1st Semester 2D Computer Graphics

Ali Hassan Hammadie

Iraq

1st Semester 2D Computer Graphics

Ali Hassan Hammadie

Part one
(2D Introduction)

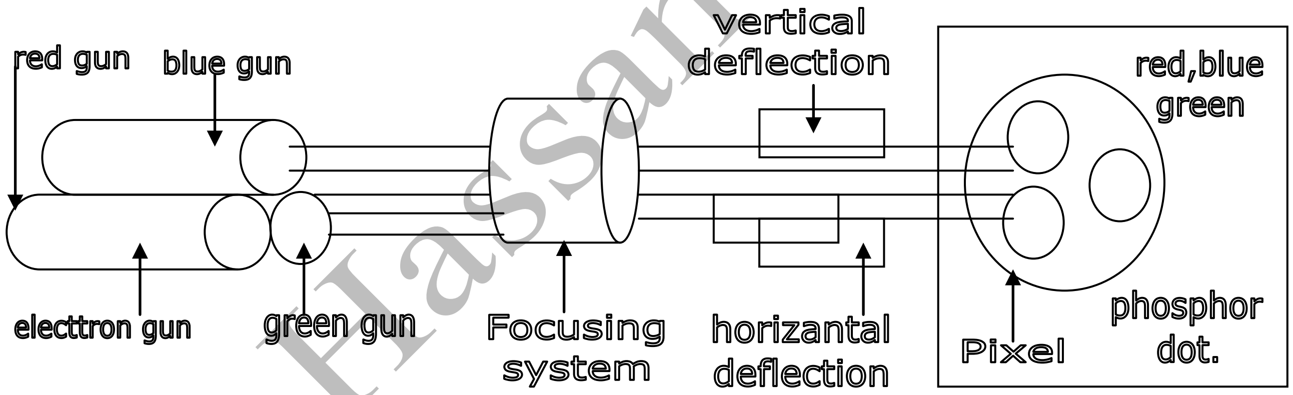
Introduction of Computer Graphics

1: <u>Computer graphics</u>: the generation, representation, manipulation, processing, or evaluation of graphic images by a computer.

- Requires more work that using test mode.
- You must develop methods for drawing lines and characters.
- Single characters is made up of many pixel arranged in pattern that forms the character.
- You can light pixel any when your display.

1.2: <u>Interactive computer graphics</u>: the observe has some control over the image, it involves communication between the computer and user, and the displayed picture is modified appropriately to signals, the computer receives from the input device e.g. (keyboard, mouse, joystick, etc.). It appears that picture is changing instantaneously in response to the observers' commands.

<u>How the cathode ray tube (CRT) works</u>:



The CRT consists of three electron guns: red, green, and blue. They emit a beam of negatively charged electrons towards a positively charged phosphor-coated screen. Along the way, the electron beam passes through the focusing system that concentrates the beam so that by the time the electrons reach the screen they have converged to a small dot.

The beam then passes through the deflection system (horizontal and vertical) which deflects the beam to strike any point on the screen. When this focused electron beams strikes the screen the phosphor emits a dot of visible light.

The video display is divided into very small dots called **"pixels"** (picture elements) each pixel is composed of a triangular pattern of red, green, and blue phosphor dot.

The CRT has three electron guns one for each of the three primary colors: red, green, and blue each electron gun hits its corresponding phosphor dot causing that particular color to appear on the screen the light on the display screen starts to fade as soon as the beam moves to another location.

So the beam must refresh the screen by illuminating pixels 30 to 60 times each second.

1.3: Generating color on a *RGB* monitors:

Each electron gun in an RGB monitor has an assigned number of bit that determines the intensities of the red, green, and blue phosphors, with one bit per gun eight-color, are possible (2^3=8).

If a fourth bit is used for the brightness or intensity of the displayed color it results in (2^4=16) possible colors. When the brightness bit equals one another set of eight bright colors is produced.

R	*G*	*B*	*Binary value*	*Color name*
0	*0*	*0*	*0*	*Black* اسود
0	*0*	*1*	*1*	*Blue* ازرق
0	*1*	*0*	*2*	*Green* اخضر
0	*1*	*1*	*3*	*Cyan* شذري
1	*0*	*0*	*4*	*Red* احمر
1	*0*	*1*	*5*	*Magenta* بنفسجي
1	*1*	*0*	*6*	*Brown* بني
1	*1*	*1*	*7*	*White* ابيض

Note: if True Color is 24 bit then displayed color = 0 to 2^{24}-1.

In RGB 24 bit ➔ Red (8 bit = 0 to 2^8-1), Green (8 bit = 0 to 2^8-1), Blue (b bit= 0 to 2^8-1)

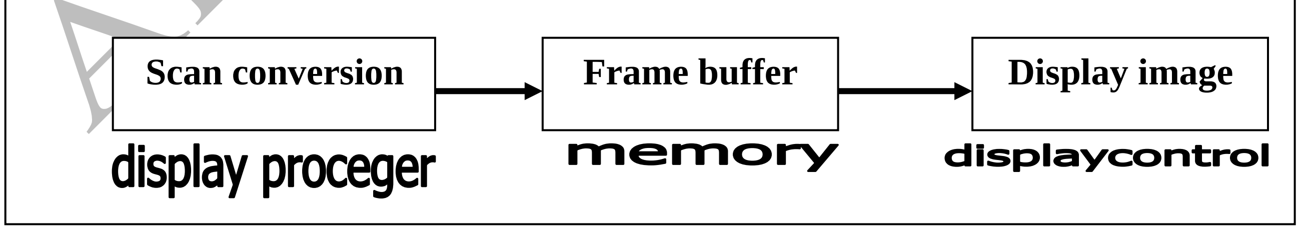
1.4: Coordinates system

- Graphics screen consists of pixels ordered in horizontal and vertical lines.
- The sizes of axes are differing.
- CGA has low-resolution pixels that are large 320 horizontally and 200 vertically.
- VGA has high-resolution pixels so small 640 vertically and 480 horizontally. But SVGA resolution pixels are (1024 × 768).
- The smaller pixel the more pixels per average and the higher quality of the graphics display.
- Each adapter has one or more graphic images.
- You must know what graphics adopt is installed and which made is active.
- To light the pixel at the right corner of the screen you must know what the screen coordinate is.

Note: **LCD**: Liquid Crystal Display, **LED**: Light-Emitting Diode, **HD**: HIGH Definition

1.5: Raster – scan display

The video display in microcomputer is divided into very small rectangle or dots these dots are referred to as picture elements or pixels. We can consider the CRT screen to consist of grid of line made up of pixels the horizontal line made up of pixels the horizontal line are called raster-scan-line and video display is referred to as raster-scan-display.



Raster scan display

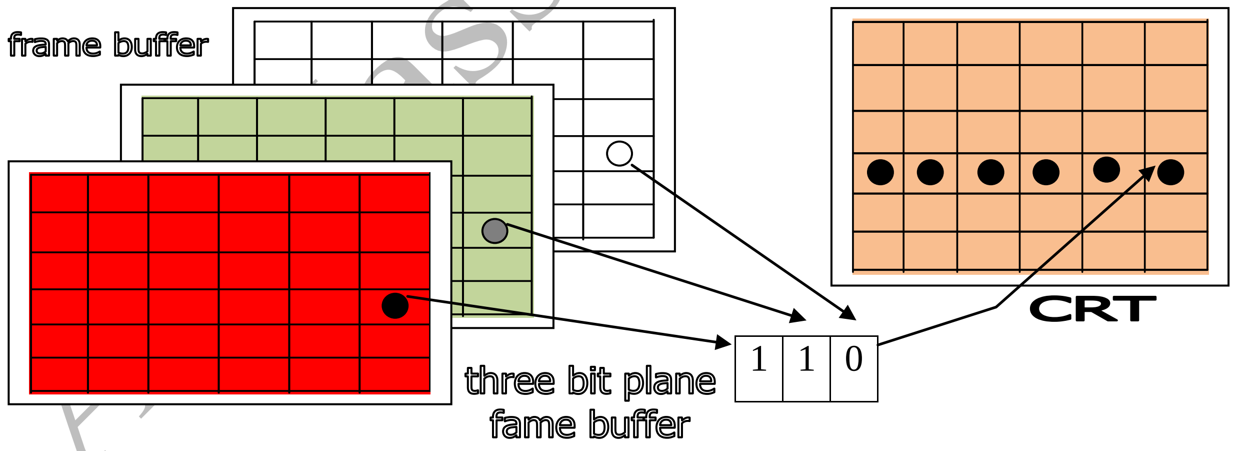
Raster scan display (n) = Scan Conversion (n) ➔Frame Buffer (n) ➔Display image (n)

1.5. A: Scan conversion

Image is usually defined in terms of equations for example x+ y=5 or graphic description such as "draw line from point to point" scan conversion is: the process of converting their abstract representation of an image into the appropriate pixel value in the frame buffer. An inexpensive microcomputer graphics system uses the CPU and library of software routines to perform scan conversion. "The process of representing continuous graphics objects as a collection of a discrete pixel is called scan conversion".

1.5. B: Frame buffer

Each screen pixel corresponds to a particular entry in a two-dimension array residing in memory. This memory is called frame buffer or bit map. Frame buffer accessible to the central processing unit (CPU) of the main computer. This allowing repaid update of the stored image. The number of rows on the frame buffer array equal the number of raster lines on a display screen. The number of columns in this array equals the number of pixels on each raster line.
The term pixel is also used to describe the row and column location in the frame buffer array that corresponds to the screen location. A size 512*512 display screen required (262144) pixels memory location.



If we wish to display a pixel on the screen a specification values is placed into the corresponding memory location in the frame buffer array.

Each pixel in the frame buffer array is composed at several bits a single bit of place frame buffer to display a color or black and white quality image with shades of gray additional bit places are needed.

0	0	1	1	0	0	0	0	0	0
1	0	0	0	0	1	1	1	0	1
1	0	1	1	0	1	0	1	1	0

Frame Buffer

➔

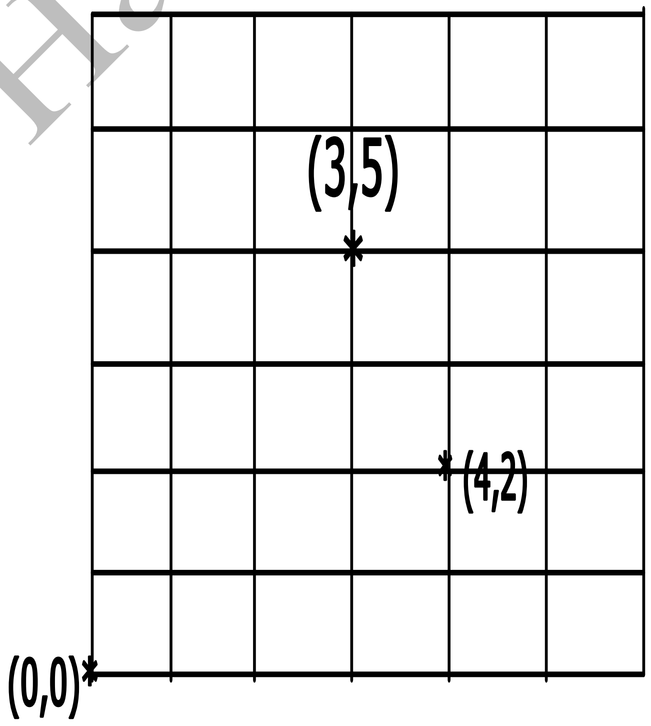
▨	▨	□	□	▨	▨	▨	▨	▨	▨
□	▨	▨	▨	▨	□	□	□	▨	□
□	▨	□	□	▨	□	▨	□	□	▨

display screen

Each screen location pixel and corresponding memory location in the frame buffer is accessed by (x, y). Integer coordinates pair. The (x) value refers to the column the (y) value refers to the row position.

1.5. C: <u>Plotting points</u>

To draw a picture on a raster display, we must determine the corresponding points in the frame buffer that make up the picture. To perform this we must write a scan conversion point-plotting algorithm. Both frame buffer and display screen are given a two-dimensional coordinates system with the origin at the lower-left corner. Each pixel is described in no negative integer (x, y) coordinates pair the x value starts at the origin (0) and increases from left to right (there is no standard for the location of two origins).



1.6: Applications of computer graphics:

1. **CAD** (Computer Aided design): use of a computer to aid in the design of product.
2. **CAM** (Computer Aided Manufacturing): use of a computer to control the manufacturing of products.
3. **CAI** (Computer Aided Instructing): use of computer to display animated pictures to illustrate educational concept.
4. **CAE (Computer Aided Engineering)** use of a computer as engineer work .
5. **Simulation**: use of a computer to experience circumstance that other wise would be too expensive or catastrophic to experience in reality E.g.: flight simulators, simulating unclear reactors.
6. **GUI** (Graphical User Interfaces): simplify the user of computer programs by giving them user friendly interfaces.
7. **Entertainment**: Computer games and movie making.
8. **Visualization:** The need for visualization today has increased dramatically and many advanced technologies will see the need for visualization. Data visualization helps us gain insights into the information to analyze and study the compatibility of the systems that we have around us.
9. **Image Processing:** Specific types of photos or photographs need to be edited to be used at various locations. One of the many techniques of computer graphics is to transform existing images into optimized ones to enhance their understanding.

1st Semester 2D Computer Graphics

Ali Hassan Hammadie

Part two (2D Vectors)

2. Vectors

The word vector is used for the type of quantity that has magnitude or size as well as direction. Vectors come in many guises: there are physical quantities such as displacement or velocity or force, where a complete description involves both direction and size, and the arrowhead its direction, then we have a vector. In this lecture, we adopt a standard notation and show vector quantities in bold type scalar quantities (like a real number) in standard type. Thus the arrow from A to B will be written AB and the arrow from A to B will be written BA; in both cases, the length of the line (the size of the vector) is the same. (in handwriting, we usually underline vector quantities).

These vector two types:

2. A. Free Vector: A. Free Vector: is shown by direction line segment whose length is a measure of its magnitude. Such as line AB, CD, and EF show figure 1 below:

Not because of E is quantity in F otherwise is false

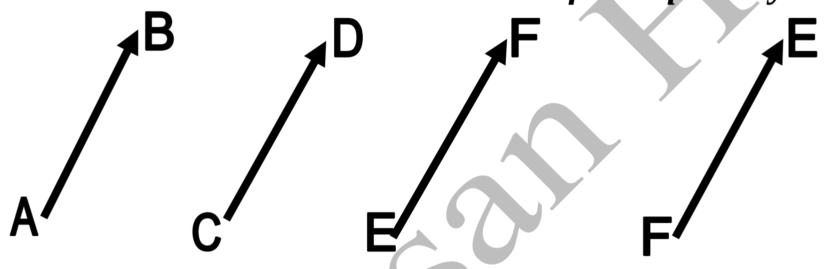
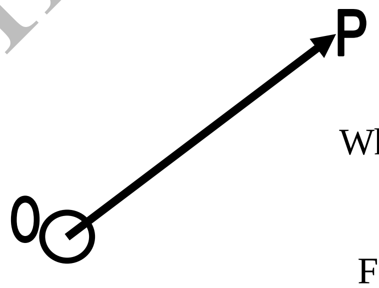


Figure 1: how that drowning of free vector and direction.

2. B) "Anchor" Vector: Its starting position at the origin (0, 0) in a particular direction, then we have a position vector. The position vector p=OP where p is specifying the position point with O is respected to the original show in figure 2 below:



Where point of O is (0,0)

Figure 2: anchor vectors

2.1: Unit vector it is a vector whose size is "1", so it can be represented by an arrow of length 1. a unit vector in the direction of the X-axis is called **i** ,and a unit vector in direction of the Y-axis is called **j**.

Note: we can write vector in this lecture either **point** such as P(5,2) or **vector row** p(5 2) or **component form** OP=5i+2j.

We note that **i= (1 0)** and **j= (0 1)** see figure 3

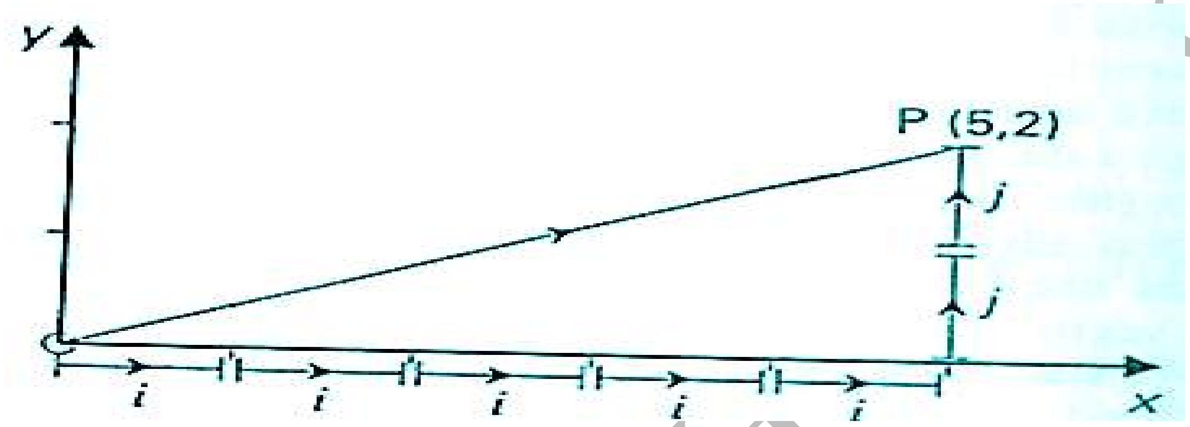


Figure 3: show that combine the unit vector **i** and **j**

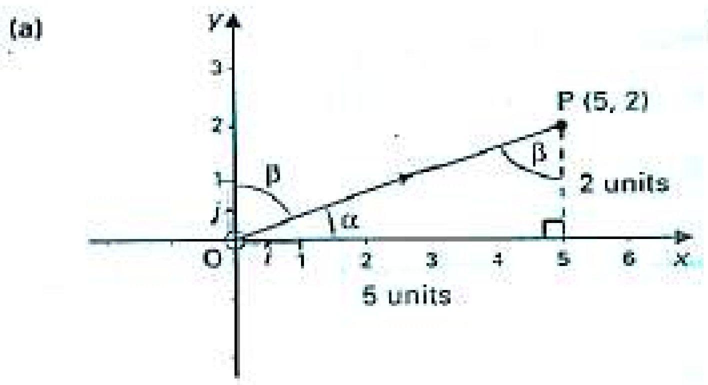
2.2: measurement associated with vectors

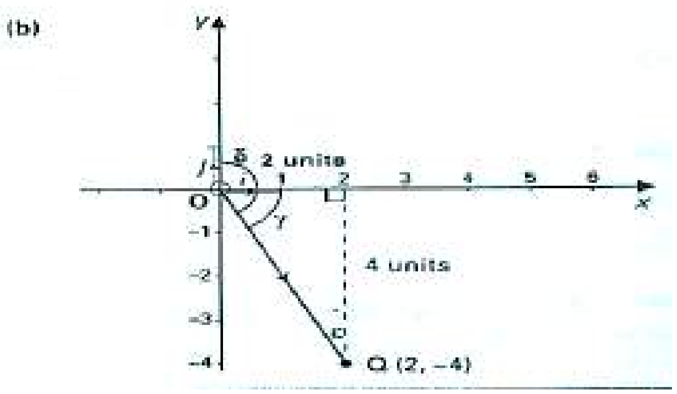
this following example of vector OP and OQ, we have shown above the there are different ways to write OP & OQ either point or row vector or component form.

2.2. A: Modulus of a vector: the modulus of a vector is given by the length of the arrow by using find length of line & term the modules of vector p is |OP|.

Example; if p (5,2) & Q(2,-4) in figure below to find modulus of two vector are:

|OP|= $\sqrt{5^2+2^2} = \sqrt{29}$ = 5.39 and |OQ| = $\sqrt{2^2+(-4)^2} = \sqrt{20}$ = 4.47





2.2. B: **unit vectors:** the unit vector in direction of OP is written $\hat{OP}$, which is calculated as following:{ $\widehat{OP}$ **= vector OP/modulus of OP}**. in preview example then $\widehat{OP}$ **= OP/|OP|**

OP/|OP| = (5i+2j)/5.39 = (5i/5.39) + (2j/5.39) = 0.93i+0.37j

Similarly we have $\hat{OQ}$ = OQ/|OQ| = (2i-4j)/4.47 = 0.45i-0.89j

2.2. C: Angles between vectors and axis

1. The angle between OP and i (X-axis) is ά, where tan ά =2/5=0.4 so ά =21.80.

2. The angle between OP and j (Y-axis) is β, where β=90-21.80 = 68.20.

H.W: **find angle between OQ and i & OQ and j.**

2.3 manipulation vectors

The system of vectors, which we have introduced above, will be seen to give us the terminology and techniques for dealing with point, line, angle, and surface too, as will be seen later. Moreover, as this system deals with geometrical quantities in numerical terms, it is extremely valuable for computer technology. To exploit their potential we must be able to manipulate vectors, so next, we look at aspects of adding and subtracting vectors and "scaling" them by a number.

2.3.A:adding vectors let see figure below two vector p, q

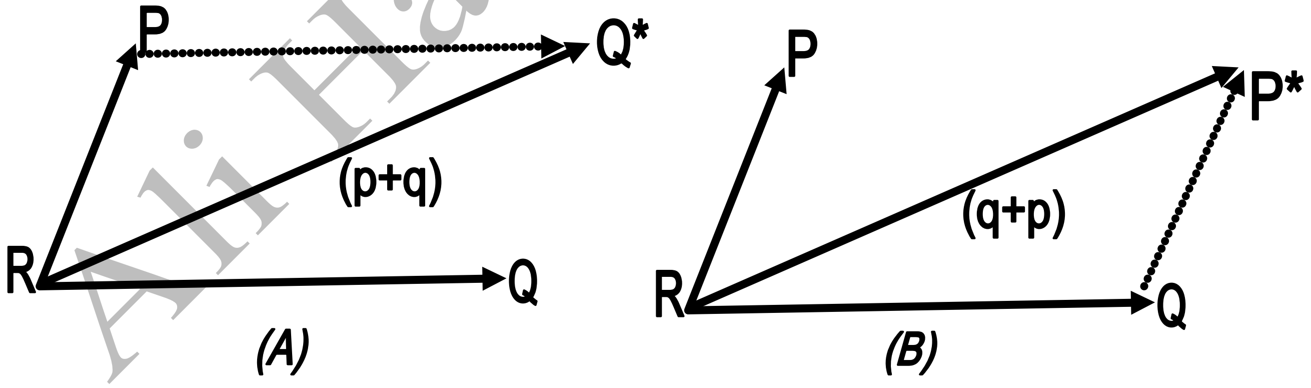


Figure 4 show adding two vectors (A) p+q (B) q+p

Example: let vector p (3, 2) and Q (5,-4) find p +q and drowning.

Sol. / P=3i+2j <u>and</u> q=5i-4j

P +q= (3i+2j) + (5i-4j)

=3i+2j+5i-4j=8i-2j

(In component form)

======================

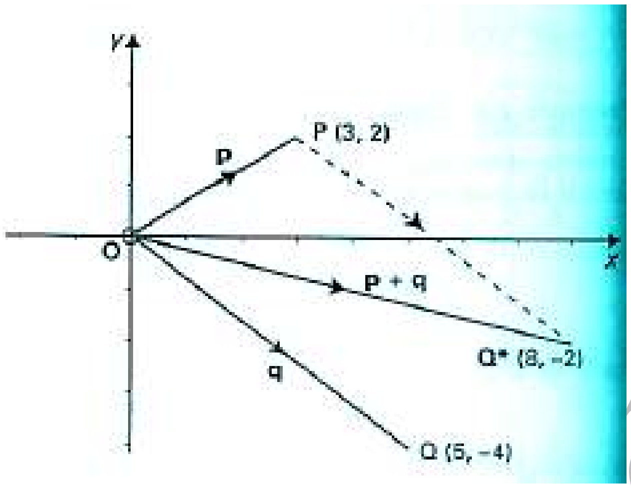


Figure 5: result adding vector p+ q to Q*

p= (3 2) and q= (5 -4) then p+ q = (3 2) + (5 -4) = (3+5) (2-4) = (8 -2) **(in row vector)**

2.3. B: <u>negative vectors and subtracting vectors</u>

-q is a negative vector, by which we mean a vector with the same magnitude as q but opposite direction, as shown in figure 6. We can calculate p-q by finding p+ (-q): we reverse the direction of q and starting from Q, we then use triangle addition.

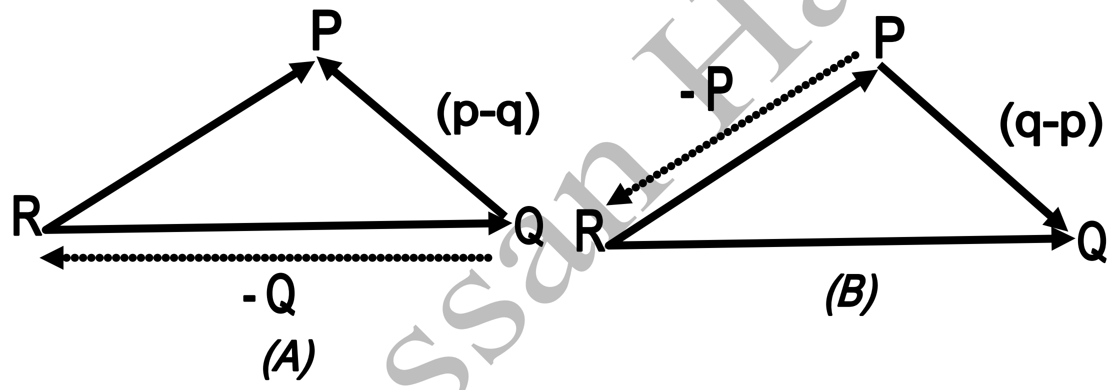


Figure 6 show drawing triangle in (A) p-q (B) q-p

Example: let p(3,2) and Q(5,-4) and drowning this vectors.

Sol. / in component form or row vector

(In component form)	(In row vector)
p=3i+2j and q=5i-4j	p=(3 2) and q=(5 -4)
p-q=(3i+2j)-(5i-4j)	p-q=(3 2)-(5 -4)
=3i+2j-5j+4j	= (3-5) (2+4)
=-2i+6j	= (-2 6)

Drowning is H.W ((coordinate the example)) answer in figure 7

Note: if we wish to determine the vector between any two points whose coordinates are known, then we use vector subtraction. Suppose we require the vector p (2, 7) to Q (4, 10): we note that direction matters, it is PQ that we want. First we identify the position vector OP=p and OQ=q, and then we use these to calculate the vector PQ using triangle addition in triangle OPQ: PQ =PO +OQ= -p+ q= -(2 7)+ (4 10) = (2 3). Similarly, we can show that QP= (-2 -3), which is as expected since this is the negative of the vector PQ.

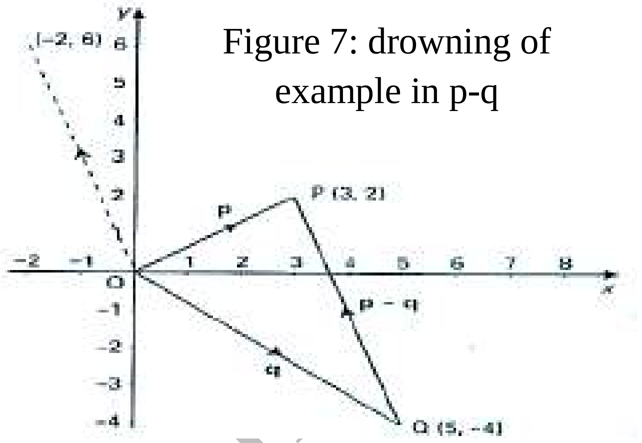


Figure 7: drowning of example in p-q

2.3.C:<u>scaling Vectors</u>

We can scale any vectors by multiplying them by scalar number (just a number). To scale it by 3 "we make it three times bigger in the same direction that is we multiply the vector by 3. that p is (4 3) as in figure 8 we have 3p=3(4 3)=(12 9), similarly a scale factor 1/2 in the same direction (1/2)p = 1/2 (4 3) = (2 3/2).

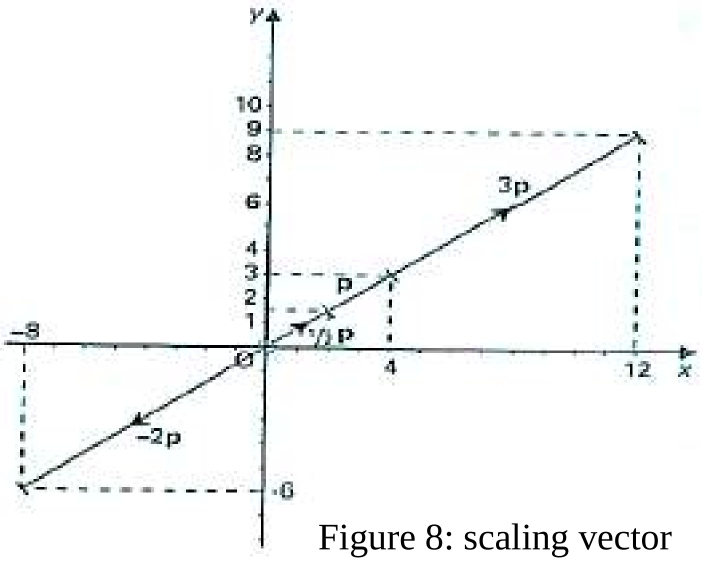


Figure 8: scaling vector

A negative scale factor reverses the direction of a vector. When the scale factor is -1, then the effect is just to reverse the vector, so that it has the same modulus but the opposite direction. If a negative scale factor is other than -1, then alters the modulus of the vector as well. Thus referring to the same vector p in figure 8: (-2) p=-2(4 3) = (-8 -6) the effects of these scaling's indicated in figure 8.

2.3. D: <u>multiplying vectors uses the "*dot Product*"</u> Purpose of the dot *product* in computer graphics we shall use this product as a way of finding the angle between two vectors, and also of showing when two vectors are perpendicular. We define their dot product by ***a.b*=|*a*| |*b*| *cos* θ; *where*** ɵ is 0<= θ <=180, and therefore to find angle between of two vectors is: $\cos\theta = \frac{a.b}{|a|*|b|} \Rightarrow \theta = \cos^{-1}(\frac{a.b}{|a|*|b|})$

<u>Note</u>: *if dot product =zero implies <u>either</u> that at least one vector is the zero <u>or</u> that the vector are perpendicular.*

We can easy calculating *dot product* **a. b** in 2D to two vectors a & b is:

a. b= (a1 * b1) + (a2 * b2); where a=a1i + a2j & b= b1i + b2j

<u>Example</u>: calculate the angle between the vector a=3i+5j and b=2i+j as shown in figure 9

Sol. / ***a.b*=|*a*| |*b*| *cos* θ** ➔|a|= $\sqrt{3^2+5^2}=\sqrt{34}$,

|b| = $\sqrt{2^2+1^2}=\sqrt{5}$ ➔

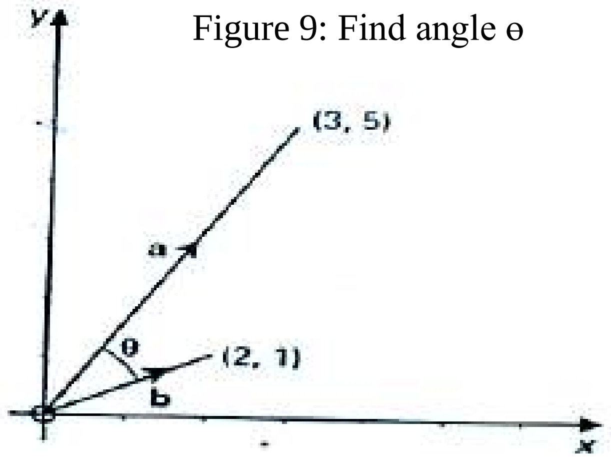
a . b = (a1*b1) + (a2*b2) = (3*2) + (5*1) =11

Cos θ = (a. b)/ (|a| |b|) = 11/ ($\sqrt{34}*\sqrt{5}$) = 0.8437

Then θ = Cos^{-1} (a. b) / (|a| |b|) ➔

θ = Cos^{-1} (11/ ($\sqrt{34}*\sqrt{5}$)) = 32.47

Figure 9: Find angle ɵ



<u>Note:</u> *then other multiplying vector is "**cross product**" this type is used in vector in 3D. These subject is useful in reach lecture 3D.*

2.3.E : <u>Multiplying vectors uses the "*Cross Product*"</u>

Purpose of cross *product* in computer graphics we shall use this product as a way of finding the perpendicular between two vectors, and also generate another vectors is perpendicular of two Vector. We define their cross product in 2D by ***a*×*b*=|*a*| |*b*| *Sin*(ɵ)*n* ; *where*** **ɵ** is angle , and **n** is unit vector of perpendicular between a & b therefore if not n(normal) ➔ **|*a*×*b*|=|*a*| |*b*| *Sin*(ɵ),** to find angle between of two vectors is : ɵ= Sin^{-1} |a×b| / (|a| |b|)

We can easy calculating *cross product* **a×b** in 2D where a = ai+ aj &b= bi +bj is:

$$\boldsymbol{a\times b}= \begin{bmatrix} ai & aj \\ bi & bj \end{bmatrix} = \boldsymbol{(ai * bj)i - (aj * bi)}j$$

Example: calculate the angle between the vector a=3i+5j and b=2i+j

$$\boldsymbol{a\times b}= \begin{bmatrix} 3 & 5 \\ 2 & 1 \end{bmatrix} = \boldsymbol{(3 * 1)i - (5* 2)}j = 3i- 10j \text{ , H.w// } \boldsymbol{b\times a}$$

2.4: Direction Cosine: it is using to find angle of any axis, thus put Axis owing as adjunctive of cosine for Example if a=3i+5j to find angle in X-axis, you need find $|a| = \sqrt{3^2+5^2} = \sqrt{34}$, therefore $\cos(\alpha)=3/\sqrt{34}$ ➔ $\alpha=\cos^{-1}(3/\sqrt{34}) \simeq 59^o.04$.

Note: - to find in Y-axis simply put j-component of vector **a** as $\cos(\beta)=5/\sqrt{34}$ ➔ $\beta=\cos^{-1}(5/\sqrt{34}) \simeq 30.96$.

V=(x ,y) ≡Xi +Yj (have Quantity ©, Direction➔)

V1	operation	V2	R
➔	±	➔	➔
©	. , × , *	➔	©➔
-1	. , × , *	➔	🡐
➔	**. (dot)**	➔	©
➔	×	➔	➔
➔	÷ , /	©	➔/©

Table: vector behavior

1st Semester 2D Computer Graphics

Ali Hassan Hammadie

Part three
(2D Line Drawing)

3. Line: a line can be described as a single point that continues for a distance, or as the connection between two points (x1, y1, z1) ,(x2, y2, z2) .
The line is one of the main components of design including principles such as shape, color, texture, value, perspective, and form. Lines can appear in many different forms some examples may be straight, curved, continuous, dotted, thick, thin, real, and implied. A line can be used to create structure and tone in illustrations and other artworks.
To find distance of line calculate $dx=X_{end} - X_{start}$, $dy=Y_{end} - Y_{start}$ and slop M=dy/dx.

3.1 drawing lines

Requirements for an acceptable line drawing algorithm

1. The line should appear to be straight.
2. The line should terminate accurately.
3. The line should have constant density.
4. The line density should be independent of line length and angle.
5. Line should be drawn rapidly.

3.2 Horizontal and vertical lines:

The simplest lines to drawn are horizontal and vertical lines, the screen coordinates of the point an **a horizontal line** *are obtained by keeping the value of y constant and repeatedly incrementing or decrementing the x value by one unit.* The following lines of code draw a horizontal line between the two points, (x1, y1), (x2, y2)

Input:(x1,y1),(x2,y2)
Output: Draw a Horizontal line Because (y2-y1)=0 ➔ { dy=0}

```
1: For x =x1 to x2 step sign(x2-x1)
2: Plot (x, y), color
3: Next x
```

Note: - sign (n) return either **1** {n>0} or **0** {n=0} or **-1** { n<0}

To draw a vertical line the x value is fixed and the y value varies.

Input:(x1,y1),(x2,y2)

Output: Draw a vertical line Because (x2-x1)=0 ➔ { dx=0}

```
    1: For y =y1 to y2 step sign (y2-y1)
    2: Plot (x, y), color
    3: Next y
```

3.3 Diagonal lines:

To draw a **diagonal line with slop equal to 1** *we repeatedly ejective Change by same unit both the x and y values from the starting to the ending pixels.* **The slop** *is defined as the change in y value divided by change in x values*: ***m= Δy/ Δx = (y2-y1)/(x2-x1)***.

The following lines of code draw a diagonal line with slop equal to +1 or -1 only:

x =x1; y =y1;While (x<=x2) { plot (x, y, color); x =x +sign(x2-x1); y=y +sign(y2-y1); }

Input :(x1, y1), (x2, y2)

Output: Draw a slop line m= -1 or m=1

```
    1: x =x1; y =y1
    2: For x =x1 to x2 step sign(x2-x1) // x=x +sgn(x2-x1)
    3: Plot (x, y), color
    4: y=y+ sign(y2-y1) //sign(y2-y1)= either +1 or -1
    5: next for // repeat in step 2 until x =x2
```

Note: ***the slop is zero*** *then* ***line is a horizontal dy(y2-y1)= (y1-y2)=0***

and if ***slop is undefined*** *then* ***line is a vertical dx(x2-x1)= (x1-x2)=0***

but if ***slop line is not equaled +1 or -1*** *then* found three methods to draw diagonal lines are:

A>Y=mX+b.

B>Simple DDA

C>Bresenhams line

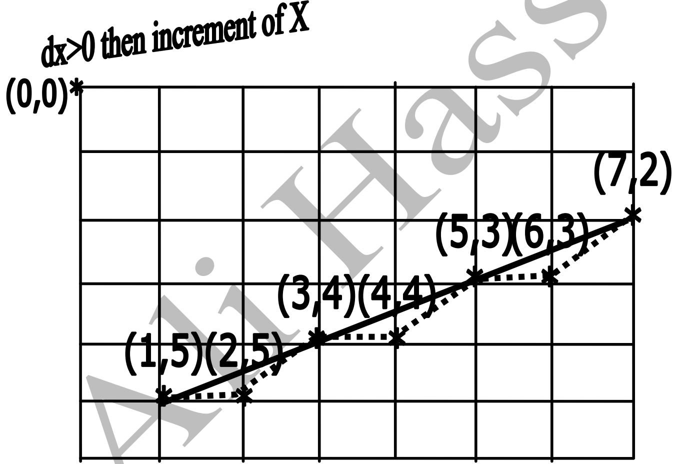
3.4 A Using line equation 'Y=mX+b': (جانب العملي فقط)

this method is advantaged the line equation to finds point where they belongs in line. The constant **b** is found by if it assigns line point in this equation then you find the value of **b**. For example if the endpoints of line is (x1, y1) (x2, y2) then the ***b = y1-mx1*** or ***b=y2-mx2***. The following algorithm is used this method below:

```
1: dx=x2-x1: dy=y2-y1
2:if (dx=0) that lead draw the vertical line & exit
3:if (dy=0) that lead draw the horizontal line & exit
4: m=dy/dx : b=y1-mx1 { OR b=y2-mx2}
5: for x=x1 to x2 step sgn(dx)  {X+ =1 or X- =1}
5.1:  y=m*x+b
5.2 plot point ( x, y), color
6: next x {goto 5 where un-finish}
```

Example: trace the line equation to draw the line that endpoints are (1, 5), (7, 2), and draw the line in screen coordinate.

dx=7-1=**6**; **dy**=2-5= **-3**; **m**= -3/6= -1/2 =**-0.5**; **b**=5- (-0.5)*1=**5.5**



X	Y	Point (x, y)	Plot in screen
1	1*-0.5+5.5	(1,5)	(1,5)
2	2*-0.5+5.5	(2,4.5)	(2,5)
3	3*-0.5+5.5	(3,4)	(3,4)
4	4*-0.5+5.5	(4,3.5)	(4,4)
5	5*-0.5+5.5	(5,3)	(5,3)
6	6*-0.5+5.5	(6,2.5)	(6,3)
7	7*-0.5+5.5	(7,2)	(7,2)

H.W/

1. Tracing same example but endpoints (7, 2) (1, 5).
2. Tracing the line (-8, -2), (4, 7) and draw line in coordinate system (4quartered)

3.5 The simple DDA

The simple Digital Differential Analyzer is a line drawing algorithm that generates lines from their differential equations. It works on the principle of simultaneously incrementing x and y by small steps proportional to the first derivatives of x and y. The slop a line between the two points (x1, y1), (x2, y2) is given by m= (y2-y)/(x2-x1).

The algorithm starts with the initial values x=x1 & y=y1, the coordinates are then incremented by Δy and Δx respectively to find the coordinates of the next point. The value of x and y are rounded to integers and a pixel is set at that point. This step is repeated until the second endpoint (x2, y2) is reached.

```
    1: dx=x2-x1, dy=y2-y1
    2: if (abs (dx)>abs (dy)) then length=abs(dx)
       Else length=abs (dy)
    3: xinc=dx/length: yinc=dy/length
    4: x=x1: y=y1
    5: for i=0 to length
     5.1: plot pixel(x, y)
    5.2: x=x +xinc ,
    5.3: y=y +yinc
 6: next i
```

Ex/: **trace the simple DDA algorithm to draw a line between the two points (23,33), (29,40).**

Sol/: dx=29-23=6

dy=40-33=7

Length=7

Xinc=6/7 =0.857

Yinc=7/7=1.

X	***Y***	***Plot(X)***	***Plot(y)***
23	***33***	***23***	***33***
23.857	***34***	***24***	***34***
24.714	***35***	***25***	***35***
25.571	***36***	***26***	***36***
26.429	***37***	***26***	***37***
27.286	***38***	***27***	***38***
28.143	***39***	***28***	***39***

H.W/ trace the simple DDA algorithm to draw a line between the two point (29, 40), (23, 33).

3.4 C **Bresenhams line drawing algorithm**:

This algorithm is designed so that each iteration changes one of the coordinate values by ±1. The other coordinate may or may not change depending on the value of an error term maintained by the algorithm. This error term records the distance measure perpendicular to the axis of greatest movement, between the exact path of the line and the actual dots generated.

If the x-axis is the axis of greatest movement, then at each iteration the x coordinates of the line are incremented, **and the slop of the line by dy/dx is added to the error term**. Whether to increment the y coordinate of the current point. A positive e value indicates that the exact path of the line lies above the current point, therefore the coordinate is incremented, and 1 is decremented from e.

If e is negative the y coordinate value is left unchanged.

And verse vise if y-axis is the greatest movement

Impartment note:-If X-axis is the greatest movement the initial **e =dy/dx**

But If Y-axis is the greatest movement the initial **e =dx/dy**

Bresnham code become when |dx |>|dy|

```
dx = X2 - X1;      dy = Y2 - Y1;
m = dy /dx;
er = m - 0.5 * sign(m);
x=x1;y=y1;
For i = 0 to abs(dx)
    Plot (x, y);
    If sign(er) = = sign (m)
        y = y + sign (dy);
        er = er - 1 * sign(m);
    Endif
    x = x + sign (dx);
  er = er + m;
Next i
```

Bresnham code become when |dy |>|dx|

```
dx = X2 - X1;      dy = Y2 - Y1;
m = dx /dy;
er = m - 0.5 * sign(m);
x=x1;y=y1;
For i = 0 to abs(dy)
    Plot (x, y);
    If sign(er) = = sign (m)
        x = x + sign (dx);
        er = er - 1 * sign(m);
    Endif
    y = y + sign (dy);
  er = er + m;
Next i
```

Ex. / trace the Bresenham algorithm to draw a line between the two points (50, 65), (59, 68).

Sol. / dx= 59-50= 9

dy= 68-65= 3

m= dy/dx =3/9 = 0.333

e=0.333 – 0.5 = -0.167

X	Y	e
50	65	-0.167 +m
51	65	0.166 -1+m
52	66	-0.501 +m
53	66	-0.168 +m
54	66	0.165 -1+m
55	67	-0.502 +m
56	67	-0.169 +m
57	67	0.164 -1+m
58	68	-0.503

this field show the e increment or unchange

Ex. / trace the Bresenhams algorithm to draw a line between the two points (1, 5), (7, 2).

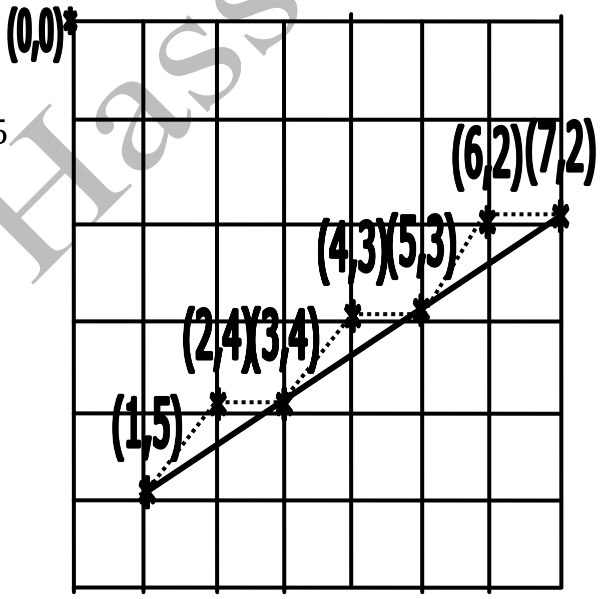
Sol. / dx= 7-1= 6;

dy= 2-5= -3

m=dy/dx=-3/6=-0.5

e= -0.5 + 0.5 = 0

Note:- try e= -0.5



X	Y	e
1	5	0 +m
2	5	-0.5 +1+m
3	4	0 +m
4	4	-0.5 +1 +m
5	3	0 +m
6	3	-0.5 +1 +m
7	2	0

EX// *Trace the line where end points (0,3), (2,-2)* by Bresnham's Method

Sol.// dx= 2-0= 2 ,dy= -2 -3= -5,

|dy|>|dx|➔Y **is Greatest move**: Change for each step ,

m=(dx/dy)= 2/-5 = -0.4➔

m<0 then e<0 then change X otherwise un-change X because less move, e=(dx/dy)+0.5= 0.1 , X = 0 , Y= 3

dx>0 then increment of X

dy<0 then decrement of Y

X	Y	e	Status of e
0	3	0.1	+(dx/dy)
0	2	-0.3	+1+(dx/dy)
1	1	0.3	+(dx/dy)
1	0	-0.1	+1+(dx/dy)
2	-1	0.5	+(dx/dy)
2	-2	0.1	Finish

Note: - need Trace by |different Greatest move | +1

Trace this line where start (3, 9) with end (-4, 4) by Bresanham drawing.
Sol// dx= -4-3 = -7, Dy=4-9= -5, length=7 because X is greatest movement ➔m=dy/dx = -5/ (-7) ➔ m= 0.714
M+ && e+ ➔ change less-movement (y-axis)

Step	e	X➔اكبر	y➔اصغر	State (e)+
0	0.214	3	9	M -1
1	-0.072	2	8	M
2	0.642	1	8	M-1
3	0.356	0	7	M-1
4	0.07	-1	6	M-1
5	-0.216	-2	5	M
6	0.498	-3	5	M-1
7	????	-4	4	Finish

1st Semester 2D Computer Graphics

Ali Hassan Hammadie

Part four
(2D Circle, Ellipse)

4. Drawing curves:

A typical point (x, y) can be described using the right angle description we refer to the vertical distance y as the perpendicular. The horizontal distance x as the base, and the distance from (0, 0) to x, y as the hypotenuse R. these distance are related by Pythagoras's theorem

R^2=X^2+Y^2 …. (1).

If any two distances in equation (1) are known this relationship can be manipulated to find third unknown distance.

First; the ratio of base to hypotenuse gives the cosine:

Cos (Ө) = X / R *….. (2)*

Second; the ratio of perpendicular to hypotenuse gives the sine:

Sin (Ө) = Y / R *….. (3)*

4.1 Drawing circles:

- A circle is specified by the coordinate of its center (xc, yc) and its radius(r).
- The circle equation is (x-xc)^2+(y- yc)^2=r^2 ….(4)

If the center of the circle is at the origin (0,0) the equation is: **x^2 + y^2 =r^2** …..(1); solving equation **(4)** for y obtain ➔ $y = yc \pm \sqrt{r^2 - (x - xc)^2}$.

Solving equation **(1)** for y obtain ➔ $y = \pm\sqrt{r^2 - x^2}$

Note: to draw a circle increment x by one unit form ***–r to r*** and so the above equation to solve for the two y value at each step (convert to integer).

Ex//circle that center (100,-20) & radius 6 units find circle points

X (-6 t0 6)	$y1 = +\sqrt{r^2 - x^2}$	$y2 = -\sqrt{r^2 - x^2}$	**X+Xc**	**y1+YC**	**y2+YC**
-6	0	0	**94**	-20	-20
-5	$y1 = +\sqrt{6^2 - (-5)^2}$	$y1 = -\sqrt{6^2 - (-5)^2}$	95	-20 + $\sqrt{11}$	-20 - $\sqrt{11}$
. . 6	H.w .. H.w	H.w .. H.w	H.w .. H.w	H.w .. H.w	H.w .. H.w

H/w: find the point of a circle where xc=20, yc=10 and r=8.

This method of drawing a circle is inefficient because

1. We are not taking advantage of symmetry of the circle.
2. the a mount of processing time required to perform the squaring and square root operation repeatedly

The algorithm in circle equation below

```
1: x= -r, dt:= 1/r   ' increment unit in circle
2: While (x<= r)
  2.1:  y1= +sqrt(r*r-x*x)
  2.2:  y2:= -sqrt(r*r-x*x);
  2.3: plot (xc +x, yc+y1)
  2.4: putpixel(xc +x, yc+y2)
  2.5  x=x+dt
3: end-while
```

Ex//Trace this circle where Center at (-20,40),radius is 7 units by circle equation

Sol// Xc= -20 , Yc = 40 , R=7 ➔ Xϵ [-7 .. 7] ➔ Y= $\pm\sqrt{r^2 - x^2}$

X	Y1=$+\sqrt{49 - X^2}$	Y2=$-\sqrt{49 - X^2}$	XC+X	Yc+Y1	YC+Y2
7	$\sqrt{49 - 7^2}$ =0	0	-20+7	40+0	40+0
6	$\sqrt{49 - 6^2} = \sqrt{13}$	$-\sqrt{13}$	-20+6	$40 + \sqrt{13}$	$40 - \sqrt{13}$
5	$\sqrt{49 - 5^2} = \sqrt{24}$	$-\sqrt{24}$	-20+5	$40 + \sqrt{24}$	$40 - \sqrt{24}$
4	$\sqrt{49 - 4^2} = \sqrt{33}$	$-\sqrt{33}$	-20+4	$40 + \sqrt{33}$	$40 - \sqrt{33}$
3	$\sqrt{49 - 3^2} = \sqrt{40}$	$-\sqrt{40}$	-20+3	$40 + \sqrt{40}$	$40 - \sqrt{40}$
2	$\sqrt{49 - 2^2} = \sqrt{45}$	$-\sqrt{45}$	-20+2	$40 + \sqrt{45}$	$40 - \sqrt{45}$
1	$\sqrt{49 - 1^2} = \sqrt{48}$	$-\sqrt{48}$	-20+1	$40 + \sqrt{48}$	$40 - \sqrt{48}$
0	$\sqrt{49 - 0^2} = 7$	-7	-20+0	40+ 7	40- 7
-1	$\sqrt{49 - (-1)^2} = \sqrt{48}$	$-\sqrt{48}$	-20-1	$40 + \sqrt{48}$	$40 - \sqrt{48}$
-2	$\sqrt{49 - (-2)^2} = \sqrt{45}$	$-\sqrt{45}$	-20-2	$40 + \sqrt{45}$	$40 - \sqrt{45}$
-3	$\sqrt{49 - (-3)^2} = \sqrt{40}$	$-\sqrt{40}$	-20-3	40+ $\sqrt{40}$	40$-\sqrt{40}$
-4	$\sqrt{49 - (-4)^2} = \sqrt{33}$	$-\sqrt{33}$	-20-4	$40 + \sqrt{33}$	$40 - \sqrt{33}$
-5	$\sqrt{49 - (-5)^2} = \sqrt{24}$	$-\sqrt{24}$	-20-5	40+ $\sqrt{24}$	40 $-\sqrt{24}$
-6	$\sqrt{49 - (-6)^2} = \sqrt{13}$	$-\sqrt{13}$	-20-6	40+ $\sqrt{13}$	40 $-\sqrt{13}$
-7	$\sqrt{49 - (-7)^2} = 0$	0	-20-7	40+ 0	40 - 0

4.2 The polar representation of circles:

A circle can be described by tracing and the path of the point (x, y) keeping R fixed and making one complete revolution of the angle Θ. One revolution is measured from 0 to 360 or radians. Which is the measure usually employed by computers from 0 to $2\prod$ Radians, from equation (2) and (3) we find:

$$X = R * \cos(\Theta) \ldots\ldots (5)$$

$$Y = R * \sin(\Theta) \ldots. (6)$$

These are equations used to trace out point defining the circumference of a circle using polar representation.

```
1: dt = 1/r ; th=0 ;  pi=3.1415
2: While (th<=2 * pi)
    2.1: x=r*cos (th) :  y:=r*sin (th)
    2.2:  Plot (xc +x, yc +y)
    2.3: th= th+ dt
3: end-while
```

Ex. / trace the algorithm that uses the polar representation to generate eight points of the circle centered at (300,150) with a radius of 5 units.

Sol. / dt= 1/r =1/5= 0.2; x= r* cos (Θ); y= r * sin (Θ);

Θ	**X**	**Y**	**x +xc**	**Y +yc**	**Plot x**	**Plot y**
0	**5**	**0**	**305**	**150**	**305**	**150**
0.2	**4.9**	**0.993**	**304.9**	**150.993**	**305**	**151**
0.4	**4.605**	**1.947**	**304.605**	**151.947**	**304**	**152**
H.W						

4.4 Incremental drawing of circles:

This method makes use of elementary results from trigonometry

If ***Xn= R* cos(Θ)*** and ***yn= R * sin(Θ)*** then

$$X_{n+1}= R* \cos(\Theta+\Delta\Theta)\dots (6)$$

And $Y= R*\sin(\Theta+\Delta\Theta)\dots (7)$

Where $\Delta\Theta$ is increment valued. The cosine and sine function can be expanded using the following standard trigonometric results:

$$\cos(\Theta+\Delta\Theta) =\cos(\Theta)\cos(\Delta\Theta) - \sin(\Theta)\sin(\Delta\Theta)\dots\dots (8)$$

$$\sin(\Theta+\Delta\Theta) =\sin(\Theta)\cos(\Delta\Theta) + \cos(\Theta)\sin(\Delta\Theta)\dots\dots (9)$$

Substituting Xn, Yn for equations (6), (7), (8), (9) gives:

$$X_{n+1} =R\cos(\Theta)\cos(\Delta\Theta) - R\sin(\Theta)\sin(\Delta\Theta)\dots\dots (10)$$

$$Y_{n+1} =R\sin(\Theta)\cos(\Delta\Theta) + R\cos(\Theta)\sin(\Delta\Theta)\dots\dots (11)$$

Substituting Xn, Yn for equation (4), (5) gives:

$$X_{n+1} =X_n \cos(\Delta\Theta) - Y_n \sin(\Delta\Theta)\dots\dots (12)$$

$$Y_{n+1} =Y_n \cos(\Delta\Theta) + X_n \sin(\Delta\Theta)\dots\dots (13)$$

The above equation show that if we know the value of cos(ΔΘ) and sin(ΔΘ) we can computer the whole sequence of points Xn (n=2, 3 …) and Yn (n=2, 3 …) from the starting point (x1, y1).The algorithm is:

```
1: x=r  : y=0 : th=0 : dt=1/r : ct=cos(dt) : st=sin(dt)
2: while (th<= 2*pi)
2.1:  plot pixel(x +xc , y +yc)
2.2: t=x
      x=x*ct - y*st;  ' Xn+1 =Xn*cos(ΔΘ) –Yn*sin(ΔΘ)
      y=y*ct + t*st;  ' Yn+1 =Yn*cos(ΔΘ)+Xn*sin(ΔΘ)
2.3:  th+=dt; 'that represent counter
3: end-while
```

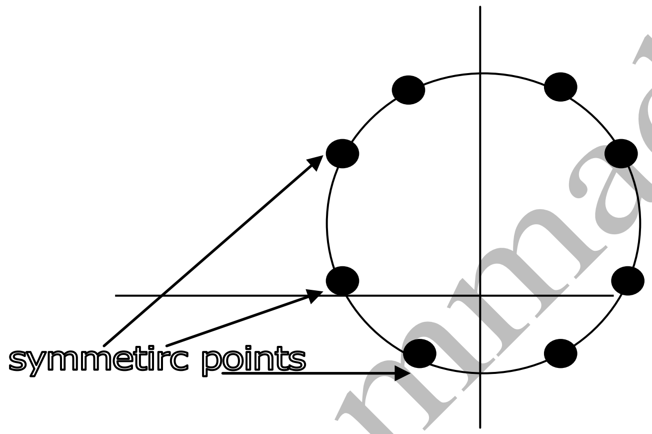
H.W/trace the algorithm that use incremental method to generate six points of the centered at (300,150) with a radius equal to 9 unit.

4.5 Symmetric of circle points:

A circle centered at the origin is defined by x^2+y^2=r^2. If a point ***(a, b)*** lies on this circle then so do seven other points ***(-a, b), (a, -b), (-a, -b), (b, a), (-b, a), (b, -a), (-b, -a), this*** can be verified by substation all eight points in to the circle equation.

We take advantage of this symmetry by calculating value for only one eight of a circle-namely an angular interval of 45.

If the first point is at (r, 0) then calculations are terminated when y=x. to find symmetric point on circle centered of (xc ,yc) we add xc to first coordinate and yc to the second coordinate for each of the eight points. And another advantage is the process is required 45 times but without symmetric points require 360 times or at least 180 times in circle equation. Then show modify preview algorithm by advantage of symmetric points.



4.5. A Symmetric in increment method of circles:

By advantage of symmetric points in circle we obtain modify algorithm is:

```
1: dt=1/r :  ct= cos(dt) :  st= sin(dt) :  x=r : y=0
2:While (y<x)
  2.1:  plot pixel(xc+ x, yc+ y),1
  2.1:plot pixel (xc+ x, yc- y),2
  2.3: plot pixel (xc- x, yc+ y),3
  2.4:plot pixel (xc- x, yc- y),4
  2.5:plot pixel (xc+y, yc+x),5
  2.6:: plot pixel (xc+y,yc-x),6
  2.7:  plot pixel (xc-y, yc+x),7
  2.8:: plot pixel (xc-y, yc-x),8
  2.9:t=x : x=x*ct –y*st :  y=y*ct +t*st
3.  end-while
```

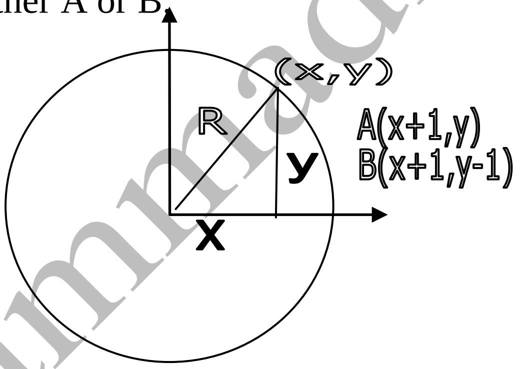
4.6 Bresenham circle algorithm:

The values of a circle centered at the origin are computed in a 45 sector from x=0 to x=y the remaining seven sectors are obtained from the eight point symmetry of the circle.

They values for this sector decrease as the x values increase if (0, r) is the starting point of the algorithm, then as x increase by one unit the y value either remains the same or is decrease by one unit. If(x, y) is a pixel on the circle, the next pixel is either A or B.

A :(x+1, y); to the right of previous point

B: (x+1, y-1); down and to the right of the previous point.



The algorithm proceeds to choose Pixel A or B by finding and comparing the distance from each pixel to the point on the circle that has x value of (x+1) these distances measure how far from the circle each pixel is the pixel with the smallest distances the best approximation on the circle. The square of the distance of pixel A from the center of the circle is **(x+1) ^2 + y^2.** The difference between this squared distance and the squared distance to the closest point on the circle is: ***d(A)= (x+1)^2 +y^2 -r^2***

for pixel B the distance is : ***d(B)= (x+1)^2 + (y-1)^2 – r^2***.

A point lies on the circle if its d value equals Ɵ, in order to determine which pixel A or B has the smallest d value we examine the sum *S= d(A)+d(B);* if S>0 we chose pixel B otherwise we choose pixel A. then algorithm:

```
1: x=0; y=r;
 2:While (x<=y)
2.1: plot pixel (xc+ x, yc+ y), 1 AND Symmetric 7 point(xc+ x, yc+ y)
2.2: da= (x+1)^2+(y)^2-(r)^2
2.3:db= (x+1)^2+ (y-1)^2 -(r)^2
2.4: s= da + db
2.5: if (s>0) y-=1
2.6 x:=x+1
3.While-end
```

Ex. /trace the Bresenham algorithm to generate six points of the circle centered at (300,150) with a radius equal to 9 unit.

Sol. / da = (x+1) ^2 + y^2 – R^2 , db= (x+1) ^2 + (y-1) ^2 – R^2

R^2 =81

x	y	x +xc	y +yc	d(A)	d(B)	s
0	9	300	159	1	-16	-15
1	9	301	159	4	-13	-9
2	9	302	159	9	-8	1
3	8	303	158	-1	-16	-17
4	8	304	158	8	-7	1
5	7	305	157	4	-9	-5
….	H.W					

Ex//Trace this circle where Center at (-20,40),radius is 7 units by Bresnham drawing

Sol// Xc= -20 , Yc = 40 , R=7 ➔ X=0➔ Y= 7

X	Y	A	d(A) $=x^2+y^2-r^2$	B	d(B) $=x^2+y^2-r^2$	Selected	X+Xc	Y+Yc
0	7	(1,7)	1+49-49➔1	(1,6)	1+36-49➔-12	A(1,7)		
1	7	(2,7)	4+49-49➔4	(2,6)	4+36-49➔-9	A(2,7)		
2	7	(3,7)	9+49-49➔9	(3,6)	9+36-49➔-4	B(3,6)		
3	6	(4,6)	16+36-49➔3	(4,5)	16+25-49➔-8	A(4,6)		
4	6	(5,6)	25+36-49➔12	(5,5)	25+25-49➔1	B(5,5)		
5	5	(6,5)	36+25-49➔12	(6,4)	36+16-49➔3	B(6,4)		
6	4	**Finish Complete all points with 8 symmetric points**						

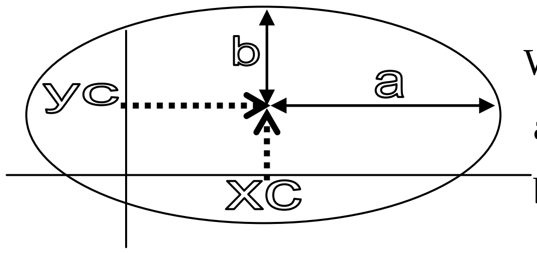
4.7 drawing ellipses:

An ellipse is a variation of a circle. Stretching a circle in one direction produces an ellipse, we shall examine ellipse that are stretched in the X or Y direction.

4.7. A The polynomial method of an ellipse:

The polynomial method of define an ellipse (figure below) is given by expression

$$\frac{(x-xc)^2}{a^2}+\frac{(y-yc)^2}{b^2}=1 \text{ Where (xc=0, yc=0)} \Rightarrow \frac{x^2}{a^2}+\frac{y^2}{b^2}=1$$



Where (xc, yc) = ellipse center.

a =length of major axis.

b =length of minor axis.

When the polynomial method is used to define an ellipse, the value of x is incremented from –a to a. for each step of x, each value of y is found by evaluating the expression

$$y=yc\pm b^2\sqrt{1-\frac{(xc-x)^2}{a^2}} \text{ If (xc=0, yc=0)} \Rightarrow y=\pm b^2\sqrt{1-\frac{x^2}{a^2}}$$

Finally, the result algorithm:

```
1: dt=1/((xr +yr)/2) : x=-xr;
2:while (x<=xr)
2.1: y1= +b*( (1- (x^2))/( xr^2))^(1/2)
2.2: y2=- b*( (1- (x^2))/( xr^2))^(1/2)
2.3: plot pixel(xc +x, yc+ y1), 1
2.4: plot pixel(xc +x, yc+ y1), 2
2.5: x=x+dt
3:wend 'goto 2
```

Note:- if a=b then polynomial convert same as circle equation

4.7. B <u>The polar representation of an ellipse</u>:

The polar equation for an ellipse centered at (xc, yc) and xr is the radius on the x-axis and yr is the radius on the y-axis.

$$X=xc+xr*\cos(\Theta)\ldots(13)$$

$$y=yc+yr*\sin(\Theta)\ldots(14)$$

The angle Θ assumes value from 0 to 2∏ radius, the values of xr and yr affect the shape of the ellipse, if yr>xr the ellipse is longer in the y direction, if xr=yr the equation produces a circle. Then algorithm is:

```
1: dt=1/ ((xr +yr)/2) :  th=0 : pi=22/7.0
2:Do While (th<=2*Pi)
2.1: x= xr*cos(th) : y= yr*sin(th)
2.2: Plot pixel(xc +x, yc +y), 8
2.3 th=th+dt
3:loop 'goto 2
```

H.w /trace the algorithm that use the polar representation to generate six points of the ellipse centered at (300, 150) with xr=10, yr=5.

4.7. C <u>Incremental method to drawing of ellipse:</u> similar of discuss in circle, but differ for equations :

$$X_{n+1}=X_n\cos(\Delta\Theta)-(Xr/Yr)*Y_n\sin(\Delta\Theta)\ldots..(16)$$

$$Y_{n+1}=Y_n\cos(\Delta\Theta)+(Yr/Xr)*X_n\sin(\Delta\Theta)\ldots..(17)$$

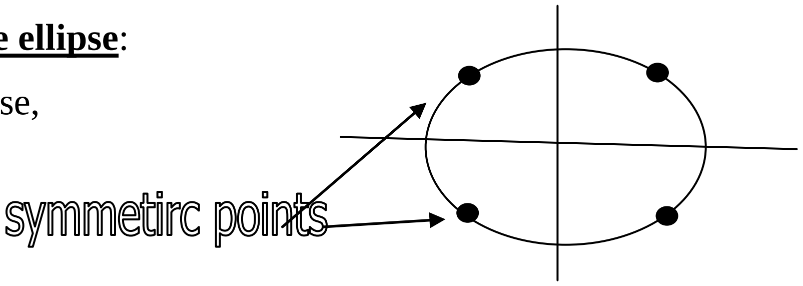
This algorithm is modify the polar representation (leave that of students to write this algorithm)

4.8. <u>Four point symmetry in the ellipse</u>:

If the point **(a, b)** lies on the ellipse,

then so do three other points:

(-a, b), (a, -b), (-a, -b)



3.9 Incremental drawing of an ellipse:

The following incremental equation for drawing an ellipse are derived from equations (12), (13)

Xn+1 =Xn cos (Δ Ɵ) – (Xr/Yr) * Yn sin (Δ Ɵ)….. (16)

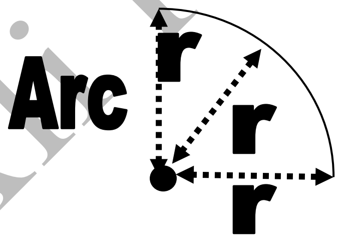
Yn+1 =Yn cos (Δ Ɵ) + (Yr/Xr) *Xn sin (Δ Ɵ)….. (17)

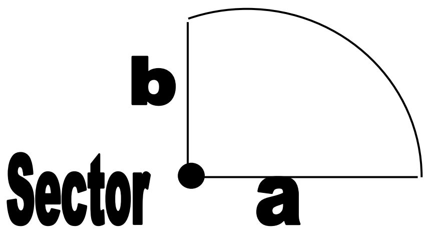
The algorithm by advantage of symmetric point for ellipse:

```
1: dt=1/((xr +yr)/2) :  ct=cos(dt) : st=sin(dt) :  x=xr : y=0
2: Do  ' Begin LOOP
2.1:plot pixel(xc +x, yc+ y), 9 :  plot pixel (xc +x, yc+ y),10
            plot pixel (xc +x, yc+y),11 :  plot pixel (xc +x, yc+ y),12
 2.2:   t=x
       x=x*ct-(xr/yr)*y*st
       y= y*ct-(yr/xr)*t*st
3: loop While( x>0 ) ' goto  step2
```

<u>Arc:</u> is part of circle but is needed start angle & end angle and distance of center of arc to surrounding is same as.

<u>Sector:</u> is part of ellipse but contains two line line1 from center to start angle & other line from center to end angle & distance between center of sector to surrounding is different as..





1st Semester 2D Computer Graphics

Ali Hassan Hammadie

Part Five
(2D Transformations)

5. 2D-Transformations: Two-dimensional transformation is the basic transformation that can be done on the image. Performing appropriate geometric or coordinate transformation on the object does the manipulation of image. Using transformation we can change the size of polygon or any object, change its position, and rotate it easily.

5.1 Fundamental Transformation

There are three basic transformations:

1. Translation (shift OR move).
2. Scaling.
3. Rotation.
4. Reflection (Mirror).
5. Shear.

5.2: Translation:

Consider a point P(x, y). we can translated it means shift it to new position p`(x`, y`) by adding tx and ty in y where Tx and Ty are translating factor.

Mathematically this can be represented as:

$$x` = x + Tx$$

$$y` = y + Ty$$

Note1: *Using coordinate system the translating factor are*

If Tx>0 then point moves to the right.

If Tx<0 then point moves to the left.

If Ty>0 then point moves to the up.

If Ty<0 then point moves to the down.

Note2: *using coordinate system in screen then translating factor are*

If TX>0 then point moves to the right.

If TX <0 then point moves to the left.

If Ty >0 then point moves to the down.

If Ty <0 then point moves to the up.

5.3: Scaling:

To changes the size of an object such that we can magnify the size or reduce it. This process is called scaling. Suppose P(x,y) is the point which we want to scale, after scaling we get new point having coordinates as: $p^s(x^s, y^s)$:➔

$$x^s = x * Sx$$

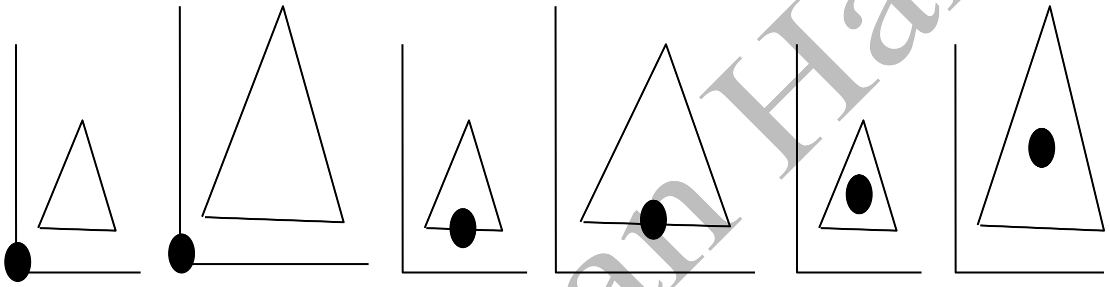
$$y^s = y * Sy$$

where *Sx* and *Sy* are scaling factors.

Whenever scaling is preformed there is one point that remains of this same location called the fixed point of scaling. If the fixed point is at the origin(0,0) a point (x, y) can be scaled by a factor Sx in the x direction and Sy in the y direction to the point (Xf ,Yf).

$x^s = x * Sx$; $y^s = y * Sy$; ***(apply in fixed point in origin point)***

If Sx<>Sy the resulting object is a distortion of the original object.



It is possible to choose any point (xf, yf) as the fixed point of scaling of scaling by performing the following steps:-

1. translate the point(xf, yf) to the origin (0,0) every point(x, y) become the new point (x`, y`) ➔ $x` = x - xf$ ; $y` = y - yf$;
2. scale the translated points with the origin as the fixed point:

   $x^s = x` * Sx$; $y^s = y` * Sy$;
3. translate the origin back to the fixed point(xf, yf):

   $x^g = x^s + xf$; $y^g = y^s + yf$;

These three steps can be combined in the following equation that scales a point (xf, yf).

$$x^g = (x - xf) * Sx + xf$$

$$y^g = (y - yf) * Sy + yf$$

5.4: Rotation:

Rotation can be performed about the origin or about a pivot point. But the rotate direction either positive oriented (anticlockwise) where angle is positive or negative oriented (clockwise) where angle is negative

5.4.A Rotation about the origin:

Mathematically the counter anticlockwise rotation of a point(x,y) about the origin an angle Θ. To produce the point (x^R, y^R) is represented by:

$$X^R = X \cos(\Theta) - Y \sin(\Theta)$$

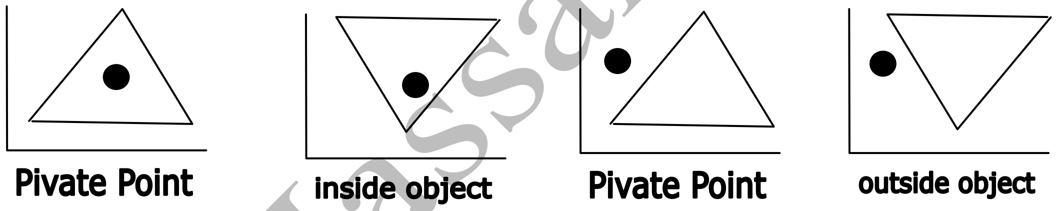
$$Y^R = Y \cos(\Theta) + X \sin(\Theta)$$

Mathematically, counter anticlockwise is considered the positive rotation direction, to rotate in a clockwise direction change to angle Θ to $-\Theta$. *Then:* **$\cos(-\Theta) = \cos(\Theta)$, $\sin(-\Theta) = -\sin(\Theta)$;** clockwise rotation can be represent by: $X^R = X \cos(\Theta) + Y \sin(\Theta)$

$$Y^R = Y \cos(\Theta) - X \sin(\Theta)$$

5.4. B Rotation about a pivot point:

After an object is rotated about a specified pivot point, it is still the same distance away from the pivot point but its orientation has been changed:



To rotate a point an angle Θ about a pivot point three steps are required:-

1) translate the pivot point (xp, yp) to the origin (0,0) this is done by translating the point (x, y) to the new point (x', y') where ➔ $x' = x - xp$, $y' = y - yp$;
2) rotate the translated point(x', y') Θ degree about the origin to obtain the new point (x^R, y^R) where

$$X^R = X' * \cos(\Theta) - Y' * \sin(\Theta).$$

$$Y^R = Y' * \cos(\Theta) + X' * \sin(\Theta).$$

3. translate the center of rotation back to the pivot point(xp,yp) thus the point (x^g, y^g) is translated to: $x^g = x^R + xp$; $y^g = y^R + yp$;

These three steps can be combined in the following equation for the counterclockwise rotation of an point by an angle Θ about pivot point (xp, yp).

➔ X^g=*(x-xp)*cos(Θ)-(y-yp)*sin(Θ)+xp*.

Y^g =*(y-yp)*cos(Θ)+(x-xp)*sin(Θ) +yp*

5.4 Inverse transformations:

The undo a transformations we perform its inverse which is the same transformation but with inverse parameters.

Translate: -H, -V.

Rotate: - Θ.

Scale: 1/Sx, 1/Sy.

ملاحظة: 1. لتحويل الزواية من مقياس الدرجات الى تقدير الدائري نقوم بضرب الزواية بـ Л ونقسمها على 180.

5.5 Mirror reflection about an axis:

The mirror reflection about the x axis is: ***xm=x ; ym= -y;***

The mirror reflection about the y axis is: ***xm= -x ; ym=y;***

The mirror reflection about the origin point is: ***xm= -x ; ym= -y;***

The mirror reflection about the line *y=x*: ***xm=y ; ym=x;***

The mirror reflection about the line *y=-x*: ***xm= -y ; ym= -x;***

Ex//Reflect (8, -11) all types.

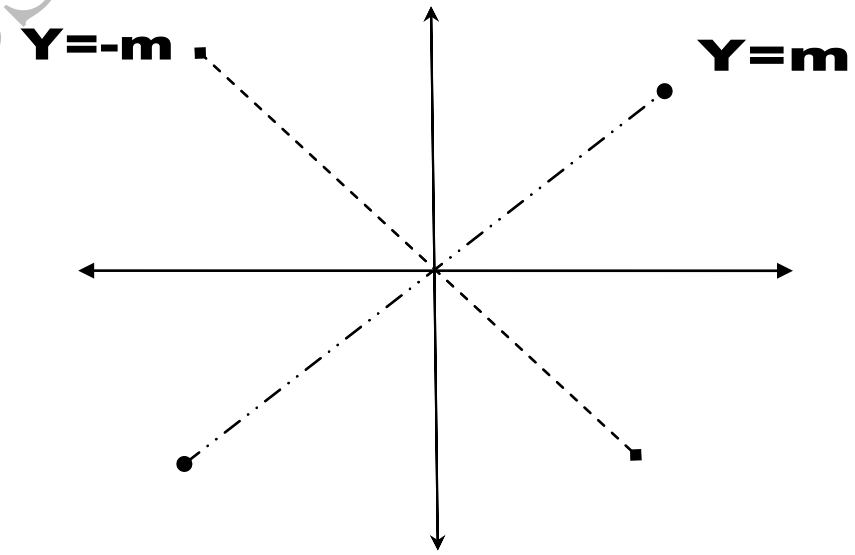
Sol// $(8,-11) \overset{x}{\Rightarrow} (8,11)$

$(8,-11) \overset{y}{\Rightarrow} (-8,-11)$

$(8,-11) \xrightarrow{Origin} (-8,11)$

$(8,-11) \xrightarrow{y=x} (-11,8)$

$(8,-11) \xrightarrow{y=-x} (11,-8)$



5.6 Matrix representation of transformations:

Each of two dimensional transformations can be represented as a product of the row vector (x y 1) and a is 3*3 matrix.

The following are the matrix representation of the transformation.

(i) Translation: $(x' \; y' \; 1) = (x \; y \; 1) * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ tx & ty & 1 \end{bmatrix}$

(ii) Anticlockwise rotation: $(x^R \; y^R \; 1) = (x \; y \; 1) * \begin{bmatrix} \cos(n) & \sin(n) & 0 \\ -\sin(n) & \cos(n) & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Note: if do you want in counter clockwise? Then, place "minus" in n value, therefore the inverse sign sin (n) in array above.

(iii) Scaling: $(x^S \; y^S \; 1) = (x \; y \; 1) * \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$

(vi) a. mirror reflection in x-axis: $(x^{mx} \; y^{mx} \; 1) = (x \; y \; 1) * \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

(v). mirror reflection in y-axis: $(x^{my} \; y^{my} \; 1) = (x \; y \; 1) * \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

(iv). mirror reflection in origin point: $(x^{mO} \; y^{mO} \; 1) = (x \; y \; 1) * \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

(iiv). mirror reflection in Line Y=X: $(x^{L+} \; y^{L+} \; 1) = (x \; y \; 1) * \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

(iiiv) mirror reflection in Line Y= -X: $(x^{L-} \; y^{L-} \; 1) = (x \; y \; 1) * \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Ex. */rotate the object defined by (43,88), (84,50), (66,72) in a rotate at antilock wise by angle 63 then scale it with the scaling factor Sx=1.3, Sy=2.where(65,-40) is fixed point. Sol/*

$$\begin{pmatrix} 43 & 88 & 1 \\ 84 & 50 & 1 \\ 66 & 72 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ -65 & 40 & 1 \end{pmatrix}\begin{pmatrix} \cos 63 & \sin 63 & 0 \\ -\sin 63 & \cos 63 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1.3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 65 & -40 & 1 \end{pmatrix}$$

H.W/ rotate the object define by (54,68), (104,66), (70,102) by counter anticlockwise by 37 degree after scaling it with the scaling factor Sx=3, Sy=2 using the fixed point(54,68) and mirror reflection in origin point.

5.7 **Mirror about arbitrary line:** arbitrary line is line where slop is neither equal 1 nor -1 & additions that any points belong in this line are not in origin points**. But** this line has contain either Two endpoints or equation of this line, the mirror of this arbitrary line that following steps:

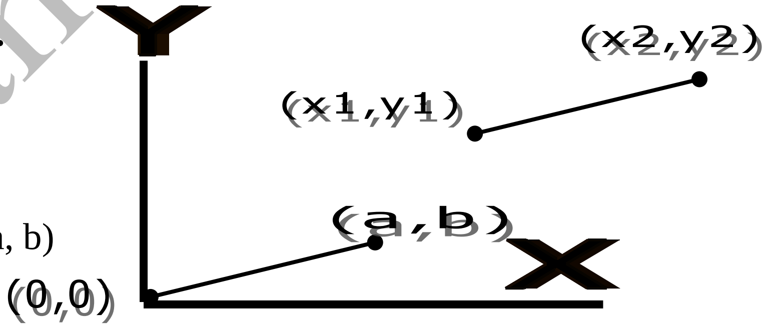
Steps of mirror:

Let line has endpoints A(x1, y1) and B(x2,y2) let a=(x2-x1), b=(y2-y1)

1. Translate the point(x1, y1) to origin.

Tr(-x1, -y1)= $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -X1 & -Y1 & 1 \end{pmatrix}$

After this translation the direction vector (a, b) define the rotation axis as follows



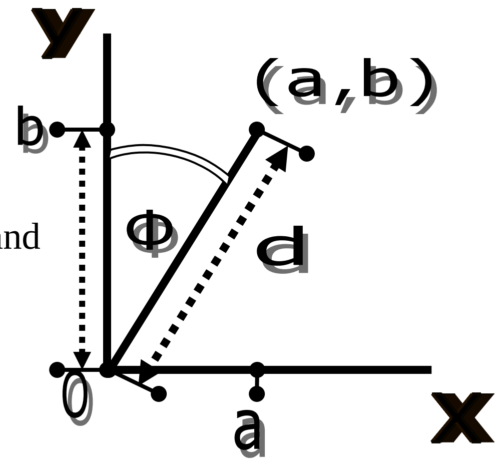
2. Rotate about the angle until the rotation axis corresponds to the y-axis.

This can be considering being a rotation about the origin.

With the axis coming out of paper

The point (a, b) is rotated anticlockwise by **Φ** degree until the line corresponds to the y-axis. We have find the sin **Φ** and cos **Φ** we find that distance from the origin to (0,b) is :

$\sqrt{a^2 + b^2} = d$➔**Sin Φ= a /d ; Cos Φ= b/d**



Substituting these values into the y-axis rotation matrix we have:

R(Φ)= $\begin{pmatrix} b/d & a/d & 0 \\ -a/d & b/d & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Now the point(a, b)has been transformed to the point (d,0).

3. Reflection about on y-axis➔ $\begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

4. Inverse step 2 by using – Φ in step 2. ➔ R(-Φ)= $\begin{pmatrix} b/d & -a/d & 0 \\ a/d & b/d & 0 \\ 0 & 0 & 1 \end{pmatrix}$

5. Inverse step 1: by change sign of movement factors➔ **Tr(x1, y1)=** $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ X1 & Y1 & 1 \end{pmatrix}$

5.8 <u>Mirror about arbitrary point:</u> let point(x, y) is arbitrary then reflection in this point has three steps:

1. Shift arbitrary point in origin the movement factor is –x, -y. ➔ **Tr(-x, -y)=** $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -X & -Y & 1 \end{pmatrix}$

2. Reflection about origin point. ➔ $\begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

3. Inverse step 1: the movement factor x, y. ➔**Tr(x, y)=** $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ X & Y & 1 \end{pmatrix}$

==

Reflect figure { P(-1,1), Q(-3,2), R(5,-2), S(-2, -4)}around line start(71,-69) and end(101,-89).

Sol/ Dx=30, Dy= -20, d= $\sqrt{1300} = 10\sqrt{13}$ **, Cos=** $\frac{2}{\sqrt{13}}$ **, Sin=** $\frac{3}{\sqrt{13}}$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ -71 & 69 & 1 \end{bmatrix} \begin{bmatrix} \frac{2}{\sqrt{13}} & \frac{3}{\sqrt{13}} & 0 \\ \frac{-3}{\sqrt{13}} & \frac{2}{\sqrt{13}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{2}{\sqrt{13}} & \frac{-3}{\sqrt{13}} & 0 \\ \frac{3}{\sqrt{13}} & \frac{2}{\sqrt{13}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 71 & -69 & 1 \end{bmatrix}$$

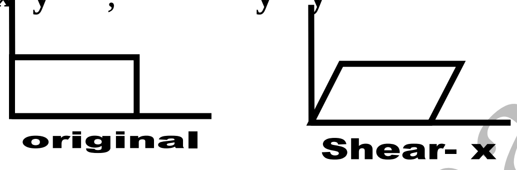
5.9 Shearing:

Shearing transformations make the objects distort their shapes in either x or y or both directions. It can be imagined as the object is made up of very thin layers and they are slid over each other fixing the base in case of single directional shearing.

1. To shear in x direction only use shearing matrix in the equation as:

x`=x +shx\*y , y`=y

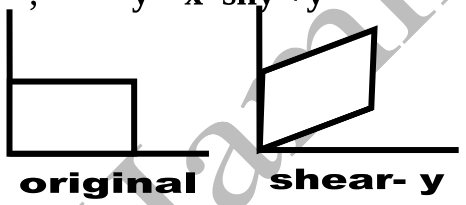
$$\begin{pmatrix} 1 & 0 & 0 \\ shx & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



2. To shear in y direction only use shearing matrix in the equation as:

x`=x , y`=x*shy +y

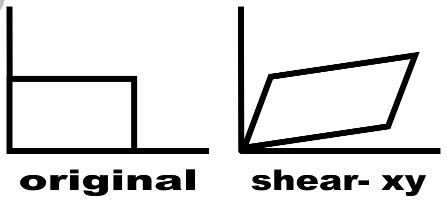
$$\begin{pmatrix} 1 & shy & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



3. To shear in x and y directions both :

x`=x +shx\*y , y`=x*shy +y;

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



All shear all types at center (0, 0)

But if you need shear at fixed Point needs {shift, Shear, return}

H.W//If you have figure { P(-1,1), Q(-3,2), R(5,-2), S(-2, -4)}, to Rotate by (31π/12) clockwise & scale in X by 7 and reduce size y by 9 with Shear in X by -1.4 and Y by 3.5 at (-10,-66),finally reflect about y=-x

1. Write in Matrix Form.

2. Final Result on this Figure

1st Semester 2D Computer Graphics

Ali Hassan Hammadie

Part six

(2D Mapping)

{Window, viewport}

6. Introduce for Window and Viewport

A window is specified by four worlds coordinate: wXmin, wXmax, wYmin, and wYmax. Similarly, a viewport is described by four normalized device coordinates: vXmin, vXmax, vYmin, and vYmax. The objective of window-to-viewport mapping is to convert the world coordinates (WX, WY) of an arbitrary point to its corresponding normalized device coordinates (VX, VY). To maintain the same relative placement of the point in the viewport as in the window, we require:

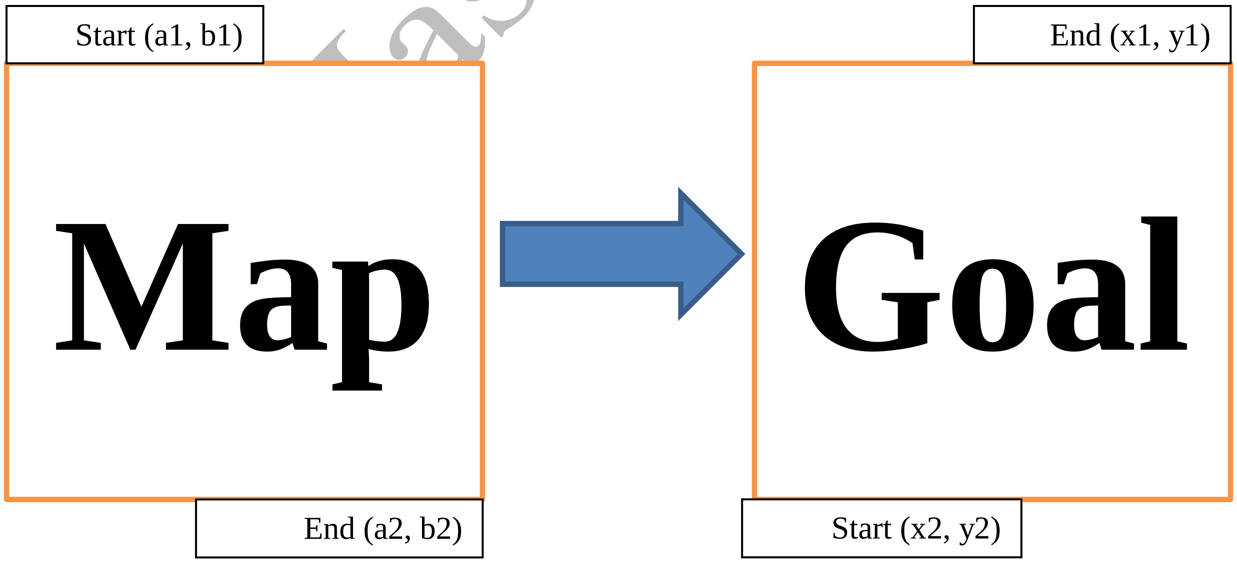
$$\frac{(W_X - W_{Xmin})}{(W_{Xmax} - W_{Xmin})} = \frac{(V_X - V_{Xmin})}{(V_{Xmax} - V_{Xmin})} \quad AND \quad \frac{(W_Y - W_{Ymin})}{(W_{Ymax} - W_{Ymin})} = \frac{(V_Y - V_{Ymin})}{(V_{Ymax} - V_{Ymin})}$$

$$VX = VXmin + (WX - WXmin) * Sx \quad where \quad Sx = \frac{(V_{Xmax} - V_{Xmin})}{(W_{Xmax} - W_{Xmin})}$$

$$VY = VYmin + (WY - WYmin) * Sy \quad where \quad Sy = \frac{(V_{Ymax} - V_{Ymin})}{(W_{Ymax} - W_{Ymin})}$$

Note: the coordinate of window is world coordinate but the view port is the coordinate x = 0 to 1 and y = 0 to 1.

This Part talk's about two worlds (windows) real world and viewport (device), how can represent real-world on device-world or device-world into real-world especially both worlds are 2D.



Look these worlds (map), (goal) may be:-

1-distance width and height are different. For example:

Map width=100, height = 56 while goal width=23, height =213

2-location start and end in both worlds are not same side direction. For example:

Map Start= (left, up), End = (right, down) while goal Start= (left, down), End= (right, up)

3- Range or boundaries of two worlds are too different. For example

Map Start= (30, -5), End = (-10, 50) ➔ X = [**30** ... **-10**], Y = [**-5** ... **50**]

While goal Start= (-9, 12), End= (-28, -70) ➔ X = [**-9** ... **-28**], Y = [**12** ... **-70**]

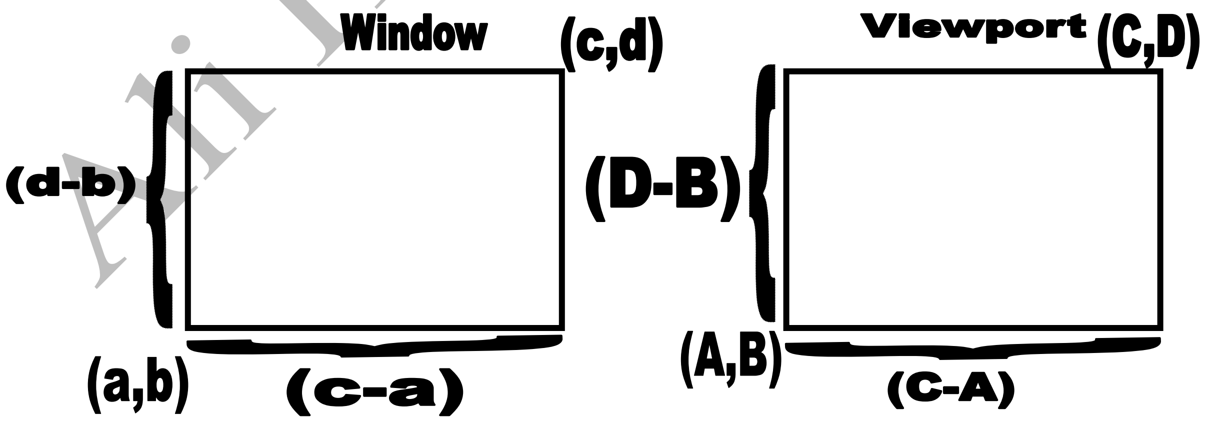
To solve this approach need three transformations to:-

A. Shift Start location Map into origin point by $Tx= -X_{start}^{map}$, $Ty= -Y_{start}^{map}$
B. Calculate Width & Height for both world and keep direction (End $_{world}$ - Start $_{world}$)

$$Sx = \frac{(X_{end}^{goal} - X_{start}^{goal})}{(X_{end}^{map} - X_{start}^{map})} , Sy = \frac{(Y_{end}^{goal} - Y_{start}^{goal})}{(Y_{end}^{map} - Y_{start}^{map})}$$

C. Return at start location of goal by $Tx= X_{start}^{goal}$, $Ty= Y_{start}^{goal}$

There three operations in the sequence. First we have the 'window shift', when we shift the start position of the window to the origin on the object; next we scale the window dimensions to the dimensions of the viewport and imagine the two origins to be coincident; finally we have the 'viewport shift' when we shift the start position of the viewport from the screen origin to its proper position. We use the coordinates and dimensions shown in figure bellow



1. And see that matrices are as follows. The window shift is given by

$$w = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -a & -b & 1 \end{pmatrix}$$

2. The scaling involves a factor of **(C-A)/(c-a)** in the x-direction and **(D-B)/ (d-b)** in the y-direction, so the matrix for local scaling is:

$$s = \begin{pmatrix} (\mathbf{C}-\mathbf{A})/(\mathbf{c}-\mathbf{a}) & 0 & 0 \\ 0 & (\mathbf{D}-\mathbf{B})/(\mathbf{d}-\mathbf{b}) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

3. The viewport shift is given by: ➔ $w = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ A & B & 1 \end{pmatrix}$

Multiplying these matrices in order we get : **M=W S V** *which is the matrix which performs this viewing transformation.*

Ex. / *find the normalization transformation that maps a window whose lower left corner is at (1,3) and upper right corner is at (3,5) onto a viewport that has lower left corner at (0.2,0.5) and upper right corner (0.8,0.9) . What position of point p in window is (2.5,3.5) onto viewport coordinate*?

Sol. The matrix for the window shift is: ➔$w = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & -3 & 1 \end{pmatrix}$

The scaling matrix is: ➔ $s = \begin{pmatrix} (\mathbf{0.8}-\mathbf{0.2})/(\mathbf{3}-\mathbf{1}) & 0 & 0 \\ 0 & (\mathbf{0.9}-\mathbf{0.5})/(\mathbf{5}-\mathbf{3}) & 0 \\ 0 & 0 & 1 \end{pmatrix}$

The matrix for the viewport shift is:➔ $v = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0.2 & 0.5 & 1 \end{pmatrix}$

When we multiply these 3 matrices together (remembering that order matters) we get **M=WSV**

$$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & -3 & 1 \end{pmatrix} \begin{pmatrix} (\mathbf{0.8}-\mathbf{0.2})/(\mathbf{3}-\mathbf{1}) & 0 & 0 \\ 0 & (\mathbf{0.9}-\mathbf{0.5})/(\mathbf{5}-\mathbf{3}) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0.2 & 0.5 & 1 \end{pmatrix}$$

The point P has coordinate (2.5, 3.5) so its homogeneous vector is [2.5 3.5 1].

We multiply this by the matrix M and obtain:

$$[2.5 \quad 3.5 \quad 1]\begin{pmatrix} 0.3 & 0 & 0 \\ 0 & 0.2 & 0 \\ -0.1 & -0.1 & 1 \end{pmatrix} = [0.65 \quad 0.6 \quad 1]$$

Thus the coordinate of P* in the viewport are (0.65, 0.6)

HW/ figure consist A= 4i+7j, B= -5i+ j, C= -i -6j is drawing on area where Starting at (6,-8), Ending at (-10, 10), what is figure points that it draw on area that is starting (-2, 7) and ending (2, 2).

1. Find Result (H.W)

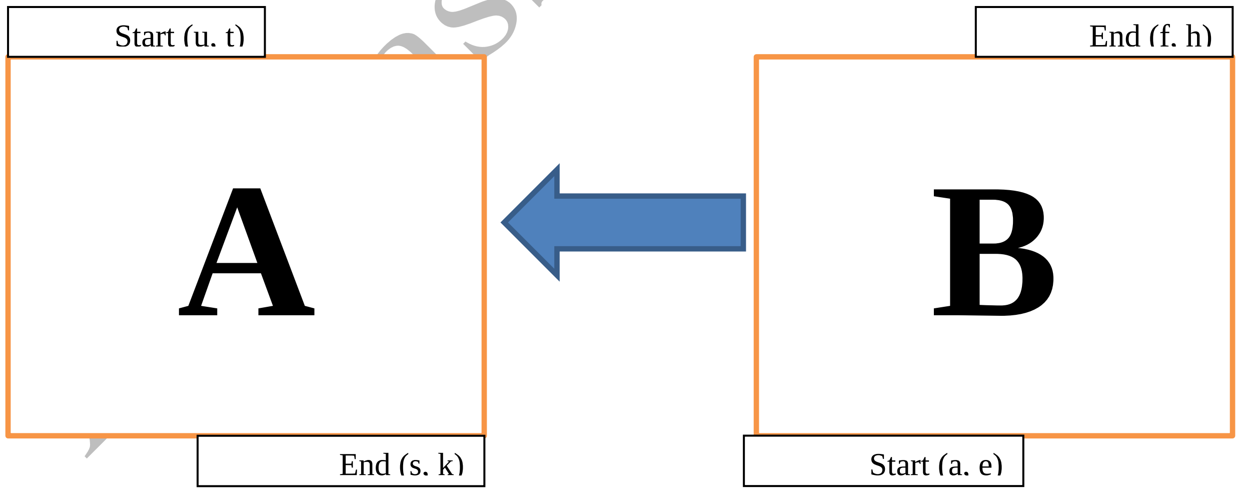
Step1: Tx= -6, Ty= 8 ➔

Step2: Sx= (2+2)/(-10-6)➔

Step3: Tx= -2 , Ty= 7➔

2. Represent in Matrix Transform.
3. U(3,-3) location on is starting (-2,7) and ending (2, 2).

H.W/ Prove Area B become Area A



- Hint: Area B (Map Area) Start (a, e)-end (f, h) while Area A (goal Area) Start (u, t)-end (s, k)

Problem how can manipulate to convert Goal Area?

1st Semester 2D Computer Graphics

Ali Hassan Hammadie

Part seven
(2D Clipping)

7. ***Clipping***

The clipping includes the point clipping, line clipping, and polygon clipping.

7.1 **Point clipping:**

Point clipping is essentially the evaluation of the following inequalities:

Xmin ≤ x ≤ Xmax *and* Ymin ≤ x ≤ Ymax

Where Xmin, Xmax, Ymin and Ymax define the clipping window. A point (x, y) is considered inside the window when the inequalities all evaluate to true.

7.2 **Line clipping:**

Lines that do not interest the clipping window are either completely inside the window or completely outside the window. On the other hand, a line that interests the clipping window is divided by the intersection point(s) into segments that are either inside or outside the window. The line clipping process is dividing into two phases:

(1) Identify those lines which intersect the clipping window and so need to be clipped and

(2) Perform the clipping.

All lines fall into one of following clipping categories:

(a) Visible—both end point of the line lie within the window.

(b) Not visible—the line definitely lies outside the window. This will occur if the line from (x1,y1) to (x2,y2) satisfies any one of the following four inequalities:

x1, x2 > Xmax* OR *y1, y2 > Ymax

***x1, x2 < Xmin* OR *y1, y2 < Ymin* 'See next section**

(c) Clipping candidate—the line is in neither category 1 and 2

If endpoint is above the window(y>Ymax) or endpoint is below the window(y<Ymin) ***the Y clipped is equaled boundaries of window*** and **X clipped is equaled *X=(y-y1)/m + x1*.**

If endpoint is right of window(x>Xmax) or endpoint is left of window(x<Xmin) ***the X clipped is equaled boundaries of window*** and **Y clipped is equaled *y=(X-x1) m + y1.***

7. 3 **Clipping Flag** to check line is inside (all or part) and outside need 4bit to flag as following

X-Min	Y-Min	X-Max	Y- Max

For suppose window (-30, 40), (40, -40) check with clipping line

Line1(70,0),(0,70) & *line2 (-50,10),(0,-30)* & **line3 (50,70),(60,-70)**

Sol:- Xmin=-30, Xmax=40, Ymin=-40, Ymax=40

Line1➔(70,0) [0010] because X>Xmax

Line1➔ (0,70) [0001] because Y>Ymax

Finally [0010] AND [0001] ➔ [0000] 'need Clipping

Line2➔(-50,10) [1000] because X<Xmin

Line2➔ (0,-30) [0000] point inside

Finally [1000] AND [0000] ➔ [0000] 'need Clipping

Line3➔(50,70) [0011] because X>Xmax & Y>Ymax

Line3➔ (50,-70) [0110] because X>Xmax & Y<Ymin

Finally [0011] AND [0110] ➔ **[0010]** it line is outside {not visible}

1001	**0001**	**0011**
1000	**0000**	**0010**
1100	**0100**	**0110**

Golden Notes

- Line Visible➔And = 0 ,Or = 0
- Line Invisible➔And < > 0 ,Or < > 0
- Otherwise Line need Clipping.
- Point line Clipping must be flag is 0000

Window area { (50,-50)- (-20,30) clip L(60,-10)-(100,-90), M(-10,-60)-(10,90) on this window

Sol// -20 ≤ x ≤ 50 AND -50≤ y ≤ 30

xmin	ymin	xmax	ymax

(60, -10)➔0010

(100,-90) ➔0110

0010 and 0110 ➔0010

0010 or 0110➔ 0110 invisible not need clipping

note

Visible➔ x1,x2 ≥ xmin AND x1,x2 ≤ xmax AND y1,y2 ≥ ymin AND y1,y2 ≤ ymax

➔P1 AND P2=0000 With P1 OR P2=0000 =➔ Visible

Invisible➔ x1,x2 < xmin OR x1,x2 > xmax OR y1,y2 < ymin OR y1,y2 > ymax

➔P1 AND P2<>0000 WITH P1 OR P2<>0000 =➔ invisible

M(-10,-60)-(10,90)➔(0100) – (0001)

(0100) AND (0001)=0000 With (0100) OR (0001)=0101 line m needs clipping

(Detection)➔ M=$\frac{-60-90}{-10-10}=\frac{15}{2}$ ➔

-10i-60j ➔Down Y=Ymin➔ Y= -50

$\frac{15}{2}=\frac{y-y1}{x-x1}$ ➔ $\frac{15}{2}=\frac{-50+60}{x+10}$➔$\frac{15}{2}=\frac{10}{x+10}$➔ x+10= $\frac{20}{15}$➔ X= $\frac{20}{15}$- - 10➔ X= $\frac{4}{3}$ - $\frac{30}{3}$= -26/3 = -8.66666

10i+90j➔ up ➔ Y=ymax ➔ y=30 ➔ $\frac{15}{2}=\frac{y-y1}{x-x1}$ ➔ $\frac{15}{2}=\frac{30-90}{x-10}$ ➔ $\frac{15}{2}=\frac{-60}{x-10}$ ➔ $x-10=\frac{-120}{15}$ =2

Clip F (-40,-60)-(10,70) ➔ (1100) – (0001) on Window{ $-20 \le x \le 50$ AND $-50 \le y \le 30$}

(1100) AND (0001) =0000 with (1100) OR (0001) =1101 line F needs clipping (Detection)

M= (-60-70)/(-40-10)=13/5

(10,70) ➔ have two locations{ either UP or Left)} assume up-side➔ Y=Ymax= 30

➔$\frac{13}{5}=\frac{30-70}{x-10}$➔$x=\frac{-200}{13}+10$➔ $\frac{-200}{13}+\frac{130}{13}$ ➔ x= $\frac{-70}{13}$➔ X≈ -5.384

Intersection point in up-side is (-5.384, 30) because it is point visible

But (-40,-60) have two locations too { either Down or Left)}

If suppose down then Y=ymin ➔Y= -50

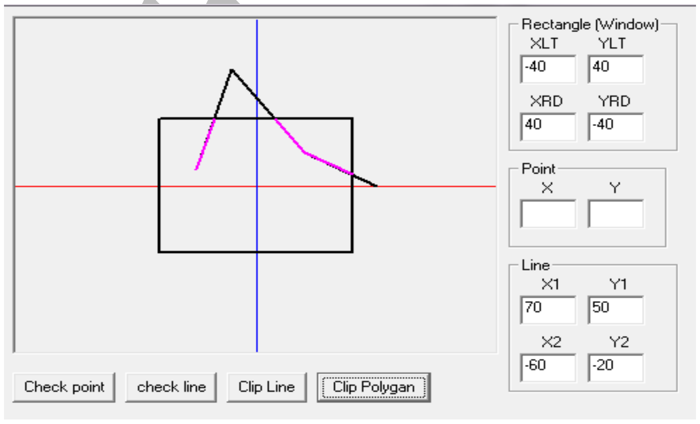
$\frac{13}{5}=\frac{-50+60}{x+40}$ ➔x = $\frac{50}{13}$ – 40 = $\frac{50}{13}$ - $\frac{520}{13}$= $\frac{-470}{13}$ ➔X ≈ -36.1538 ➔ X < Xmin (Discard)يهمل

Because $-20 \le x \le 50$ and intersection(-36.1538, -50) in not Visible (Suppose Down Discard)

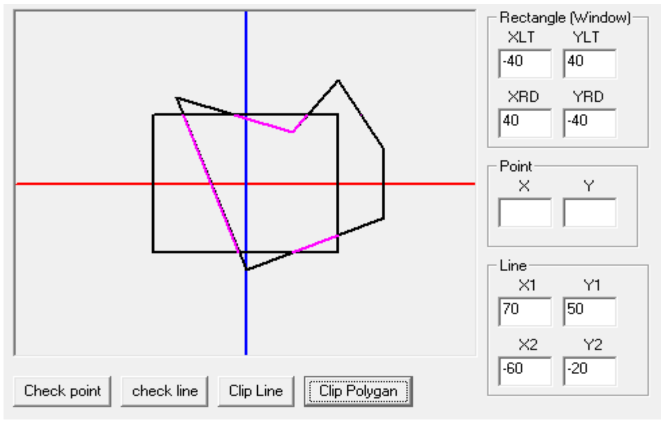
Therefore Guarantee suppose is left then x=xmin ➔X= -20

$\frac{13}{5}=\frac{y+60}{-20+40}$ ➔y = $\frac{260}{5}$ – 60 = 52 - 60 = –8 ➔y = -8

Intersection point in Left-side is (-20, -8) because it is Visible



Rectangle (Window)
XLT -40 YLT 40
XRD 40 YRD -40
Point
X Y
Line
X1 70 Y1 50
X2 -60 Y2 -20



Rectangle (Window)
XLT -40 YLT 40
XRD 40 YRD -40
Point
X Y
Line
X1 70 Y1 50
X2 -60 Y2 -20

1st Semester 2D Computer Graphics

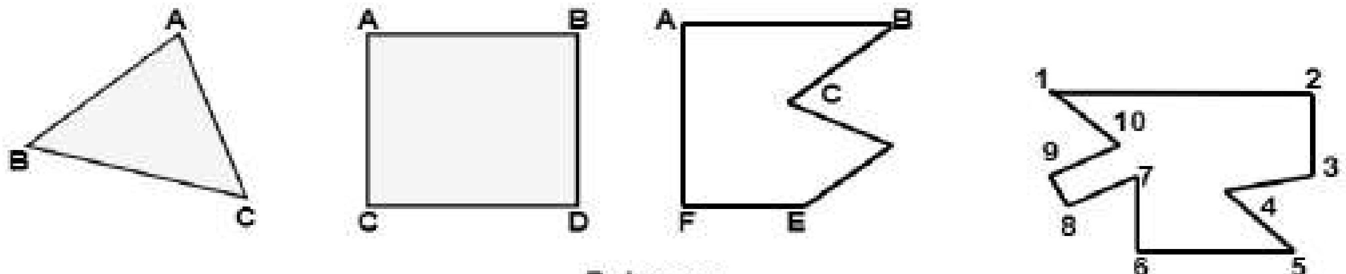
Ali Hassan Hammadie

Part eight
(2D Polygon)

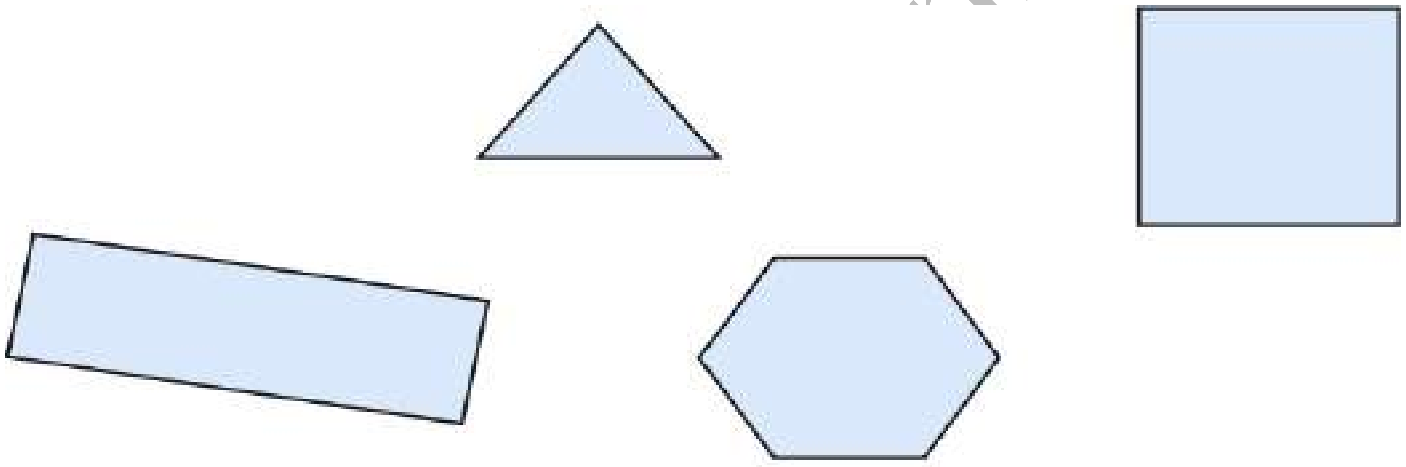
8. Polygon is a representation of the surface. It is primitive which is closed in nature. It is formed using a collection of lines. It is also called as many-sided figure. The lines combined to form polygon are called sides or edges. The lines are obtained by combining two vertices.

Example of Polygon: {**Triangle, Rectangle, Hexagon, Pentagon …etcs}**

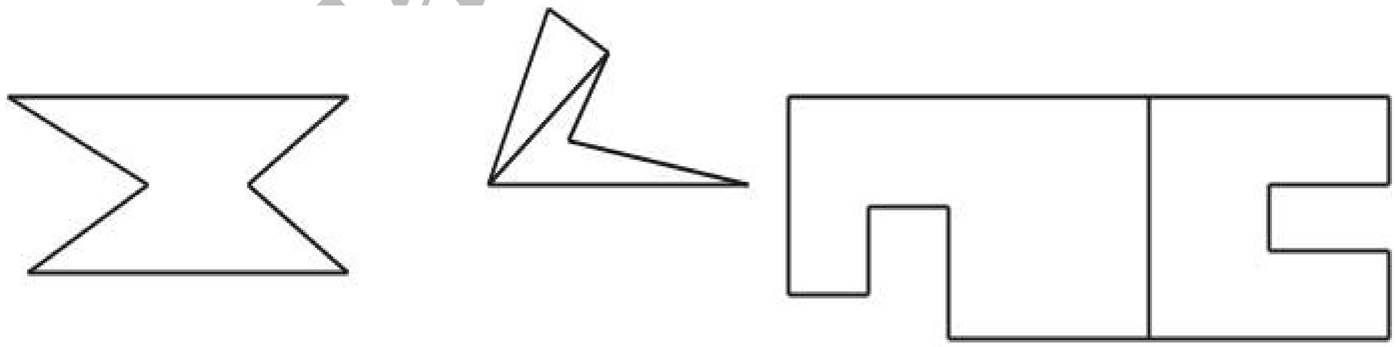
Following figures shows some polygons.



Polygons



Convex Polygon



Concave Polygon

Types of Polygons

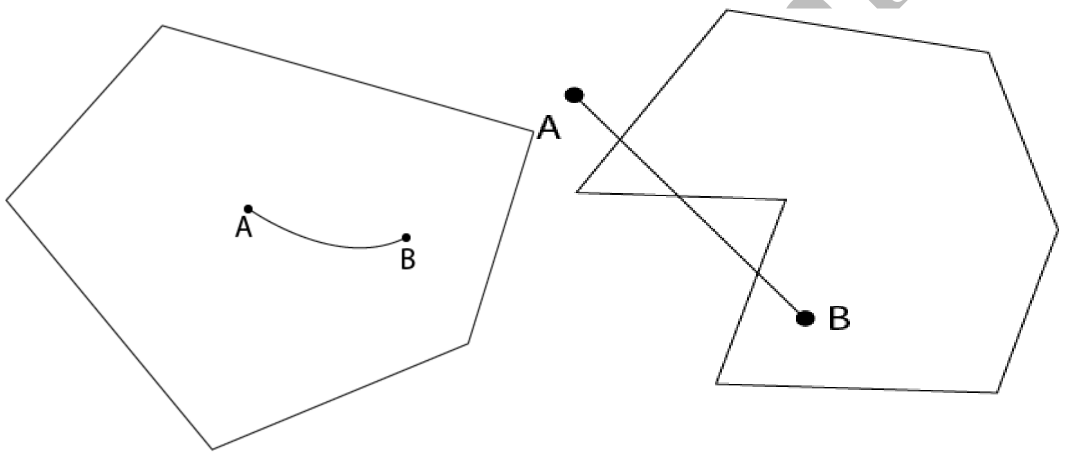
1. Concave
2. Convex

A polygon is called convex of line joining any two interior points of the polygon lies inside the polygon. A non-convex polygon is said to be concave. A concave polygon has one interior angle greater than 180°. So that it can be clipped into similar polygons.

8.1 <u>Convex</u> if we take any two points inside of polygon and connect among by line and this line is fully inside the polygon.

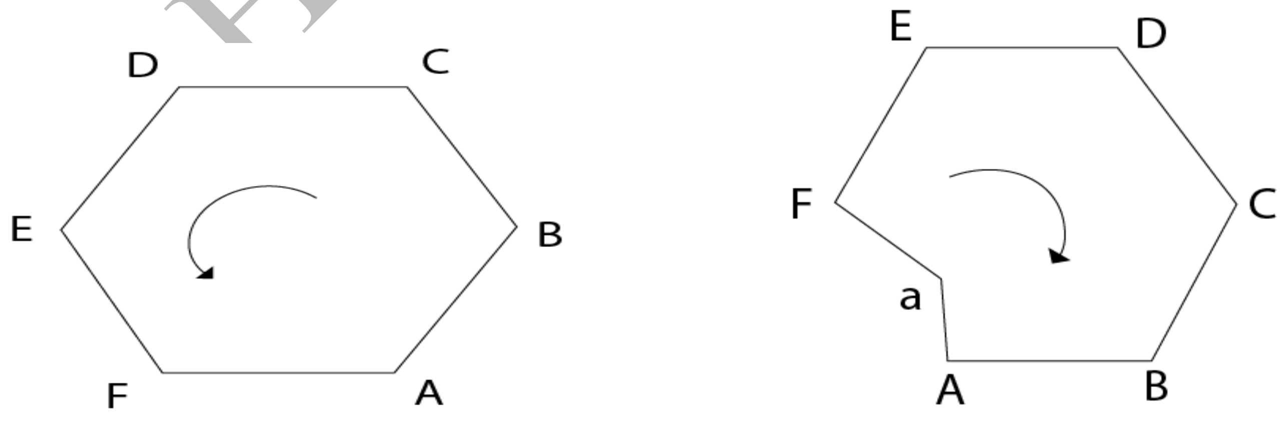
8.2 <u>Concave</u> if we take any two points inside of polygon and connect among by line and this line is not fully inside the polygon.

<u>Note:</u> *the head of polygon is called by* ***Vertices.***



Convex polygon Concave polygon

A polygon can be positive or negative oriented. If we visit vertices and vertices visit produces counterclockwise circuit, then orientation is said to be positive.



Polygon with positive orientation Polygon with negative orientation

If the sequences of capes of the polygon are anticlockwise called **positive orientation** otherwise called **Negative orientation** to know the point is inside the polygon we need:

1- Know the polygon positive orientation or Negative orientation.

2- Determine the point is inside of polygon by the equation:

$$C=(x2-x1)(y-y1)-(y2-y1)(x-x1)$$

If value of **C** is **positive** then point in left side otherwise is right side, if polygon is positive orientation then the point in left side is inside in polygon but if point in right side that is outside polygon and similar inverse of polygon is negative orientation.

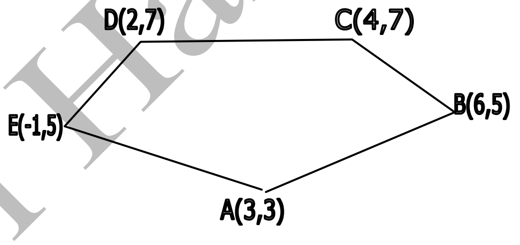
Ex./ polygon (A,B,C,D,E) the capes is A(3,3) ,B(6,5) ,C(4,7) ,D(2,7) ,E(-1,5) show the line GF is inside of polygon where G(7,4) and F(4,3)

Sol./ CF=(6-3)(3-3)-(5-3)(4-3)=-2

CF is right side of AB then it is outside

CG=(6-3)(4-3)-(5-3)(7-3)=-5

CG is right side of AB then it is outside



Finally: O+,C+$\rightarrow$left side :- inside *where O is orientation* {+,-}

O+,C- $\rightarrow$right side :- outside *& C value of Equation*

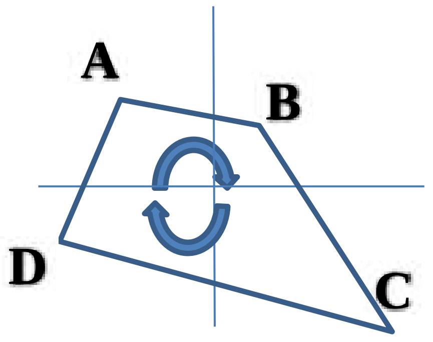
O-,C+$\rightarrow$left side :- outside

O-,C- $\rightarrow$right side :- inside

Polygon Consist of { A(-3,6), B(1, 5), C(10,-7),D(-8,-4) } Prove P(-10,2) inside or not in Negative oriented and T(9,4) inside or not Positive oriented

Table 1		
Part	**dx**	**dy**
AB	4	-1
BC	9	-12
CD	-18	3
DA	5	10

Tabel2		
Edge	**dx**	**dy**
BA	-4	1
CB	-9	12
DC	18	-3
AD	-5	-10



وفق الرسم فانه جدول رقم واحد هو Negative Oriented و جدول الثاني هو Positive Oriented

Table 1➔ -O		
Part	**dx**	**dy**
AB	4	-1
BC	9	-12
CD	-18	3
DA	5	10

Tabel2➔ +O		
Edge	**dx**	**dy**
BA	-4	1
CB	-9	12
DC	18	-3
AD	-5	-10

P(-10,2) ON Negative-Oriented ➔ x= -10, y=2

C=dx(y-y1) – dy (x – x1) ➔ X1=X_A, Y1=Y_A

C_{AB} =4 (2-6) – (-1) (-10+3) ➔ C_{AB} = -16-7➔

C_{AB} = -23 & -O ➔ Check next part polygon

X1= X_B, Y1=Y_B

C_{BC} =9 (2-5) – (-12) (-10-1) ➔ C_{BC} = -27-132➔

C_{BC} = -159 & -O ➔ Check next part polygon

X1= X_C, Y1=Y_C

C_{CD} = (-18)(2+7) – 3 (-10-10) ➔ C_{CD} = -162+60➔

C_{CD} = -102 & -O ➔ Check next part polygon

X1= X_D, Y1=Y_D

C_{DA} =5(2+4) – 10 (-10+8) ➔ C_{DA} = 30+20➔ C_{DA} = 50 & -O ➔ outside C+ , O-

Therefore P(-10,2) OUTSIDE ON POLYGON Because Left on Part DA of polygon

Table 2 ➔ +O		
Part	**Dx**	**Dy**
BA	-4	1
CB	-9	12
DC	18	-3
AD	-5	-10

Check T(9,4) inside or not positive oriented ➔ X=9, Y=4

C=dx(y-y1) – dy (x – x1) ➔ X1=X_B , Y1=Y_B

C_{BA} = -4 (4-5) – 1 (9-1) ➔ C_{BA} = 4-8➔ C_{BA} = - 4 & +O ➔ Outside

T(9,4) OUTSIDE Because Right on Part BA of polygon { C-, +O }

H.W// Check W (-1, 1) inside or not in polygon for any oriented

Note // point inside satisfy all Parts polygon same as Sign of C-value

∀ {C_1^-, C_2^- … C_n^-} in –O ➔ inside, Else ➔ outside

∀ {C_1^+, C_2^+… C_n^+} in +O ➔ inside, Else ➔ outside

8.3 Detect Polygon Oriented

How is can detect oriented of polygon depends of sequence vertices', it clockwise (negative oriented) or anti-clockwise (positive oriented). Suppose polygon consist of {P1, P2, P3 …Pn} to check orientation select three point successive to examination by $O = \begin{pmatrix} \mathbf{1} & \boldsymbol{X_i} & \boldsymbol{Y_i} \\ \mathbf{1} & \boldsymbol{X_{i+1}} & \boldsymbol{Y_{i+1}} \\ \mathbf{1} & \boldsymbol{X_{i+2}} & \boldsymbol{Y_{i+2}} \end{pmatrix}$

A(-3,6), B(1, 5), C(10,-7),D(-8,-4)

O1=$\begin{bmatrix} 1 & XA & YA \\ 1 & XB & YB \\ 1 & XC & Yc \end{bmatrix}$ ➔ **O1=**$\begin{bmatrix} 1 & -3 & 6 \\ 1 & 1 & 5 \\ 1 & 10 & -7 \end{bmatrix}$=

1{(1*-7)-(5*10)} – (-3) {(1*-7)-(5*1)} +6{(1*10)-(1*1)}

= -57-36+54= -39 ➔ Oriented Negative ➔Clockwise on A➔B➔C

O2=$\begin{bmatrix} 1 & XB & YB \\ 1 & XC & YC \\ 1 & XD & YD \end{bmatrix}$ ➔ **O2=**$\begin{bmatrix} 1 & 1 & 5 \\ 1 & 10 & -7 \\ 1 & -8 & -4 \end{bmatrix}$**= -189 ➔ Oriented Negative**

➔Clockwise on B➔C➔D

++

T(1, 5), S(-3,6), U(-8,-4) , V(10,-7)

O=$\begin{bmatrix} 1 & XT & YT \\ 1 & XS & YS \\ 1 & XU & YU \end{bmatrix}$ ➔ **O=**$\begin{bmatrix} 1 & 1 & 5 \\ 1 & -3 & 6 \\ 1 & -8 & -4 \end{bmatrix}$ =

1{(-3*-4)-(6*-8)} –1{(6*1)-(1*-4)} +5{(1*-8)-(-3*1)}

= 60-10-25= 25 ➔Anti-CLOCKWISE {Positive Oriented} T➔S➔U

==

A(-1,12),B(-13,9),C(-13,-16),D(-7,-6),E(8,-19),F(-4,5),G(8,8)

Oriented of polygon..?

Sol// O=$\begin{bmatrix} 1 & XA & YA \\ 1 & XB & YB \\ 1 & XC & Yc \end{bmatrix}$ ➔ **O=**$\begin{bmatrix} 1 & -1 & 12 \\ 1 & -13 & 9 \\ 1 & -13 & -16 \end{bmatrix}$ =

1{(-13*-16)-(9*-13)} –(-1){(1*-16)-(9*1)}+12{(1*-13)-(-13*1)}

= 325-25 +0= 300 ➔ Oriented POSITIVE ➔ANTI- Clockwise ➔ A➔B➔C

EX// A(-1,12), G(8,8), F(-4,5), E(8,-19), D(-7,-6), C(-13,-16), B(-13,9)

ORIENTED OF POLYGON

$O1=\begin{vmatrix}1 & -1 & 12\\ 1 & 8 & 8\\ 1 & -4 & 5\end{vmatrix} = -75$ **➔ Oriented NEGATIVE ➔ Clockwise A→G→F**

$O2=\begin{vmatrix}1 & 8 & 8\\ 1 & -4 & 5\\ 1 & 8 & -19\end{vmatrix} = 324$ **➔ Oriented Positive ➔ counter Clockwise** G→F→E

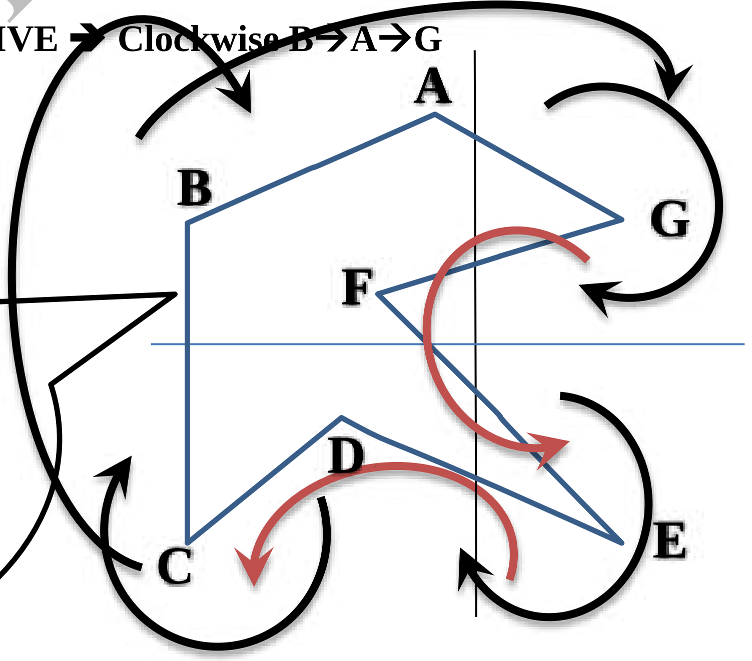
$O3=\begin{vmatrix}1 & -4 & 5\\ 1 & 8 & -19\\ 1 & -7 & -6\end{vmatrix} = -204$ **➔ Oriented NEGATIVE ➔ Clockwise F→E→D**

$O4=\begin{vmatrix}1 & 8 & -19\\ 1 & -7 & -6\\ 1 & -13 & -16\end{vmatrix} = 228$ **➔ Oriented Positive ➔ counter Clockwise** E→D→C

$O5=\begin{vmatrix}1 & -7 & -6\\ 1 & -13 & -16\\ 1 & -13 & 9\end{vmatrix} = -150$ **Oriented NEGATIVE ➔ Clockwise D→C→B**

$O6=\begin{vmatrix}1 & -13 & -16\\ 1 & -13 & 9\\ 1 & -1 & 12\end{vmatrix} = -300$ **Oriented NEGATIVE ➔ Clockwise C→B→A**

$O7=\begin{vmatrix}1 & -13 & 9\\ 1 & -1 & 12\\ 1 & 8 & 8\end{vmatrix} = -75$ **Oriented NEGATIVE ➔ Clockwise B→A→G**



This Polygon is not convex because of same Edge is different orientation