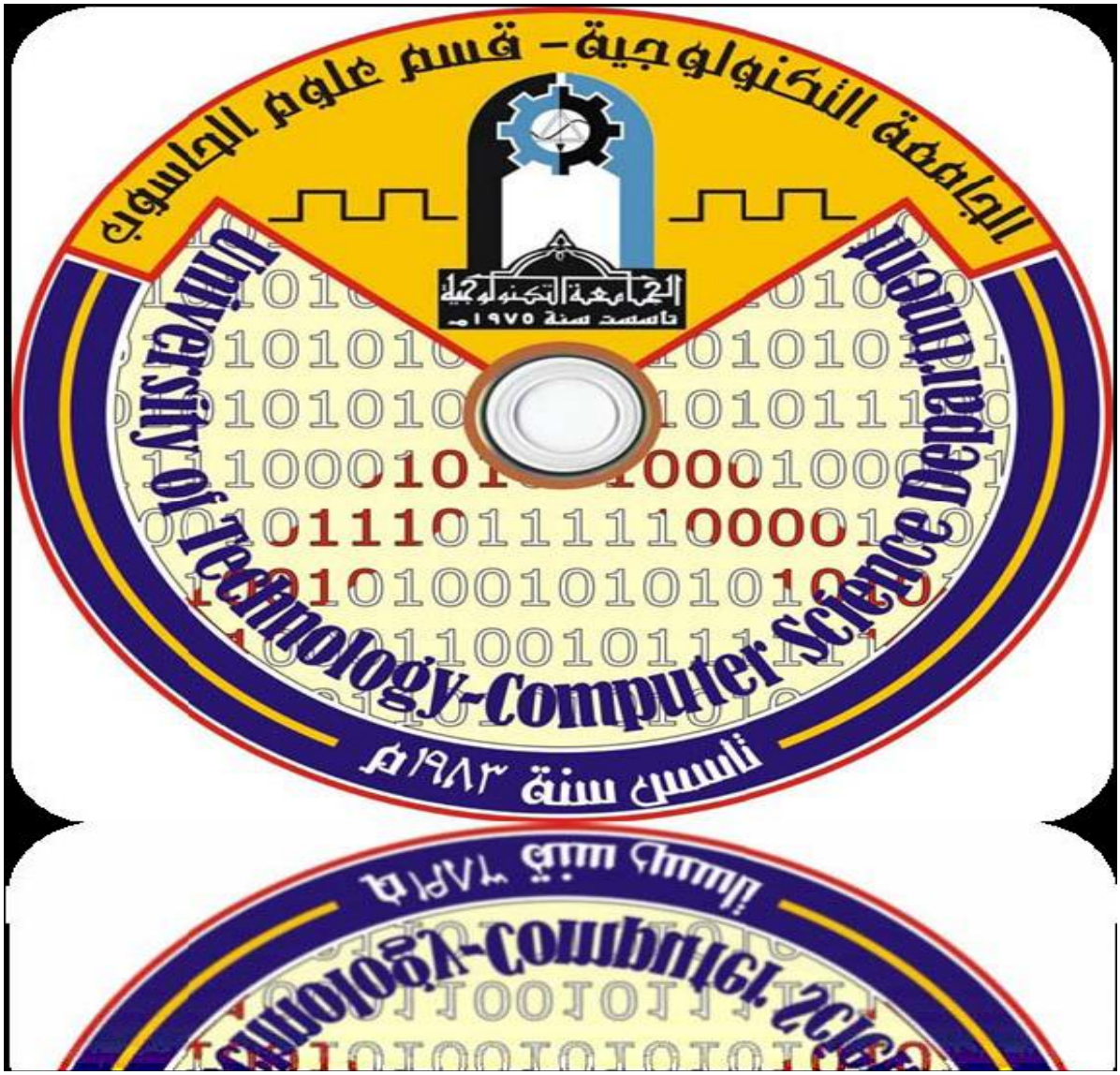


3rd (Computer Programmatic, Information Systems, Artificial Intelligent, Multimedia)

Computer Graphics

رسوم الحاسوب

أعداد :م .علي حسن حمادي

2016-2015

ALI HASSAN HAMMEDIE
COMPUTER GRAPHICS LECTURS
COMPUTER SCIENCE DEPARTMENT
UINVERSITY OF TECHNOLOGY
IRAQ-BAGHDAD

(Part One)

Introduction
Computer Graphic

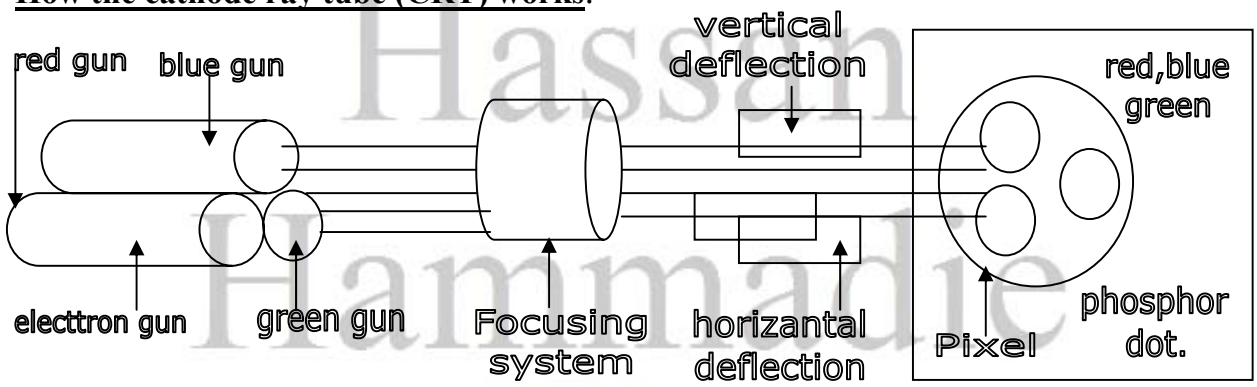
Introduction of Computer Graphics

Computer graphics: 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.

Interactive computer graphics: 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 input device e.g. (keyboard, mouse, joystick,..etc). It appears that picture is changing instantaneously in response to the observers commands.

How the cathode ray tube (CRT) works:



The CRT consists of three electron guns: red, green and blue. They emit a beam of negative 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 deflect the beam to strike any point on the screen.

When this focused electron beam strikes the screen the phosphor emits as dot of visible light.

The video display is divided in to 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.

Generating color on a *RGB* monitors:

Each electron gun in a ***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).

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***

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.

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)

Coordinates system

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

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

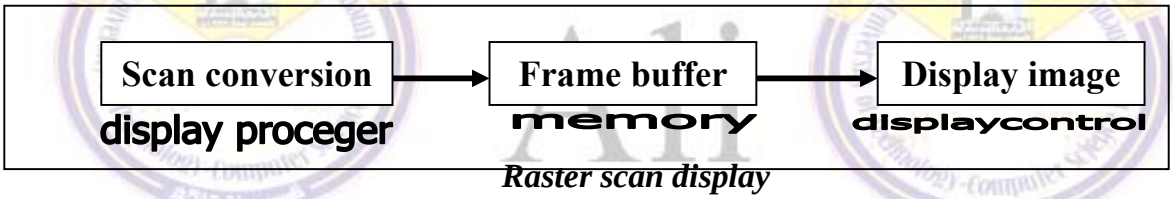
Raster – scan display

The video display in microcomputer is divided into very small rectangle or dots these dots are refferred 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 reffered to as raster-scan-display.

Frame buffer

Each screen pixel corresponds to particular entry in 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 compute. This allowing repaid up date of the stored image. Number of row on the frame buffer array equal the number of raster lines on display screen. Number of columns in this array equals the number of pixels on each raster line.

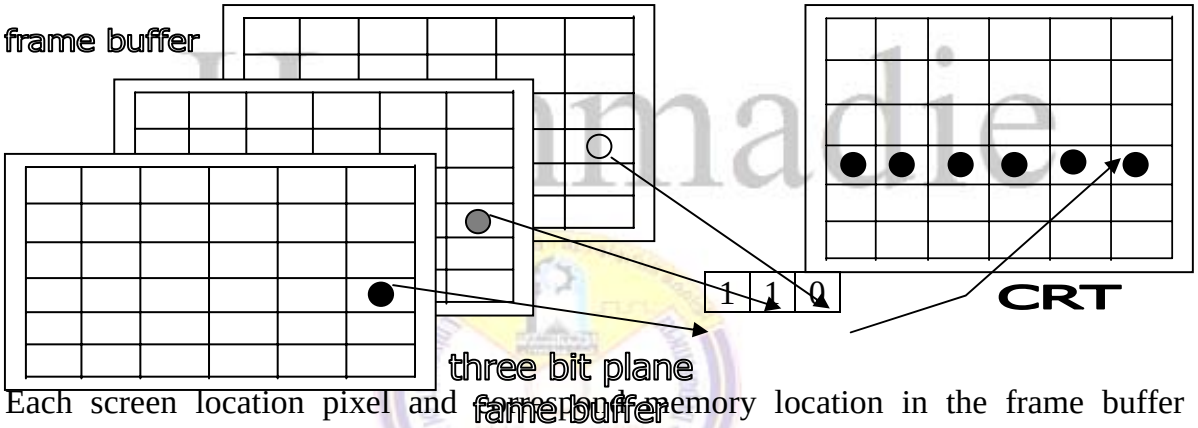
The term pixel is also used to describe the row and column location in the frame buffer array that corresponds to the screen location. A size 512*512 display screen required (262144) pixels memory location.



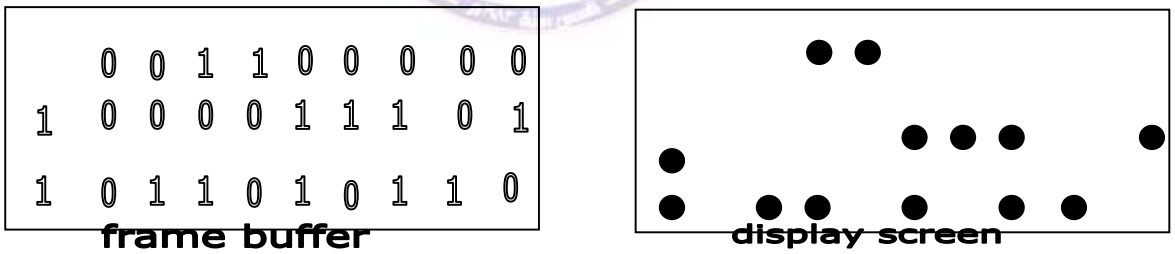
Raster scan display

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 a number of bits a single bit place frame buffer in order to display a color or black and white quality image with shades of gray additional bit places are needed.



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

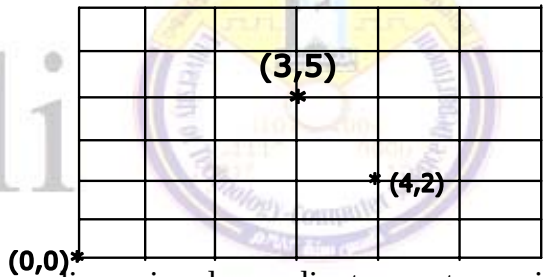


Scan conversion

Image are usually defined in term 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 appropriates pixel value in the frame buffer. In expensive micro computer graphics system use the CPU and library of software routines to perform scan conversion. "The process of representing continuous graphics objects as a collection of discrete pixel is called scan conversion".

Plotting points

In order 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 scan conversion point-plotting algorithm. Both frame buffer and display screen are given a two-dimensional coordinates system with origin at the lower left corner. Each pixel is described in no negative integer (x, y) coordinates pair the x value start at the origin (0) and increase from left to right(there is no standard for the location of two origin).



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.

ALI HASSAN HAMMEDIE
COMPUTER GRAPHICS LECTURS
COMPUTER SCIENCE DEPARTMENT
UINVERSITY OF TECHNOLOGY
IRAQ-BAGHDAD

(Part Two)

Vectors

2.1Vectors

The word vector is used for 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 it direction, then we have a vector. In this lecture we adopt a standard notation and show vector quantities in **bold** type scalar quantities (like real number) in standard type. Thus the arrow from A to B will 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:

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:

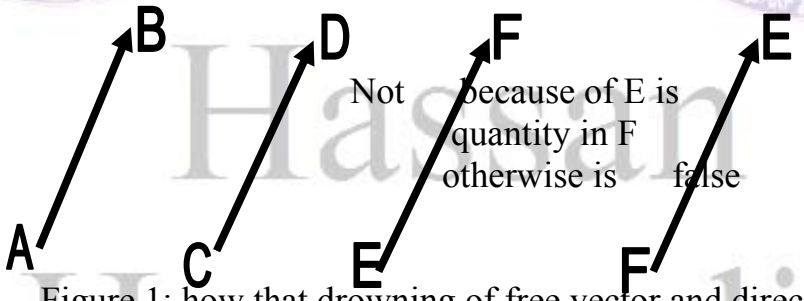


Figure 1: how that drowning of free vector and direction.

B) "Anchor" Vector:

Its starting position at the origin (0, 0) in 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 origin show in figure 2 below:

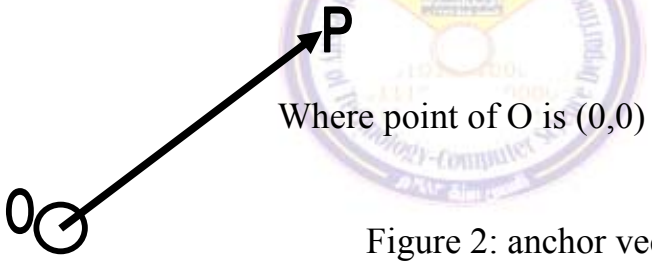


Figure 2: anchor vectors

2.2: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

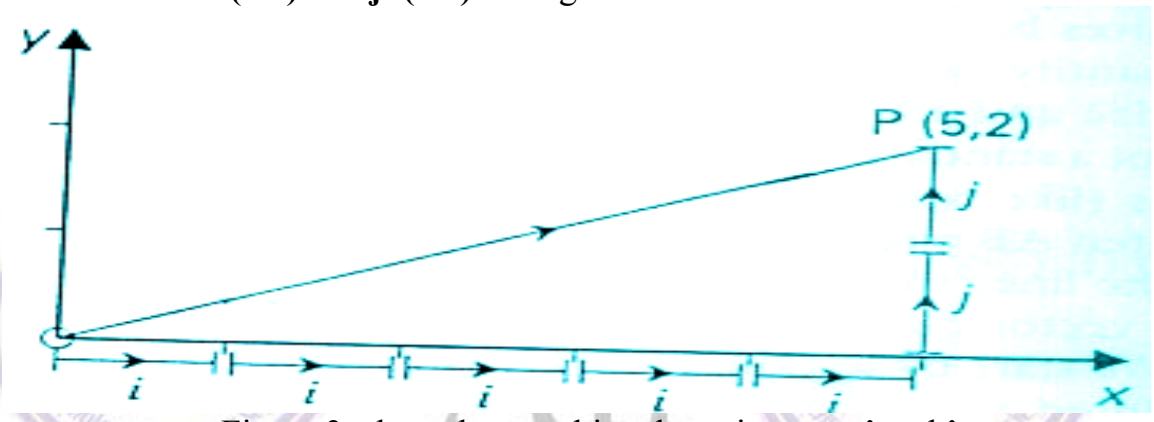


Figure 3: show that combine the unit vector **i** and **j**

2.3:measurement associated with vectors
this following example of vector OP and OQ, we have shown above the there are different way to write OP & OQ either point or row vector or component form.

2.3.1 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 $|OQ| = \sqrt{2^2+(-4)^2} = \sqrt{20} = 4.47$

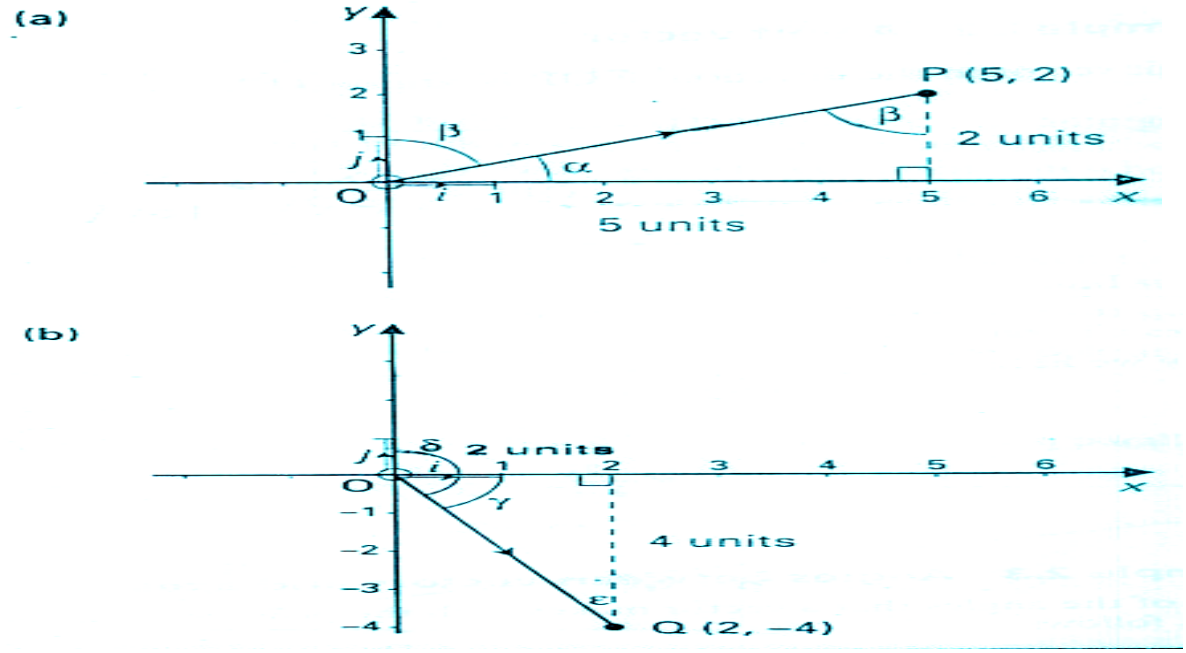


Figure 4: example vectors p,q

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

$\hat{OP}$ = **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

2.3.3 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.

2.4 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.

2.4.1:adding vectors let see figure below two vector p, q

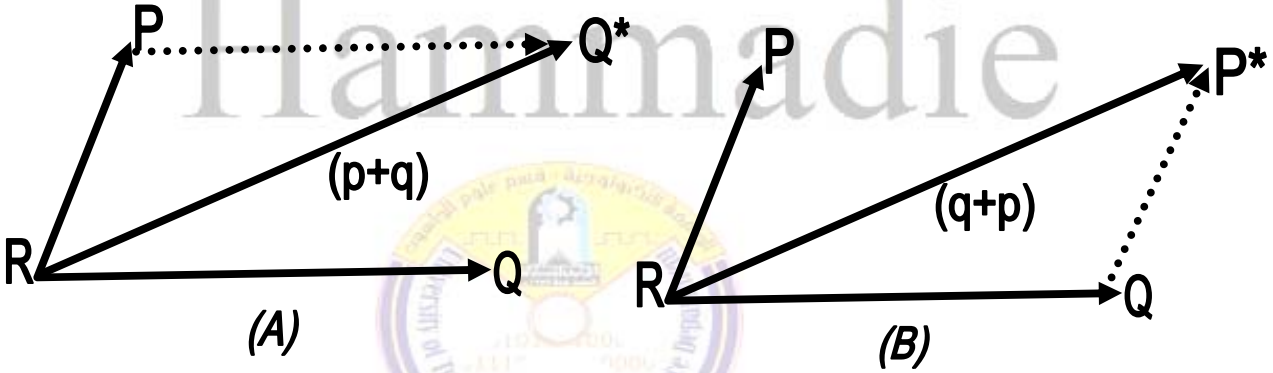


Figure 5 show adding two vectors (A) p+q, (B) q+p

Example: let vector p (3, 2) and Q (5,-4) find p+q and drowning.
Sol. / P=3i+2j and q=5i-4j
p+q= (3i+2j) + (5i-4j) =3i+2j+5i-4j=8i-2j **(in component form)**

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

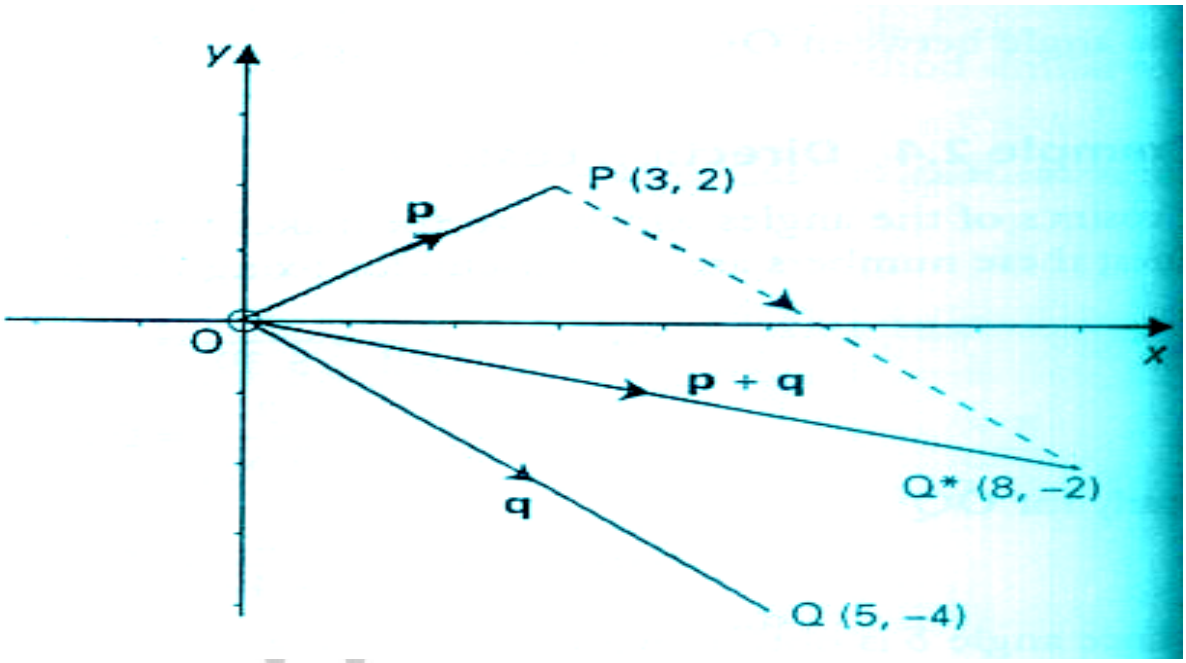


Figure 6: result adding vector p+ q to Q*

2.4.2:negative vectors and subtracting vectors

-q is a negative vector, by which we mean a vector with the same magnitude as **q** but **opposite direction**, as shown figure 7. we are able calculate **p-q** by finding **p+(-q)**: we reverse the direction of **q** and starting from **Q,** we then use triangle addition.

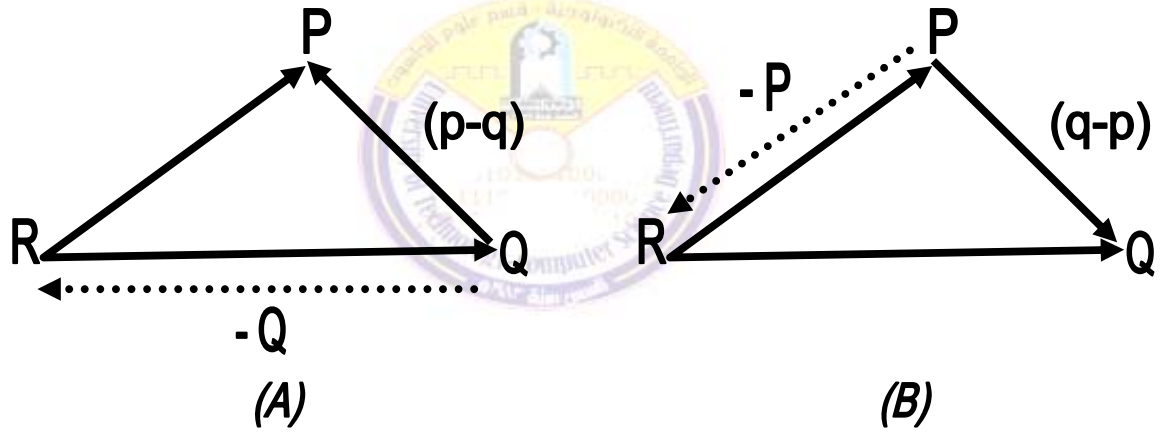


Figure 7 show drawing triangle in (A) p-q (B) q-p

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-q=(3i+2j)-(5i-4j) =3i+2j-5j+4j =-2i+6j	p=(3 2) and q=(5 -4) p-q=(3 2)-(5 -4) = (3-5) (2+4) = (-2 6)

Drowning is H.W ((coordinate the example)) answer in figure 10

Note: if we wish to determine the vector between any two point whose coordinate 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.

2.4.3:scaling Vectors

we can scale any vectors by multiplying it 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 same direction (1/2)p=1/2 (4 3) = (2 1(1/2)).

A negative scale factor reverses the direction of a vector. When the scale factor is -1, than 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 -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 scalings are indicated in figure 8.

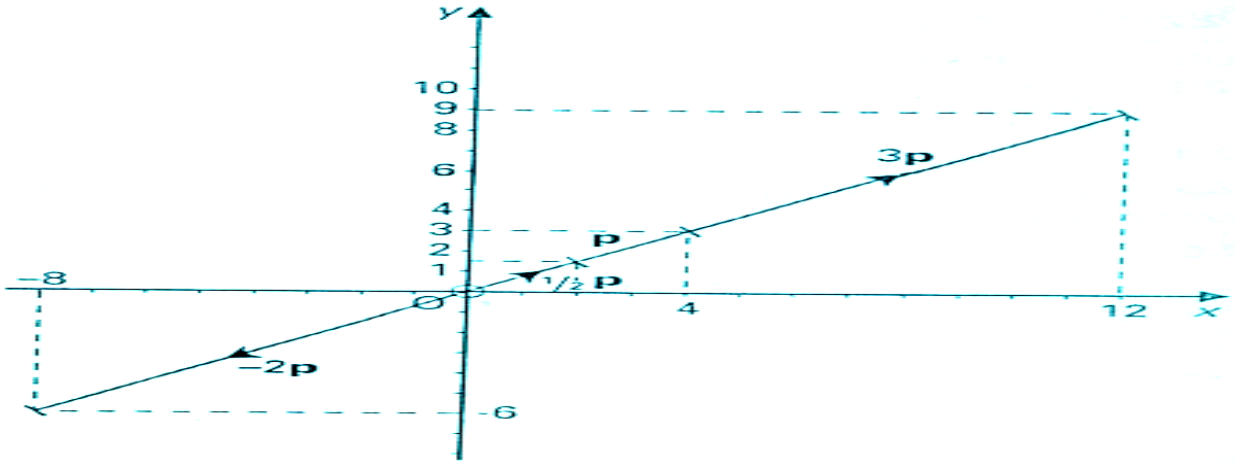


Figure 8: scaling vector

2.4.4:multiplying vectors uses the "*dot Product*"

purpose of *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 ɵ ; where*** ɵ is 0<= ɵ <=180, and therefore to find angle between of two vectors is :

cos ɵ = (a.b) / (|a| |b|)

note: *if dot product =zero implies either that at least one vector is the zero or 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

Example*:* calculate the angle between the vector a=3i+5j and b=2i+j as shown in figure 9 below:

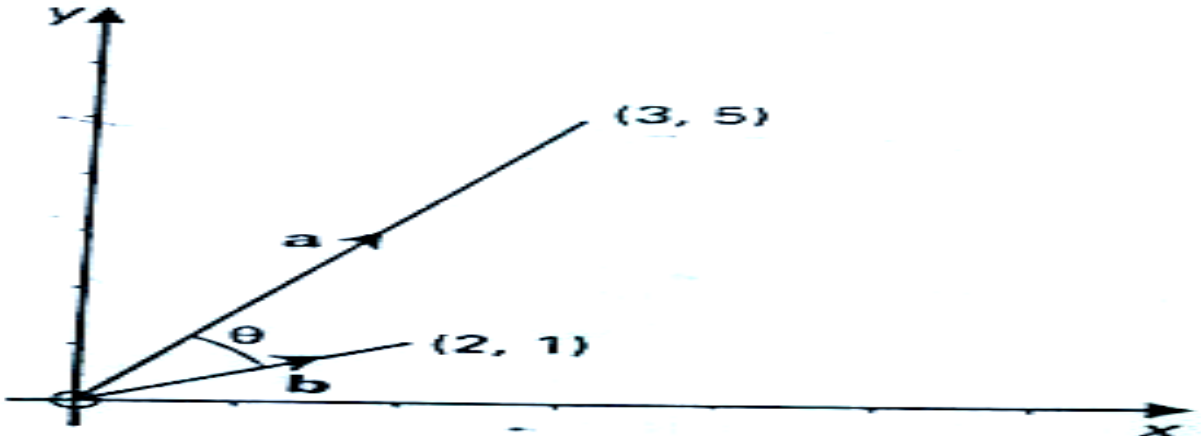


Figure 9: example to find angle ɵ

Sol. / a . b = |a| |b| cos ɵ

|a|= $\sqrt{3^2+5^2} = \sqrt{34}$ And |b| = $\sqrt{2^2+1^2} = \sqrt{5}$

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

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

Then ɵ= Cos^{-1} (a. b) / (|a| |b|) ➔ ɵ= Cos^{-1} 11/ ($\sqrt{34} * \sqrt{5}$) = 32.47

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

.....

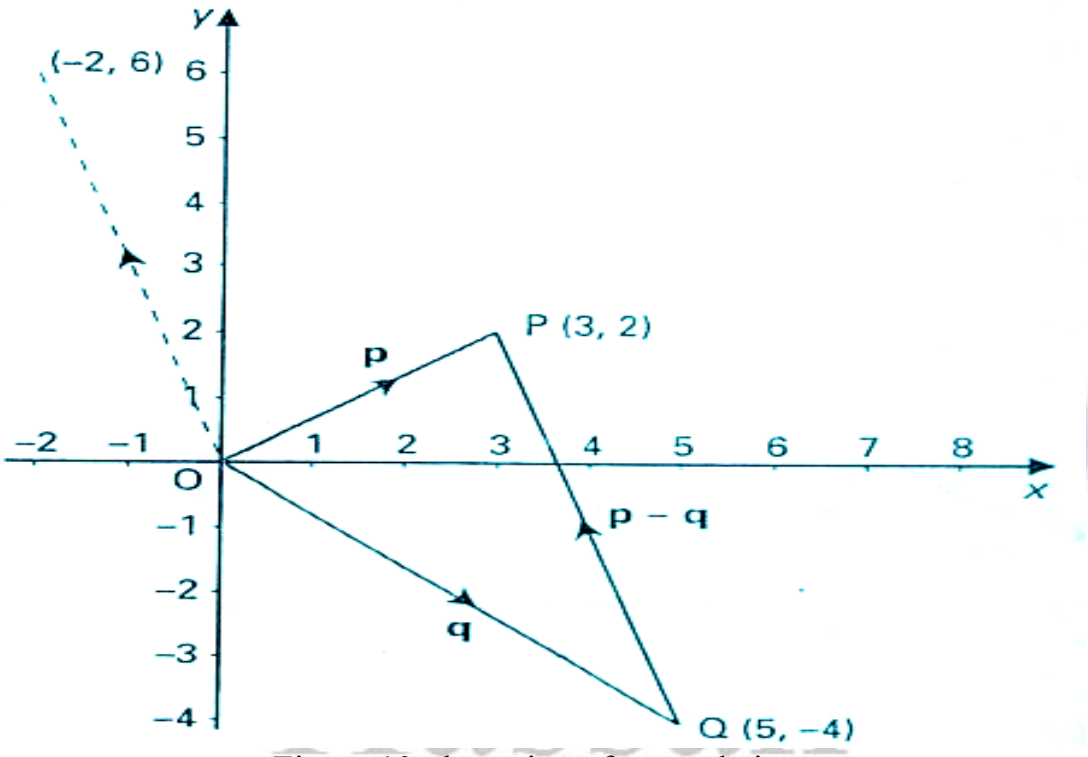


Figure 10: drowning of example in p-q

2.4.5:Multiplying vectors uses the "*Cross Product*"

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 $\boldsymbol{a}\times\boldsymbol{b}=|\boldsymbol{a}|\ |\boldsymbol{b}|\ \boldsymbol{Sin(\theta)n}$ *; where* **ɵ** is angle , and **n** is unit vector of perpendicular between a & b therefore if not n(normal) ➔ $|\boldsymbol{a}\times\boldsymbol{b}|=|\boldsymbol{a}|\ |\boldsymbol{b}|\ \boldsymbol{Sin(\theta)}$,
to find angle between of two vectors is : $\theta = \mathrm{Sin}^{-1}\ |a\times b|\ /\ (|a|\ |b|)$
We can easy calculating *cross product* **a×b** in 2D where a = ai+ aj &b= bi +bj is:

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

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

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

2.5: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 $\mathrm{Cos}(\beta)=5/\sqrt{34}$ ➔ $\beta=\cos^{-1}(5/\sqrt{34}) \simeq 30.96$.

ALI HASSAN HAMMEDIE
COMPUTER GRAPHICS LECTURS
COMPUTER SCIENCE DEPARTMENT
UINVERSITY OF TECHNOLOGY
IRAQ-BAGHDAD

(Part Three)

Line Drawing Algorithms

3.1 plotting points:

We will use the Properties in V.B on Picturebox or Form "Pset" to plot points on the screen. For Example on PictureBox:-

Picture1.Pser(X, Y), [color] ; where color is: ***0 to 2^{24} -1*** or Using RGB Function as following **RGB(R,G,B).**

R is red base color rang on [0..255]

G is green base color rang on [0..255]

B is Blue base color rang on [0..255]

* finally combination color ***is [0 .. 2^{24} -1***] **color**

3.2 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.

3.3 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

1:For x =x1 to x2 step sgn(x2-x1)

2: Plote Pixel (x, y) ,color) // pset in V.B

3: goto 1

Note:- sgn(n) in VB 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

1:For y =y1 to y2 step sgn(y2-y1)

2: Plote Pixel (x, y) ,color) // pset in V.B

3: goto 1

3.4 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) {Putpixel (x, y, color);

x+ =sgn(x2-x1); y+ =sgn(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 sgn(x2-x1) // x=x +sgn(x2-x1)

3: Plote Pixel (x, y) ,color) // pset (x,y),color in V.B

4: y=y+ sgn(y2-y1) //sgn(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':

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

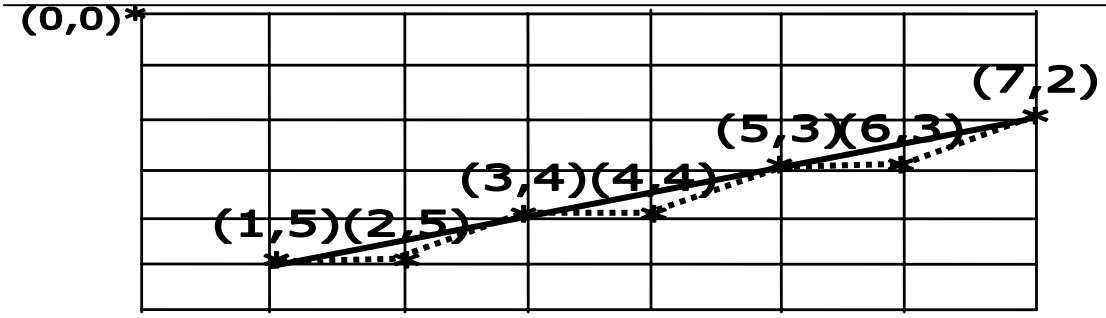
6: next x {goto 5 where un-finish}

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**

dx>0 then increment of X

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)

3.4 B <u>the simple DDA</u>

The simple *Digital Differential Analyzer* is a line drawing algorithm that generates line from their differential equations. It work 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,y,) ,(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 next points coordinates. 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 end point (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 : 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 <u>Bresenhams line drawing algorithm</u>:

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 record the distance measure perpendicular to the axis of greatest movement, between the exact path of the line and the actual dots generated.

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

*<u>If **e** is negative the y coordinate **value is left unchanged**.</u>*

And verse vise if y-axis is greatest movement

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

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

'this algorithm is X-axis greatest movement and slop is positive

1:dx=x2-x1: dy=y2-y1: e=(dy/dx)-0.5: x=x1: y=y1;

2:For i=0 to |dx|

2.1: Plot pixel (x, y,)

2.2: If (e>0) then if (y1>y2) y=y-1

Else y=y+1

e=e-1

2.3: if(x1>x2) then x=x-1

else x=x+1

2.4: e=e+ (dy/dx)

3: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

Note: this algorithm uses in **slop of line positive** but in **negative slop** replaces *e>0 to* ***e<0***, and *e=e-1 to* ***e=e+1*** and coordinate y=y+1 to y=y-1 Or x=x+1 to x=x-1 to obtain algorithms:

1:dx=x2-x1: dy=y2-y1: e=(dy/dx)+0.5: x=x1: y=y1;

2:For i=0 to |dx|

2.1: Plot pixel (x, y,)

2.2: If (e<0) then if (y1>y2) y=y-1

Else y=y+1

e=e+1

2.3: if(x1>x2) then x=x-1

else x=x+1

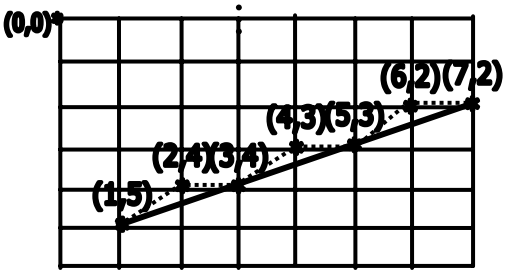
2.4: e=e+ (dy/dx)

3:next i

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
{Negative slop} uses algorithm modify of Bresenhams
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

ALI HASSAN HAMMEDIE
COMPUTER GRAPHICS LECTURS
COMPUTER SCIENCE DEPARTMENT
UINVERSITY OF TECHNOLOGY
IRAQ-BAGHDAD

(Part Four)

Circle ,Ellipse Drawing Algorithms

4.1 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 phethagoras's theorem

$$R^2=X^2+Y^2 \quad \ldots.(1)\ .$$

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:

$$Cos(\theta) = X / R \quad \ldots.. (2)$$

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

$$Sin(\theta) = Y / R \quad \ldots.. (3)$$

4.2 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** …..(5); solving equation **(4)** for y obtain $\mathbf{y=yc\pm\sqrt{r^2-(x-xc)^2}}$ solving equation **(5)** for y obtain $\mathbf{y=\pm\sqrt{r^2-(x)^2}}$

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. //

recommend using program

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) : y2:= -sqrt(r*r-x*x);***

2.2: plot pixel(xc +x, yc+y1) : putpixel(xc +x, yc+y2)

2.3 x=x+dt

3: goto 2

4.3 <u>the polar representation of circles</u>:

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∏ Radians. From equation (2) and (3) we find:

$X = R * \cos(\Theta)$ …… *(4)*

$Y = R * \sin(\Theta)$ …. *(5)*

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

1: dt = 1/r : th=0 : pi=22/7.0

2: While (th<=2 * pi)

2.1: x=r*cos (th) : y:=r*sin (th)

2.2: Plot pixel(xc +x, yc +y)

2.3: th= th+ dt

3: goto 2

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

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*

Xn+1= R* cos(Θ+Δ Θ) *... (6)*

And ***Y= R*sin(Θ+Δ Θ)*** *... (7)*

Where *ΔΘ* is increment valued. The cosine and sine function can be expanded using the following standard trigonometric results:

Cos (Θ+Δ Θ) =cos(Θ)cos(ΔΘ) - sin(Θ)sin (ΔΘ)….. (8)

Sin (Θ+Δ Θ) =sin(Θ)cos(ΔΘ) + cos(Θ)sin (ΔΘ)….. (9)

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

Xn+1 =Rcos(Θ)cos (ΔΘ) – Rsin(Θ)sin(ΔΘ)….. (10)

Yn+1 =Rsin(Θ)cos (ΔΘ) + Rcos(Θ)sin(ΔΘ)….. (11)

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

Xn+1 =Xn cos(ΔΘ) – Yn sin(ΔΘ)….. (12)

Yn+1 =Yn cos(ΔΘ) + Xn sin(ΔΘ)….. (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:goto 2 ' if not reach 360^o

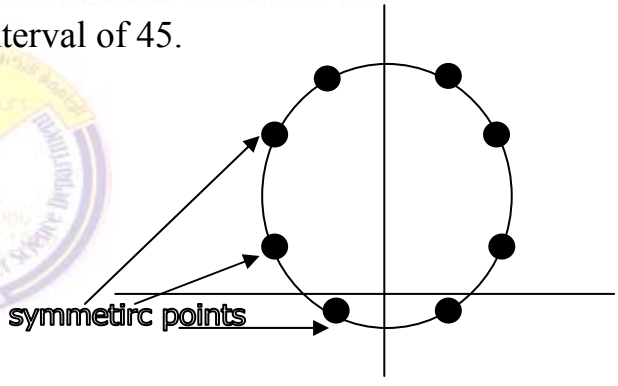
H.W/trace the algorithm that use incremental method to generate six points of the centered at(300,150)with a radius equal to 9 unit.

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: plot pixel (xc+ x, yc- y),2

plot pixel (xc- x, yc+ y),3 plot pixel (xc- x, yc- y),4

plot pixel ***(xc+y, yc+x)****,5: plot pixel* ***(xc+y,yc-x)****,6*

plot pixel ***(xc-y, yc+x)****,7: plot pixel* ***(xc-y, yc-x)****,8*

*2.2t=x : x=x*ct –y*st : y=y*ct +t*st*

3. goto2

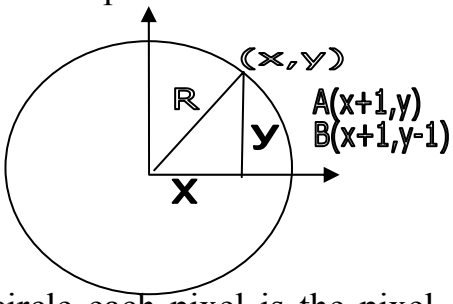
4.6 Bresenham circle algorithm:

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 Θ, 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.Wend 'goto 2

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

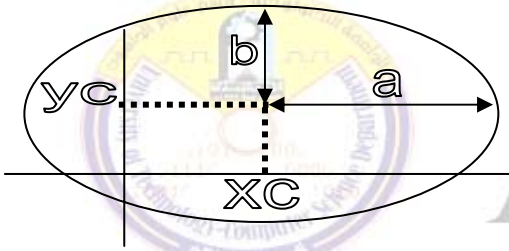
x	y	x +xc	y +yc	d(A)	d(B)	s
0	9	300	159	1	-16	-15
1	9	301	159	4	-13	-9
2	9	302	159	9	-8	1
3	8	303	158	-1	-16	-17
4	8	304	158	8	-7	1
5	7	305	157	4	-9	-5

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 $\frac{(x- xc)^2}{a^2} + \frac{(y- yc)^2}{b^2} = 1$



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

$y= yc \pm b\sqrt{1- \frac{(x- xc)^2}{a^2}}$ and 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 : plot pixel(xc +x, yc+ y1), 2

2.4: x=x+dt

3:wend 'goto 2

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

4.7. B The polar representation of an ellipse:

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 (Θ)… (14)*

*y=yc +yr*sin (Θ)… (15)*

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 :

$$Xn+1 = Xn \cos(\Delta\Theta) - (Xr/Yr) * Yn \sin(\Delta\Theta) \ldots\ldots (16)$$

$$Yn+1 = Yn \cos(\Delta\Theta) + (Yr/Xr) * Xn \sin(\Delta\Theta) \ldots\ldots (17)$$

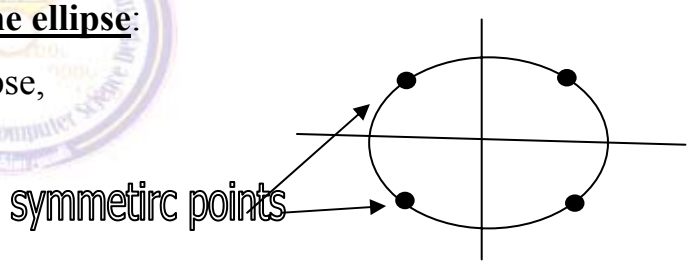
This algorithm is modify the polar representation (leave that of students to write this algorithm)

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)



3.5. A <u>Incremental drawing of an ellipse:</u>

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

$$Xn+1 = Xn \cos(\Delta\Theta) - (Xr/Yr) * Yn \sin(\Delta\Theta) \ldots\ldots (16)$$

$$Yn+1 = Yn \cos(\Delta\Theta) + (Yr/Xr) * Xn \sin(\Delta\Theta) \ldots\ldots (17)$$

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

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

Sector: 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..

ALI HASSAN HAMMEDIE
COMPUTER GRAPHICS LECTURS
COMPUTER SCIENCE DEPARTMENT
UINVERSITY OF TECHNOLOGY
IRAQ-BAGHDAD

(Part Five)

2D

Transformation

5.1 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.

5.2 Fundamental Transformation

There are three basic transformations:

1. Translation(shift OR move).
2. Scaling.
3. Rotation.

5.2 A: 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.

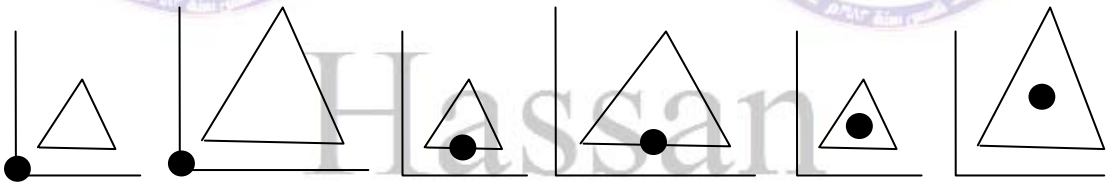
5.2 B: Scaling:

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:

$$x^s = x` * Sx; \quad y^s = y` * Sy ;$$

3. translate the origin back to the fixed point(xf, yf):

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

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

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

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

5.2 C: 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

5.2. C.1 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:

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

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

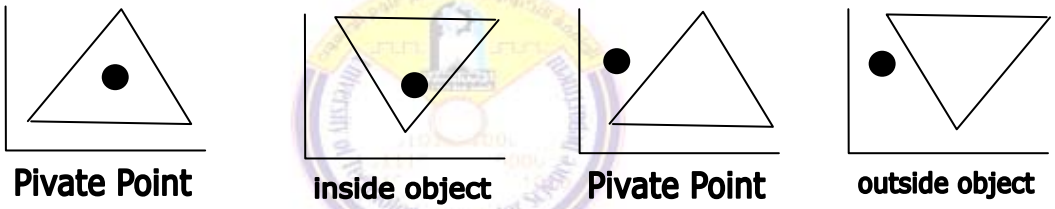
Mathematically, counter anticlockwise is considered the positive rotation direction, to rotate in a clockwise direction change to angle Θ to $-\Theta$. *Then:* **$\cos(-\Theta) = \cos(\Theta)$, $\sin(-\Theta) = -\sin(\Theta)$;** clockwise rotation can be represent by:

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

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

5.3. C.2 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 :-

$$x' = x - xp \quad , \quad y' = y - yp ;$$

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: $\mathbf{x^g = x^R + xp}$; $\mathbf{y^g = y^R + yp}$;

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

$Y^g = (y-yp)*\cos(\Theta)+(x-xp)*\sin(\Theta)+yp$

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:

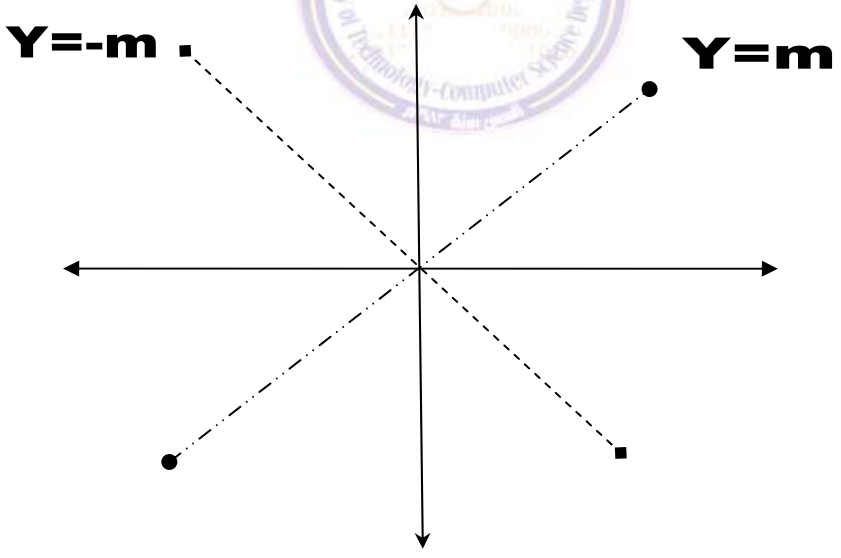
The mirror reflection about the x axis is: ***xm=x ; ym=-y;***

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;***



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: $(x^{mx} \; y^{mx} \; 1) = (x \; y \; 1) * \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

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

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

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

c. 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 anticlock wise by angle 63 then scale it with the scaling factor Sx=1, Sy=2.where(65,40) is fixed point,*

1. translate the pivot point to the origin.

$$\begin{bmatrix} 43 & 88 & 1 \\ 84 & 50 & 1 \\ 66 & 72 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -65 & -40 & 1 \end{bmatrix} = \begin{bmatrix} -22 & 48 & 1 \\ 19 & 10 & 1 \\ 1 & 32 & 1 \end{bmatrix}$$

2. rotate in a anticlockwise direction by 63^o.

$$\begin{bmatrix} -22 & 48 & 1 \\ 19 & 10 & 1 \\ 1 & 32 & 1 \end{bmatrix} * \begin{bmatrix} \cos(63) & \sin(63) & 0 \\ \sin(63) & \cos(63) & 0 \\ 0 & 1 & \end{bmatrix}$$

3. Scale with Sx=1, Sy=2. *(H.W)*

4. Translate the origin back to the pivot point. *(H.W)*

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 <u>Mirror about arbitrary line:</u> 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:

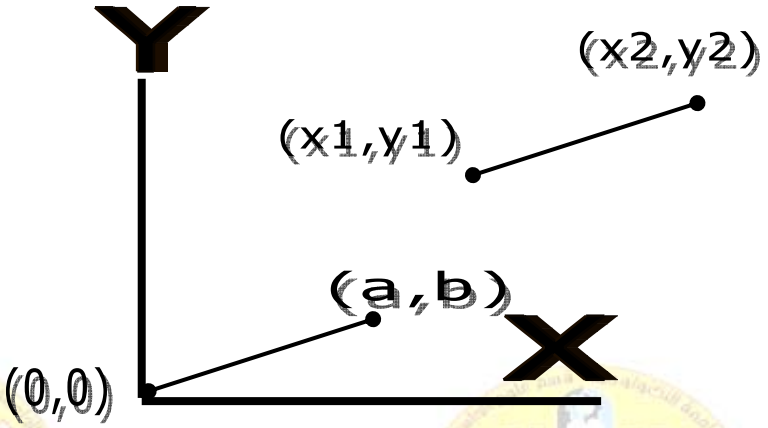
<u>Steps of mirror:</u>

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.

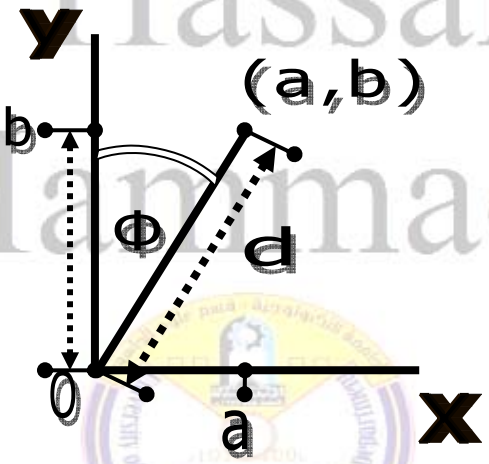
$$\mathbf{Tr(-x1, -y1)=} \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{-x1} & \mathbf{-y1} & \mathbf{1} \end{bmatrix}$$

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



a) 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:

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

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

3. Reflection about x-axis OR y-axis:using mirror matrix in(section 5.5)

4. Inverse step 2 by using – Φ in step 2.

5. Inverse step 1: by change sign of movement factors

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.

2. Reflection about origin point. [*Or Y-axis ,Or X-axis****]***

3. Inverse step 1: the movement factor x, y.

==========

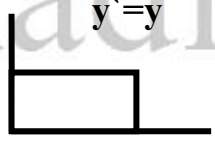
5.9 <u>Shearing</u>:

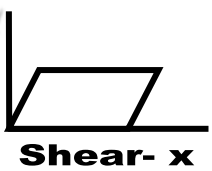
Shearing transformations makes the objects to distort their shapes in either x or y or both the directions. It can be imagined as the object is made up of very thin layers and they are slided over each other fixing the base in case of single directional shearing.

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

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

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

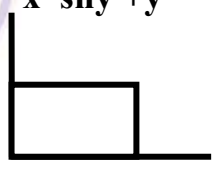


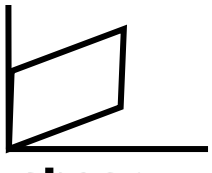


2. To shear in y direction only use shearing matrix in the equation as:

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

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





3. To shear in x and y directions both :

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

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

- If you need shear at fixed Point needs{shift ,Shear, return}

ALI HASSAN HAMMEDIE
COMPUTER GRAPHICS LECTURS
COMPUTER SCIENCE DEPARTMENT
UINVERSITY OF TECHNOLOGY
IRAQ-BAGHDAD

(Part Six)

Mapping ,Clipping

6.A>Introduce for Window and Viewport

A windows is specified by four world 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-veiwport mapping is to convert the world coordinates **(WX, WY)** of an arbitrary point to its corresponding normalized device coordinates **(VX, VY)**. In order to maintain the same relative placement of the point in the viewport as in the window, we require:

$$\frac{WX - WXmin}{WXmax - WXmin} = \frac{VX - VXmin}{VXmax - VXmin} \quad \textit{and}$$

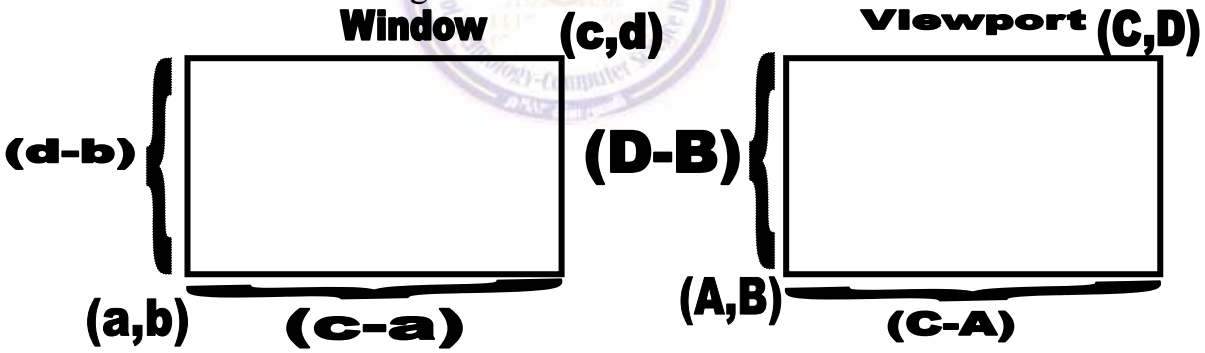
$$\frac{WY - WYmin}{WYmax - WYmin} = \frac{VY - VYmin}{VYmax - VYmin} \quad \textit{thus}$$

$$VX = VXmin + (WX - WXmin) * Sx$$
$$VY = VYmin + (WY - WYmin) * Sy \quad \textit{where}$$

$$Sx = \frac{VXmax - VXmin}{WXmax - WXmin} \quad \textit{and} \quad Sy = \frac{VYmax - VYmin}{WYmax - WYmin}$$

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

There three operations in the sequence. First we have the 'window shift', when we shift the lower left corner 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 lower left corner of the viewport from the screen origin to its proper position. We use the coordinates and dimensions shown in figure bellow



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

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

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:

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

The viewport shift is given by:

$$V=\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ A & B & 1 \end{pmatrix}$$

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:

$$W=\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ -1 & -3 & 1 \end{pmatrix}$$

The scaling matrix is :

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

$$S=\begin{pmatrix} 0.3 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The matrix for the viewport shift is

$$V=\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0.2 & 0.5 & 1 \end{pmatrix}$$

When we multiply these three matrices together(remembering that order matters) we get **M=W S V**

$$=\begin{pmatrix} 0.3 & 0 & 0 \\ 0 & 0.2 & 1 \\ -0.1 & -0.1 & 1 \end{pmatrix}$$

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:

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

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

HW/ a viewport that has lower left corner at (0,0) and upper right corner (1/2, 1/2).

6.B>Clipping

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

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.

Line clipping:

Line 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 y1, y2 > Ymax

x1, x2 < Xmin 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***.

Note:-if m=0 then X=X1 suppose "down or X2 suppose "up

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.***

Note:-if m=∞ then y=y1 suppose "left or Y2 suppose "right

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

Note: If Result And (bitwise)=0 then need clipping {cate.1 , cate. 3}

otherwise is Fully external of window Coordinate {not visible cate. 2}

6.C>Polygon

Convex if we take any two point inside of polygon and connect among by line and this line is fully inside the polygon.

Concave if we take any two point inside of polygon and connect among by line and this line is not fully inside the polygon.

Note: *the head of polygon is called by* ***Vertices***

If the sequence of capes of polygon in anticlockwise called **positive orientation** otherwise called **Negative orientation.**

To Known the point is inside the polygon we need :

1- known the polygon **positive orientation** or **Negative orientation.**

2- Determine the point is inside of polygon by 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.

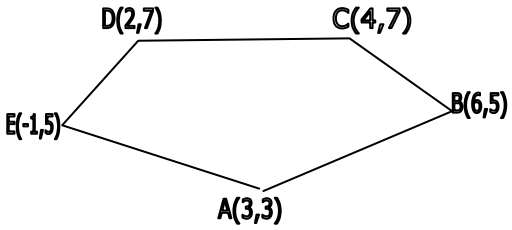
Ex./ polygon (A,B,C,D,E) the capes is A(3,3) ,B(6,5) ,C(4,7) ,D(2,7) ,E(-1,5) show the line GF is inside of polygon where G(7,4) and F(4,3)

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+➔left side :- inside *where O is orientation {+,-}*

O+,C- ➔right side :- outside *& C value of C_equation*

O-,C+➔left side :- outside

O-,C- ➔right side :- inside

Note:-it is useful for polygon clipping

HW/ consider the GF where coordinate G(4,5) and F(3,5)

Ex. *Suppose window where left up (1,8) and right down (5,2) and need clipping polygon where Vertices:-*

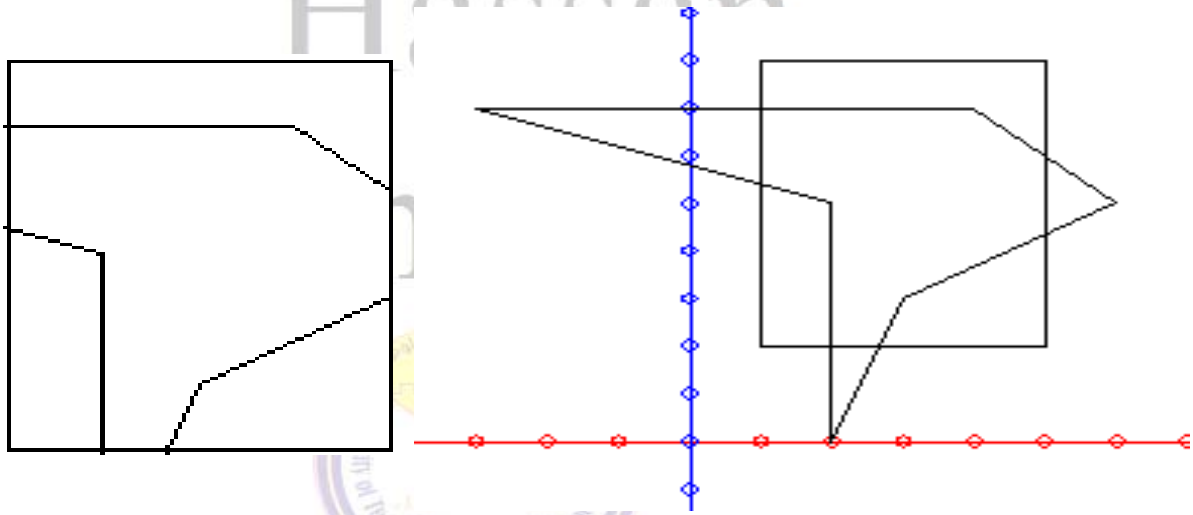
{ (3,3), (6,5), (4,7), (-3,7), (2,5), (2,0), (3,3)} what happened then new polygon uses vertices that inside in window.

Sol:/

Step1:- check *all points and classification outside or inside.*{Polygon}

Step2:- *if all vertices inside then no need clipping but if all vertices outside therefore, window is polygon clipping otherwise check edges of polygon by using line clipping.*

Step3:- in this step all edge of polygon are category 3 { neither visible nor invisible } in line clipping.



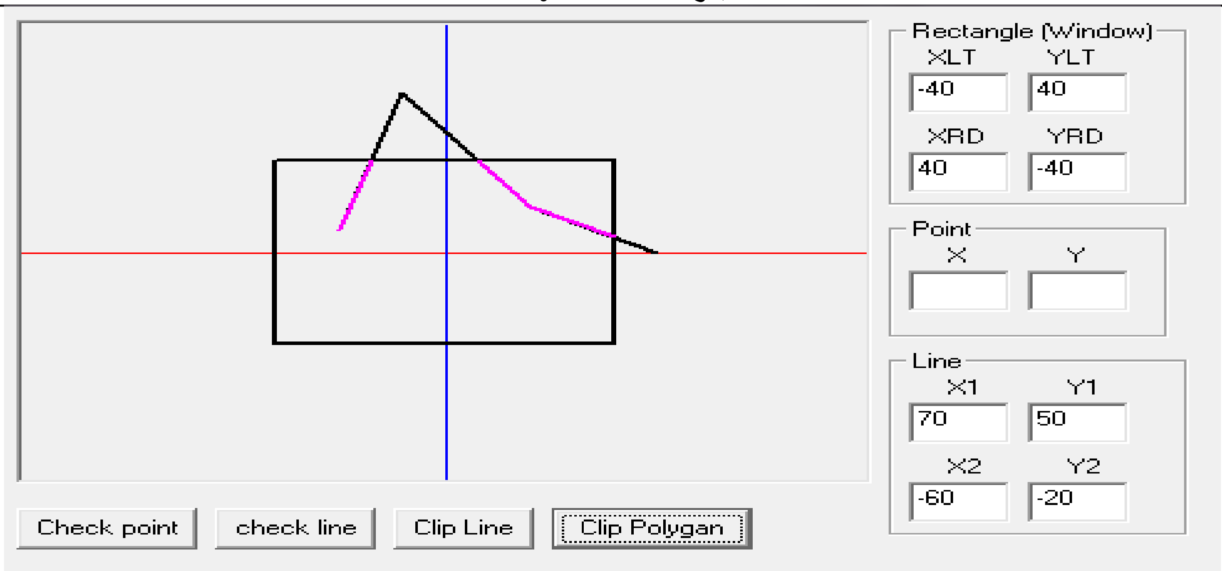
Let draw see shape ****Design in V.B by L. Ali Hassan Hammadie*

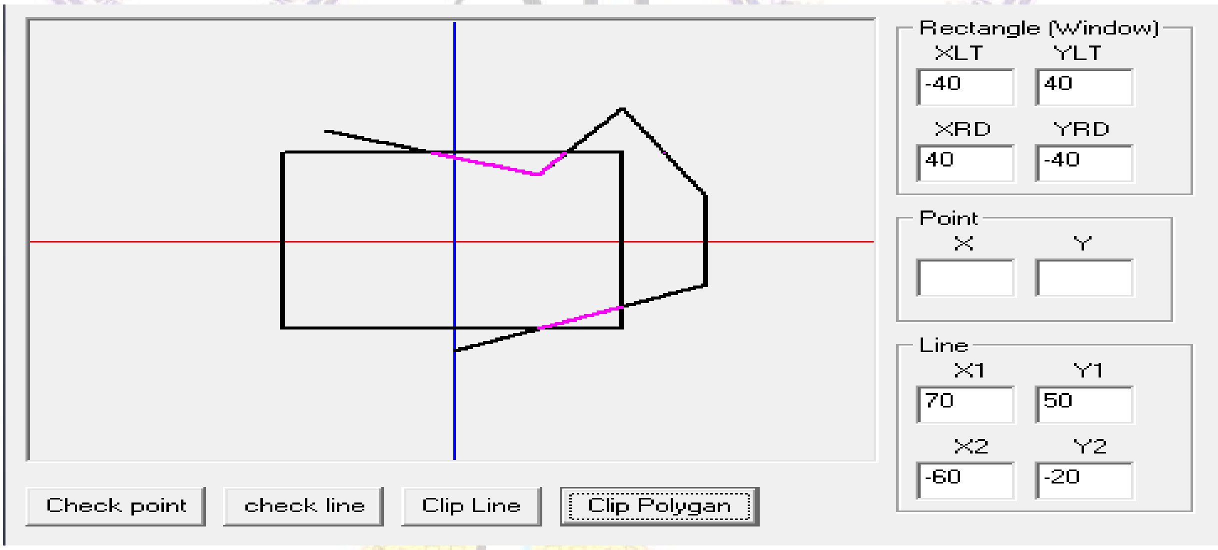
See vertices (6,5) & (-3,7) & (2,0) are outside window.

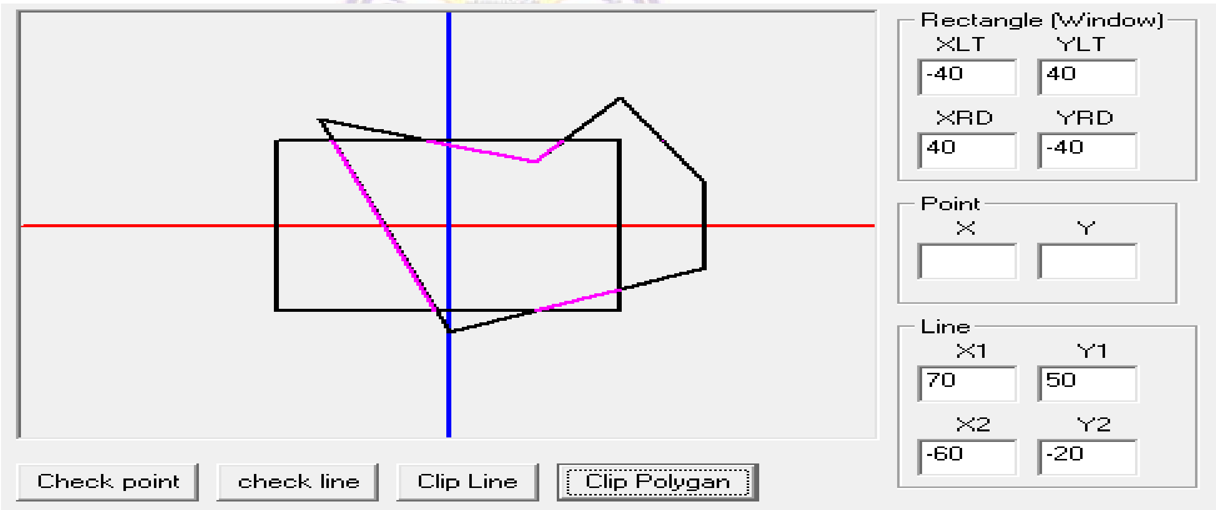
Then See edge (2,0),(3,3)&(3,3),(6,5)&(6,5),(4,7)&(4,7),(-3,7)& (-3,7),(2,5) & (2,5),(2,0) are category 3 in line clipping.

Finally :- need value for all X-market for new polygon vertices'.

H.W:- that leave student to do procedure and give vertices of polygon clipping. See result Below :-







Design in V.B by L. Ali Hassan Hammadie

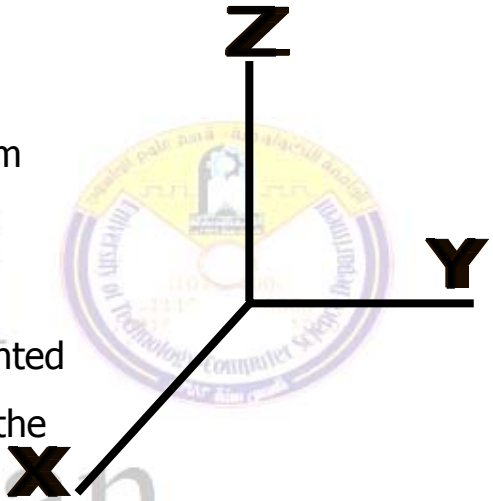
ALI HASSAN HAMMEDIE
COMPUTER GRAPHICS LECTURS
COMPUTER SCIENCE DEPARTMENT
UINVERSITY OF TECHNOLOGY
IRAQ-BAGHDAD

(Part Seven)

3D system & Transformation & Projection

Three-dimensional Transformation

- The world composed of three-dimensional images.
- Objects have height, width, and depth.
- The computer uses a mathematical model to create the image.



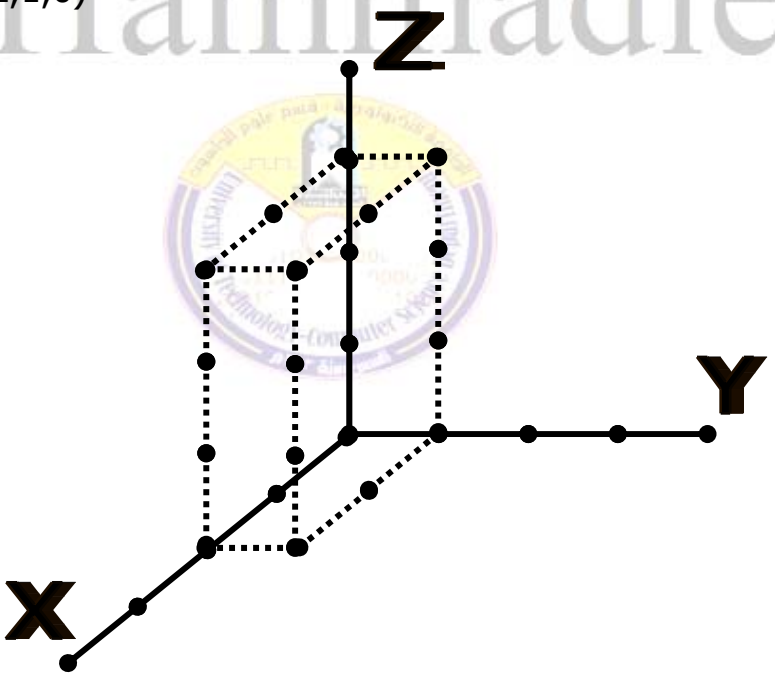
7.1:Coordinate System:

A three dimensional coordinate system can be view as an extension of the two dimensional 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

Ex./ Draw the figure: (0,0,3), (0,1,3), (2,0,3), (2,1,3), (0,1,0) (2,0,0), (2,1,0)



7.2 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

Example; if p(5,2,3) & Q(2,-4, -4) in figure below to find modulus of two vector are: $|P|= \sqrt{5^2 + 2^2 + 3^2} = \sqrt{38}$ And $|Q| = \sqrt{2^2 + (-4)^2 + (-4)^2} = \sqrt{36}$

7.3 unit vectors: the unit vector in direction of OP is written $\hat{OP}$, which is calculated as following : OP = vector OP/modulus of OP. in preview example then $\hat{OP}$ **= OP/|OP|=**$5i/\sqrt{38} + 2j/\sqrt{38} + 3k/\sqrt{38}$, $\hat{OQ}$ (H.W)

7.4 Angles between vectors and axis: using Direction Cosine

If P=Pi +Pj +Pk need $|P|=\sqrt{Pi^2 + Pj^2 + Pk^2}$

A. Angle between P with X-axis ➔ $\alpha=\cos^{-1}(Pi/|P|)$

B. Angle between P with X-axis ➔ $\beta=\cos^{-1}(Pj/|P|)$

C. Angle between P with X-axis ➔ £=$\cos^{-1}(Pk/|P|)$

Note:- Unit Vector is Direction Cosine for all Axis Depend of Components

7.5 adding vectors let P=P1i +P2j +P3k and Q=Q1i +Q2j +Q3k

P+Q=Q+P=(P1+Q1)i+(P2+Q2)j+(P3+Q3)k

7.6 subtracting vectors let P=P1i +P2j +P3k and Q=Q1i +Q2j +Q3k

P-Q= -(Q-P)=(P1-Q1)i+(P2-Q2)j+(P3-Q3)k

7.7:scaling Vectors let P=P1i +P2j +P3k nP= nP1i +nP2j +nP3k

7.8:**multiplying vectors uses the "*dot Product*"** let P=P1i +P2j +P3k and Q=Q1i +Q2j +Q3k ➔ ***P.Q=Q.P=(P1*Q1)+(P2*Q2) +(P3*Q3)=m***

7.9:multiplying vectors uses the "*Cross Product*" let a = ai+ aj+ ak

And b= bi +bj +bk is:

Where a×b= $\begin{bmatrix} +i & -j & +k \\ ai & aj & ak \\ bi & bj & bk \end{bmatrix}$

= [(aj * bk) - (ak * bj)]i -[(ai * bk) - (ak * bi)]j+[(ai * bj) - (aj * bi)]k

7.10:Transformation:

Transformations of 3 dimensions are simply extension of two dimension transformation.

A three-dimensional point (x, y, z) will be associated with homogeneous row vector [x, y, z, 1]. We can represent all three-dimensional linear transformation by multiplication of 4*4 matrix.

The new coordinate of a translate point can be calculate by using transformation.

$$T: \begin{cases} \underline{X} = X + a \\ \underline{Y} = Y + b \\ \underline{Z} = Z + c \end{cases}$$

$$(\underline{X}\ \ \underline{Y}\ \ \underline{Z}\ \ 1) = (X\ \ Y\ \ Z\ \ 1) * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ a & b & c & 1 \end{bmatrix}$$

7.11:Scaling:

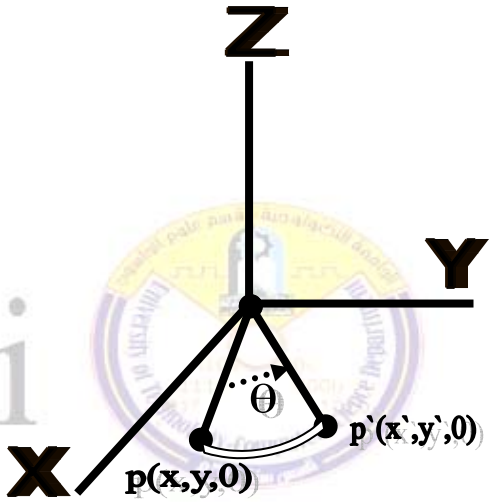
- Allows for a contraction or stretching in any of the x, y, or z direction. To scale an object:
    1. Translate the fixed point to the origin.
    2. Scale the object.
    3. Perform the inverse of the original translation.
- The scaling matrix with scale factors Sx, Sy, Sz in x, y, z direction is given by the matrix

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

$$S: \begin{cases} \underline{X} = Sx * X \\ \underline{Y} = Sy * Y \\ \underline{Z} = Sz * Z \end{cases}$$

$$(\underline{X}\ \ \underline{Y}\ \ \underline{Z}) = (X\ \ Y\ \ Z) * \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & Sz \end{bmatrix}$$

7.12:Rotation:



Rotation in three dimensions is considerably more complex than rotation in two dimensions.

In two dimensions, a rotation is prescribed by an angle of rotation θ and center of rotation p.

three dimensional rotations require the prescription of an angle of rotation and an axis of rotation. The canonical rotations are define when one of the positive x, y, or z coordinate axes is chosen as the axis of rotation. Then the construction of the rotation transformation proceeds just kike that of a rotation in two dimensions about the origin see figure above.

Rotation about the z-Axis

$$R\ \theta, z = \begin{cases} x` = x\cos\theta - y\sin\theta \\ y` = x\sin\theta + y\cos\theta \\ z` = z \end{cases}$$

Rotation about the y-Axis An analogous derivation leads to

$$R\ \theta, y = \begin{cases} x` = x\cos\theta - z\sin\theta \\ y` = y \\ z` = x\sin\theta + z\cos\theta \end{cases}$$

Rotation about the x-Axis

Similarly

$$R\ \theta, x = \begin{cases} x` = x \\ y` = y\cos\theta - z\sin\theta \\ z` = y\sin\theta + z\cos\theta \end{cases}$$

note that the direction of positive angle of rotation is chosen in accordance to the right-hand rule with respect to the axis of rotation. The corresponding matrix transformations are:

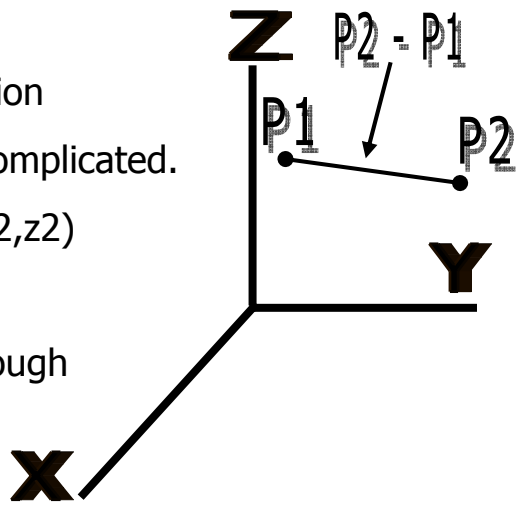
$$R\,\theta, z = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R\,\theta, y = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R\,\theta, x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}$$

The general use of rotation about an axis ***L*** can be built up from these canonical rotations using matrix multiplication in next section.

7.13:Rotation about an arbitrary Axis



- It is like a rotation in the two-dimension about an arbitrary point but it is more complicated.
- Two points P1(x1,y1,z1) and P2(x2,y2,z2) Define a line.
- The equation for the line passing through these Point are :

x= (x2 - x1) t + x1

y= (y2 - y1) t + y1 t: real value [0 to 1]

z= (z2 – z1) t + z1

- Let a=(x2 – x1) & b=(y2 – y1) & c=(z2 – z1)

then the equation of line becomes

x=at + x1 & y=bt +y1 & z=ct + z1 the difference P2 – P1 = (x2 – x1) (y2 – y1) (z2 – z1) = (a, b, c) is the direction vector from P1 to P2 along the line through P1 and P2.

A line can be defined by a point on (x, y, z) and by a direction (a, b, c)

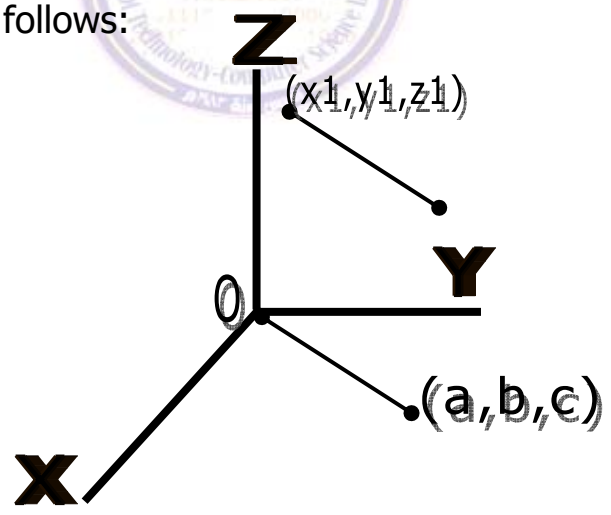
Steps of rotation:

Let (x1, y1, z1) be a point through which the rotation axis passes with (a, b, c) direction. A rotation of angle θ about an arbitrary axis is:

1. Translate the point(x1, y1, z1) to origin.

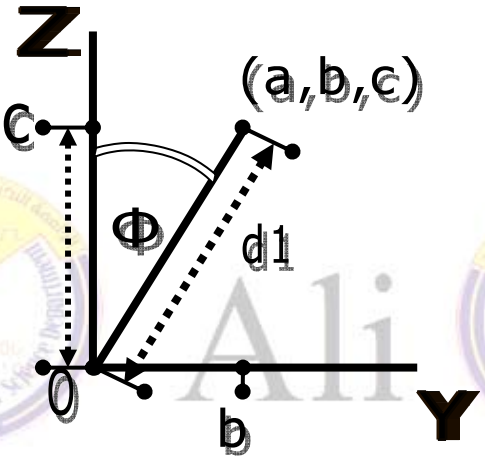
$$Tr(-x1, -y1, -z1)= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x1 & -y1 & -z1 & 1 \end{bmatrix}$$

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



2. Rotate about the x-axis until the rotation axis corresponds to the z-axis.

This can be consider to be a rotation about the origin. With the axis coming out of paper



When the rotation axis is projected onto the x,z plane, any point on it has x coordinate equal to zero. In particular a=0.

- The point (0,b,c) is rotated **Φ** degree until the line corresponds to the z-axis. We have find the sin **Φ** and cos **Φ** we find that distance from the origin to (0,b,c) is : $\sqrt{\mathbf{b^2 + C^2 = d1}}$

Sin Φ= b/d1 ; Cos Φ= c/d1

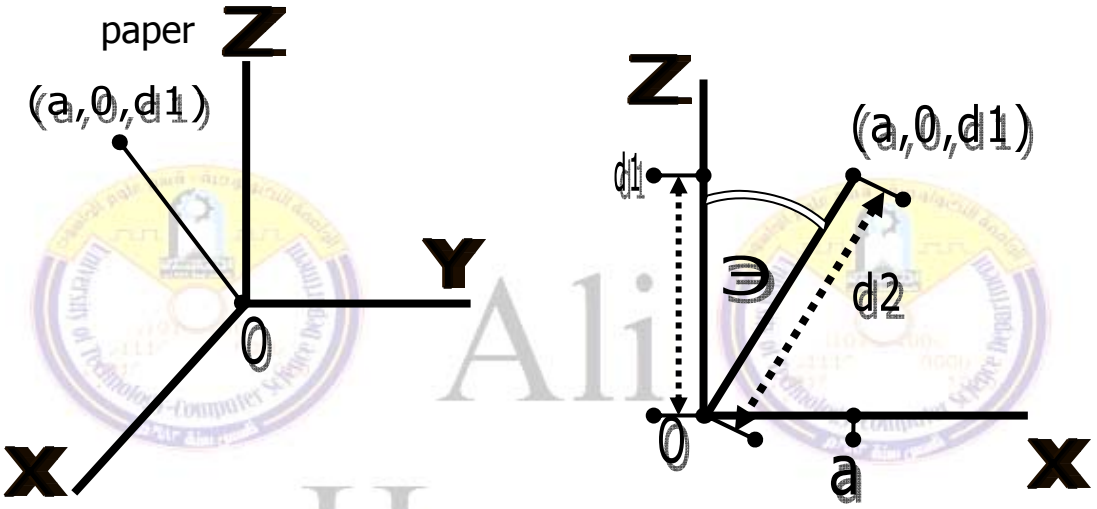
Substituting these values into the x-axis rotation matrix we have:

$$\mathbf{Rx(\Phi)} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{c/d1} & \mathbf{b/d1} & \mathbf{0} \\ \mathbf{0} & \mathbf{-b/d1} & \mathbf{c/d1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

Now the point(o,b,c)has been transformed to the point (0,0,d1) but since the rotation about the x-axis doesnot change the x coordinate value the point (a,b,c) is now at location (a, 0, d1).

3. Rotate about the y-axis until the rotation axis corresponds to the z-axis.

Since (a,0,d1) lies in the x, z plane we can visualize this as rotation about the origin with the y-axis coming out of the paper



A rotation of angle Ə in clockwise direction, we need to compute *sin Ə, cos Ə where:*

$$d2 = \sqrt{a^2 + d1^2} = \sqrt{a^2 + b^2 + c^2}$$ *thus:*

$$\sin Ə = a/d2 \; ; \quad \cos Ə = d1/d2 \quad \text{and}$$

$$\sin(-Ə) = -a/d2 \; ; \quad \cos(-Ə) = d1/d2$$

Substituting the value into y rotation matrix given:

$$Ry(-Ə) = \begin{bmatrix} d1/d2 & 0 & a/d2 & 0 \\ 0 & 1 & 0 & 0 \\ -a/d2 & 0 & d1/d2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4. ***Rotate about the z-axis angle*** *ϴ*. This require the Rz(ϴ) matrix
5. ***perform the inverse rotation of step (3)*** **.** requires Ry(Ə)

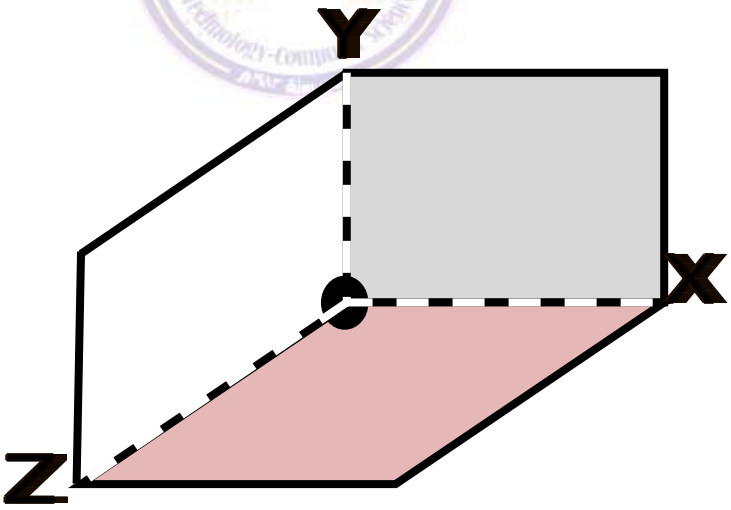
6. ***Perform the inverse rotation of step (2).*** Requires Rx(-Φ)
7. ***Perform the inverse translation of step (1).*** Require Tr (x1,y1,z1)

The composite transformation is:

Tr(-x1,y1,z1) * Rx(Φ) * Ry(-Э) * Rz(∈) * Ry(Э) * Rx(-Φ) *

7.14 Mirror in 3D :- Mirror in 3D can category into:-

- In Original Point: (x, y ,z)➔ **(-x,-y,-z)**
- In main Axis **X or Y or Z**
  - **In X-axis: (x, y, z)➔ (x,-y,-z)**
  - **In Y-axis: (x, y, z)➔ (-x, y,-z)**
  - **In Z-axis: (x, y, z)➔ (-x,-y, z)**
- In Plane **XY ,YZ , XZ**
  - **In Plane XY: (x, y, z)➔ (x, y, -z)**
  - **In Plane YZ: (x, y, z)➔ (-x, y, z)**
  - **In Plane XZ: (x, y, z)➔ (x, -y, z)**

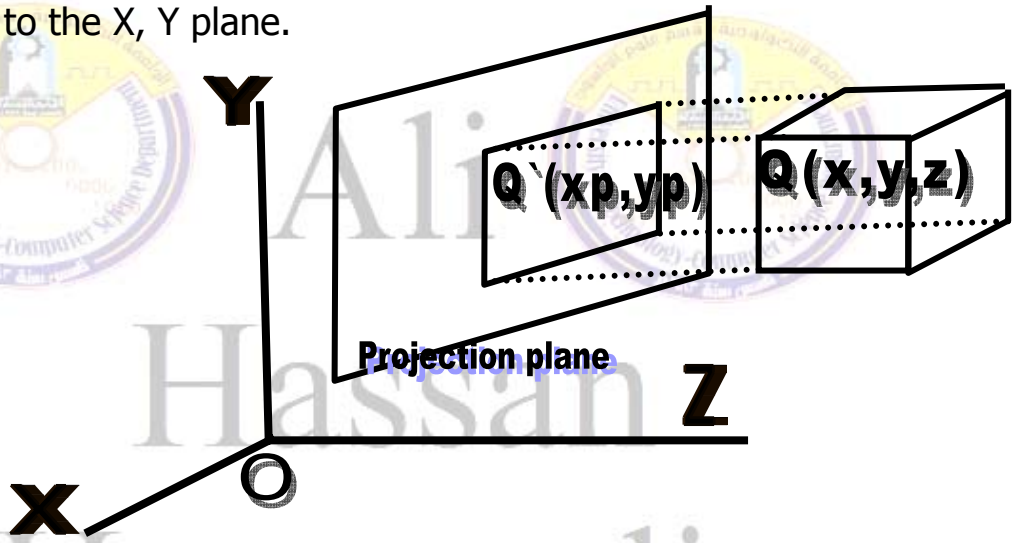


7.15>Projects

A projection is transformations that perform a conversion from three-dimension representation to a two dimension representation.

7.14.*A Parallel (orthogonal) projection:*

A parallel projection is to discard one of the coordinate. Like dropping the Z coordinate and project the X, Y, Z coordinate system in to the X, Y plane.



The projection of a point Q(x, y, z) lying on the cube is point Q`(xp, xy) in the x, y plane where a line passing through Q and parallel to the Z-axis intersect the X, Y plane these parallel line called projectors and we get ***Xp=X ; Yp=Y .***

- Straight lines are transformed into straight lines.
- Only endpoints of a line in three-dimension are projected and then draw two-dimensional line between these projected points.
- The major disadvantages of parallel projection is its lack of depth information.

Explanation:

- Let [xp yp zp] is a vector of the direction of projection.
- The image is to be projected onto the x y plane.
- If we have a point on the object at (x1, y1, z1) we wish to determine where the projected point (x2, y2) will lie.
- The equation for a line passing through the point (x,y,z) and in the direction of projection

X= x1 + xp * u

Y= y1 + yp * u

Z= z1 + zp * u

If Z=0 then u= -z1/zp

Substituting this into the first two equation:

X2= x1 – z1 (xp / zp)

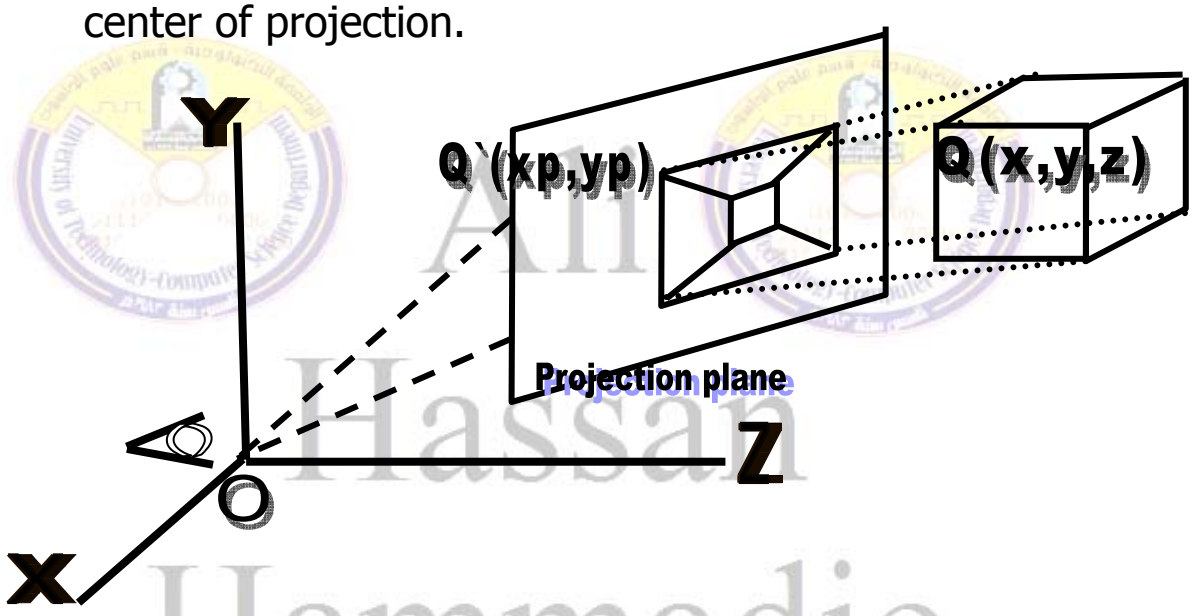
Y2 = y1 – z1 (yp / zp)

Written in matrix form we set:

$$[x2\ y2\ z2\ 1]=[x1\ y1\ z1\ 1]*\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -xp/zp & yp/zp & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

7.14.B Perspective projection

- The further a way an object is from the viewer the smaller it appears.
- These provide the viewer with a depth cue.
- All line are converging at a single point called the center of projection.



If the center of projection is at (xc, yc, zc) and the point on the object is (x1, y1, z1) then the projection ray will be the line containing these point and will give by:

$$X = xc + (x1 - xc)\,u$$

$$Y = yc + (y1 - yc)\,u$$

$$Z = zc + (z1 - zc)\,u$$

The projection point (x2, y2) will be the point where this line intersects the xy plane.

The third equation tells us that u for this intersection point (Z=0) is **u= - zc/(z1-zc)**

substituting into the first two equation gives:

$$x2 = xc - zc\ [\ (x1 - xc)/(z1 - zc)\ ]$$

$$y2 = yc - zc\ [\ (y1 - yc)/(z1 - zc)\ ]$$

this can be written as:

$$x2 = (\ xc * z1 - x1 * zc\ ) / (\ z1 - zc\ )$$

$$y2 = (\ yc * z1 - Y1 * zc\ ) / (\ z1 - zc\ )$$

this projection can be put into the form of transformation matrix.

$$P = \begin{bmatrix} -Zc & 0 & 0 & 0 \\ 0 & -Zc & 0 & 0 \\ Xc & Yc & 0 & 1 \\ 0 & 0 & 0 & -Zc \end{bmatrix}$$

It is equivalent from of the projection transformations

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -Zc/Xc & -Zc/Yc & 0 & -1/Zc \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Note: If Q(x, y, z) be a point that project to the point Q`(xp, yp) in center of projection (0, 0, D) where is distance from the eye to the projection plane the perspective transformation

$$xp = (D * x) / (z + D)$$

$$yp = (D * y) / (z + D)$$

$$zp = 0$$

The perspective transformation matrix

$$P=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/D \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

7.14.*c* Oblique projection

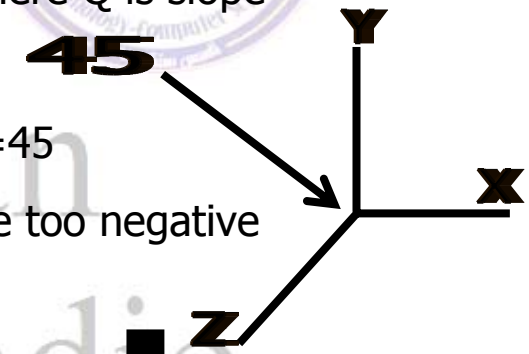
That show 3D reality by equation:-

X'=X+(Z*cos Q) & Y'=Y+(Z*sin Q) where Q is slope

Z-Axis of coordinate as following:-

X'=X+(Z*-0.7) & Y'=Y+(Z*-0.7) ➔Q=45

Sin45=cos45 ≈0.7 in three quarter are too negative



Matrix representation

$$[X'\ Y'\ Z'] = [X\ Y\ Z] * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -CosQ & -sinQ & 1 \end{bmatrix}$$

$$= [\ X-Z*cosQ \quad Y-Z*sinQ \quad 0\ ]$$

if need Distance only add distance as d in $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ d*-CosQ & d*-SinQ & 1 \end{bmatrix}$

$$=[\ X-d*Z*cosQ \quad Y-d*Z*sinQ \quad 0\ ]$$

ALI HASSAN HAMMEDIE
COMPUTER GRAPHICS LECTURS
COMPUTER SCIENCE DEPARTMENT
UINVERSITY OF TECHNOLOGY
IRAQ-BAGHDAD

(Part Eight)

Curve Spline & 3D Shapes

Spline Curve

This Chapter talk method for curve drawing & curve fitting are

{Bezier Curve, B-spline curve , Cubic curve }

8.1: Bezier Curve uses a sequence of control points, P_1, P_2, P_3, P_4 to construct a well defined curve $P(t)$ at each value of t from *0* to *1*.This provides a way to generate a curve from a set of points. Changing the points will change the curve. $P(t)$ is defined as:

$$P(t)=(1-t)^3P_1+3(1-t)^2tP_2+3(1-t)t^2P_3+t^3P_4 \ldots\ldots\ldots\ldots (1)$$

Code Segment :- ***Let X1,X2,x3,X4 & Y1,Y2,Y3,Y4 are control points***

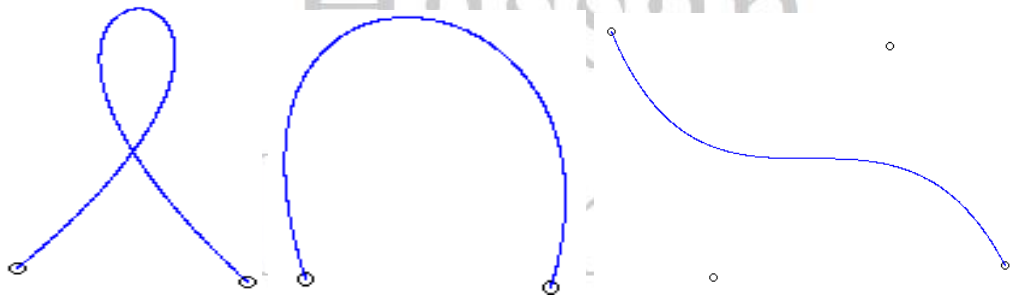
For t = 0 To 1 Step 0.0001 "to smooth

```
x = (1 - t) ^ 3 * X1 + 3 * (1 - t) ^ 2 * t * X2 + 3 * (1 - t) * t ^ 2 * x3 + t ^ 3 * X4
y = (1 - t) ^ 3 * Y1 + 3 * (1 - t) ^ 2 * t * Y2 + 3 * (1 - t) * t ^ 2 * y3 + t ^ 3 * y4
```

plot point (x, y)

Next t

Finally : the first and last points are fitting but other are effected not fitting.



8.2: B-spline Curve:- uses a sequence of control points, P_1, P_2, P_3, P_4 to construct a well defined curve of degree three, at each value of ***t*** from *0* to *1*.This provides a way to generate a curve from a set of points. Changing the points will change the curve. $F(t)$ defined as

$$F(t)=\frac{1}{6}(1-t)^3p_1+\frac{1}{6}\{3t^3-6t^2+4\}p_2+\frac{1}{6}\{-3t^3+3t^2+3t+1\}p_3+\frac{1}{6}t^3p_4 \ldots\ldots\ldots(2)$$

Code Segment :- Let X1,X2,x3,X4 & Y1,Y2,Y3,Y4 are control points

For t = 0 To 1 Step 0.0001

```
x =((1-t)^3*X1 + (3*t^3 -6*t^2 +4)*X2 + (-3*t^3 +3*t^2 +3*t+1)*x3 + t^3*x4) / 6
y =((1-t)^3*Y1 + (3*t^3 - 6*t^2 +4)*Y2 + (-3*t^3 +3*t^2 +3*t +1)*y3 +t^3*y4) / 6
```

plot point (x, y)

Next t

Finally : the B-spline curve is not fitting any control point but it inside curve points grouping

8.3: <u>Cubic Curve:-</u> *n points curve points that enable fitting all curve points where*
$F(t)=(t)^3 a_i+(t)^2 b_i+ (t) c_i+ e_i$. **<u>where</u>** t=[0..1] and $F(t)= e_{i+1}$
$a_i=(D_{i+1} - D_i) /6$. & $b_i= D_i / 2$. & . $c_i=(x_{i+1} - x_i)-(2D_i + D_{i+1}) /6$. Or
$c_i=(y_{i+1} - y_i)-(2D_i + D_{i+1}) /6$. & $e_i= x_i$ or y_i
$Dx_i= [(x_{i+1} - x_i)- (x_i - x_{i-1})] * (3/2)$ where $Dx_{\text{start point}} = 0$ & $Dx_{\text{end point}} = 0$
$Dy_i= [(y_{i+1} - y_i)- (y_i - y_{i-1})] * (3/2)$ where $Dy_{\text{start point}} = 0$ & $Dy_{\text{end point}} = 0$

<u>How can find this</u>

$F(t)=(t)^3 a_i+(t)^2 b_i+ (t) c_i+ P_i$ **…..(1)**

$F'(t)=3(t)^2 a_i+2(t) b_i+ c_i$ **…….(2)**

$F''(t)=6(t) a_i+2 b_i$ **…. (3)** → $F''(0)=D_i$ & $F''(1)=D_{i+1}$

let t=0 in equ.(3) ➔ $D_i=0+2b_i$ ➔ **$b_i=D_i/2$ …..(4)** where $D_i=F''(0)$

let t=1 in equ.(3)➔$D_{i+1}=6a_i+D_i$➔**$a_i=(D_{i+1}-D_i)/6$ …..(5)** where $D_{i+1}=F''(1)$

apply equ.(4,5) in equ(1) in t=1 then

$$P_{i+1}=\frac{D_{i+1}-D_i}{6}+\frac{D_i}{2}+C_i+P_i ==> (P_{i+1}-P_i)=(\frac{D_{i+1}+2D_i}{6})+C_i ===>$$

$$C_i=(P_{i+1}-P_i)-(\frac{D_{i+1}+2D_i}{6}).....(6)$$

'<u>step 1</u>: WHERE np = number of control points

```
dx(1) = 0: dx(np) = 0: dy(1) = 0: dy(np) = 0
  For i = 2 To np - 1
    dx(i) = ((X(i + 1) - X(i)) - (X(i) - X(i - 1))) * (3 / 2)
    dy(i) = ((Y(i + 1) - Y(i)) - (Y(i) - Y(i - 1))) * (3 / 2)
  Next i
```

'<u>step 2:</u> ' find a,b,c,e for x in all points

```
For j = 1 To np - 1
  ax(j) = (dx(j + 1) - dx(j)) / 6.0                          : bx(j)=dx(j)/2
  cx(j) = ((X(j + 1) - X(j))) + ((-2 * dx(j) - dx(j + 1)) / 6.0) : ex(j)=X(j)
        'find a,b,c,e for  y for all points
  ay(j) = (dy(j + 1) - dy(j)) / 6.0                          : by(j)=dy(j)/2
  cy(j) = ((Y(j + 1) - Y(j))) + ((-2 * dy(j) - dy(j + 1)) / 6.0) : ey(j) = Y(j)
Next j
```

'<u>step 3</u> apply equ.(1)

```
For P = 1 To np
 For T = 0 To 1 Step 0.0001
   xp = (T ^ 3) * ax(P) + (T ^ 2) * bx(P) + (T) * cx(P) + ex(P)
   yp = (T ^ 3) * ay(P) + (T ^ 2) * by(P) + (T) * cy(P) + ey(P)
   plote point (xp, yp) ' draw Curve points
  Next T
Next P
End Sub
```

Let see figure

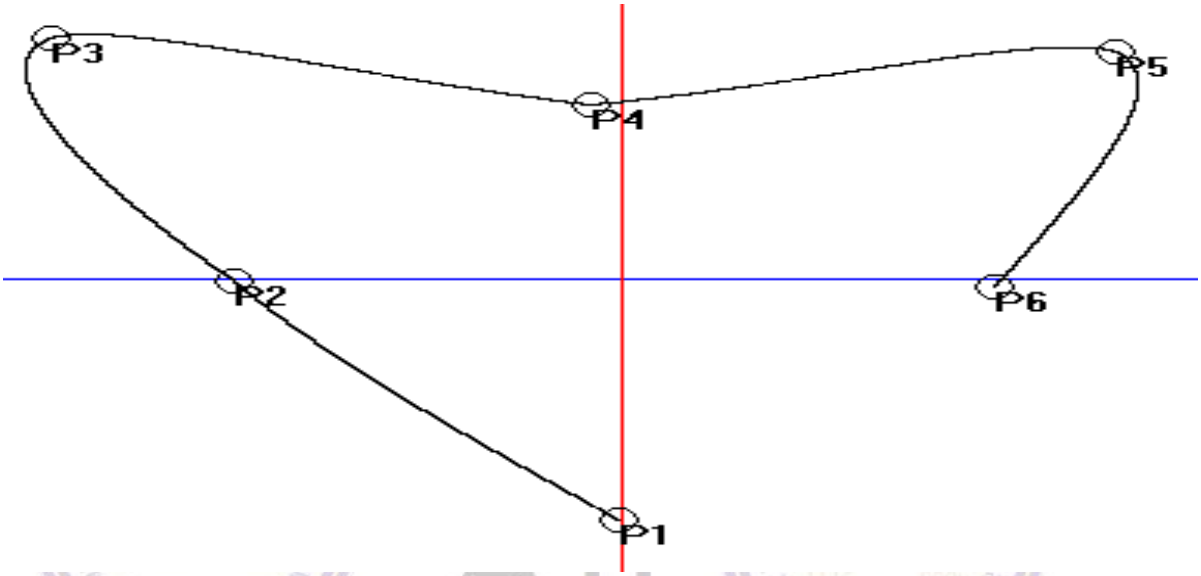


Figure A. design in V.B by L. Ali Hassan Hammadie

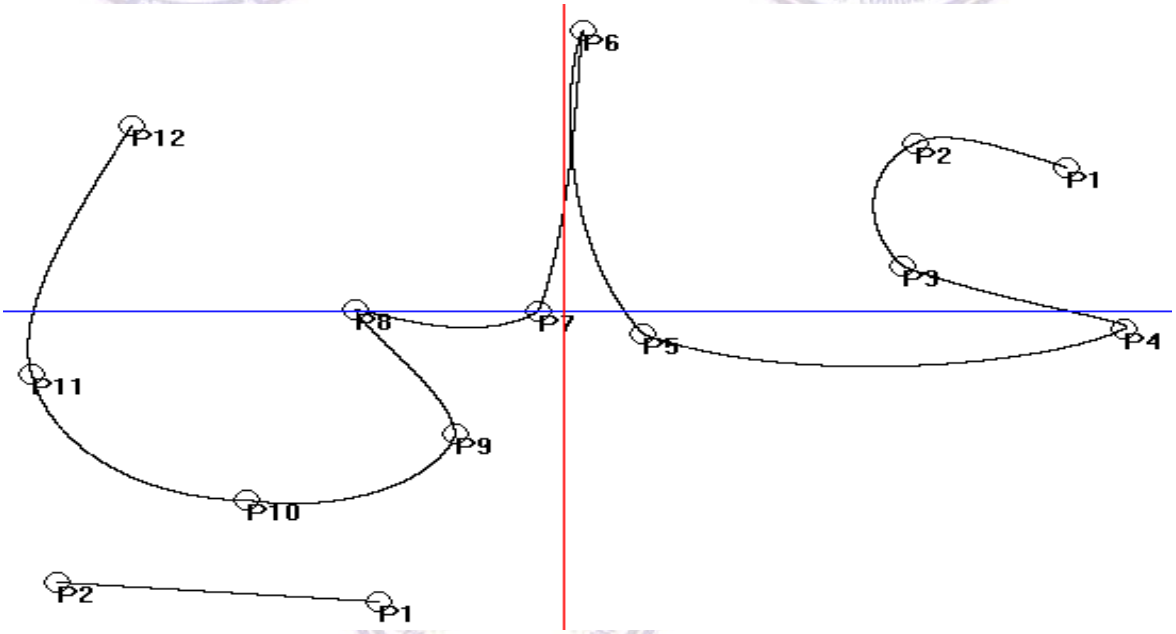


Figure B. design in V.B by L. Ali Hassan Hammadie

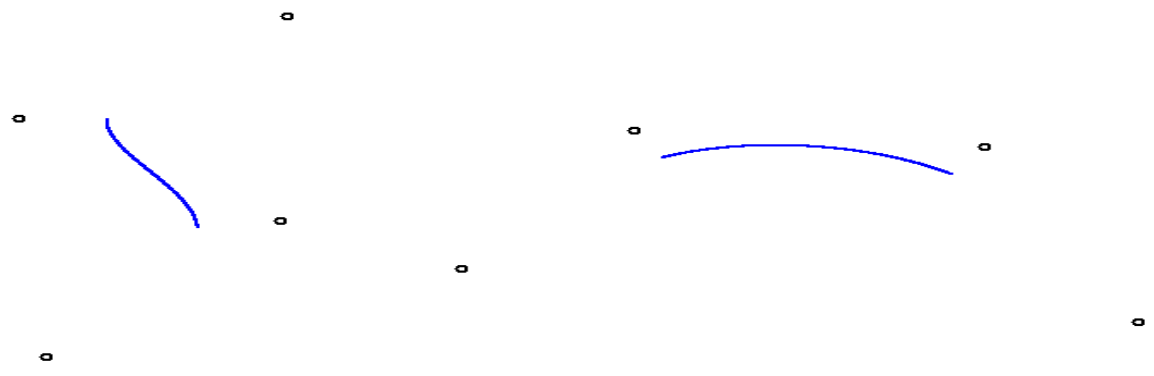


Figure C. B-spline

$F_i(t)=(t)^3 a_i+(t)^2 b_i+(t) c_i+ e_i$. **where** t=[0..1] & $F_i(t)$ *is* $\{X_{i+1}$ *OR* $Y_{i+1}\}$

$a_i=(D_{i+1} - D_i)/6$.

$b_i= D_i / 2.$ & .

$c_i=(x_{i+1} - x_i)-(2D_i + D_{i+1})/6.$ OR $c_i=(y_{i+1} - y_i)-(2D_i + D_{i+1})/6.$

$e_i= X_i$ OR Y_i

How To find a_i, b_i, c_i

F(t)=(t)³ ai+(t)² bi+ (t) ci+ Pi …..(1)

F'(t)=3(t)²ai+2(t) bi+ ci …….(2)

F''(t)=6(t) ai+2 bi …. (3) ➔ *F''(0)=Di & F''(1)=Di+1*

let t=0 in equ.(3) ➔ Di=0+2bi ➔ **bi=Di/2 …..(4)** where Di=F''(0)

let t=1 in equ.(3) ➔Di+1=6ai+Di➔**ai=(Di+1-Di)/6 …..(5)** where Di+1=F''(1)

apply equ.(4,5) in equ(1) in t=1 because if t=0 therefore $F_i(t)$ *is* $\{X_i$ *OR* $Y_i\}$ **then**

$$P_{i+1}=\frac{D_{i+1}-D_i}{6}+\frac{D_i}{2}+C_i+P_i \Rightarrow (P_{i+1}-P_i)=(\frac{D_{i+1}+2D_i}{6})+C_i \Rightarrow$$

$$C_i=(P_{i+1}-P_i)-(\frac{D_{i+1}+2D_i}{6}).....(6)$$

To find *Di*➔ **need Solve**

$$h_iD_i+2(h_i+h_{i+1})D_{i+1}+h_{i+1}D_{i+2}=6\left(\frac{P_{i+2}-P_{i+1}}{h_{i+1}}-\frac{P_{i+1}-P_i}{h_i}\right)$$

$$D_1+4D_2+D_3=6(P_3-2P_2+P_1)$$
$$D_2+4D_3+D_4=6(P_4-2P_3+P_2)$$
$$D_3+4D_4+D_5=6(P_5-2P_4+P_3)$$
$$Then \Rightarrow D_{n-2}+4D_{n-1}+D_n=6(P_n-2P_{n-1}+P_{n-2})$$

where $h_i= t_{i+1}- t_i=1$

Suppose n=5 the matrixes to solve are:

Gray Raw Special Case in raw(1) & raw(n)

$$\begin{bmatrix}1&0&0&0&0\\1&4&1&0&0\\0&1&4&1&0\\0&0&1&4&1\\0&0&0&0&1\end{bmatrix} * \begin{bmatrix}D_1\\D_2\\D_3\\D_4\\D_5\end{bmatrix} = 6 \begin{bmatrix}0\\P_3-2P_2+P_1\\P_4-2P_3+P_2\\P_5-2P_4+P_3\\0\end{bmatrix}$$

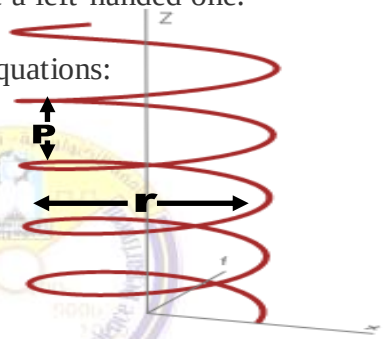
Note:- D_1 & $D_n = 0$

Finally:- therefore need Elimination methods to solve To Find D_i ***or using TDMA***

3D Shape { Helix , Sphere }

Helix definition

A helix is a type of smooth curve in three-dimensional space. It has the property that the tangent line at any point makes a constant angle with a fixed line called the axis. Helices can be either right-handed or left-handed. With the line of sight along the helix's axis, if a clockwise screwing motion moves the helix away from the observer, then it is called a right-handed helix; if towards the observer, then it is a left-handed helix. A right-handed helix cannot be turned or flipped to look like a left-handed one. Most machine screw threads are right-handed helices.
A cylindrical helix may be described by the following parametric equations:



X = Xc + r * Cos(t)

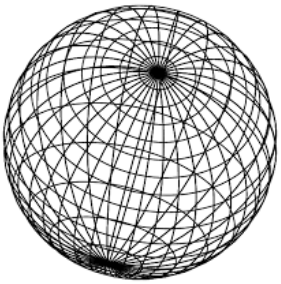
Y =Yc + r * Sin(t)

z = Zc + p * (t) ' it's round about Z-axis

where t [angle] € (-∞,∞)

(Xc, Yc, Zc) is center of Helix

Sphere (from Greek sphaira, "globe, ball) :- *From Wikipedia, the free encyclopedia*
It is a perfectly round geometrical object in three-dimensional space that is the surface of a completely round ball,. Like a circle, which geometrically is a 2D object, a sphere is defined mathematically as the set of points that are all at the same distance **r** from a given point, but in 3D space. This distance **r** is the radius of the ball, and the given point is the center of the mathematical ball. The longest straight line through the ball, connecting two points of the sphere, passes through the center and its length is thus twice the radius; it is a diameter of the ball.



```
For k = 0 To 360 Step m          ' m is a texture height line
  For n = 0 To 360 Step v        'v is rings circle width line
    Y = r * Sin(n * pi / 180) * Cos(k * pi / 180)
    X = r * Sin(n * pi / 180) * Sin(k * pi / 180)
    z = r * Cos(n * pi / 180)
   'z-rotation
     X2 = X * Cos(az) - Y * Sin(az)      ' az :- angle rotate about Z-axis
     Y2 = X * Sin(az) + Y * Cos(az)
   'x-rotation
    z2 = z * Cos(ax) - Y2 * Sin(ax)      ' ax :- angle rotate about X-axis
    Y1 = z * Sin(ax) + Y2 * Cos(ax)
  'y-rotation
    X1 = X2 * Cos(ay) - z2 * Sin(ay)     ' ay :- angle rotate about Y-axis
    z1 = X2 * Sin(ay) + z2 * Cos(ay)
    picture1.PSet (X1 + (z1 * -0.7), Y1 + (z1 * -0.7))     ' using oblique Projection
  Next n
Next k
```