Computer graphics: The term computer graphics involves using a computer to create and hold pictorial information and also to adept and manipulate thedisplay in different ways.

What is computer graphics

الرسم على الحاسوب هو طريقة حديثة لتقنية ادخال واخراج الاشكال والصور المرئيه على الحاسوب بما فيها من تكوين ومعالجة وعرض الصور باستخدام الحاسب , فالرسم على الحاسب هو حقل جديد من حقول علم الحاسوب وهو سريع التوسع, وله مصطلحات وطرق ومكونات مادية خاصة به.
الرسم بالحاسوب يمثل احدث التطورات في تحسين كفاءة الاتصالات بين البشر والحاسوب , اي انها اصبحت اسرع بكثير فالصورة تثمن بالاف الكلمات لان العين البشريه يمكنها استيعاب معلومات من شكل مرسوم على جهاز عرض او من منضر ثلاثي الابعاد بصوره اكبر من المعلومات المكتوبه سواء كانت جدول من الارقام او رساله نصية .
الكلفه العالية لتقنية الرسم على الحاسوب جعلت انتشاره محدود لسنوات عديده , مع انخفاض الكلفه اصبحت تلعب دورا كبيرا ومهما في الحياة العملية.

Applications:

رسومات الحاسوب تدخل في مجالات متعددة منها على سبيل المثال:

- Computational biology
- Computational physics
- Computer-aided design (CAD)
- Computer simulation
- Digital art
- Education
- Graphic design
- Information visualization
- Video Games
- Web design
- Architecture

سوف نستعرض واحد من هذه البرامج الواسعة الاستخدام:

Computer aided design (CAD):

يعتبر نظامي التصميم المسند بالحاسب والتصنيع المسند بالحاسب CAD/CAM من الانظمه التي تنمو فيها رسوم الحاسب بشكل واسع ,فمثلا الدوائر الالكترونية في الدوائر المتكاملة IC يمكن ان تخزن كرسم على الحاسب بملف في ذاكرة الحاسب الثانوية .

باستخدام ال التصميم المسند بالحاسوب CAD يمكن توفير الكثير من الجهد والوقت ,حيث يوفر هذا البرنامج دوائر الكترونية جاهزه يمكن تنزيلها على لوحة العمل في الحاسوب من خلال سحبها بواسطة الفارة وعمل الدائرة الالكترونية ومن ثم اجراء التعديلات عليها والاختبارات قبل التصنيع وهذا مايعرف بال computer aided design.

اما ال CAM والمعروف ب compuuter aided manufacturing فيستخدم لاسعاف المصنع , فالتصميم يرسم على الحاسب ثم يتم تصويره وتصغيره (اما بالتصوير او بالحاسوب)ويستخدم كقناع لحفر الدوائر الالكترونية .

Classification of computer graphics

يمكن تصنيف الرسوم الى نوعين وهي:

Non interactive graphics: وهي رسوم غير تفاعلية يمكن توليدها في الحاسوب ولا توفر امكانية تفاعل المستخدم معها , مثل العاب الفيديو والاعلانات المتحركة .

Interactive graphics: هي الرسوم التي توفر للمستخدم بعض السيطرة على الصورة بتجهيزه باحدى اجهزة الادخال مثل الفارة او القلم الضوئي .

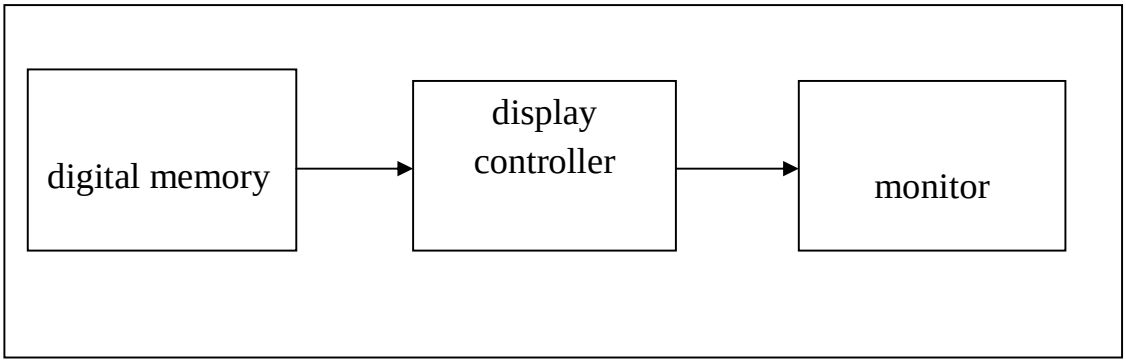
توفر الرسوم التفاعلية اتصال باتجاهين بين الحاسبة والمستفيد ,بالاعتماد على الاشارة المرسله من جهاز الادخال يمكن تحديث الصورة الضاهرة بطريق مناسبة .

بالنسبة للمستفيد فان الصورة سوف تتغير فوريا استجابة لايعازاته.

How the interactive graphics display works:

لاجل بناء وعرض الصورة على شاشة الحاسوب يجب توفر ثلاث عناصر اساسيه وهي:

1- A digital memory, or frame buffer.
وفيها يتم خزن الصورة الضاهره على شكل مصفوفة ثنائية البعد من القيم الرقمية.
2- A monitor.
وفيها تجري عملية عرض الصورة وهي تشبه التلفاز العادي مع عدم احتوائها على دوائر الاستقبال وضبط الموجه.
3- A display controller.
يمثل واجهة بسيطه (interface) يقوم بتحرير محتويات الذاكرة الرقمية (digital memory) الى شاشة العرض (moniter).



يجب ان يتم ارسال الصوره بصورة مستمره ومتوالية الى شاشة العرض ,على الاقل 30 مرة بالثانية.
في داخل frame buffer يتم خزن الصوره على شكل نموذج من الارقام الثنائية (bibnary digital number) والتي تمثل مصفوفة مستطيله من العناصر الاساسية للصوره (picture element) والتي تسمى pixel والذي يعتبراصغر عنوان على شاشة الحاسوب.

في ابسط حالة لخزن الصورة من الابيض والاسود يمكن تمثيل ال(pixels) في ال (frame buffer) بالرقم zero للون الاسود و 1 للون الابيض .

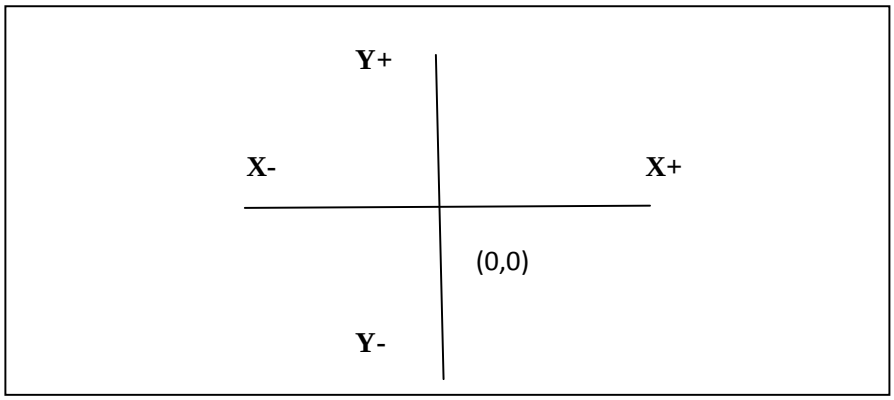
This mean:-
When the bit = 0 that mean black
When the bit= 1 that mean white

يقوم ال display controller بقراءة ال bytes المتعاقبه من ال الذاكرة وتحويل قيمها الى ماينا ضرها من اشارة الفيديو(التناضرية) وهذه الاشارات يتم تغذيتها الى شاشة العرض لانتاج نموذج الصورة من الابيض والاسود.

- عند الحاجه الى تغير الصورة الضاهرة يجب تغير محتويات frame buffer لتمثل الشكل الجديد.

Cartesian coordinate system

A coordinate system provide a framework for translating geometric ideas into numerical expressions.In a two-dimensional plane, we pick any point and single it out as a reference point called the origin. Through the origin we construct two perpendicular number lines called axes. These are labeled the X axis and the Y axis. Any point in two dimensions in this X-Y plane can be specified by a pair of numbers, the first number is for the X axis, and the second number is for the Y axis.



Graphical user interface:

The aspects of the user interface of a program are the parts of the program that link the user to the computer and enable him to control it.
A good user interface makes the program not only easy to use and learn but also easier to operate and more efficient and vis versa.

هو الجزء المسؤول عن ربط المستخدم بالحاسوب ويمكنه من السيطرة عليه , واجهة المستخدم الجيدة هي التي تجعل البرنامج سهلا للاستخدام والتعلم وتوفر واجهة تشغيل للرسومات اكثر كفائة .

Graphics Devices:

تعتبر ال plotters , laser printer plotters و films , storge tube ,

من الامثلة على اجهزة عرض الصورة, تدعم اغلب انظمة raster scan cathode Ray tube(CRT) رسوم الحاسوب النوع raster scan cathode Ray tube(CRT) .

يوجد ثلاث انواع مشهورة من تقنية اجهزة العرض (CRT) وهي:

1- Direct view storage tube display.

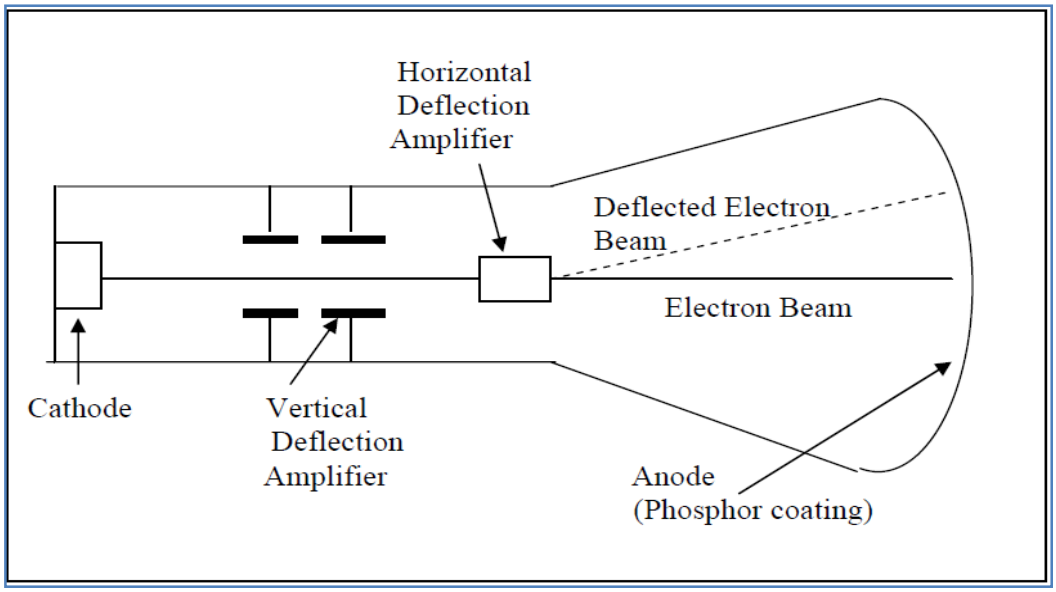
2- Calligraphic refresh display.

3- Raster scan refresh display.

قبل البدء باستعراض تقنيات ال CRT يجب معرفة وفهم اساسيات ال CRT.

cathode ray tube (CRT):

صمام الشعاع الكاثودي(*cathode ray tube* **)** :هو صمام إلكتروني ينتج فيضا من الإلكترونات على هيئة شعاع دقيق, وكما مبين في الشكل ادناه.



ويستخدم هذا الصمام الإلكتروني في أجهزة الرادار وفي أجهزة التلفزة . في تلك الأجهزة تتحكم أقطاب كهربية موجبة الشحنة وتعمل على انحراف شعاع الإلكترونات الناشئة من القطب السالب (كاثود cathode) ليقع على شاشة الصمام فيحدث نقطة مضيئة . وبتغيير مقدار الشحنة على أقطاب التحكم يمكن انحراف الشعاع الإلكتروني ليرسم منحنى أو شكلا هندسيا منتظما. وفي أجهزة التلفزة و في شاشات الحاسوب تستعمل لولبيات لإنتاج مجال كهرومغناطيسي يمكن بواسطتها التحكم في انحراف شعاع الإلكترونات لتغطية مدي زوايا كبيرة .

يتكون صمام الشعاع الكاثودي من أنبوبة مغلقة من الزجاج تتسع بشكل قمعي وتنتهي بشاشة للعرض . والأنبوب مفرغ من الهواء . يوجد في أول الأنبوب الكاثود أو المهبط على هيئة فتيل يولد الإلكترونات)، ويخرج شعاع الإلكترنات منجذبا نحو المصعد الموجب الشحنة ويمر بعدة أقطاب تعمل على تركيز الشعاع في بؤرة عند لقائه للشاشة .

- يزوّد الأنبوب بملفات كهربائية Deflection Coils (أنظر الشكل أعلاه) تعمل على توليد مجال كهرومغناطيسي ذو تردد سريع يعمل بدوره على توجيه شعاع الإلكترون ، وانتقاله متتابعا على الشاشة من نقطة إلى نقطة ، حتى يغطي مساحة الشاشة في عدد نحو 625 * 1024 من النقاط لتكوين الصورة.

- **Type of graphics devices**

1- Storage Tube Graphics Displays

This storage tube display, also called a bistable storage tube, can be considered a CRT with a long persistence phosphor. A line or character will remain visible up to a hour until erased. To draw a line on the display the intensity of the electron beam is increased sufficiently to cause the phosphor to assume its permanent bright "storage". The display is erased by flooding the entire tube with a specific voltage which causes the phosphor to assume its dark state.

هو نوع خاص من انواع ال CRT , تعتبر الشاشة كنوع من انواع الذاكرة لذلك تسمى ب **Storage Tube ,** لان الشاشه تبقى محتفضة بالسطوع بعد تسليط شعاع الالكترونات عليها لفتره طويله تمتد الى ساعه لحين مسح الشاشه بتسليط فيض من الشحنات ذات فولتية معينه .استخدم بكثرة بداية ال 1960 وخاصة في المحطات الطرفية computer terminal.

- لاحاجة الى ذاكرة الوصول العشوائي (RAM) والتي كانت مكلفة في ذالك الوقت.

Features of storage tube graphics display

1- Storage tube display is flicker free

لا تتميز بالسطوع .

2- Capable of displaying an unlimited number of vectors

القابلية على توليد عدد غير محدود من المتجهات(الخطوط).

3- Resolution is typically 1024 X 1024 addressable points on an 8 X 8 inch square or 4096 X 4096 on either 14 X 14 or an 18 X 18 inch square.

3- Display of dynamic motion or animation is not possible.

عدم امكانية عرض الرسوم التفاعلية.

5- A storage tube display is a line drawing or random scan display. This means that a line (vector) can be drawn directly from any addressable point to any other addressable point.

This device plots continuous lines and curves rather than separate pixels.

قابليتها على رسم اي متجه مباشرة من اي نقطه الى نقطة اخرى بدلا من النقاط المفصوله .

6- A storage tube display is easier to program than a calligraphic or raster scan refresh display.

تتميز بسهولتها برمجيا بالمقارنه مع التقنيات الاخرى.

7- The level of interactivity is lower than with either a refresh or raster scan display.

مستوى التفاعلية فيها قليل مقارنة بالتقنيات الاخرى.

2-The Calligraphic Refresh graphics display

A Calligraphic (line drawing or vector) refresh CRT display uses a very short persistence phosphor. These displays are frequently called random scan display. Because of the short persistence of the phosphor, the picture painted on the CRT must be repainted or refreshed many times each second. The minimum refresh rate is at least 30 times each second. Refresh rates much lower than 30 times each second result in a flickering image.

The basic calligraphic refresh display requires two elements in addition to the CRT. These are the **display buffer** and the **display controller**. The display buffer is contiguous memory containing all the information required to draw the picture on the CRT. The display controller's function is to repeatedly cycle through this information at the refresh rate. Two factors which limit the complexity (number of vectors displayed) of the picture are the size of the display buffer and the speed of the display controller. A further limitation is the speed at which picture information can be processed.

Features of calligraphic refresh displays

1- It is a vector graphics display
2- Resolution is the same as storage tube display
3-Employee the concept of picture segmentation that support the interactive graphics programs.

توضف مفهوم ال segmentation والذي بدوره يوفر ويدعم التفاعلية للرسوم.

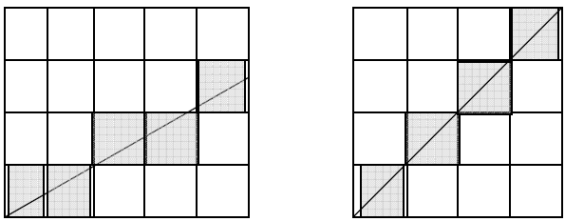
3- Raster refresh graphics display

A raster CRT graphics devices can be considered a matrix of discrete cells each of which can be made bright. Thus it is a point plotting devices.

It is not possible except in special cases to directly draw a straight line from one addressable point, or pixel in the matrix to another addressable point, or pixel. The line can only approximated by a series of dots (pixels) close to the path of the line. Only in the special cases of completely horizontal, vertical or 45 degree lines will a straight line result. All other lines will appear as a series of stair steps. *This is called aliasing*.

تتميز هذه التقنية بعرض الصورة على شكل مصفوفه من الخلايا المنفصلة ,كل خلية تتميز بسطوع معين ,هي الية رسم النقطة .

من غير الممكن رسم خط مباشر من نقطه الى اخرى الا في حالات خاصة ومنها وجود مسار مغلق عمودي او افقي او بدرجة 45 درجه والتي تسمى ب ***aliasing*** وكما مبين ادناه.



The most common method of implementing a raster CRT graphics device utilize a frame buffer. A frame buffer is a large,contiguous piece of computer memory. As a minimum there is one memory bit for each location or pixel in the raster. This amount of memory is called a bit plane.
A 512 X 512 element square raster requires 2^18 memory bits in a single bit plane. The picture is built up in the frame buffer 1 bit at a time.

تتميز هذه التقنية بوجوب توفر ال **frame buffer** والذي يمثل مواقع خزنية متجاوره في ذاكرة الحاسوب. اقل قيمه يمكن تمثيلها لهذه التقنية**(raster)** هي واحد bit لكل pixel والتي تمثل اللون الابيض والاسود.

Resolution

The maximum number of points (pixels) which a line can have is a measure of the resolution of the display device. The greater the number of points, the higher the resolution**. Resolution** is the number of visibly distinct dots that can be displayed in a given area of the screen

Line Drawing algorithms

العنصر الاساسي لرسم اي خط على شاشة الرسم هو النقطة(pixel) والتي تعتبر اصغر وحدة عرض حيث تتكون كل شاشات العرض البياني من عدد كبير من النقاط مرتبة بشكل خطوط افقية وعمودية , الفرق بينهما هو في حجم النقطة ,ففي مهايء الرسم CGA ذو نمط التحليل الواطئ تكون النقاط كبيرة نسبيا وهي 320 *200 اما مهايء الرسم VGA (Very High Resolution Graphic Adapter) ذو قوة التحليل العالية ونقاط صغيرة (640*480) . كلما زاد عدد النقاط كلما زادت الدقة.

لتحديد اي نقطة على الشاشة يتم تحديد قيم المواقع المقابله لها في الذاكرة المؤقتة(frame buffer) حيث ان كل نقطة يتم الوصول اليها بزوج من القيم الصحيحة غير السالبة (X,Y) ويتم اضاءتها بلون مختلف عن لون الخلفية .

لأضاءة نقاط مختلفة على مواقع مختلفة على شاشة الرسم يستخدم الاجراء :

Procedure PutPixel (X,Y: integer, Color:word);

حيث يتم بهذا الاجراء اضاءة النقطة بالموقع (x,y) واللون color بلغة باسكال.

Example:

PutPixel(10,10,yellow);

هنالك عدة امور تحدد جودة اي خوارزمية لرسم الخط وهي :

1 – يجب ان تضهر بصورة مستقيمة.

2 – يجب ان تنتهي الخطوط بصورة دقيقه.

3 – يجب ان تكون لها كثافة ثابته.

4 – سرعة خوارزمية الرسم.

1: straight line may be defined by two endpoints and an equation

If the two endpoints used to specify a line are (X1,Y1) and (X2,Y2) ,then the equation of the line is used to describe the X , Y coordinates of all the pointes that lie between these two endpoints.

The equation of the straight line is :

$$Y = A + BX$$

Where (B) is the slope of the line $B = \frac{\Delta Y}{\Delta X}$

and (A) is the Y intercept.

The Y values of the pointes of the line can be calculated using the above equation, by incrementing X values from X1 to X2 and substitute it in the line equation.

Note : The slope between any point (X,Y) on the line and (X1,Y1) is the same as the slope between (X2,Y2) and (X1,X2).

$$\frac{Y-Y1}{X-X1} = \frac{Y2-Y1}{X2-X1}$$

2 : DDA (Digital Differential Analyzer) algorithm

The DDA algorithm generates lines from their differential equations.

We calculate the length of the line in the X direction (number of pointes) by the equation :

$$ABS\ (X2-X1)$$

and calculate the length of the line in the Y direction (number of pointes) by the Equation :

$$ABS\ (Y2-Y1)$$

Where *ABS* is a function takes the positive of the arguments.

The Length estimates is equal to the larger of the magnitudes of the above two equations.

The increment steps (dX and dY) are used to increment the X and Y coordinates for the next pointes to be plotted

$$dX = \frac{X2 - X1}{\text{Larger Length}} \qquad dY = \frac{Y2 - Y1}{\text{Larger Length}}$$

Algorithm DDA

```
Start
    If ABS( X2-X1) > ABS (Y2-Y1) Then
                    Length=ABS (X2-X1)
                Else
                    Length=ABS (Y2-Y1)

    dX = (X2-X1) / Length

    dY = (Y2-Y1) / Length

    X=X1+ 0.5
    Y=Y1+ 0.5
    For I=1 to Length
        Begin
            Plot(Int(X) , Int (Y))
            X=X+dX
            Y=Y+dY
        End

Finish
```

Example 1 : Consider the line from (0,0) to (5,5) Use DDA to rasterize the line.

Sol 1 :

X1=0 ; Y1=0 ; X2=5 ; Y2=5 ; Length=5

dX=1 ; dY=1 ; X=0.5 ; Y=0.5

I	Plot	X	Y
		0.5	0.5
1	(0,0)	1.5	1.5
2	(1,1)	2.5	2.5
3	(2,2)	3.5	3.5
4	(3,3)	4.5	4.5
5	(4,4)	5.5	5.5

Note : the integer part of X and Y are used in plotting the line.

This would normally have the effect of truncating rather then rounding so we initialize the DDA with the value 0.5 in each of the fractional parts to achieve true rounding. One advantage of this arrangement is that it allows us to detect changes in X and Y and hence to avoid plotting the same point twice.

Features of DDA

1- The algorithm is orientation dependent

2- The end point accuracy deteriorates

3- The algorithm suffer from the fact that it must be performed using floating point arithmetic

تعمل هذه الخوارزمية بشكل جيد عندما يكون $dx >= dy$ ولكنها بطيئة جداً على الحاسوب حيث أنها تتطلب عمليات على أرقام ذات فاصلة عائمة .أما إذا كان $dx < dy$ فإن المستقيم يصبح خشناً جداً .

3 : Bresenham's algorithm

Bresenham algorithm seeks to select the optimum raster locations to represent a straight line. To accomplish this the algorithm always increment by one unit in either X or Y depending on the slop of the line. The slop of the line represent the error between the location of the real line and the location of the drawn line.

The algorithm is cleverly constructed so that only the sign of this error need be examined.

Bresenham algorithm for the first octant : { $0 \leq \Delta Y \leq \Delta X$ }

```
Start

 X=X1
Y=Y1

      DX=X2-X1
      DY=Y2-Y1

      E= (DY / DX) – 0.5

      For I=1 to DX
      Begin
            Plot (X,Y)
            While (E ≥ 0 )
                  Y=Y+1
                  E=E-1
            End While

            X=X+1
            E=E + ( DY / DX )
    End
Finish
```

4 : Integer Bresenham algorithm

Bresenham algorithm requires the use of floating point arithmetic and division to calculate the slop of the line and to evaluate the error term. The speed of the algorithm can be increased by using integer arithmetic and eliminating the division.

Since only the sign of the error term is important , the simple transformation

$$E = E * 2 * \Delta X$$

of the error term in the previous algorithm yields an integer algorithm. This allows the algorithm to be efficiently implemented in hardware.

So : $E = \{ (DY / DX) - 0.5 \} * 2\Delta X \longrightarrow E = 2\Delta Y - \Delta X$

$E = \{ E-1 \} * 2\Delta X \longrightarrow E = E - 2\Delta X$

$E = \{ E + (DY / DX) \} * 2\Delta X \longrightarrow E = E + 2\Delta Y$

Bresenham's integer algorithm for the first octant { $0 \leq \Delta Y \leq \Delta X$ }

i.e. slop between zero and one

Example 3 : Consider the line from (0,0) to (5,5) Rasterize the line with Bresenham algorithm

Sol 3 : X=0 ; Y=0 ; DX=5 ; DY=5 ; E=0.5

I	Plot	E	X	Y
		0.5	0	0
1	(0,0)	-0.5	0	1
		0.5	1	1
2	(1,1)	-0.5	1	2
		0.5	2	2
3	(2,2)	-0.5	2	3
		0.5	3	3
4	(3,3)	-0.5	3	4
		0.5	4	4
5	(4,4)	-0.5	4	5
		0.5	5	5

Start

X=X1

Y=Y1

dX=X2-X1

dY=Y2-Y1

E= 2 ΔY - ΔX

For I=1 to dX

Begin

Plot (X , Y)

While (E ≥ 0)

Begin

Y=Y+1

E= E - 2 ΔX

End While

X=X+1

E = E+ 2 ΔY

Next I ***Finish***

5 : General Bresenham's algorithm

A full implementation of Bresenham algorithm requires modification for lying in the other octanats. These can easily be developed by considering quadrant in which the line lies and its slope. When the absolute magnitude of the slop of the line is > 1 ,Y is incremented by one end Bresenham error is used to determine when to increment X. Whether X or Y incremented by +(-) 1 depends on the quadrant.

General Bresenham's algorithm for all quadrants

```
X=X1
Y=Y1
dX=Abs (X2-X1)
dY=Abs(Y2-Y1)
S1=Sign (X2-X1)
S2=Sign (Y2-Y1)
If dY > dX Then
        Begin
                T=dX : dX=dY : dY=T : Interchange=1
        End
        Else
                Interchange =0
End If
        E= 2 dy - dx
        For I=1 to dX
                Plot (X,Y)
                While ( E ≥ 0)
                        Begin
                                If Interchange=1        Then X=X + S1
                                        Else Y= Y + S2
                                End If
                                E= E - 2 dx
                End While
```

If Interchange=1 Then Y=Y + S2
Else X=X + S1
End If
E = E+ 2 dy
Next I

Finish

Example 4 : Draw the line from (0,0) to (-8,-4) using General Bresenham algorithm
Sol 4 : X=0 ; Y=0 ; dX=8 ; dY=4 ; S1=-1 ; S2=-1 Because dX>dY then Interchange=0 ; E=0

I	Plot	E	X	Y
		0	0	0
1	(0,0)			
		-16	0	-1
		-8	-1	-1
2	(-1,-1)			
		0	-2	-1
3	(-2,-1)			
		-16	-2	-2
		-8	-3	-2
4	(-3,-2)			
		0	-4	-2
5	(-4,-2)			
		-16	-4	-3
		-8	-5	-3
6	(-5,-3)			
		0	-6	-3
7	(-6,-3)			
		-16	-6	-4
		8	-7	-4
8	(-7,-4)			
		0	-8	-4

Circle Drawing algorithms

1- Bresenham algorithm for circle drawing(Mid point algorithm):

To begin with circle drawing, note that only one octant of the circle need to be generated. The other parts can be obtained by successive reflections. To drive Bresenham circle generation algorithm , consider the first quadrant of the origin centered circle. Notice that, if the algorithm begins at X=0 and Y=R (R is the radius) , then for clockwise generation of the circle Y is a monotonically decreasing function of X in the first quadrant. Similarly if the algorithm begins at Y=0 and X=R then for counterclockwise generation of the circle, X is a decreasing function of Y.

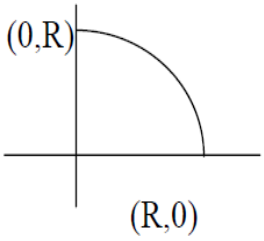
For clockwise generation of the circle there are only three possible selections for the next pixel which best represent the circle:

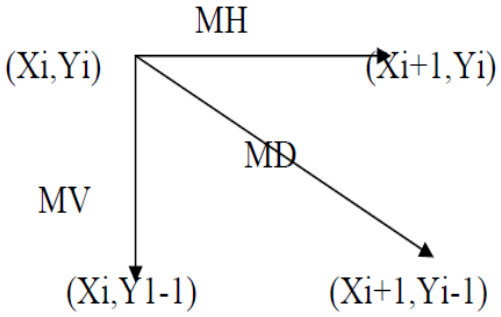
1- Horizontal to the right

2- Diagonally downward to the right

3- Vertically downward

These are labeled MH , MD , MV





The algorithm chose the pixel (the movement)which minimize the square of the distance between one of these pixels and the true circle

The distance in the three cases are measured by :

Case 1 : MH = | (Xi+1)^2 + (Yi)^2 - R^2 |

Case 2 : MD = | (Xi+1)^2 + (Yi-1)^2 - R^2 |

Case 3 : MV = | (Xi)^2 + (Yi-1)^2 - R^2 |

The difference between the square of the distance from the center of the circle to the diagonal pixel at (Xi+1 , Yi-1) and the distance to a point on the circle R^2 is

Di= (Xi+1)^2 + (Yi-1)^2 – R^2

1- If Di<0 then the diagonal point (Xi+1,Yi-1) is *inside* the actual circle i.e. we use case 1 or case 2 .

It is clear that either the pixel at (Xi+1,Yi) == MH or that the pixel at (Xi+1,Yi-1) == MD must be chosen.

ə = | (Xi+1)^2 + (Yi)^2 – R^2| - | (Xi+1)^2 + (Yi-1)^2 – R^2|

if ə <0 then the difference from the actual circle to the diagonal pixel (MD) is greater than that to the horizontal pixel (MH).

If ə <0 choose MH (Xi+1,Yi)

If ə >0 choose MD (Xi+1, Yi-1)

The horizontal move has been selected when ə=0 ; i.e. when the distance are equal.

2- if Di>0 then the diagonal point (Xi+1,Yi-1) is *outside* the actual circle i.e. we use case 2 or case 3 .

β = | (Xi+1)^2 + (Yi-1)^2 – R^2 | - | (Xi)^2 + (Yi-1)^2 + R^2 |

if β <=0 choose MD (Xi+1, Yi-1)

if β >0 choose MV (Xi , Yi-1)

3- if Di=0 then we choose the pixel at (Xi+1 , Yi-1) i.e. MD

Summery

1- Di < 0

ə ≤ 0 , then choose pixel at (Xi+1 , Yi) i.e. MH

ə > 0 , then choose pixel at (Xi+1 , Yi-1) i.e. MD

2- Di > 0

β ≤ 0 , then choose pixel at (Xi+1 , Yi-1) i.e. MD

β > 0 , then choose pixel at (Xi , Yi-1) i.e. MV

3- Di = 0

choose pixel at (Xi+1 , Yi-1) i.e. MD

Bresenham circle algorithm (for the first quadrant)

Xi=0

Yi=R

Di= 2 (1-R)

Limit = 0

1: Plot (Xi , Yi)

If Yi ≤ Limit then goto 4

If Di < 0 then goto 2

If Di > 0 then goto 3

If Di = 0 then goto 20

2: ə = 2Di + 2Yi -1

If ə ≤ 0 then goto 10

If ə > 0 then goto 20

3: β= 2Di – 2X1 -1

If β ≤ 0 then goto 20

If β > 0 then goto 30

10: Xi = Xi + 1 { MH }

Di = Di + 2Xi + 1

Goto 1

20: Xi = Xi +1 { MD }

Yi = Yi - 1

Di = Di + 2Xi – 2Yi + 2

Goto 1

30: Yi = Yi -1 { MV }

Di = Di – 2Yi + 1

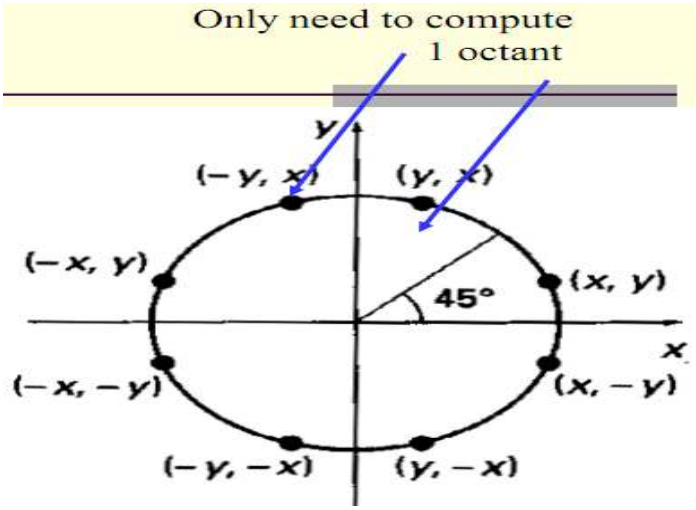
Goto 1

4: Finish

Example :Draw a circle with R=4

Plot (X,Y)	Di	ə	β	Xi	Yi
	-6	-	-	0	4
(0,4)	-3	-5	-	1	4
(1,4)	-3	1	-	2	3
(2,3)	4	-1	-	3	3
(3,3)	1	-	1	3	2
(3,2)	9	-	-5	4	1
(4,1)	10	-	9	4	0
(4,0)	-3				

This algorithm drawled one quadrant ,and if we want to draw another quadrants we should us symmetry property on the circle that shows below .



Thus, at every stage, we can generate 8 points, as shown below

(xc+x,yc+y) , (xc-x,yc+y) , (xc+x,yc-y), (xc-x,yc-y) ,

(xc+y,yc+x) , (xc+y,yc-x) , (xc-y,yc-x) , (xc-y,yc-x)

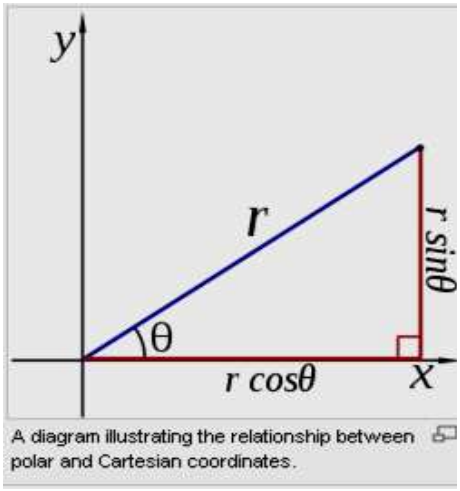
2- Parametric polar representation for circle:

The two polar coordinates *r* and θ can be converted to the Cartesian coordinates *x* and *y* by using the trigonometric functions sine and cosine:

$$X = Xc + R * Cos(ø)$$

$$Y = Yc + R * Sin(ø)$$

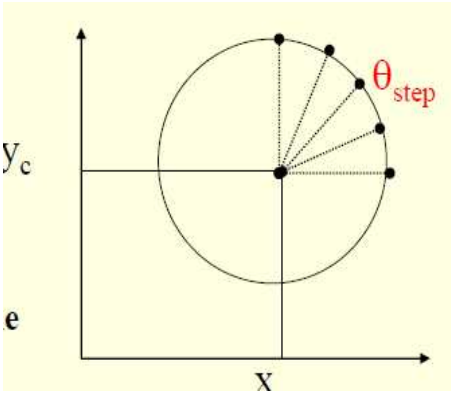
Where ø is measured in radians .



A diagram illustrating the relationship between polar and Cartesian coordinates.

Allows a uniform spacing via the ø variable.

ø step size is often set to be 1/r.



ملاحظة: عند عمل البرنامج لرسم الدائرة يجب تحويل الزاوية من القياس الستيني الى القياس نصف القطري ويتم ذلك من خلال المعادلة التالية .

ø = الزاوية * Π / 180

عندما Π = 3.14285

Question : write procedure to draw circle by using polar representation .

Spiral:

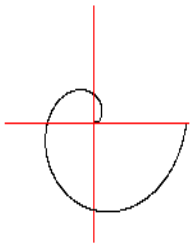
What is a spiral? A spiral is a curve in the plane or in the space, which runs around a centre in a special way.

You can add half circles growing step by

step to get spirals.

Spiral can be plotted by gradually increasing

Radius where plotting a circle .

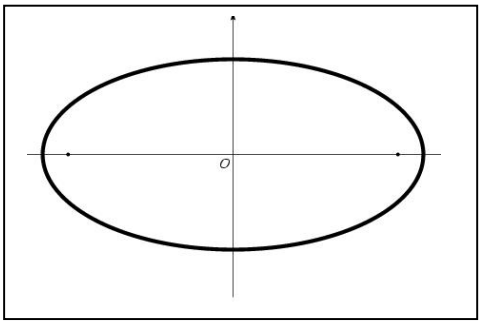


Q: write procedure to draw spiral ?

```
Procedure spiral (xc , yc: integer ; r:real);
 Var
     n,x,y:real;       i:integer;
     Begin
          i:=-1;
          While (r < 200) do
          Begin
              i:=i+1;n:=i*pi/180 ;                R:=R+0.02;
              x:=xc+r*cos(n);                       y:=yc+r*sin(n);
              putpixel(trunc( x),trunc( y),red);        delay(1000);
           End;
      End;
```

Ellipse: An ellipse is a smooth closed curve which is symmetric about its horizontal and vertical axes.

An ellipse can be represented *parametrically* by the equations x = a cos θ and y = b sin θ, where x and y are the rectangular coordinates of any point on the ellipse, and the parameter θ is the angle at the center measured from the x-axis anticlockwise. This is one method of drawing an ellipse, called the *concentric circle*method



x = xc + A* cos θ

y = yc + B* sin θ

Q: write procedure to draw ellipse ?

```
procedure spiral(xc,yc,a,b:integer);
 Var
     n,x,y:real;       i:integer;
     Begin
          i:=-1;
          While(i< 360) do
          Begin
             i:=i+1;n:=i*pi/180 ;
             x:=xc+a*cos(n);                  y:=yc+b*sin(n);
             putpixel(trunc( x),trunc( y),red);                delay(1000);
          end;         end;
```

Chapter 2

TWO-DIMENSIONAL TRANSFORMATION

Introduction

As stated earlier, Computer Aided Design consists of three components, namely, **Design** (Geometric Modeling), **Analysis** (FEA, etc), and **Visualization** (Computer Graphics). Geometric Modeling provides a mathematical description of a geometric object - point, line, conic section, surface, or a solid. Visualization deals with creation of visual effects, e.g., creation of pie charts, contour plots, **shading**, **animation**, etc. Computer graphics provides visual displays and manipulations of objects, e.g., transformation, editing, printing, etc. Fortran and visual C languages are used to effect these operations. Transformation is the backbone of computer graphics, enabling us to manipulate the shape, size, and location of the object. It can be used to effect the following changes in a geometric object:

- Change the location
- Change the Shape
- Change the size
- Rotate
- Copy
- Generate a surface from a line
- Generate a solid from a surface
- Animate the object

Two-Dimensional Transformation

Geometric transformations have numerous applications in geometric modeling, e.g., manipulation of **size**, **shape**, and **location** of an object. To develop an easier understanding of transformations, we will first study the two-dimensional transformations and then extend it to the study of three-dimensional transformations. Until we get to the discussion of surfaces and solids, we will limit our discussion of transformation to only the simple cases of scaling, translation, rotation, and the combinations of these. Applications of transformations will become apparent when we discuss the surface and solid modeling.

There are two types of transformations:

Modeling Transformation: this transformation alters the coordinate values of the object. Basic operations are scaling, translation, rotation and, combination of one or more of these basic transformations. Examples of these transformations can be easily found in any commercial CAD software. For instance, AutoCAD uses SCALE, MOVE, and ROTATE commands for scaling, translation, and rotation transformations, respectively.

Visual Transformation: In this transformation there is no change in either the geometry or the coordinates of the object. A copy of the object is placed at the desired sight, without changing the coordinate values of the object. In AutoCAD, the ZOOM ,and the motion of a car in a scene, we can keep the car fixed while moving the background scenery examples of visual transformation.

Basic Modeling Transformations

There are three basic modeling transformations: **Scaling**, **Translation**, and **Rotation**. Other transformations, which are modification or combination of any of the basic transformations, are **Shearing**, **Mirroring**, **copy**, etc.

Let us look at the procedure for carrying out basic transformations, which are based on matrix operation. A transformation can be expressed as

$$[P^*] = [P]\,[T]$$

where, $[P^*]$ is the new coordinates matrix

[P] is the original coordinates matrix, or points matrix

[T] is the transformation matrix

With the z-terms set to zero, the P matrix can be written as,

$$[P] = \begin{pmatrix} x_1 & y_1 & 0 \\ x_2 & y_2 & 0 \\ x_3 & y_3 & 0 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 0 \end{pmatrix} \qquad (2.1)$$

The size of this matrix depends on the geometry of the object, e.g., a point is defined by a single set of coordinates (x_1, y_1, z_1), a line is defined by two sets of coordinates (x_1, y_1, z_1) and (x_2, y_2, z_2), etc. Thus a point matrix will have the size 1x3, line will be 2x3, etc. A transformation matrix is always written as a 4x4 matrix, with a basic shape shown below,

$$[T] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \textbf{(2.2)}$$

Values of the elements in the matrix will change according to the type of transformation being used, as we will see shortly. The transformation matrix changes the size, position, and orientation of an object, by mathematically adding, or multiplying its coordinate values. We will now discuss the mathematical procedure for translation,scaling, and rotation transformations.

1- Scaling

Scaling is the process of expanding or compressing the dimensions of an object (changing the size of an object) , There are two types of scaling transformations: uniform and non-uniform. In the uniform scaling, the coordinate values change uniformly along the x, y, and z coordinates, where as, in non- uniform scaling, the change is not necessarily the same in all the coordinate directions, In scaling transformation, the original coordinates of an object are multiplied by the given scale factor (SF).

If SF (scale factor) > 1 then the object is enlarged
If SF (scale factor) < 1 then the object is compressed
If SF (scale factor) = 1 then the object is unchanged

1-1 Uniform Scaling

For uniform scaling, the scaling transformation matrix is given as

$$[T] = \begin{pmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Here, s is the scale factor.

1-2 Non-Uniform Scaling

Matrix equation of a non-uniform scaling has the form:

$$[T] = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (2.4)$$

where, s_x, s_x, s_x are the scale factors for the x, y, and z coordinates of the object.

Homogeneous Coordinates

Before proceeding further, we should review the concept of homogeneous coordinate system. Since the points matrix has three columns for the x, y, and z values, and a transformation matrix is always 4x4 matrix, the two matrices are incompatible for multiplication. A matrix multiplication is compatible only if the number of columns in the first matrix equals the number of row in the second matrix. For this reason, a points matrix is written as,

$$[P] = \begin{pmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & z_n & 1 \end{pmatrix}$$

Here, we have converted the Cartesian coordinates into homogeneous coordinates by adding a 4^{th} column, with unit value in all rows. When a fourth column, with values of 1 in each row, is added in the points matrix, the matrix multiplication etween the [P] and [T] becomes compatible. The values (x_1, y_1, z_1, 1) represent the coordinates of the point (x_1, y_1, z_1), and the coordinates are called as homogeneous coordinates. In homogeneous coordinates, the points (2,3,1), (4,6,2), (6,9,3), (8,12,4), represent the same point (2,3,1), along the plane z = 1, z = 2, z = 3, and z = 4, respectively. In our subsequent discussion on transformation, we will use homogeneous coordinates.

Example 1: **If the triangle A(1,1), B(2,1), C(1,3) is scaled by a factor 2, find the new coordinates of the triangle**.

Solution: Writing the points matrix in homogeneous coordinates, we have

$$[P] = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 2 & 1 & 0 & 1 \\ 1 & 3 & 0 & 1 \end{pmatrix}$$

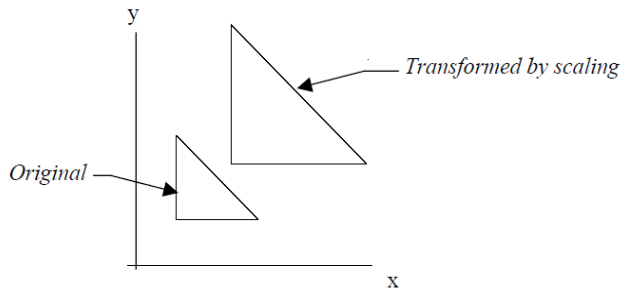
and the scaling transformation matrix is,

$$[T_s] = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The new points matrix can be evaluated by the equation

$[P^*] = [P]\,[T]$, and by substitution of the P and T values, we get

$$P^* = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 2 & 1 & 0 & 1 \\ 1 & 3 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 2 & 0 & 1 \\ 4 & 2 & 0 & 1 \\ 2 & 6 & 0 & 1 \end{pmatrix}$$



Note that the new coordinates represent the original value times the scale factor. The old and the new positions of the triangle are shown in the figure.

Notice that after a scaling transformation is performed, the new object is located at a different position relative to the origin. In fact, in scaling transformation the only point that remains fixed is the origin.

If we want to let one point of an object that remains at the same location (fixed), scaling can be performed by three steps:

1- Translate the fixed point to the origin, and all the points of the object must be moved the same distance and direction that the fixed point moves.

2- Scale the translated object from step one

3- Back translate the scaled object to its original position

Example 2: Scale the rectangle (12,4),(20,4),(12,8),(20,8) with SX=2,SY=2
so the point (12,4) being the fixed point.

Solution:
1- Translate the object with TX= -12 and TY= -4 so the point (12,4) lies on the origin

(12,4) ===> (0,0)
(20,4) ===> (8,0)
(12,8) ===> (0,4)
(20,8) ===> (8,4)

2- Scale the object by SX=2 and SY=2

(0,0) ===> (0 , 0)
(8,0) ===> (16 , 0)
(0,4) ===> (0 , 8)
(8,4) ===> (16 , 8)

3- Back translate the scaled object with TX= 12 and TY= 4

(0 , 0) ===> (12 , 4)
(16 , 0) ===> (28 , 4)
(0 , 8) ===> (12 ,12)
(16,8) ===> (28 ,12)

2- Translation Transformation

In translation, every point on an object translates exactly the same distance. The effect of a translation transformation is that the original coordinate values increase or decrease by the amount of the translation along the x, y, and z-axes. For example, if line A(2,4), B(5,6) is translated 2 units along the positive x axis and 3 units along the positive y axis, then the new coordinates of the line would be

A'(2+2, 4+3), B'(5+2, 6+3) or
A'(4,7), B'(7,9).

The transformation matrix has the form:

$$[T_t] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x & y & 0 & 1 \end{pmatrix} \qquad (2.3)$$

where, x and y are the values of translation in the x and y direction, respectively. For translation transformation, the matrix equation is

$$[P^*] = [P]\,[T_t] \qquad (2.4)$$

where, $[T_t]$ is the translation transformation matrix.

Example 2: Translate the rectangle (2,2), (2,8), (10,8), (10,2) 2 units along x-axis and 3 units along y-axis.

Solution: Using the matrix equation for translation, we have

$[P^*] = [P]\,[T_t]$, substituting the numbers, we get

$$[P^*] = \begin{pmatrix} 2 & 2 & 0 & 1 \\ 2 & 8 & 0 & 1 \\ 10 & 8 & 0 & 1 \\ 10 & 2 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 2 & 3 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 4 & 5 & 0 & 1 \\ 4 & 11 & 0 & 1 \\ 12 & 11 & 0 & 1 \\ 12 & 5 & 0 & 1 \end{pmatrix}$$

Note that the resultant coordinates are equal to the original x and y values plus the 2 and 3 units added to these values, respectively.

3- Rotation

We will first consider rotation about the z-axis, which passes through the origin (0,0,0), since it is the simplest transformation for understanding the rotation transformation. Rotation about an arbitrary axis, other than an axis passing through the origin, requires a combination of three or more transformations, as we will see later.

When an object is rotated about the z-axis, all the points on the object rotate in a circular arc, and the center of the arc lies at the origin. Similarly, rotation of an object about an arbitrary axis has the same relationship with the axis, i.e., all the points on the object rotate in a circular arc, and the center of rotation lies at the given point through which the axis is passing.

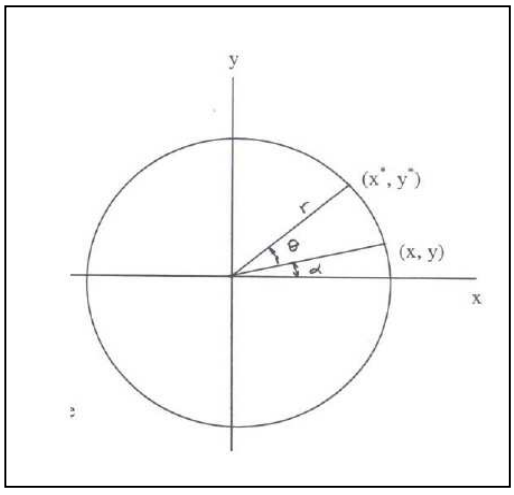
In rotation, the object is rotated Ǿ about the origin. The convention is that the direction of rotation is counterclockwise if Ǿ is a positive angle and clockwise if Ǿ is a negative angle.

Derivation of the Rotation Transformation Matrix (about the origin)

Using trigonometric relations, as given below, we can derive the rotation transformation matrix. Let the point P(x, y) be on the circle, located at an angle α, as shown. If the point P is rotated an additional angle θ, the new point will have the coordinates (x*, y*). The angle and the original coordinate relationship is found as follows.

$$\left.\begin{aligned} x &= r\cos\alpha \\ y &= r\sin\alpha \end{aligned}\right\}$$ Original coordinates of point P.

$$\left.\begin{aligned} x^* &= r\cos(\alpha+\theta) \\ y^* &= r\sin(\alpha+\theta) \end{aligned}\right\}$$ The new coordinates.



where, α is the angle

between the line joining the initial position of the point and the x-axis, and θ is the angle between the original and the new position of the point.

Using the trigonometric relations, we get,

$x^* = r(\cos\alpha \cos\theta - \sin\alpha \sin\theta) = x\cos\theta - y\sin\theta$
$y^* = r(\cos\alpha \sin\theta + \sin\alpha \cos\theta) = x\sin\theta + y\cos\theta$

In matrix form we can write these equations as

$$[x^* \; y^*] = [x \; y] \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \qquad (2.8)$$

In general, the points matrix and the transformation matrix given in equation (2.8) are re-written as:

$$[x^* \quad y^* \quad 0 \quad 1] = [x \quad y \quad 0 \quad 1] \begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (2.9)$$

Thus, a point or any object can be rotated about the z-axis (in 2-D) and the new coordinates of the object found by the product of the points matrix and the rotation matrix, derived here.

Rotation of an Object about an Arbitrary Axis (about a specific point)

Rotation of a geometric model about an arbitrary axis, other than any of the coordinate axes, involves several rotational and translation transformations. When we rotate an object about the origin (in 2-D), we in fact rotate it about the z-axis. Every point on the object rotates along a circular path, with the center of rotation at the origin. If we wish to rotate an object about an arbitrary axis, which is

perpendicular to the xy-plane, we will have to first translate the axis to the origin and then rotate the model, and finally, translate so that the axis of rotation is restored to its initial position. If we erroneously use the equation (2.9) directly, to rotate the object about a fixed axis, and skip the translation of this point to the origin, we will in fact end up rotating the object about the z-axis, and not about the fixed axis.

Thus, the rotation of an object about an arbitrary axis, involves three steps:

Step 1: Translate the fixed axis so that it coincides with the z-axis

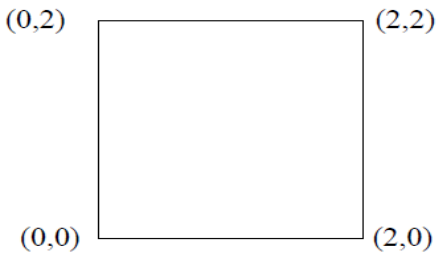
Step 2: Rotate the object about the axis

Step 3: Translate the fixed axis back to the original position.

Note: When the fixed axis is translated, the object is also translated. The axis and the object go through all the transformations simultaneously.

We will now illustrate the above procedure by the following example.

Example 3**: Rotate the rectangle (0,0), (2,0), (2, 2), (0, 2) shown below, 30^0** about its centroid and find the new coordinates of the rectangle.



Solution: Centric of the rectangle is at point (1, 1). We will first translate the centered to the origin, then rotate the rectangle, and finally, translate the rectangle so that the centered is restored to its original position.

1. Translate the centered to the origin: The matrix equation for this step is

$$[P^*]_1 = [P]\,[T_t], \text{ where } [P] = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 1 \\ 2 & 2 & 0 & 1 \\ 0 & 2 & 0 & 1 \end{pmatrix}$$

$$\text{and} \qquad [T_t] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1 & -1 & 0 & 1 \end{pmatrix}$$

2. **Rotate the Rectangle 30^0 About the z-axis:** The matrix equation for this step is given as

$[P^*]_2 = [P^*]_1\,[T_r]$, where, $[P^*]_1$ is the resultant points matrix obtained in step 1, and $[T_r]$ is the rotation transformation, where $\theta = 30^0$. The transformation matrix is,

$$[T_r]_\theta = \begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} .866 & .5 & 0 & 0 \\ -.5 & .866 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3. **Translate the Rectangle so that the Centerior Lies at its Original Position:** The matrix equation for this step is

$[P^*]_3 = [P^*]_2 [T_{-t}]$, where $[T_{-t}]$ is the reverse translation matrix, given as

$$[T_{-t}] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

Now we can write the entire matrix equation that combines all the three steps outlined above. The equation is,

$[P^*] = [P] [T_t] [T_r] [T_{-t}]$

Substituting the values given earlier, we get,

$$[P^*] = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 1 \\ 2 & 2 & 0 & 1 \\ 0 & 2 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1 & -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos 30^0 & \sin 30^0 & 0 & 0 \\ -\sin 30^0 & \cos 30^0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} x$$

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

$$= \begin{pmatrix} 0.6340 & -0.3660 & 0 & 1 \\ -0.3660 & 1.3660 & 0 & 1 \\ 1.3660 & 2.3660 & 0 & 1 \\ -0.3660 & 1.3660 & 0 & 1 \end{pmatrix}$$

The first two columns represent the new coordinates of the rotated rectangle.

Note:: Rotation in clockwise direction :In order to rotate in clockwise direction, the matrix will be

$$[x^* \quad y^* \quad 0 \quad 1] = [x \quad y \quad 0 \quad 1] \begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (2.9)$$

Combined Transformations

Most applications require the use of more than one basic transformation to achieve desired results. As stated earlier, scaling with an arbitrarily fixed point involves both scaling and translation. And rotation around a given point, other than the origin, involves rotation and translation. We will now consider these combined transformations.

Scaling With an Arbitrary Point

In uniform scaling, all points and their coordinates are scaled by a factor s. Therefore, unless the fixed point is located at (0, 0), it will be moved to a new location with coordinates s-times x and s-times y. To scale an object about a fixed point, the fixed point is first moved to the origin and then the object is scaled. Finally, the object is translated or moved so that the fixed point is restored to its original position. The transformation sequence is,

$$[P^*] = [P]\,[T_t]\,[T_s]\,[T_{-t}]$$

Where, $[T_t]$ is the translation transformation matrix, for translation of the fixed point to the origin,

$[T_s]$ is the scaling transformation matrix, and

$[T_{-t}]$ is the reverse translation matrix, to restore the fixed point to its original position.

Note: The order of matrix multiplication progresses from left to right and the order should not be changed.

The three transformation matrices $[T_t]$ $[T_s]$ $[T_{-t}]$ can be concatenated to produce a single transformation matrix, which uniformly scales an object while keeping the pivot point fixed. Thus, the resultant, concatenated transformation matrix for scaling is,

$$[T_s]_R = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x & -y & 0 & 1 \end{pmatrix} \begin{pmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x & y & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ x\text{-}sx & y\text{-}sy & 0 & 1 \end{pmatrix} \qquad (2.10)$$

The concatenated equation can be used directly instead of the step-by-step matrix solution. This form is preferable when writing a CAD program.

Example : Given the triangle, described by the homogeneous points matrix below, scale it by a factor 3/4, keeping the centroid in the same location. Use (a) separate matrix operation and (b) condensed matrix for transformation.

$$[P] = \begin{pmatrix} 2 & 2 & 0 & 1 \\ 2 & 5 & 0 & 1 \\ 5 & 5 & 0 & 1 \end{pmatrix}$$

Solution

(a) The centroid of the triangle is at,

x = (2+2+5)/3 = 3, and y = (2+5+5)/3 = 4 or the centroid is C(3,4).

We will first translate the centroid to the origin, then scale the triangle, and finally translate it back to the centroid. Translation of triangle to the origin will give,

$$[P^*]_1 = [P]\,[T_t] = \begin{pmatrix} 2 & 2 & 0 & 1 \\ 2 & 5 & 0 & 1 \\ 5 & 5 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & -4 & 0 & 1 \end{pmatrix} = \begin{pmatrix} -1 & -2 & 0 & 1 \\ -1 & 1 & 0 & 1 \\ 2 & 1 & 0 & 1 \end{pmatrix}$$

Scaling the triangle, we get,

$$[P^*]_2 = [P^*]_1\,[T_s] = \begin{pmatrix} -1 & -2 & 0 & 1 \\ -1 & 1 & 0 & 1 \\ 2 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} .75 & 0 & 0 & 0 \\ 0 & .75 & 0 & 0 \\ 0 & 0 & .75 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} -0.75 & -1.5 & 0 & 1 \\ -0.75 & 0.75 & 0 & 1 \\ 1.5 & 0.75 & 0 & 1 \end{pmatrix}$$

Translating the triangle so that the centroid is positioned at (3, 4), we get

$$[P^*] = [P^*]_2\,[T_{-t}] = \begin{pmatrix} -.75 & -1.5 & 0 & 1 \\ -.75 & .75 & 0 & 1 \\ 1.5 & .75 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 3 & 4 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2.25 & 2.5 & 0 & 1 \\ 2.25 & 4.75 & 0 & 1 \\ 4.5 & 4.75 & 0 & 1 \end{pmatrix}$$

(b) The foregoing set of three operations can be reduced to a single operation using the condensed matrix with x = 3, and y = 4. See equation (2.10) on page 16.

$$[P^*] = [P]\,[T_{cond}] = \begin{pmatrix} 2 & 2 & 0 & 1 \\ 2 & 5 & 0 & 1 \\ 5 & 5 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0.75 & 0 & 0 & 0 \\ 0 & 0.75 & 0 & 0 \\ 0 & 0 & 0.75 & 0 \\ 3-0.75(3) & 4-0.75(4) & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 2.25 & 2.5 & 0 & 1 \\ 2.25 & 4.75 & 0 & 1 \\ 4.5 & 4.75 & 0 & 1 \end{pmatrix}$$

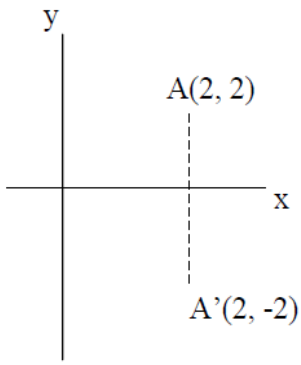
Mirroring (Reflection):

In modeling operations, one frequently used operation is mirroring an object. Mirroring is a convenient method used for copying an object while preserving its features. The mirror transformation is a special case of a negative scaling, as will be explained below.

Let us say, we want to mirror the point A(2,2) about the x-axis, as shown in the figure.The new location of the point, when reflected about the x-axis, will be at (2, -2). The point matrix [P*] = [2 -2] can be obtained with the matrix transformation given below.

$$[P^*] = [2 \quad 2 \quad 0 \quad 1] \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= [2 \quad -2 \quad 0 \quad 1]$$



The transformation matrix above is a special case of a non-uniform scaling

with s_x =1 and s_y = -1. We can extend this concept to mirroring around the y, z, and any arbitrary axis, as will be explained in the following discussion.

4-1 Mirroring About an Arbitrary Plane

If mirroring is required about an arbitrary plane, other than one defined by the coordinate axes, translation and/or rotation can be used to align the given plane with one of the coordinate planes. After mirroring, translation or rotation must be done in reverse order to restore the original geometry of the axis.

We will use the figure shown below, to illustrate the procedure for mirroring an object about an arbitrary plane. We will mirror the given rectangle about a plane passing through the line AB and perpendicular to the xy-plane. It should be noted that in each of the transformations, the plane and the rectangle have a fixed relationship, i.e., when we move the plane (or line AB, the rectangle also moves with it. A step-by-step procedure for mirroring the rectangle about the plane follows.

Note: We are using line AB to represent the plane, which passes through it. Mirroring can be done only about a plane, and not about a line.

Step 1: Translate the line AB (i.e., the plane) such that it passes through the origin, as shown by the dashed line.