y2) then if $X1 \neq X2$

$$M = \frac{(Y2 - Y1)}{(X2 - X1)}$$

Then any point (X, Y) on the line is

$$M = \frac{(Y - Y1)}{(X - X1)}$$

★ If the line segment crosses a left or right window edge then $X1 \neq X2$ and M has non denominator.

★ If the line crosses a top or bottom widow edge then $Y1 \neq Y2$ and the reciprocal of the slop 1/m has a nonzero denominator.

The slop M is obtained from the two given endpoints.

★ If we are testing against a left or right direction the X value is known (the left or right edge value).

The X value is substituted into the equation for Y.

$$Y = M * (X - X_1) + Y_1$$

For top and bottom the Y value known then:

$$X = 1/M * (Y - Y_1) + X_1$$

c) Polygon Clipping Algorithm :

Case – 1 :

If the first point and the second point inside the window then store second point.

Case – 2:

If the first point inside and the second point outside the window then store the intersection.

Case – 3:

If the first point and the second point outside the window then nothing.

Case – 4:

If the first point outside and the second point inside the window then store the intersection and the second point .

The input polygon is: V1 . V2 . V3 . V4

- عندما نعمل Clipping بالنسبة إلى Left تكون النتيجة هي V1 P2 P1 V2.
- عندما نعمل Clipping بالنسبة إلى Right تكون النتيجة هي P1 V2 V1 P2.
- عندما نعمل Clipping بالنسبة إلى TOP تكون النتيجة هي V2 V1 P2 P1.
- عندما نعمل Clipping بالنسبة إلى Bottom تكون النتيجة هي P2 P1 V2 V1.

نطبق العملية 4 مرات (Bottom , right , top , left)

Left : - V2 P1 P2 V1

Bottom : - P1 P2 V1 V2

Right : - P2 V1 V2 P1

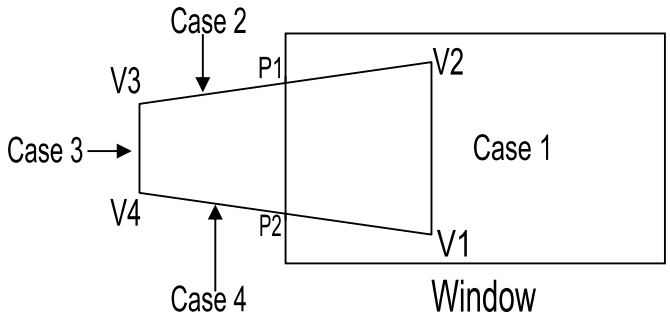
Top : - V1 V2 P1 P2

- Clips a polygon against each edge of the window.

- For each edge it inputs a list of vertices and outputs a new list of vertices.

- The input list is a sequence of consecutive vertices of the polygon obtained from the previous edge clipping.

Example :



There are 4 possible cases:

Case 1:

First and second V1,V2 inside the window, V2 is sent to the output list.

Case 2:

First vertex V2 inside and the second vertex V3 outside the window, the intersection point (P1) of the side of the polygon joining the vertices and the edge is added to the output list.

Case 3:

Both vertices V3, V4 outside the window and no point is output.

Case 4:

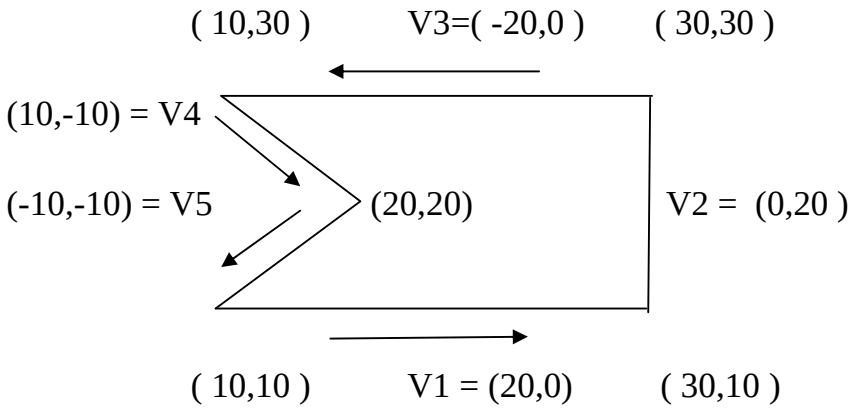
First vertex V4 outside the window and the second vertex V1 inside the window, the intersection point (P2) and the second vertex V1 are added to the output list.

The result of this left clipping is the transformation of

input list {V1, V2, V3, V4} to the

output list {V1,V2, P1, P2}.

Convex and Concave Window:



V1 = (x2 - x1 , y2 - y1) = (30 – 10 , 10 – 10) = (20 , 0)

V2 = (x2 - x1 , y2 - y1) = (30 – 30 , 30 – 10) = (0 ,20)

V3 = (x2 - x1 , y2 - y1) = (10 – 30 , 30 – 30) = (-20 ,0)

V4 = (x2 - x1 , y2 - y1) = (20 – 10 , 20 – 30) = (10 ,-10)

V5 = (x2 - x1 , y2 - y1) = (10 – 20 , 10 – 20) = (-10 ,-10)

$$V1 \cdot V2 = \begin{vmatrix} 20 & 0 \\ 0 & 20 \end{vmatrix} = 400$$

$$V2 \cdot V3 = \begin{vmatrix} 0 & 20 \\ -20 & 0 \end{vmatrix} = 400$$

$$V3 \cdot V4 = \begin{vmatrix} -20 & 0 \\ 10 & -10 \end{vmatrix} = 200$$

$$V4 \cdot V5 = \begin{vmatrix} 10 & -10 \\ -10 & -10 \end{vmatrix} = 200$$

$$V5 \cdot V1 = \begin{vmatrix} -10 & -10 \\ 20 & 0 \end{vmatrix} = 200$$

H . W (10) :

1. A window has been set to set-window (2,5,3,7), and a view port has been set to set-viewport (0,0.5,0.5,1). Find the viewing transformation.

2 . We have a set window (5,10,5,10), check the points:

a) P1(6,7).

b) P2(8,16), were inside the rectangular window or not (By Low)?

3 . We have a set window (5,10,5,10), clipping the point:

a) P1(7,7)

b) P2(18,20)

c) P3(8,16)

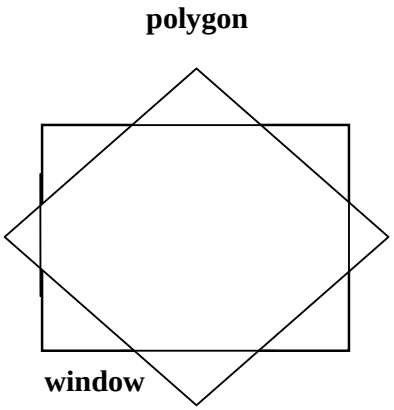
4 . If the clipping window is XL = 5, XR = 10, YB = 5, YT = 10. Clip the lines, using simple visibility algorithm:

a) AB = (6,6) – (8,7)

b) EF = (14,8) – (12,14)

c) IJ = (1,6) – (7,8)

5 . Clip the following figure using *polygon clipping algorithm*.



6 . Determine the window of the end points P1(5,5) , P2(20,5) , P3(18,10) , P4(20,20) , P5(5,20) , P6(10,10).

Aspect Ratio

Aspect Ratio: is the ratio of the horizontal width to the vertical to the vertical height.

- The horizontal and vertical plots of an equal number of pixels have different lengths.
- The ratio is a consequence of non – square pixels and rectangular display screen.

Example :

If we plot eight pixels horizontally on the display screen and then we measure the line, we find that it is 0 .3 cm wide. If we plot eight pixels vertically on the display screen and then we measure the line, we find that its 0.4 cm height.

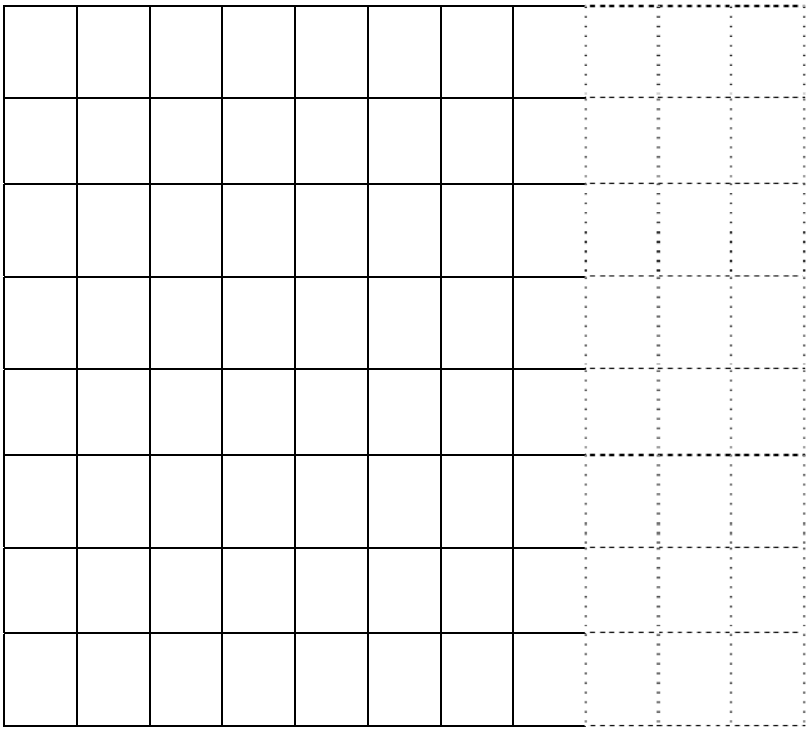
The **A.R = 0.4 / 0.3 = 1.33 .**

New number of pixels = Old number of pixels * AR .

New number of pixels = 8 * 1.33

= 10.64 ≈ 11 pixels.

11 – 8 = 3 pixels will be added to the horizontal line.



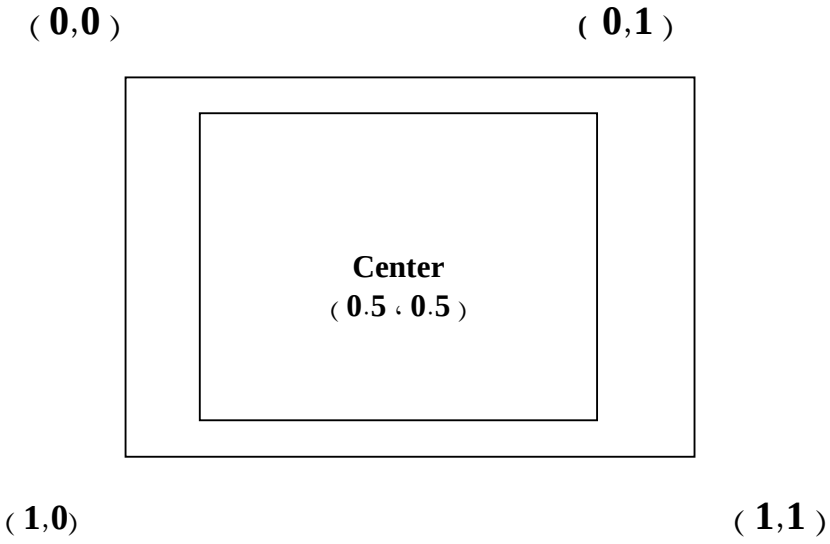
Graphics Primitives

The normalized device coordinate :

Different display devices may have different screen sizes as measured in pixels. If we wish our program to be devices independent we should specify the coordinates in some units other than pixels and then use the interpreter to convert these coordinates to the appropriates values for particular display, we are using device independent units are called the ***normalized device coordinates***.

In these units, the screen measures (1) unit wide and (1) unit high, the lower left corner of the screen is the origin , and the right upper corner is the point (1 , 1) , the point (0.5 , 0.5) is the center of the screen no matter what the physical or resolution of the actual display device may be .

Appropriate pixel values for the particular display we are using the device Independent units are called the normalized devices coordinates.



Normalized device coordinates

The interpreter uses a simple linear formula to convert from normalized device coordinates to the actual device coordinates suppose that for actual display the index of the left most pixel is the (**width – start**) and that there are (**width**) pixels in the horizontal direction .

Suppose also that the bottom most pixel is (**high – start**) and the number of Pixels in the vertical direction is (**height**). In the normalized coordinates the screen is one unit wide, so the normalized coordinates it is (**width**) units wide, so the normalized **X** position should by multiplied by (**width / 1**) to convert it actual screen units . At position **Xn = 0** in normalized coordinates we should get **Xs = width – start** in actual screen coordinate, so the conversion formula should be**:**

Xs = width * xn + width – start.

Similarly for the vertical direction

Ys =height * Yn + height – start

$$\frac{Xs}{Xn} - = \frac{width}{1}$$

Xs = width * Xn + width – start

Screen **normalized**

Normalization

هي عملية تحويل إحداثيات الشاشة إلى إحداثيات جديدة وتفيد في عملية تنفيذ البرامج على جميع أنواع الشاشات ومهما كان حجم الشاشة .

وتجري عملية التحويل وكالاتي: -

- أخذ جميع إحداثيات الصورة على max (x-1) و max (y-1) وتكون محصورة ما بين (0 , 1) سواء على المحور x أو المحور Y .

Example :

If we have points P1(0,0) , p2(200,200) p3(1023,1023) , convert these points from screen 1024 * 1024 to appropriate coordinates in screen 800*600 ?

العملية الأولى :- تقسيم إحداثيات الشاشة المصدر ناقص واحد.

MaxX = 1023(1024 = 0..1023).

MaxY = 1023(1024 = 0..1023).

We use the equation

Xr = X / maxX ; Yr = Y / maxY

For all points so we get:-

p1 * = (0/1023 ،0/1023) ≈ (0,0)

P2 * = (200/1023 ،200/1023) ≈ (0.195,0.195)

P3 * = (1023/1023 ،1023/1023) ≈ (1,1)

العملية الثانية :- نضرب النسبة في الشاشة الجديدة ناقص واحد .

MaxX = 799(800 = 0..799).

MaxY = 599(600 = 0..599).

By using the equation

X = Xr * maxX ; Y = Yr * maxY

For all points so we get:-

p1 * = (0*799, 0*599) ≈ (0,0)

P2 * = (0.195*799 , 0.195*599) ≈ (156 , 116)

P3 * = (1*799, 1*599) ≈ (799,599)

So that the points will be:-

1024 * 1024	**800 * 600**
(0,0)	(0,0)
(200,200)	(156,116)
(1023,1023)	(799,599)

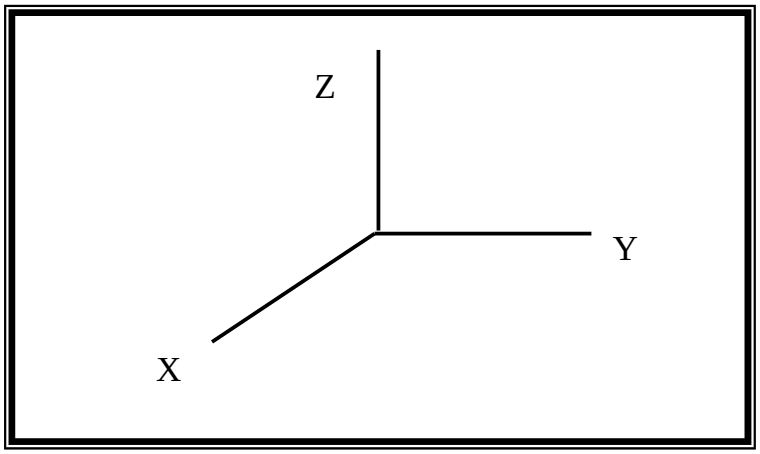
THREE – DIMENSIONAL TRANSFORMATIONS

The world composed of three – dimensional images so the object has height, width and depth. The computer uses a mathematical model to create the image.

1 - Coordinate system:

A three- dimensional coordinate system can be view as an extension of the – two –-dimension coordinate system.

The third – dimension depth is represented by the **Z – axis** which is at right angle to the **X , Y** coordinate plane. A point can be described by triple **(X , Y , Z)** of coordinate values.



2 . The Transformations:

2 . 1 Translation :-

A point (x , y , z) is translated to a new position (x_1 , y_1 , z_1) by move it dx units in the X – direction and by units in the Y – direction and dz units in Z – direction. Mathematically this can be represented as:-

$$X_1 = X + dx$$
$$Y_1 = Y + dy$$
$$Z_1 = Z + dz$$

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 \end{bmatrix} = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ dx & dy & dz & 1 \end{bmatrix}$$

2 . 2 Scaling :-

Increasing the distance, between the points describing the object can make an object. In general, we can change the size of an object, or the entire image, by multiplying the distance between points by an enlargement or reduction factor. This factor is called the **scaling factor,** and the operation that the size is called **scaling**. If the scaling factor is greater than 1, the object is enlarge, if the factor is less than 1, the object is made smaller, a factor of 1 has no effect on the object. Whenever scaling is performed, there is one point that remains at the same location. This is called **fixed point** of the scaling transformation.

a) To scale an object from a origin point:-

We used the following matrix.

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

b) To scale an object from fixed point (x_p , y_p , z_p), we perform the following three steps:

1. Translate the point (x_p , y_p , z_p) to the origin. Every point (x , y , z) is moved to a new point (x_p , y_p , z_p):

$$x_1 = x - x_p$$
$$y_1 = y - y_p$$
$$z_1 = z - z_p$$

2. Scale these translate points with the origin as the fixed points:

$$x_2 = x_1 \times S_x$$
$$y_2 = y_1 \times S_y$$
$$z_2 = z_1 \times S_z$$

3. Translate the origin back to the fixed point (x_p , y_p , z_p):

$$x_3 = x_2 + x_p$$
$$y_3 = y_2 + y_p$$
$$z_3 = z_2 + z_p$$

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 \end{bmatrix} = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_p & -y_p & -z_p & 1 \end{bmatrix} \times \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x_p & y_p & z_p & 1 \end{bmatrix}$$

2 . 3 Rotation :-

a) Rotation about X – axis:

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

b) Rotation about Y – axis:

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

c) Rotation about Z – axis:

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

Example (1):

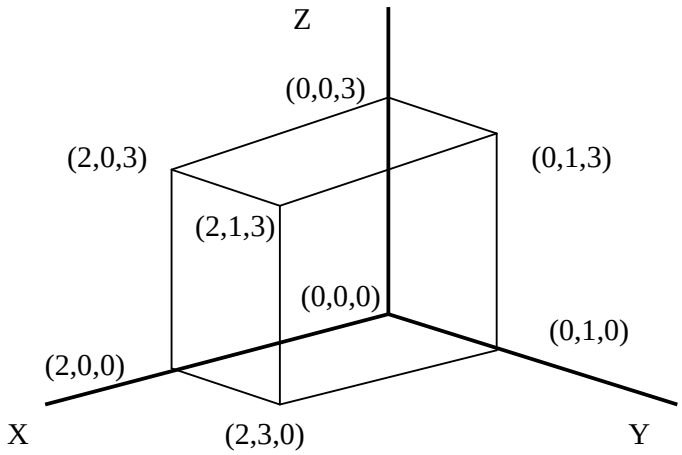
Draw the figure (0 , 0 , 0) , (0 , 1 , 0) ,(0 , 1 , 3) , (0 , 0 , 3) , (2 , 0 , 0) , (2 , 3 , 0), (2 , 0 , 3) , (2 , 1 , 3), and find: **Not: sin (90) = 1 , cos (90) = 0.**

a) Translate it to the point (0 , 3 , 0).

b) Scaling 4 times its size.

c) Rotate its (90^o) about the Z – axis.

Solution:



a) Translate it to the point (0 , 3 , 0).

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & 3 & 1 \\ 2 & 0 & 0 & 1 \\ 2 & 3 & 0 & 1 \\ 2 & 0 & 3 & 1 \\ 2 & 1 & 3 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 3 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 3 & 0 & 1 \\ 0 & 3 & 0 & 1 \\ 0 & 4 & 3 & 1 \\ 0 & 3 & 3 & 1 \\ 2 & 3 & 0 & 1 \\ 2 & 6 & 0 & 1 \\ 2 & 3 & 3 & 1 \\ 2 & 4 & 3 & 1 \end{bmatrix}$$

b) Scaling 4 times its size.

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

c) Rotate its (90^o) about the Z – axis.

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & 3 & 1 \\ 2 & 0 & 0 & 1 \\ 2 & 3 & 0 & 1 \\ 2 & 0 & 3 & 1 \\ 2 & 1 & 3 & 1 \end{bmatrix} \times \begin{bmatrix} \cos(90) & \sin(90) & 0 & 0 \\ -\sin(90) & \cos(90) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 1 \\ -1 & 0 & 3 & 1 \\ 0 & 0 & 3 & 1 \\ 0 & 2 & 0 & 1 \\ -3 & 2 & 0 & 1 \\ 0 & 2 & 3 & 1 \\ -1 & 2 & 3 & 1 \end{bmatrix}$$

H . W (11):

Draw the figure A (4 , 4 , 0) , B (-3 , 3 , 4) , C (-2 , 3 , 3), D (3 , -3 , 4) , E(3 , -2 , 3) and find:- **Not: sin (180) = 0 , cos (180) = -1**

a) Translate above shape to point (-3 , 4 , 3).

b) Scaling the figure, twice in X direction, three in Y direction and once in Z direction.

c) Rotate the figure (180^o) about X – axis.

Ibn Al-Haitham Education College

Computer Science Department

Computer Graphics / Third class

Subject name	***No. of hours***
1 – Introduction to Computer Graphics.	2
1-1) Overview.	
1-2) Application of Graphics .	
1-3) Remarks.	
1. Mode.	
-Text mode. -Graphics mode.	
2. Picture Elements.	
3. Raster-Scan Display.	
4. Resolution.	
1-4) The Graphic Display.	
1. Frame buffer.	
2. Display Controller.	
3. Scan Conversion Algorithms.	
1-5) Graphic Devices.	
2- Drawing Elementary Figure.	2
2-1) Plotting Point.	
2-2) Line Drawing Algorithms.	
2-2-1) Draw Horizontal Line.	
2-2-2) Draw Vertical Line.	
2-2-3) Draw Diagonal Line.	
2-2-4) DDA Algorithm.	
2-2-5) Bresenham`s Algorithm.	2
2-2-6) Integer Bresenham`s Algorithm.	
2-2-7) General Bresenham`s Algorithm.	2
2-3) Circle Drawing.	2
2-3-1) Circle General Algorithm.	
2-3-2) Circle General Bresenham`s Algorithm.	2
2-3-3) Circle Drawing by Using Circle Equation.	2
2-3-4) Ellipse.	
3- Two Dimensional Geometric Transformations.	2
3-1) Introduction.	
3-1-1) Translation.	
3-1-2) Rotation.	2
3-1-3) Scaling.	2

Subject name	*No. of hours*
3-1-4) Reflection.	2
3-1-5) Shearing.	
3-2) Matrix Representation of Transformations.	2
3-2-1) Translation.	
3-2-2) Rotation.	2
3-2-3) Scaling.	2
3-2-4) Reflection.	2
3-2-5) Shearing.	
4- Two Dimensional Viewing Transaction and Clipping Windows and View Ports.	2
4-1) Viewing Transformation.	
4-2) Clipping.	2
4-2-1) Rectangular Clipping Windows.	
a) Point Clipping.	
b) Line Clipping.	2
- Simple Visibility Algorithm.	
- Find Intersection Points.	2
1) Midpoint Subdivision.	
2) Line Intersections and Clipping.	
c) Polygon Clipping Algorithm.	2
4-2-2) Convex and Concave Window.	
5- Aspect Ration.	2
5-1) Graphics Primitives.	2
5-2) Normalized Device Coordinates.	2
- Normalization.	
6- Three Dimensional Transformations.	2
6-1) Coordinate System.	
6-2) The Transformations.	
a) Translation.	
b) Rotation.	2
c) Scaling.	
7- 3D Models.	8
7-1) Why do you see the movie in 3D?	
7-2) 3D Modeling.	
7-3) 3D Modeling Operations.	
7-4) Usage of 3D Modeling.	
7-5) 3D Models Features.	
7-6) The Process of 3D Modeling.	
7-7) 3D Models Creating Method.	
7-8) Stereoscopy.	