UNIT-1
Computer Graphics

Introduction:

It is difficult to display an image of any size on the computer screen. This method is simplified by using Computer graphics. Graphics on the computer are produced by using various algorithms and techniques.

Computer Graphics involves technology to access. The Process transforms and presents information in a visual form. The role of computer graphics is insensible. In today life, computer graphics has now become a common element in user interfaces, T.V. commercial motion pictures.

Computer Graphics is the creation of pictures with the help of a computer. The end product of the computer graphics is a picture it may be a business graph, drawing, and engineering.

In computer graphics, two or three-dimensional pictures can be created that are used for research. Many hardware devices algorithm has been developing for improving the speed of picture generation with the passes of time. It includes the creation storage of models and image of objects. These models for various fields like engineering, mathematical and so on.

"It is the use of computers to create and manipulate pictures on a display device. It comprises of software techniques to create, store, modify, represents pictures."

Advantages of Computer Graphics:

The advantages of computer graphics are as follows:

1. It provides tools for producing pictures not only of concrete real world objects but also of abstract, synthetic objects.
2. It has the ability to show moving pictures and thus it is possible to produce animations with computer graphics.
3. With computer graphics user can also control the animation speed, portion of the view, the goniometric relationship the object in the scene to one another, the amount of details shown and on.
4. The computer graphics provides tool called Motion dynamics. With this tools user can move and tumble objects with respects to a stationary observer. or he can make objects stationary and the viewer moving around them. A typical example is walk through made by builder to show flat interior and building surroundings. In many case it is also possible to move both objects and viewer.
5. The computer graphics also provides facility called update dynamics. With up date dynamics it is possible to change the shape, color or other properties of the objects being viewed.
6. With the recent development of digital signal processing (DSP) and audio synthesis chip the interactive graphics can now provide audio feedback along with the graphical feedbacks to make the simulated environment even more realistic.

Disadvantages of Computer Graphics:

Computer Graphics comes with some demerits of complexity, time-consuming, cost, expertise, time to time repair, and updates.

1. **Complexity:** The graphical system demand of users for getting advanced tools is increasing. It is making graphical applications heavy and complex for a beginner user. Users have to participate in graphical application training. Some of the software is so complex that they require an expert to install the system.
2. **Expensive:** Modern computer graphics software and tools are very expensive for an individual user. To learn those expensive tools, they also have to participate in training which increases the cost.
3. **Limitations:** Computer Graphics can be interactive or non-interactive but they cannot have the intelligence to understand real-world situations. It cannot run on the basis of real-life principles. Users have to learn to use graphical software to achieve the objective of designing.
4. **Cost Ineffective:** It is true that every software needs to be updated and changed on a regular basis. This requires the user to keep on updating the software being used. This software comes with limited time purchase that means users have to renew the graphical product again and again when their subscription is finished.
5. **Increase Heat and Weight of the System:** Graphics is the processing and memory unit of the computer which consumes high energy and makes the computer costlier. We can simply compare a laptop with V-RAM and without V-RAM on the basis of price and specs. A computer with video graphics is around 20k more expensive than a normal one. CPU (an AMD Ryzen 9 3900x) can consume around 140W at stock while an Nvidia GeForce GT 1030 will take up to 30W within minimum specs.

   We can also compare GPU (RTX 2080 ti) which can take up to 373W with an Intel Core i7 8559U which ships a meager 28W. But a normal laptop without V-RAM only consumes 20-50 Watts of energy. High Specs graphics computers are provided with a supercooling system so that to protect any hardware damage due to heat. The maximum temperature exerted by GPU is 95°C-105°C, at this temperature the system turns off automatically to reduce the damage to the hardware.

Applications of Computer Graphics:

Applications of Computer Graphics are as follows:

1. **Education and Training:** Computer-generated model of the physical, financial and economic system is often used as educational aids. Model of physical systems, physiological system, population trends or equipment can help trainees to understand the operation of the system.

   For some training applications, particular systems are designed. For example Flight Simulator.

   Flight Simulator: It helps in giving training to the pilots of airplanes. These pilots spend much of their training not in a real aircraft but on the ground at the controls of a Flight Simulator.

   Advantages:

   - Fuel Saving

- Safety
- Ability to familiarize the training with a large number of the world's airports.

2. **Use in Biology:** Molecular biologist can display a picture of molecules and gain insight into their structure with the help of computer graphics.
3. **Computer-Generated Maps:** Town planners and transportation engineers can use computer-generated maps which display data useful to them in their planning work.
4. **Architect:** Architect can explore an alternative solution to design problems at an interactive graphics terminal. In this way, they can test many more solutions that would not be possible without the computer.
5. **Presentation Graphics:** Example of presentation Graphics are bar charts, line graphs, pie charts and other displays showing relationships between multiple parameters. Presentation Graphics is commonly used to summarize
    - Financial Reports
    - Statistical Reports
    - Mathematical Reports
    - Scientific Reports
    - Economic Data for research reports
    - Managerial Reports
    - Consumer Information Bulletins
    - And other types of reports
6. **Computer Art:** Computer Graphics are also used in the field of commercial arts. It is used to generate television and advertising commercial.
7. **Entertainment:** Computer Graphics are now commonly used in making motion pictures, music videos and television shows.
8. **Visualization:** It is used for visualization of scientists, engineers, medical personnel, business analysts for the study of a large amount of information.
9. **Educational Software:** Computer Graphics is used in the development of educational software for making computer-aided instruction.
10. **Printing Technology:** Computer Graphics is used for printing technology and textile design.

Example of Computer Graphics Packages:

1. LOGO
2. COREL DRAW
3. AUTO CAD
4. 3D STUDIO
5. CORE

6. GKS (Graphics Kernel System)
7. PHIGS (Programmer's Hierarchical Interactive Graphics Standard)
8. CAM (Computer Graphics Metafile)
9. CGI (Computer Graphics Interface)

Interactive and Passive Graphics:

1. **Non-interactive or Passive Computer Graphics:**

   In non-interactive computer graphics, the picture is produced on the monitor, and the user does not have any controlled over the image, i.e., the user cannot make any change in the rendered image. One example of its Titles shown on T.V.

   Non-interactive Graphics involves only one-way communication between the computer and the user, User can see the produced image, and he cannot make any change in the image.

2. **Interactive Computer Graphics:**

   In interactive Computer Graphics user have some controls over the picture, i.e., the user can make any change in the produced image. One example of it is the ping-pong game.

   Interactive Computer Graphics require two-way communication between the computer and the user. A User can see the image and make any change by sending his command with an input device.

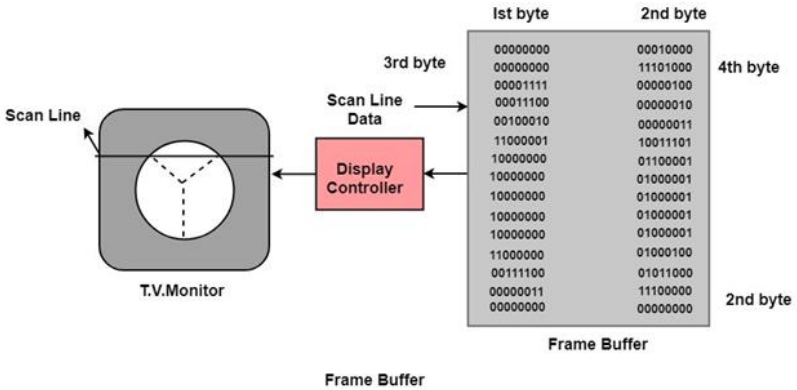
   Advantages of Interactive Computer Graphics:

   1. Higher Quality
   2. More precise results or products
   3. Greater Productivity
   4. Lower analysis and design cost
   5. Significantly enhances our ability to understand data and to perceive trends.

Working of Interactive Computer Graphics:

The modern graphics display is very simple in construction. It consists of three components:

1. Frame Buffer or Digital Memory
2. A Monitor likes a home T.V. set without the tuning and receiving electronics.
3. **Display Controller or Video Controller:** It passes the contents of the frame buffer to the monitor.



Frame Buffer

Frame Buffer: A digital frame buffer is large, contiguous piece of computer memory used to hold or map the image displayed on the screen.

- At a minimum, there is 1 memory bit for each pixel in the raster. This amount of memory is called a bit plane.
- A 1024 x 1024 element requires 2^{20} (2^{10}=1024;2^{20}=1024 x 1024)sq.raster or 1,048,576 memory bits in a single bit plane.
- The picture is built up in the frame buffer one bit at a time.
- ∵ A memory bit has only two states (binary 0 or 1), a single bit plane yields a black and white (monochrome display).
- As frame buffer is a digital device write raster CRT is an analog device.

Properties of Video Monitor:

1. **Persistence:** Persistence is the duration of phosphorescence. Different kinds of phosphors are available for use in CRT. Besides color, a major difference between phosphor in their persistence how they continue to emit light after the electron beam is removed.
2. **Resolution:** Use to describe the number of pixels that are used on display image.
3. **Aspect Ratio:** It is the ratio of width to its height. Its measure is unit in length or number of pixels.

$$\text{Aspect Ratio} = \frac{WidthUnit}{HeightUnit}$$

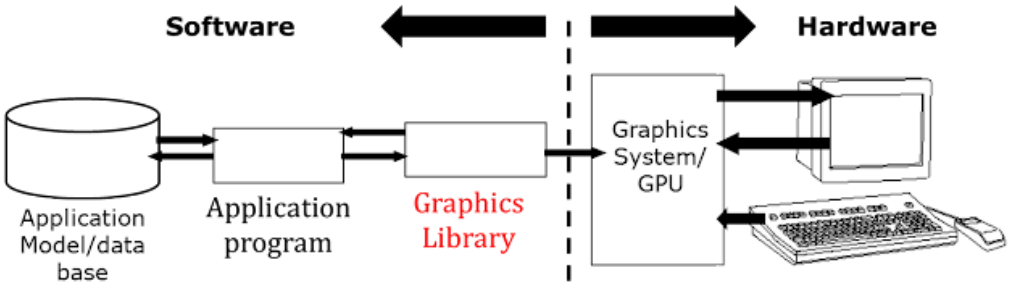
Representative Uses of Computer Graphics :-

- **User interfaces**: GUI, etc.
- **Business, science and technology**: histograms, bar and pie charts, etc.
- **Office automation and electronic publishing**: text, tables, graphs, hypermedia systems, etc.
- **Computer-aided design (CAD)**: structures of building, automobile bodies, etc.
- **Simulation and animation for scientific visualization and entertainment**: flight simulation, games, movies, virtual reality, etc.
- **Art and commerce**: terminals in public places such as museums, etc.
- **Cartography**: map making.

Conceptual Framework for Interactive Graphics

- **Graphics library/package** : is intermediary between application and display hardware (Graphics System)
- **Application program:** maps application objects to views (images) of those objects by calling on graphics library. Application model may contain lots of non-graphical data (e.g., non-geometric object properties)
- **User interaction:** results in modification of model and/or image

This hardware and software framework is more than 4 decades old but is still useful.



Classification of Hardware and Software for Computer Graphics:

Hardware:

1. **Mouse-** Used to navigate around digital graphics software. A mouse is also used to navigate around a computer, and to launch programs/applications.
2. **Monitor/Screen-** Displays a picture/image that you may be editing using digital graphics software.
3. **RAM (Random Access Memory –** Makes your computer faster. The more RAM you have the quicker your PC will be. Therefore when using digital graphics software, the software is more likely to run smoother and quicker.
4. **CPU (Central Processing Unit)-** Executes commands that are given to it. For example in digital graphics, if you type in some code, your CPU then process's that and then executes the command.
5. **Hard Disk Drive (HDD) & Solid State Drive (SSD)-** Both SSD's and HDD's store data or memory. However HDD's are slower than SSD's. It's also worth noting SSD's cann**ot** defragged, as they defrag automatically. If you try to format a SSD, it will probably not work. Either a SSD or HDD can be used in digital graphics, as they both store data/memory (or your work).
6. **Digital Camera-** Used to take pictures or videos. These videos/pictures can then be used in digital graphics once they have been uploaded to your computers HDD/SDD.
7. **Printer-** Used to print out pictures or documents. This can be used in digital graphics, to print out your work.
8. **Scanner-** Used to scan photos or documents into a computer. You can then use these pictures in digital graphics software.
9. **USB (Universal Serial Bus)-** Used to transfer data/memory around portably. This can be used in digital graphics to transfer your finished pictures or videos around.
10. **Graphics Tablet-** Graphic tablets used to navigate around graphics software or to enhance, manipulate or create digital images.
11. **CD Rom-** Used in reading discs. You put the disc in the CD rom drive and it reads it, and whatever is on there, whether it be photos or a program it will open or install.

Software:

1. **Adobe Photoshop (Raster Graphics)-** Used to enhance and manipulate photos and to create original digital art.
2. **Adobe Illustrator (Vector Graphics)-** A program that is used by both artists and graphic designers to create vector images. These images will then be used for company logos, promotional material both in print and digital form**.**

Scan Conversion:

It is a process of representing graphics objects a collection of pixels. The graphics objects are continuous. The pixels used are discrete. Each pixel can have either on or off state.

The circuitry of the video display device of the computer is capable of converting binary values (0, 1) into a pixel on and pixel off information. 0 is represented by pixel off. 1 is represented using pixel on. Using this ability graphics computer represents picture having discrete dots.

Any model of graphics can be reproduced with a dense matrix of dots or points. Most human beings think graphics objects as points, lines, circles, ellipses.

Advantage of developing algorithms for scan conversion

1. Algorithms can generate graphics objects at a faster rate.
2. Using algorithms memory can be used efficiently.
3. Algorithms can develop a higher level of graphical objects.

Examples of objects which can be scan converted

1. Point
2. Line
3. Sector
4. Arc
5. Ellipse
6. Rectangle
7. Polygon
8. Characters
9. Filled Regions

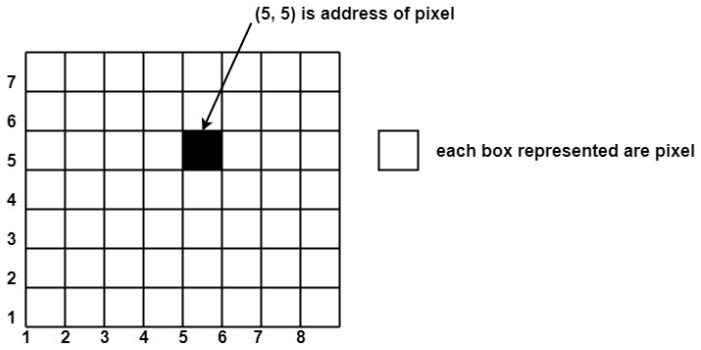
The process of converting is also called as rasterization. The algorithms implementation varies from one computer system to another computer system. Some algorithms are implemented using the software. Some are performed using hardware or firmware. Some are performed using various combinations of hardware, firmware, and software.

Pixel or Pel:

The term pixel is a short form of the picture element. It is also called a point or dot. It is the smallest picture unit accepted by display devices. A picture is constructed from hundreds of such pixels. Pixels are generated using commands. Lines, circle, arcs, characters; curves are drawn with closely spaced pixels. To display the digit or letter matrix of pixels is used.

The closer the dots or pixels are, the better will be the quality of picture. Closer the dots are, crisper will be the picture. Picture will not appear jagged and unclear if pixels are closely spaced. So the quality of the picture is directly proportional to the density of pixels on the screen.

Pixels are also defined as the smallest addressable unit or element of the screen. Each pixel can be assigned an address as shown in fig:



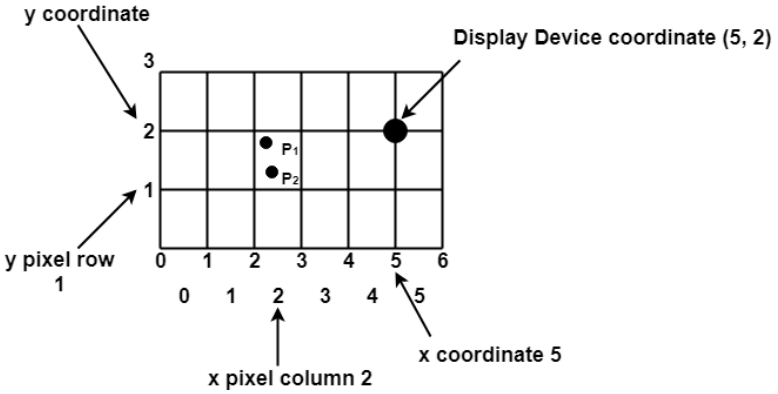
Different graphics objects can be generated by setting the different intensity of pixels and different colors of pixels. Each pixel has some co-ordinate value. The coordinate is represented using row and column.

P (5, 5) used to represent a pixel in the 5th row and the 5th column. Each pixel has some intensity value which is represented in memory of computer called a **frame buffer**. Frame Buffer is also called a refresh buffer. This memory is a storage area for storing pixels values using which pictures are displayed. It is also called as digital memory. Inside the buffer, image is stored as a pattern of binary digits either 0 or 1. So there is an array of 0 or 1 used to represent the picture. In black and white monitors, black pixels are represented using 1's and white pixels are represented using 0's. In case of systems having one bit per pixel frame buffer is called a bitmap. In systems with multiple bits per pixel it is called a pixmap.

Scan Converting a Point:

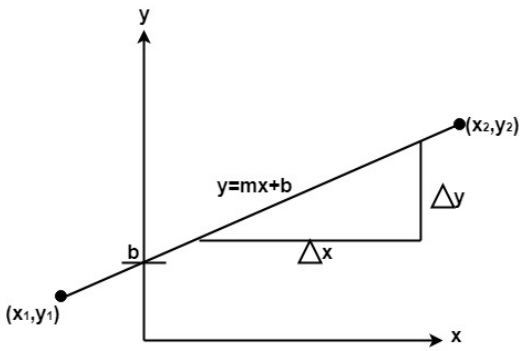
Each pixel on the graphics display does not represent a mathematical point. Instead, it means a region which theoretically can contain an infinite number of points. Scan-Converting a point involves illuminating the pixel that contains the point.

Example: Display coordinates points $P_1(2\frac{1}{4}, 1\frac{3}{4})$ and P_2 $(2\frac{2}{3}, 1\frac{1}{4})$ as shown in fig would both be represented by pixel (2, 1). In general, a point p (x, y) is represented by the integer part of x & the integer part of y that is pixels [(INT (x), INT (y).



Scan Converting a Straight Line:

A straight line may be defined by two endpoints & an equation. In fig the two endpoints are described by (x_1,y_1) and (x_2,y_2). The equation of the line is used to determine the x, y coordinates of all the points that lie between these two endpoints.



Using the equation of a straight line, y = mx + b where m = $\frac{\Delta y}{\Delta x}$ & b = the y interrupt, we can find values of y by incrementing x from x =x_1, to x = x_2. By scan-converting these calculated x, y values, we represent the line as a sequence of pixels.

Properties of Good Line Drawing Algorithm:

1. **Line should appear straight:** We must appropriate the line by choosing addressable points close to it. If we choose well, the line will appear straight, if not, we shall produce crossed lines.

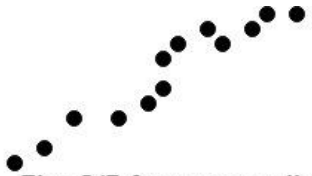


Fig: O/P from a poor line generating algorithm

The lines must be generated parallel or at 45° to the x and y-axes. Other lines cause a problem: a line segment through it starts and finishes at addressable points, may happen to pass through no another addressable points in between.

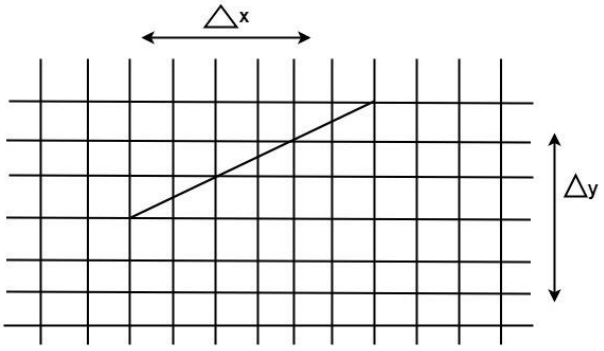


Fig: A straight line segment connecting 2 grid intersection may fail to pass through any other grid intersections.

2. **Line should terminate accurately:** Unless lines are plotted accurately, they may terminate at the wrong place.

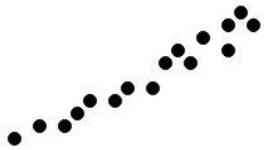


Fig: Uneven line density caused by bunching of dots.

3. **Line should have constant density:** Line density is proportional to the no. of dots displayed divided by the length of the line. To maintain constant density, dots should be equally spaced.
4. **Line density should be independent of line length and angle:** This can be done by computing an approximating line-length estimate and to use a line-generation algorithm that keeps line density constant to within the accuracy of this estimate.
5. **Line should be drawn rapidly:** This computation should be performed by special-purpose hardware.

Algorithm for Line Drawing:

- Most Line Drawing algorithms use incremental method to draw a line.
- In these methods line starts with starting point, then a fix increment is added to the current point to get the next point on line.
- It is continued till the end of line.

Incremental Algorithm:

1. CurrPosition = Start
   Step = Increment
2. If (|CurrPosition-End| < Accuracy) then go to step 5
   (It checks whether the current position is reached upto approximate end point if yes, line drawing is completed).
   If (CurrPosition < End) goto step 3
   (Start < End)
   If (CurrPosition > End) goto step 4
   (Start > End)
3. CurrPosition = CurrPosition + Step
   Goto Step 2
4. CurrPosition = CurrPosition - Step
   Goto Step 2
5. Stop

Algorithm:

1. Direct use of Line equation
2. Digital Differential Analyzer (DDA)
3. Bresenham's Line Algorithm

1. **Direct use of Line equation:**

It is the simplest form of conversion. First of all scan P_1 and P_2 points. P_1 has co-ordinates (x_1, y_1) and P_2 has (x_2, y_2).

Then $m = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$ and $b = y_1 + mx_1$

If the value of $|m| \leq 1$ for each integer value of x, but do not consider x_1 and x_2

If value of |m|>1 for each integer value of y. But do not consider y_1 and y_2

Algorithm for drawing line using equation:

Step1: Start Algorithm

Step2: Declare variables x_1,x_2,y_1,y_2,dx,dy,m,b

Step3: Enter values of x_1,x_2,y_1,y_2.

The (x_1,y_1) are co-ordinates of a starting point of the line.

The (x_2,y_2) are co-ordinates of a ending point of the line.

Step4: Calculate dx = x_2- x_1

Step5: Calculate dy = y_2-y_1

Step6: Calculate m = $\frac{dy}{dx}$

Step7: Calculate b = y_1-m* x_1

Step8: Set (x, y) equal to starting point, i.e., lowest point and x_{end} equal to largest value of x.

If dx < 0

then x = x_2

y = y_2

x_{end}= x_1

If dx > 0

then x = x_1

y = y_1

x_{end}= x_2

Step9: Check whether the complete line has been drawn if x=x_{end}, stop

Step10: Plot a point at current (x, y) coordinates

Step11: Increment value of x, i.e., x = x+1

Step12: Compute next value of y from equation y = mx + b

Step13: Go to Step9.

Example: A line with starting point as (0, 0) and ending point (6, 18) is given. Calculate value of intermediate points and slope of line.

Solution: P_1 (0,0) P_7 (6,18)

$$x_1 = 0, y_1 = 0, x_2 = 6, y_2 = 18$$

$$m = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1} = \frac{18-0}{6-0} = \frac{18}{6} = 3$$

We know equation of line is

$$y = mx + b$$

$$y = 3x + b \text{ ----------(1)}$$

put value of x from initial point in equation (1), i.e., (0, 0) x =0, y=0

0 = 3 x 0 + b

0 = b $\Longrightarrow$ b=0

put b = 0 in equation (1)

y = 3x + 0

y = 3x

Now calculate intermediate points

Let x = 1 $\Longrightarrow$ y = 3 x 1 $\Longrightarrow$ y = 3

Let x = 2 $\Longrightarrow$ y = 3 x 2 $\Longrightarrow$ y = 6

Let x = 3 $\Longrightarrow$ y = 3 x 3 $\Longrightarrow$ y = 9

Let x = 4 $\Longrightarrow$ y = 3 x 4 $\Longrightarrow$ y = 12

Let x = 5 $\Longrightarrow$ y = 3 x 5 $\Longrightarrow$ y = 15

Let x = 6 $\Longrightarrow$ y = 3 x 6 $\Longrightarrow$ y = 18

So points are P_1 (0,0)

P_2 (1,3)

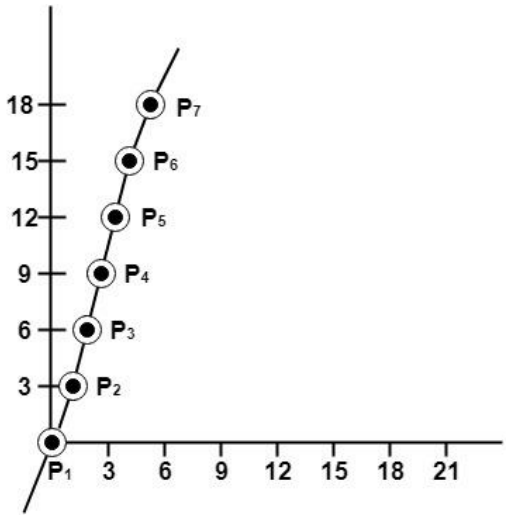
P_3 (2,6)

P_4 (3,9)

P_5 (4,12)

P_6 (5,15)

P_7 (6,18)



2. **Digital Differential Analyzer (DDA):** DDA stands for Digital Differential Analyzer. It is an incremental method of scan conversion of line. In this method calculation is performed at each step but by using results of previous steps.
   - Slope intercept form of the line is

     y = mx + c

     where m – slope of the line

c - intercept of line at Y-Axis

$$m = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1} \quad \text{----------(1)}$$

- The above differential equation can be used to obtain a rasterized line.
- For any given x, interval Δx along a line, we can compute the corresponding y, interval Δy from equation (1)

$$\Delta y = \frac{y_2 - y_1}{x_2 - x_1} \Delta x = m.\Delta x \text{----------(2)}$$

- Similarly, we can obtain the x interval Δx corresponding to a specified Δy as

$$\Delta x = \frac{x_2 - x_1}{y_2 - y_1} \Delta y = \frac{\Delta y}{m} \text{----------(3)}$$

- Once, the interval is known, the values for next x and next y on the straight line can be obtained as follows:

$$x_{i+1} = x_i + \Delta x$$

$$x_{i+1} = x_i + \frac{x_2 - x_1}{y_2 - y_1} \Delta y \text{----------(4)}$$

$$y_{i+1} = y_i + \Delta y$$

$$y_{i+1} = y_i + \frac{y_2 - y_1}{x_2 - x_1} \Delta x \text{----------(5)}$$

- Equation (4) and (5) represent a recursion relation for successive values of x and y along the required line. Such a way of rasterizing a line is called a Digital Differential Analyzer (DDA).
- For simple DDA, either Δx or Δy, whichever is larger is chosen as one raster unit, i.e.

  If $|\Delta x| = |\Delta y|$ then

  $\Delta x = 1$

  Else $\Delta y = 1$

  With this simplification, if $\Delta x = 1$

  $$y_{i+1} = y_i + \frac{y_2 - y_1}{x_2 - x_1}$$

  $$x_{i+1} = x_i + 1$$

  If $\Delta y = 1$ then

  We have, $y_{i+1} = y_i + 1$

  $$x_{i+1} = x_i + \frac{x_2 - x_1}{y_2 - y_1}$$

Algorithm:

Step1: Start Algorithm

Step2: Declare x1,y1,x2,y2,dx,dy,x,y as integer variables.

Step3: Enter value of x1,y1,x2,y2.

Step4: Calculate dx = x2-x1

Step5: Calculate dy = y2-y1

Step6: If ABS (dx) > ABS (dy)

Then step = abs (dx)

Else step = abs (dy)

Step7: xinc=dx/step

yinc=dy/step

assign x = x1

assign y = y1

Step8: Set pixel (x, y)

Step9: x = x + xinc

y = y + yinc

Set pixels (Round (x), Round (y))

Step10: Repeat step 9 until x = x2

Step11: End Algorithm

Advantage:

1. It is a faster method than method of using direct use of line equation.
2. This method does not use multiplication theorem.
3. It allows us to detect the change in the value of x and y ,so plotting of same point twice is not possible.
4. This method gives overflow indication when a point is repositioned.
5. It is an easy method because each step involves just two additions.

Disadvantage:

1. It involves floating point additions rounding off is done. Accumulations of round off error cause accumulation of error.
2. Rounding off operations and floating point operations consumes a lot of time.
3. It is more suitable for generating line using the software. But it is less suited for hardware implementation.

Example 1: **Calculate the points between the starting point (5, 6) and ending point (8, 12).**

Solution-

Given-

Starting coordinates = (X_0, Y_0) = (5, 6)

Ending coordinates = (X_n, Y_n) = (8, 12)

Step-01:

Calculate ΔX, ΔY and M from the given input.

- $\Delta X = X_n - X_0 = 8 - 5 = 3$
- $\Delta Y = Y_n - Y_0 = 12 - 6 = 6$
- $M = \Delta Y / \Delta X = 6 / 3 = 2$

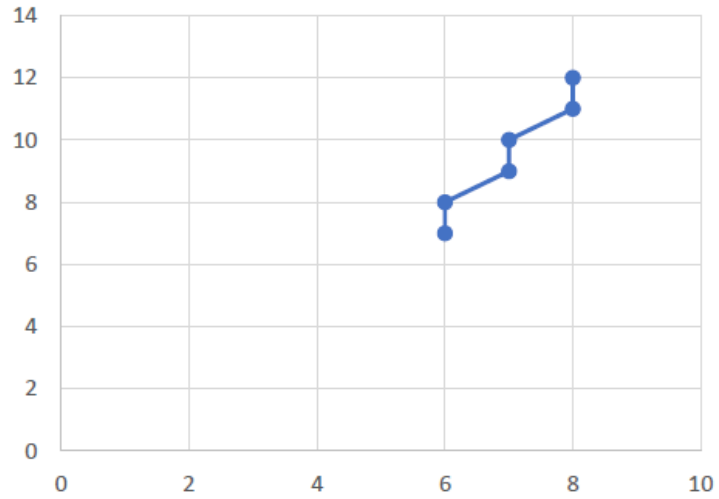
Step-02:

Calculate the number of steps.

As $|\Delta X| < |\Delta Y| = 3 < 6$, so number of steps = $\Delta Y = 6$

Step-03:
As M > 1, so case-03 is satisfied.
Now, **Step-03** is executed until Step-04 is satisfied.

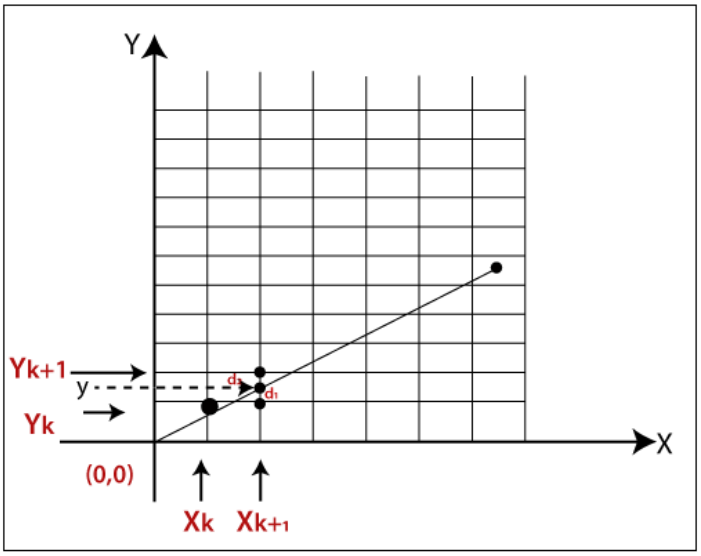
X_p	Y_p	x_{p+1}	Y_{p+1}	Rounding-off (X_{p+1}, Y_{p+1})
5	6	5.5	7	(6,7)
		6	8	(6,8)
		6.5	9	(7,9)
		7	10	(7,10)
		7.5	11	(8,11)
		8	12	(8,12)



3. Bresenham's Line Algorithm:

Bresenham's line algorithm uses only integer addition and subtraction and multiplication by 2. The computer can perform the operations of integer addition and subtraction very rapidly. The computer is also time-efficient when performing integer multiplication by powers of 2. Therefore, it is an efficient method for scan converting straight lines.

The basic principle of Bresenham's line algorithm is to select the optimum raster locations to represent a straight line. To accomplish this, the algorithm always increments either x or y by one unit depending on the slope of line. The increment in other variable is determined by examining the distance between the actual line location and the nearest pixel.



As shown in the above figure let, we have initial coordinates of a line = (x_k, y_k)

The next coordinates of a line = (x_{k+1}, y_{k+1})

The intersection point between y_k and y_{k+1} = y

Let we assume that the distance between *y* and $y_k = d_1$

The distance between *y* and $y_{k+1} = d_2$

Now, we have to decide which point is nearest to the intersection point.

If m < 1

then x = $x_k + 1$ (Unit Interval)

y = $y_k + 1$ (Unit Interval)

As we know the equation of a line-

y = mx +b

Now we put the value of x into the line equation, then

$$y = m(x_k + 1) + b \text{ ---------- (1)}$$

The value of $d_1 = y - y_k$

Putting the value of y, we get $d_1 = m(x_k + 1) + b - y_k$

Now, $d_2 = y_{k+1} - y$

$d_2 = y_k + 1 - m(x_k + 1) - b$

Now, we calculate the difference between d_1 and d_2

If $d_1 < d_2$

Then $y_{k+1} = y_k$ (we will choose the lower pixel as shown in the figure)

If $d_1 >= d_2$

Then $y_{k+1} = y_k + 1$ (we will choose the upper pixel as shown in the figure)

Now we calculate the between two separations $d_1 - d_2$

$d_1 - d_2 = m(x_k + 1) + b - y_k - y_k - 1 + m(x_k + 1) + b$

$2m(x_k + 1) - 2y_k + 2b - 1$

We know that slope $m = \frac{\Delta y}{\Delta x}$

Multiply Δx at both sides, we got

$$\Delta x(d_1 - d_2) = \Delta x(2m(x_k + 1) - 2y_k + 2b - 1)$$

We consider $\Delta x(d_1 - d_2)$ as decision parameter p_k so

$$P_k = \Delta x(d_1 - d_2)$$

$$P_k = \Delta x(2m(x_k + 1) - 2y_k + 2b - 1)$$

$$P_k = 2m\Delta x x_k + 2m\Delta x - 2\Delta x y_k + \Delta x(2b - 1)$$

$$P_k = 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + \Delta x(2b - 1)$$

Next coordinates of P_k

$$P_{k+1} = 2\Delta y x_{k+1} + 2\Delta y - 2\Delta x y_{k+1} + \Delta x(2b - 1)$$

Difference between $P_{k+1} - P_k$

$$P_{k+1} - P_k = 2m\Delta x(x_{k+1} - x_k) - 2\Delta x(y_{k+1} - y_k)$$

$$P_{k+1} = P_k + 2m\Delta x(x_{k+1} - x_k) - 2\Delta x(y_{k+1} - y_k)$$

Bresenham's Algorithm

Step 1: Input the two end-points of line, storing the left end-point in (x_0,y_0)

Step 2: Plot the point (x_0,y_0)

Step 3: Calculate constants $\Delta x, \Delta y$ and obtain the starting value for the decision parameter as P_0

$$P_0 = 2\Delta y - \Delta x$$

Step 4: At each x_k along the line, starting at $k = 0$ perform the following test

If $P_k < 0$, the next point to plot is $(x_{k+1}, y_k$ and

$$P_{k+1} = P_k + 2\Delta y$$

Otherwise ($P_k \geq 0$, the next point to plot is $(x_{k+1}, y_{k+1}$

$$P_{k+1} = P_k + 2\Delta y - 2\Delta x$$

Example: Calculate the points between the starting coordinates (9, 18) and ending coordinates (14, 22) using Bresenham's line algorithm.

Solution: Given-

Starting Coordinates (x_0,y_0) = (9, 18)

Ending Coordinates (x_n,y_n) = (14, 22)

Step 1: Calculate $\Delta y, \Delta x$ from the given input

$$\Delta x = x_n - x_0 = 14 - 9 = 5$$

$$\Delta y = y_n - y_0 = 22 - 18 = 4$$

Step 2: Calculate Decision Parameter

$$P_k = 2\Delta y - \Delta x = 2 \times 4 - 5$$

$P_k = 3$

Step 3: $P_k \geq 0$ so case-2 is satisfied

Thus $P_{k+1} = P_k + 2\Delta y - 2\Delta x$

$3 + 2 \times 4 - 2 \times 5 = 3 + 8 - 5$

$= 1$

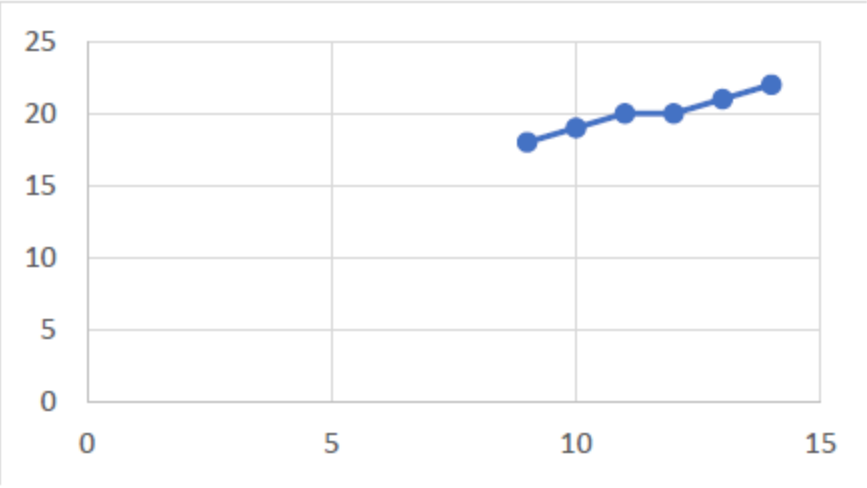
$x_{k+1} = x_k + 1 = 9 + 1 = 10$

$y_{k+1} = y_k + 1 = 18 + 1 = 19$

Similarly, Step-03 is executed until the end point is reached or number of iterations equals to 4 times.

No. of iterations : $\Delta x - 1 = 5 - 1 = 4$

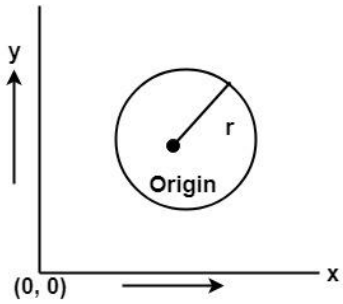
P_k	P_{k+1}	x_{k+1}	y_{k+1}
		9	18
3	1	10	19
1	-1	11	20
-1	7	15	20
7	5	13	21
5	3	14	22



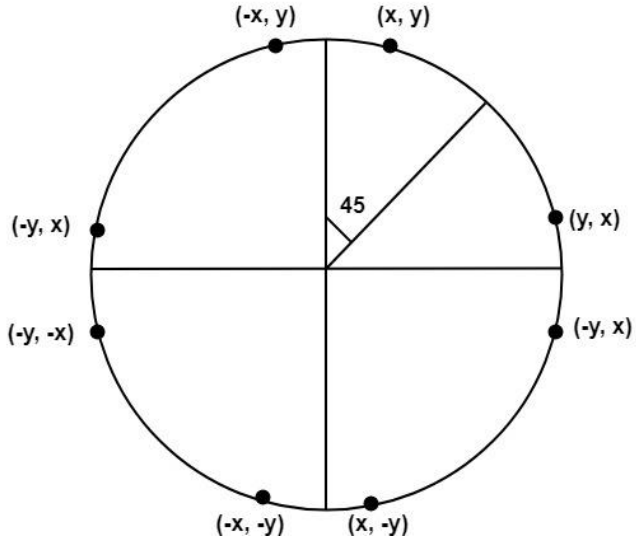
Scan Converting a Circle:

Circle is an eight-way symmetric figure. The shape of circle is the same in all quadrants. In each quadrant, there are two octants. If the calculation of the point of one octant is done, then the other seven points can be calculated easily by using the concept of eight-way symmetry.

For drawing, circle considers it at the origin. If a point is $P_1(x, y)$, then the other seven points will be



P2 (x, -y)
P3(-x, -y)
P4 (-x, y)
P5 (y, x)
P6 (y, -x)
P7 (-y, -x)
P8 (-y, x)



So we will calculate only 45°arc. From which the whole circle can be determined easily.

If we want to display circle on screen then the putpixel function is used for eight points as shown below:

putpixel (x, y, color)

putpixel (x, -y, color)

putpixel (-x, y, color)

putpixel (-x, -y, color)

putpixel (y, x, color)

putpixel (y, -x, color)

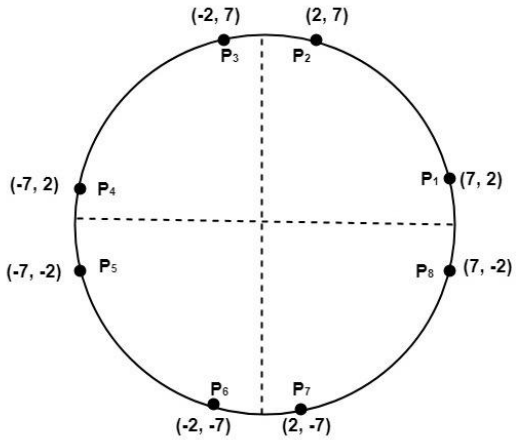
putpixel (-y, x, color)

putpixel (-y, -x, color)

Example: Let we determine a point (2, 7) of the circle then other points will be (2, -7), (-2, -7), (-2, 7), (7, 2), (-7, 2), (-7, -2), (7, -2)

These seven points are calculated by using the property of reflection. The reflection is accomplished in the following way:

The reflection is accomplished by reversing x, y co-ordinates.



Eight way symmetry of a Circle

There are two standards methods of mathematically defining a circle centered at the origin.

1. Defining a circle using Polynomial Method
2. Defining a circle using Polar Co-ordinates

Defining a circle using Polynomial Method:

The first method defines a circle with the second-order polynomial equation as shown in fig:

$$y^2 = r^2 - x^2$$

Where x = the x coordinate
y = the y coordinate
r = the circle radius

With the method, each x coordinate in the sector, from 90° to 45°, is found by stepping x from 0 to $\frac{r}{\sqrt{2}}$ & each y coordinate is found by evaluating $\sqrt{r^2 - x^2}$ for each step of x.

Algorithm:

Step1: Set the initial variables

r = circle radius

(h, k) = coordinates of circle center

x = 0

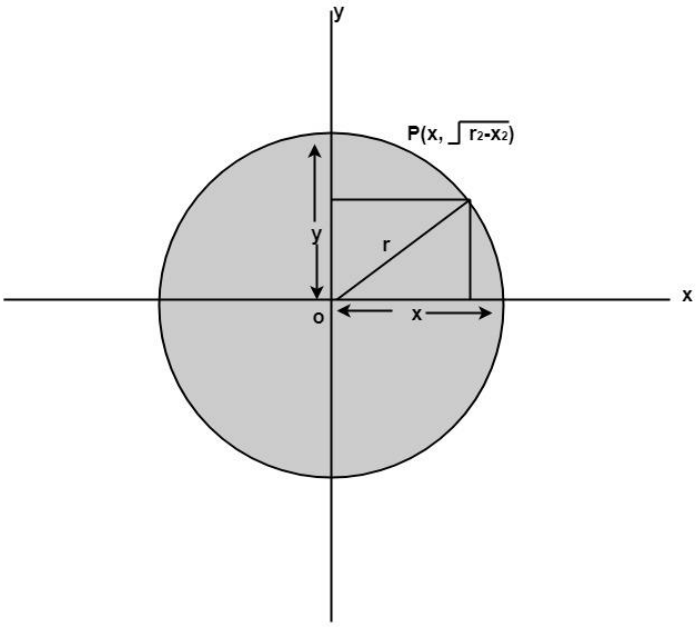
I = step size

$x_{end} = \frac{r}{\sqrt{2}}$

Step2: Test to determine whether the entire circle has been scan-converted.

If $x > x_{end}$ then stop

Step3: Compute $y = \sqrt{r^2 - x^2}$



Step4: Plot the eight points found by symmetry concerning the center (h, k) at the current (x, y) co-ordinates.

Plot (x + h, y +k)	Plot (-x + h, -y + k)
Plot (y + h, x + k)	Plot (-y + h, -x + k)
Plot (-y + h, x + k)	Plot (y + h, -x + k)
Plot (-x + h, y + k)	Plot (x + h, -y + k)

Step5: Increment x = x + i

Step6: Go to step (ii).

Defining a circle using Polar Co-ordinates:

The second method of defining a circle makes use of polar coordinates as shown in fig:

$x = r \cos \theta$ $y = r \sin \theta$
Where θ = current angle
r = circle radius
x = x coordinate
y = y coordinate

By this method, θ is stepped from 0 to $\frac{\pi}{4}$ and each value of x & y is calculated.

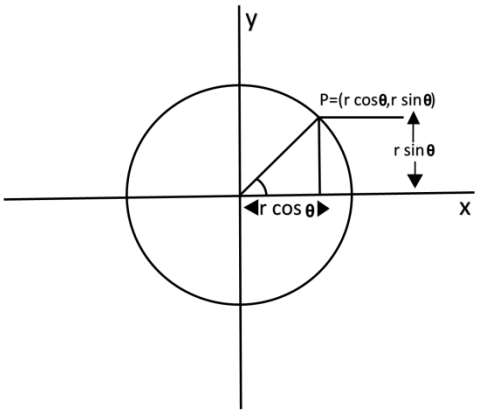
Algorithm:

Step1: Set the initial variables:

r = circle radius
(h, k) = coordinates of the circle center
i = step size

$$\theta_{end} = \frac{\frac{22}{7}}{4}$$

$$\theta = 0$$



Step2: If $\theta > \theta_{end}$ then stop

Step3: Compute

x = r * cos θ y=r*sin θ

Step4: Plot the eight points, found by symmetry i.e., the center (h, k), at the current (x, y) coordinates.

Plot (x + h, y +k) Plot (-x + h, -y + k)
Plot (y + h, x + k) Plot (-y + h, -x + k)
Plot (-y + h, x + k) Plot (y + h, -x + k)
Plot (-x + h, y + k) Plot (x + h, -y + k)

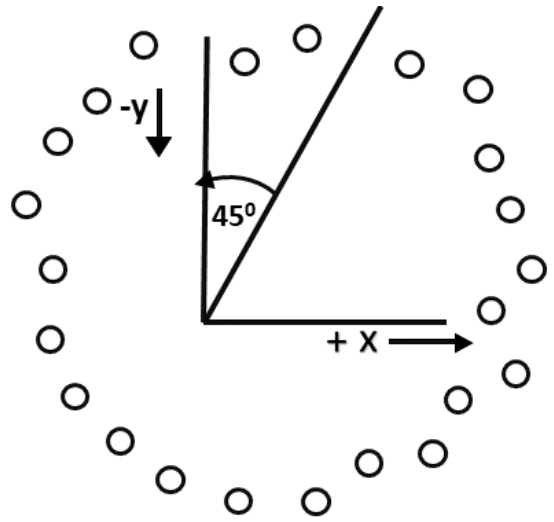
Step5: Increment θ = θ + i

Step6: Go to step (ii).

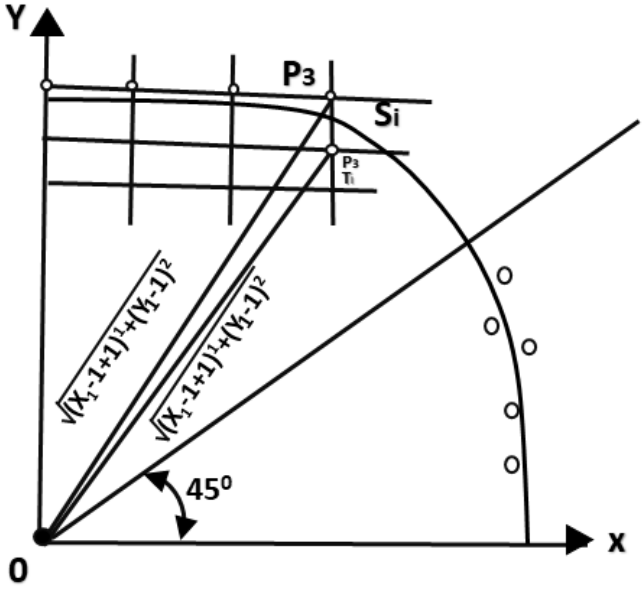
Bresenham's Circle Algorithm:

- This is the most efficient method of drawing the circle. It avoids the use of trigonometric and power functions. It performs the computations with integers and simple additions and subtractions.
- It works on the basic principle of obtaining the next pixel values by means of decision variable and generates the circle with eight symmetry property.
- At each step when we obtain a new pixel/point, there are going to be three possibilities:
  - The point may be on the circle.
  - The point may be outside the circle.
  - The point may be inside the circle.
- Scan-Converting a circle using Bresenham's algorithm works as follows:

- Points are generated from 90° to 45°
- moves will be made only in the +x and -y directions



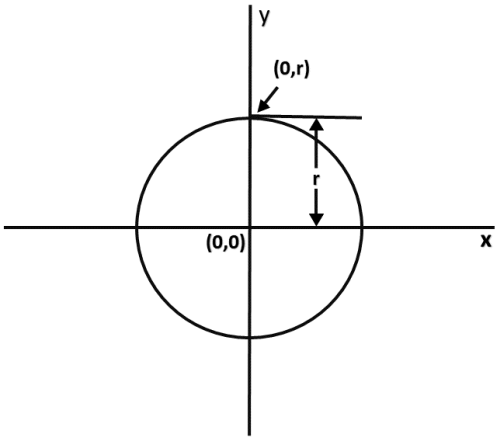
- The best approximation of the true circle will be described by those pixels in the raster that falls the least distance from the true circle. We want to generate the points from 90° to 45°.



- Assume that the last scan-converted pixel is P_1 as shown in fig. Each new point closest to the true circle can be found by taking either of two actions:
  - Move in the x-direction one unit or
  - Move in the x- direction one unit & move in the negative y-direction one unit.
- Let D (S_i) is the distance from the origin to the true circle squared minus the distance to point P_3 squared. D (T_i) is the distance from the origin to the true circle squared minus the distance to point P_2 squared.
- Therefore, the following expressions arise.

$$D(S_i) = (x_{i-1} + 1)^2 + {y_{i-1}}^2 - r^2$$
$$D(T_i) = (x_{i-1} + 1)^2 + (y_{i-1} - 1)^2 - r^2$$

- Since D (S_i) will always be +ve & D (T_i) will always be -ve, a decision variable d may be defined as follows:



d_i=D (S_i)+ D (T_i)

$$d_i = (x_{i-1} + 1)^2 + {y_{i-1}}^2 - r^2 + (x_{i-1} + 1)^2 + (y_{i-1} - 1)^2 - r^2$$

From this equation, we can drive initial values of d_i as

If it is assumed that the circle is centered at the origin, then at the first step x = 0 and y = r.

Therefore,

$d_i=(0+1)^2+r^2 -r^2+(0+1)^2+(r-1)^2-r^2$
$=1+1+r^2-2r+1-r^2$
$= 3 - 2r$

Thereafter, if $d_i < 0$, then only x is incremented.

$$x_{i+1} = x_i + 1 \wedge d_{i+1} = d_i + 4x_i + 6$$

And if $d_i \geq 0$ then x and y are incremented

$$x_{i+1} = x_i + 1 \wedge y_{i+1} = y_i + 1$$

$$d_{i+1} = d_i + 4(x| \quad |i - y_i) + 10$$

Algorithm:

Step1: Start Algorithm

Step2: Declare p, q, x, y, r, d variables
p, q are coordinates of the center of the circle
r is the radius of the circle

Step3: Enter the value of r

Step4: Calculate d = 3 - 2r

Step5: Initialize $x = 0 \wedge y = r$

Step6: Check if the whole circle is scan converted
If x > = y
Stop

Step7: Plot eight points by using concepts of eight-way symmetry. The center is at (p, q). Current active pixel is (x, y).

putpixel (x+p, y+q)
putpixel (y+p, x+q)
putpixel (-y+p, x+q)
putpixel (-x+p, y+q)
putpixel (-x+p, -y+q)
putpixel (-y+p, -x+q)
putpixel (y+p, -x+q)
putpixel (x+p, -y-q)

Step8: Find location of next pixels to be scanned

If $d < 0$
then $d = d + 4x + 6$
increment $x = x + 1$
If $d \geq 0$
then $d = d + 4 (x - y) + 10$
increment $x = x + 1$
decrement $y = y - 1$

Step9: Go to step 6

Step10: Stop Algorithm

Example: Plot 6 points of circle using Bresenham Algorithm. When radius of circle is 10 units. The circle has centre (50, 50).

Solution: Let r = 10 (Given)

Step1: Take initial point (0, 10)

$d = 3 - 2r$
$d = 3 - 2 * 10 = -17$
$d < 0 \therefore d = d + 4x + 6$
$= -17 + 4 (0) + 6$
$= -11$

Step2: Plot (1, 10)

$d = d + 4x + 6 (\because d < 0)$
$= -11 + 4 (1) + 6$
$= -1$

Step3: Plot (2, 10)

$d = d + 4x + 6 (\because d < 0)$
$= -1 + 4 \times 2 + 6$
$= 13$

Step4: Plot (3, 9) d is > 0 so x = x + 1, y = y - 1

$d = d + 4 (x-y) + 10 (\because d > 0)$
$= 13 + 4 (3-9) + 10$
$= 13 + 4 (-6) + 10$
$= 23-24=-1$

Step5: Plot (4, 9)

d = -1 + 4x + 6

= -1 + 4 (4) + 6

= 21

Step6: Plot (5, 8)

d = d + 4 (x-y) + 10 (∵ d > 0)

= 21 + 4 (5-8) + 10

= 21-12 + 10 = 19

So P_1 (0,0)⟹(50,50)

P_2 (1,10)⟹(51,60)

P_3 (2,10)⟹(52,60)

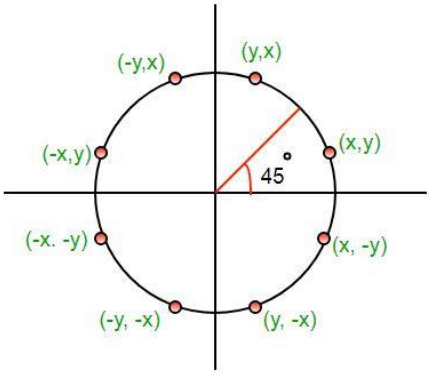
P_4 (3,9)⟹(53,59)

P_5 (4,9)⟹(54,59)

P_6 (5,8)⟹(55,58)

Midpoint Circle Algorithm:

The **mid-point** algorithm is used to calculate all the perimeter points of the circle in the **first octant** and then print them along with their mirror points in the other octants. This will work because a circle is symmetric about its centre.



- For a given radius r and screen centre position (x_0,y_0), we can first setup our algorithm to calculate pixel positions around a circle path centred at co-ordinate origin (0, 0).
- Each calculated position (c, y) is moved to a proper screen position by adding x_c to x and y_c to y along the circle section from x = 0 to x = y in the first quadrant.
- The slope of the curve varies from 0 to -1. Therefore, we can take unit steps in the position x direction over this octant and use a decision parameter to determine which of two possible values of y positions is closer to the circle path at each step.
- Positions in the other seven octants are obtained by symmetry.
- As we know that the equation of a circle is x2 +y2 =r2 when the centre is (0, 0).
- To apply the mid-point method, we defined a circle function

$$f_{(x,y)=x^2+y^2-r^2}$$

- Any point (x, y) on the boundary of the circle with the radius r satisfies the equation

$$f_{(x,y)=0}$$

$$f_{(x,y)}\begin{cases} 0,if(x,y)isinsidetheboundary \\ 0,if(x,y)isontheboundary \\ 0,if(x,y)isoutsidetheboundary \end{cases}$$

- The midpoint between two candidate pixels at sampling position x_{k+1}. Assuming we have just plotted the pixel (x_k, y_k).
- We need to determine whether the pixel at position (x_{k+1}, y_k) or one at position $(x| \quad |k+1, y_{k-1})$ is closer to the circle.
- Our decision parameter is the circle function at midpoint between these two pixels:

$$p_k = f_{\circ}\left(x_k+1, y_k-\frac{1}{2}\right)$$

- Where p_k is a decision parameter and $\frac{1}{2}$ is taken in $p_k = f_{\circ}\left(x_k+1, y_k-\frac{1}{2}\right)$
- the equation because it is a midpoint value through which it is easy to calculate value of y_k and y_{k-1}.

$$p_k = (x_k+1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2$$

- If $p_k < 0$, then midpoint is inside the circle in this condition we select y is y_k otherwise we will select next y as $y_k - 1$
- Successive decision parameters are obtained using incremental calculations. We obtain a recursive expression for the next decision parameter by evaluating the circle function at the sampling position $x_{k+1} + 1 = x_k + 2$:

$$p_{k+1} = f_{\circ}\left(x_{k+1}+1, y_{k+1}-\frac{1}{2}\right)$$

$$p_{k+1} = [(x_k+1)+1]^2 + \left(y_{k+1} - \frac{1}{2}\right)^2 - r^2$$

$$p_{k+1} - p_k = 2(x_k+1) + \left(y^2{}_{k+1} - y^2{}_k\right) - (y_{k+1} - y_k) + 1$$

$$p_{k+1} = p_k + 2(x_k+1) + \left(y^2{}_{k+1} - y^2{}_k\right) - (y_{k+1} - y_k) + 1$$

- Where y_{k+1} is either y_k or y_{k-1} depending on the sign of p_k
- Increments for obtaining p_{k+1} are either $2x_{k+1} + 1$ or $2x_{k+1} + 1 - 2y_{k+1}$
- Evaluation of the terms $2x_{k+1}$ and $2y_{k+1}$ can also be done incrementally as

$$2x_{k+1} = 2x_k + 2$$

$$2y_{k+1} = 2y_k - 2$$

- At the start position (0, r) these two terms have the values 0 and 2r, respectively. Each successive value is obtained by adding 2 to the previous values of 2x and subtracting 2 from the previous value of 2y.
- The initial decision parameter is obtained by evaluating the circle function at the start position (x0,y0) = (0, r).

$$p_0 = f_{\circ}\left(1, r-\frac{1}{2}\right)$$

$$p_0 = 1 + \left(r - \frac{1}{2}\right)^2 - r^2$$

$$p_0 = \frac{5}{4} - r$$

- If the radius r is specified as an integer we can simply round p_0 to

$$p_0 = 1 - r$$

- Since all increments are integers.

Algorithm:

- Input radius r and circle centre (x_c, y_c) and obtain the first point on the circumference of a circle centred on the origin as

$$(x_c, y_c) = (0, r)$$

- Calculate the initial value of the decision parameters as

$$p_0 = \frac{5}{4} - r$$

- At each x_k position, starting at k = 0, perform the following test: If $p_k < 0$ the next point along the circle centred on (0, 0) is $(x_{k+1}, y_k)$and
$$p_{k+1} = p_k + 2x_{k+1} + 1$$
- Otherwise, the next point along the circle is $(x_k + 1, y_k - 1)$ and
$$p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$$
Where $2x_{k+1} = 2x_k + 2$ and $2y_{k+1} = 2y_k - 2$
- Determine the symmetry points in other seven octants.
- Move each calculated pixel position (x, y) onto the circular path centred on (x0,y0) and plot the coordinate value
$$x = x + x_c$$
$$y = y + y_c$$
- Repeat step 3 through 5 until $x \geq y$.

Example: Given the centre point coordinates (0, 0) and radius as 10, generate all the points to form a circle.

Solution:

Given- Centre Coordinates of a circle (x_0,y_0) = (0, 0)

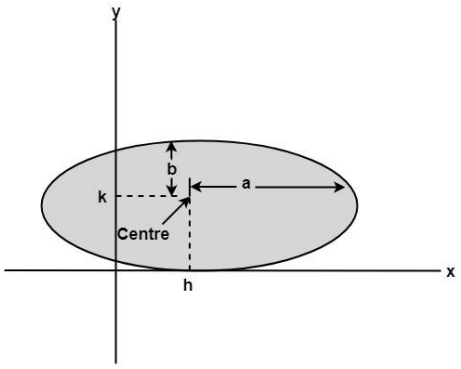
Radius of the circle = 10

1. Assign the starting point coordinates (x_0,y_0) as
$$x_0 = 0$$
$$y_0 = r = 10$$
2. Calculate the value of decision parameter as p_0 as
$$p_0 = 1 - r$$
$$p_0 = 1 - 10$$
$$p_0 = -9$$
3. As $p_{initial} < 0$ so case 1 is satisfied.
Thus,
$$x_{k+1} = x_k + 1 = 0 + 1 = 1$$
$$y_{k+1} = y_k = 10$$
$$p_{k+1} = p_k + 2x_{k+1} + 1 = -9 + 2 \times 1 + 1 = -6$$
4. This step is not applicable here as the given centre point coordinates is (0, 0).
5. Step 3 is executed similarly until $x_{k+1} \geq y_{k+1}$ as follows:

p_k	p_{k+1}	(x_{k+1}, y_{k+1})
		(0, 10)
-9	-6	(1, 10)
-6	-1	(2, 10)
-1	6	(3, 10)
6	-3	(4, 9)
-3	8	(5, 9)
8	5	(6, 8)
Algorithm Terminates These are all points for Octant-1		

Scan Converting an Ellipse:

The ellipse is also a symmetric figure like a circle but is four-way symmetry rather than eight-way.



1. **Polynomial Method:**

The polynomial method of defining an ellipse is given by the expression

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

Where (h, k) = ellipse center

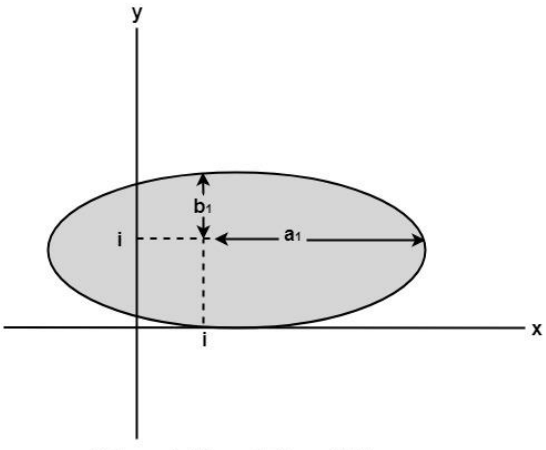
a = length of major axis b = length of minor axis

The ellipse has a major and minor axis. If a and bare major and minor axis respectively. The centre of ellipse is (h, k). The value of x will be incremented from h to a and value of y will be calculated using the following formula

$$y=b\sqrt{1-\frac{(x-h)^2}{a^2}}+k$$

Drawback of Polynomial Method:

- It requires squaring of values. So floating point calculation is required.
- Routines developed for such calculations are very complex and slow.



Polynomial Description of Ellipse

Algorithm:

1) Set the initial variables: a = length of major axis; b = length of minor axis; (h, k) = coordinates of ellipse center; x = 0; i = step; x_{end} = a.
2) Test to determine whether the entire ellipse has been scan-converted. If $x>x_{end}$, stop.
3) Compute the value of the y coordinate:

$$y=b\sqrt{1-\frac{x^2}{a^2}}$$

4) Plot the four points, found by symmetry, at the current (x, y) coordinates:

Plot (x + h, y + k) Plot (-x + h, -y + k) Plot (-y - h, x + k) Plot (y + h, -x + k)

5) Increment x; x = x + i.
6) Go to step 2.

2. Trigonometric Method:

The following equation defines an ellipse trigonometrically as shown in fig:

$x = a * \cos(\theta) + h$ and

$y = b * \sin(\theta) + k$

where (x, y) = the current coordinates

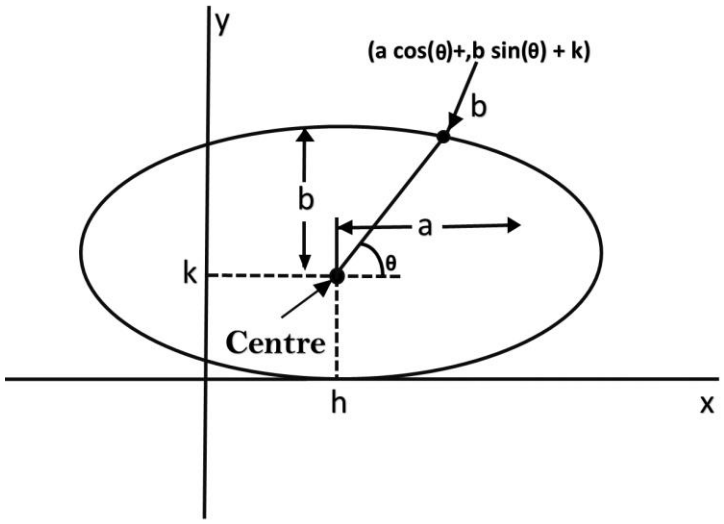
a = length of major axis

b = length of minor axis

θ= current angle

(h, k) = ellipse center

In this method, the value of θ is varied from 0 to $\frac{\pi}{2}$ radians. The remaining points are found by symmetry.



Drawback:

- This is an inefficient method.
- It is not an interactive method for generating ellipse.
- The table is required to see the trigonometric value.
- Memory is required to store the value of θ.

Algorithm:

Step1: Start Algorithm

Step2: Declare variable x_1,y_1,aa_1,bb_1,aa_2,bb_2,fx,fy,p1,a1,b1

Step3: Initialize x_1=0 and y_1=b/* values of starting point of circle */

Step4: Calculate $aa_1=a_1*a_1$
Calculate $bb_1=b_1* b_1$
Calculate $aa_2=aa_1*2$
Calculate $bb_2=bb_1*2$

Step5: Initialize fx = 0

Step6: Initialize fy = aa2* b_1

Step7: Calculate the value of $p_1$and round if it is integer

$p_1=bb_1-aa_1* b_1+0.25* aa_1$

Step8: While (fx < fy)

{

Set pixel (x_1,y_1)

Increment x i.e., x = x + 1

Calculate fx = fx + bb_2

If (p1 < 0)

Calculate p1 = p1 + fx + bb_1/

else

{

Decrement y i.e., y = y-1

Calculate fy = fy – bb_2;

p1=p1 + fx + bb_1-fy

}

}

Step9: Setpixel (x_1,y_1)

Step10: Calculate p1=bb_1 (x+.5)(x+.5)+aa(y-1)(y-1)-aa_1*bb_1

Step 11: While (y_1>0)

{

Decrement y i.e., y = y-1

fy=fx-aa_2/

if (p1>=0)

p1=p1 - fx + aa_1/

else

{

Increment x i.e., x = x + 1

fx= fx+bb_2

p1=p1+fx-fy-aa_1

}

}

Set pixel (x_1,y_1)

Step12: Stop Algorithm

3. **Ellipse Axis Rotation:**

Since the ellipse shows four-way symmetry, it can easily be rotated. The new equation is found by trading a and b, the values which describe the major and minor axes. When the polynomial method is used, the equations used to describe the ellipse become

$$\frac{(x-h)^2}{a^2}+\frac{(y-k)^2}{b^2}$$

Where (h, k) = ellipse center

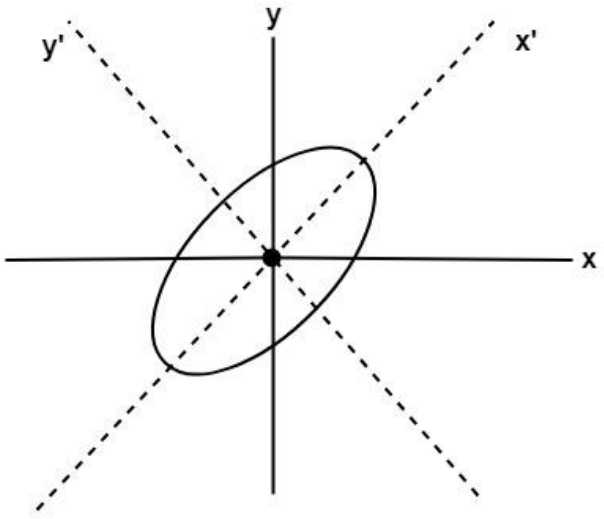
a = length of major axis b = length of minor axis

In the trigonometric method, the equations are
$x = b \cos(\theta) + h$ and $y = a \sin(\theta) + k$

Where (x, y) = current coordinates
a = length of the major axis
b = length of the minor axis
θ = current angle
(h, k) = ellipse center

Assume that you would like to rotate the ellipse through an angle other than 90 degrees. The rotation of the ellipse may be accomplished by rotating the x &y axis α degrees.

$x = a \cos(\theta) - b \sin(\theta + \alpha) + h$
$y = b (\sin \theta) + a \cos(\theta + \alpha) + k$



Rotation of an Ellipse

4. Midpoint Ellipse Algorithm:

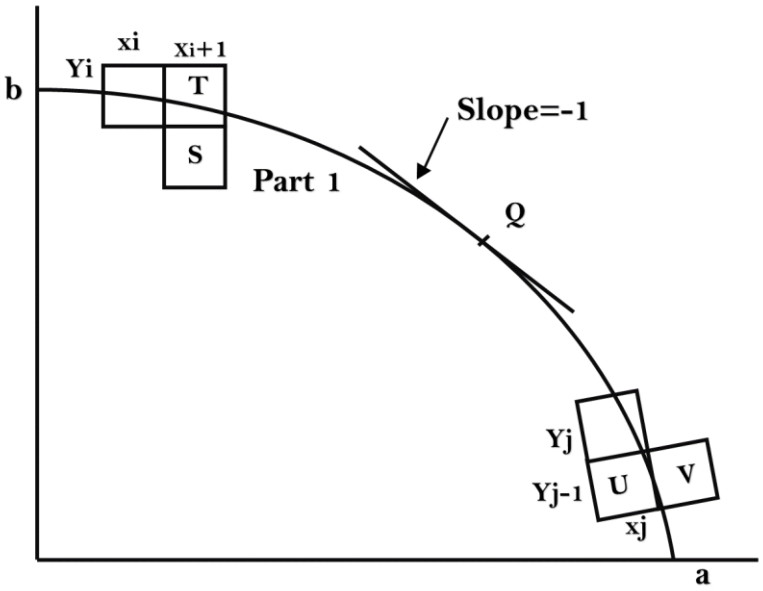
This is an incremental method for scan converting an ellipse that is centered at the origin in standard position i.e., with the major and minor axis parallel to coordinate system axis. It is very similar to the midpoint circle algorithm. Because of the four-way symmetry property we need to consider the entire elliptical curve in the first quadrant.
Let's first rewrite the ellipse equation and define the function f that can be used to decide if the midpoint between two candidate pixels is inside or outside the ellipse:

$$f(x,y) = b^2x^2 + a^2y^2 - a^2b^2 = \begin{cases} 0, if(x,y) isinsidetheellipseboundary \\ 0, if(x,y) isontheellipseboundary \\ 0, if(x,y) isoutsidetheellipseboundary \end{cases}$$

Now divide the elliptical curve from (0, b) to (a, 0) into two parts at point Q where the slope of the curve is -1.

Slope of the curve is defined by the f(x, y) = 0 is $\frac{dy}{dx} = \frac{-fx}{fy}$

where fx and fy are partial derivatives of f(x, y) with respect to x and y respectively.



We have $2b^2x, fy = 2a^2\frac{y \wedge dy}{dx} = \frac{-2b^2x}{2a^2y}$. Hence we can monitor the slope value during the scan conversion process to detect Q. Our starting point is (0, b).

Suppose that the coordinates of the last scan converted pixel upon entering step i are (x_i,y_i). We are to select either T $(x_{i+1}),y_i)$ or S (x_{i+1},y_{i-1}) to be the next pixel. The midpoint of T & S is used to define the following decision parameter.

$$pi = f\left(x_i + 1, y_i - \frac{1}{2}\right)$$

$$pi = b^2(x_i + 1)^2 + a^2\left(y_i - \frac{1}{2}\right)^2 - a^2b^2$$

If pi<0, the midpoint is inside the curve and we choose pixel T.

If pi>0, the midpoint is outside or on the curve and we choose pixel S.

Decision parameter for the next step is:

$$p_{i+1} = f\left(x_{i+1} + 1, y_{i+1} - \frac{1}{2}\right)$$

$$p_{i+1} = b^2(x_{i+1} + 1)^2 + a^2\left(y_{i+1} - \frac{1}{2}\right)^2 - a^2b^2$$

Since $x_{i+1} = x_i + 1$**,** we have

$$p_{i+1} - pi = b^2$$

$$p_{i+1} = pi + 2b^2x_{i+1} + b^2 + a^2\left[\left(y_{i+1} - \frac{1}{2}\right)^2 - \left(y_i - \frac{1}{2}\right)^2\right]$$

If T is chosen pixel (pi<0), we have $y_{i+1} = y_i$

If S is chosen pixel (pi>0) we have $y_{i+1} = y_i - 1$. Thus we can express p_{i+1} in terms of pi and $(x| \quad |i+1, y_{i+1})$

$p_{i+1} = pi + 2b^2 x_{i+1} + b^2$ if $pi < 0$

$p_{i+1} = pi + 2b^2 x_{i+1} + b^2 - 2a^2 y_{i+1}$ if $pi > 0$

The initial value for the recursive expression can be obtained by the evaluating the original definition of pi with (0, b):

$$p_1 = b^2 + a^2\left(b - \frac{1}{2}\right)^2 - a^2 b^2$$

$$p_1 = -a^2 b + a^2/4$$

Suppose the pixel (x_j y_j) has just been scan converted upon entering step j. The next pixel is either U (x_j ,y_j-1) or V (x_j+1,y_j-1). The midpoint of the horizontal line connecting U & V is used to define the decision parameter:b^2

$$q_j = f\left(x_j + \frac{1}{2}, y_j - 1\right)$$

$$q_j = b^2\left(x_j + \frac{1}{2}\right)^2 + a^2(y_j - 1)^2 - a^2 b^2$$

If q_j<0, the midpoint is inside the curve and we choose pixel V.

If $q_j \geq 0$, the midpoint is outside the curve and we choose pixel U. Decision parameter for the next step is:

$$q_{j+1} = f\left(x_{j+1} + \frac{1}{2}, y_{j-1} + 1\right)$$

$$q_{j+1} = b^2\left(x_{j+1} + \frac{1}{2}\right)^2 + a^2(y_{j+1} - 1)^2 - a^2 b^2$$

Since, $y_{j+1} = y_j - 1$, we have

$$q_{j+1} - q_j = b^2\left[\left(x_{j+1} + \frac{1}{2}\right)^2 - \left(x_j + \frac{1}{2}\right)^2\right] + a^2\left[(y_{j+1} - 1)^2 - (y_j - 1)^2\right]$$

$$q_{j+1} = q_j + b^2\left[\left(x_{j+1} + \frac{1}{2}\right)^2 - \left(x_j + \frac{1}{2}\right)^2\right] - 2a^2 y_{j+1} + a^2$$

If V is chosen pixel (qj<0), we have $x_{j+1} = x_j$

If U is chosen pixel (pi>0) we have $x_{j+1} = x_j$. Thus we can express q_{j+1} in terms of q_j and (x_{j+1}, y_{j+1})

$q_{j+1} = q_j + 2b^2 x_{j+1} - 2a^2 y_{j+1} + a^2$ if qj < 0

$q_{j+1} = q_j - 2a^2 y_{j+1} + a^2$ if qj>0

The initial value for the recursive expression is computed using the original definition of qj. And the coordinates of (x_k y_k) of the last pixel choosen for the part 1 of the curve:

$$q_1 = f\left(x_k + \frac{1}{2}, y_k - 1\right)$$

$$q_1 = b^2\left(x_k + \frac{1}{2}\right)^2 - a^2(y_k - 1)^2 - a^2 b^2$$

Algorithm:

1. Input r_x, r_y and ellipse center (x_0, y_0) and obtain the first point on an ellipse centered on the origin as $(x_0, y_0) = (0, r_y)$
2. Calculate the initial value of f_0 the decision parameter as
$$p1_0 = r^2{}_y - r^2{}_x r_y + \frac{1}{4} r^2{}_x$$
3. At each x_k position starting at k = 0, perform the following test: If $p1_k < 0$, the next point along the ellipse centered on (0, 0) is (x_{k+1}, y_k) and
$$p1_{k+1} = p1_k + 2r^2{}_y x_{k+1} + r^2{}_y$$
Otherwise, the point along the circle is, $(x_{k+1},\ y_{k-1})$
And $p1_{k+1} = p1_k + 2r^2{}_y x_{k+1} - 2r^2{}_x y_{k+1} + r^2{}_y$
With $2r^2{}_y x_{k+1} = 2r^2{}_y x_k + 2r^2{}_y$
$2r^2{}_x y_{k+1} = 2r^2{}_x y_k - 2r^2{}_x$
4. Calculate the initial value of decision parameter using the last point $(x_0, y_0$
$$p2_0 = r^2{}_y \left(x_0 + \frac{1}{2}\right)^2 + r^2{}_x (y_0 - 1)^2 - r^2{}_x r^2{}_y$$
5. At each y_k position starting at k = 0, perform the following test: If $p^2{}_k > 0$ the next point along the ellipse centered on (0, 0) is $(x_k, y_k - 1)$ and
$$p2_{k+1} = p2_k - 2r^2{}_y x_{k+1} + r^2{}_x$$
Otherwise, the next point along the circle is $(x_k + 1, y_k - 1)$ and
$$p2_{k+1} = p2_k + 2r^2{}_y x_{k+1} + r^2{}_x - 2r^2{}_y x_{k+1}$$
Using the same incremental, calculate x and y
6. Determine the symmetry quadrant in the other three quadrants
7. Move each calculated pixel position (x, y) onto the elliptical path centered on (x_c, y_c) and plot the coordinate values:
$$x = x_c \wedge y = y + y_c$$
8. Repeat the steps from 1 until $2r^2{}_y x \geq 2r^2{}_x y$