Prof. Jelili O. Oyelade
Mr. Henry Ogbu

Computer Graphics and Animation (CSC433)

What is a computer graphics?

Computer graphics is a sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content. Although the term often refers to the study of three-dimensional computer graphics, it also encompasses two-dimensional graphics and image processing.

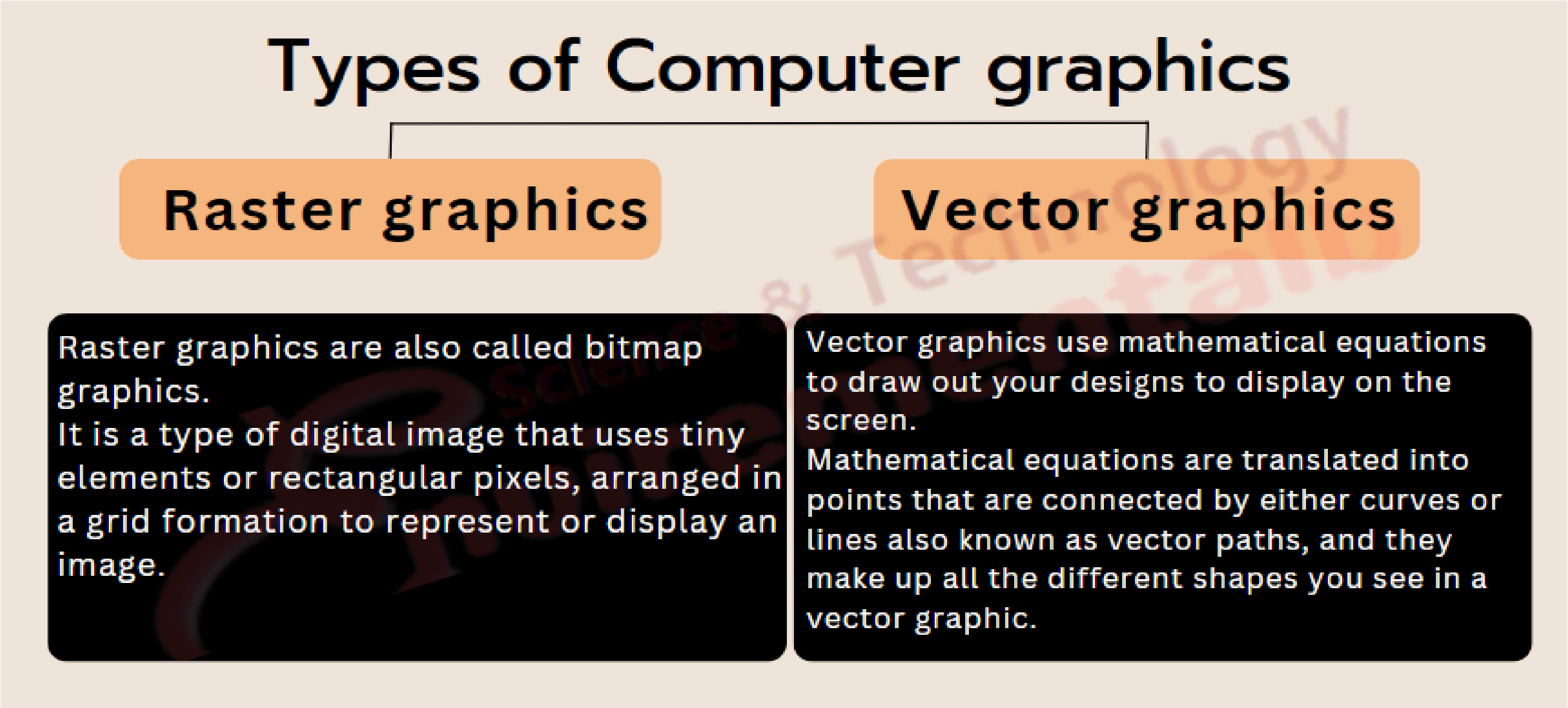
It is used in digital photography, film and television, video games, and on electronic devices and is responsible for displaying images effectively to users. Think of computer graphics as the intersection of design and computer science, with the purpose of delighting and engaging audiences.

- **Computer Graphics is a technology enabling visual content to be drawn, displayed, or manipulated on computer screens with the help of programming languages.**
- Mathematical algorithms for representing, transforming, and rendering visual elements like pixels are used by Computer Graphics techniques such as geometry and modeling, animation, image processing, and rendering.

Types of Computer graphics

They are-

- **Raster (Bitmap) graphics**
- **Vector graphics**



Types of Computer graphics

Raster graphics

Vector graphics

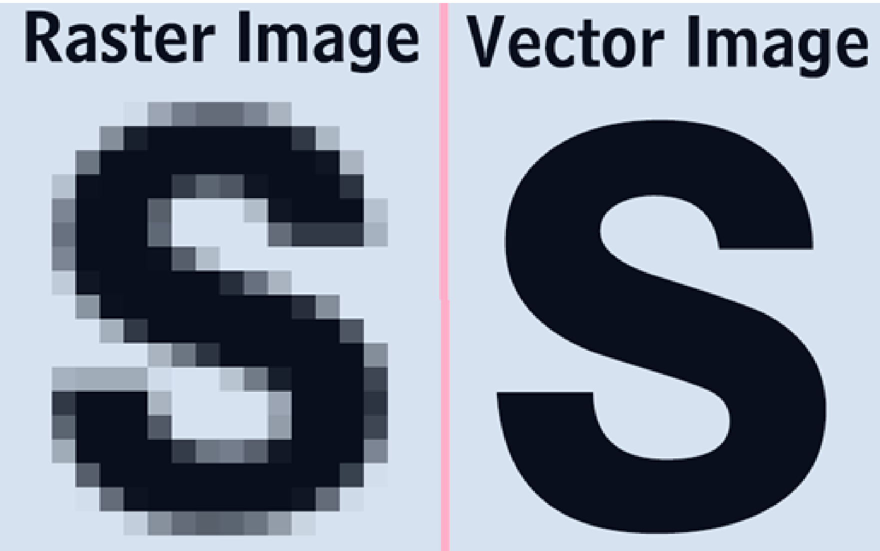
Raster graphics are also called bitmap graphics.
It is a type of digital image that uses tiny elements or rectangular pixels, arranged in a grid formation to represent or display an image.

Vector graphics use mathematical equations to draw out your designs to display on the screen.
Mathematical equations are translated into points that are connected by either curves or lines also known as vector paths, and they make up all the different shapes you see in a vector graphic.

Raster (Bitmap) Graphics

- **Raster graphics** are also called **bitmap graphics.** It is a type of digital image that uses tiny elements or rectangular pixels, arranged in a grid formation to represent or display an image.

- Because the format can support a wide range of colors and depict subtle graduated tones:
  - it is well suited for displaying continuous-tone images such as shaded drawings, photographs along with other detailed images.



File Formats for Raster Images

Raster files are saved in various formats:

- **.png** (Portable Network Graphic)
- **.jpg** (Joint Photographics Expert Group)
- **.gif** (Graphics Interchange Format)
- **.pdf** (Portable Document Format)
- **.tiff** (Tagged Image File Format)
- **.psd** (Adobe Photoshop Document)
- **.bmp** (Bitmap Image File)

Benefits of Raster (Bitmap) Graphics

- Raster files are very easy to construct through existing pixel information stored in a sequence in memory.
- Retrieval of pixel information stored in a raster file is easy while using a collection of coordinates that enables the information to be characterized in the grid form.
- Pixel values can be changed separately if available or as huge sets by changing a gradient.
- Raster graphics or **bitmap graphics** can display on external devices like CRTs and printers in spot format.

Vector graphics

- Vector graphics use mathematical equations to draw out your designs to display on the screen. Mathematical equations are translated into points that are connected by either curves or lines also known as vector paths, and they make up all the different shapes you see in a vector graphic.

- Vector graphics technique scaled to any size without sacrificing or disturbing image quality as well as maintaining a small file size. Common vector file formats are cgm .svg, dog, .eps, and .xml.

File Formats for Vector Images

Vector files can be saved or edited in these formats:

- **.pdf** (Portable Document Format; only when saved from vector programs)
- **.svg** (Scalable Vector Graphic)
- **.ai** (Adobe Illustrator document)
- **.eps** (Encapsulated PostScript)

Application of Computer Graphics

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.

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 (Language of graphics-oriented)
2. COREL DRAW
3. AUTO CAD
4. 3D STUDIO
5. CORE (a core is a small CPU or processor built into a big CPU or CPU socket)
6. GKS (Graphics Kernel System)
7. PHIGS (Programmer's Hierarchical Interactive Graphics System)
8. CAM (Computer Graphics Metafile)
9. CGI (Computer Graphics Interface)

Interactive and Passive Graphics

(a) 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.

(b) 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. It allows the users to actively engage with the digital content they work with which may lead to better interactive visual experiences.

- It encompasses **interactive communication between users and visual content in real-time using different input devices.** These devices include touchscreens, keyboards, mice, game controllers, and VR controllers.

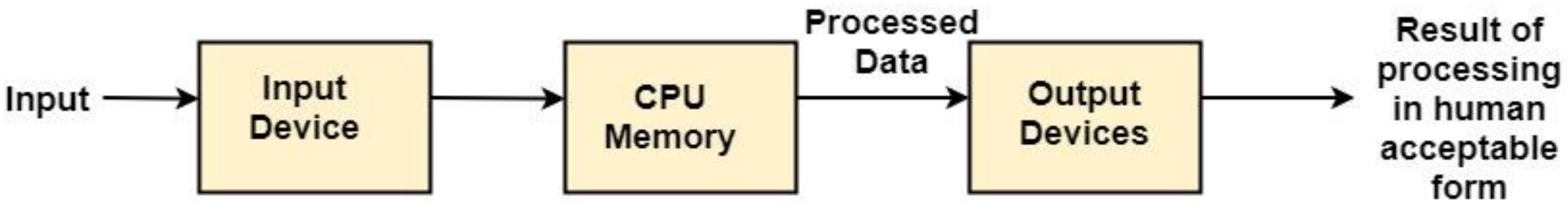
- Interactive Computer Graphics require two-way communication between the computer and the user. A User can see the image and make any change by sending his command with an input device.

Advantages:

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.

Input Devices

The Input Devices are the hardware that is used to transfer transfers input to the computer. The data can be in the form of text, graphics, sound, and text. Output device display data from the memory of the computer. Output can be text, numeric data, line, polygon, and other objects.



Some of these Devices are:

1. Keyboard
2. Mouse
3. Trackball
4. Spaceball
5. Joystick
6. Light Pen
7. Digitizer
8. Touch Panels
9. Voice Recognition
10. Image Scanner
11. etc.

Output Devices

It is an electromechanical device, which accepts data from a computer and translates them into form understand by users.

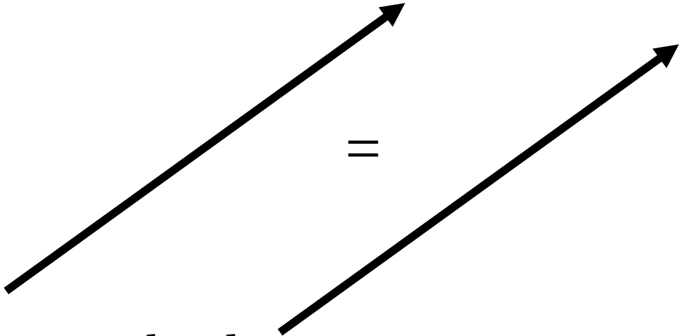
Following are Output Devices:

1. Printers
2. Plotters

Ingredients of Computer Graphics

- Many graphics concepts need basic mathematics like **linear algebra, Geometry, etc.**
  - Vectors (dot products, cross products, …)
  - Matrices (matrix-matrix, matrix-vector mult., …)
  - For example, a point is a vector, and an operation like translating or rotating points on object can be matrix-vector multiply

Vectors

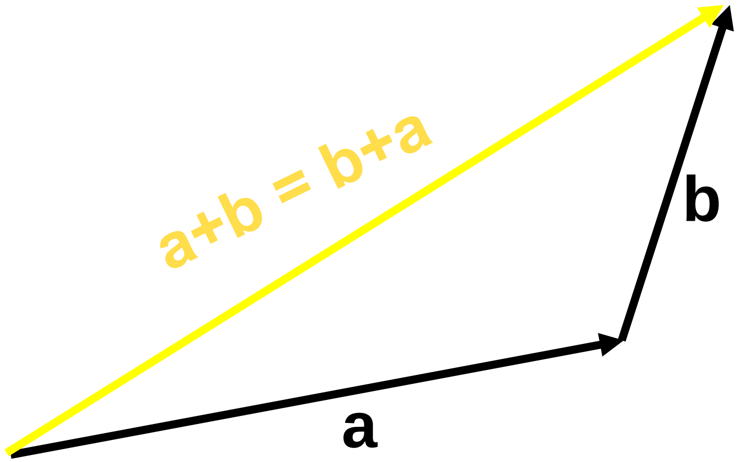


- Length and direction. Absolute position not important

Usually written as $\vec{a}$ or in bold. Magnitude written as $\|\vec{a}\|$

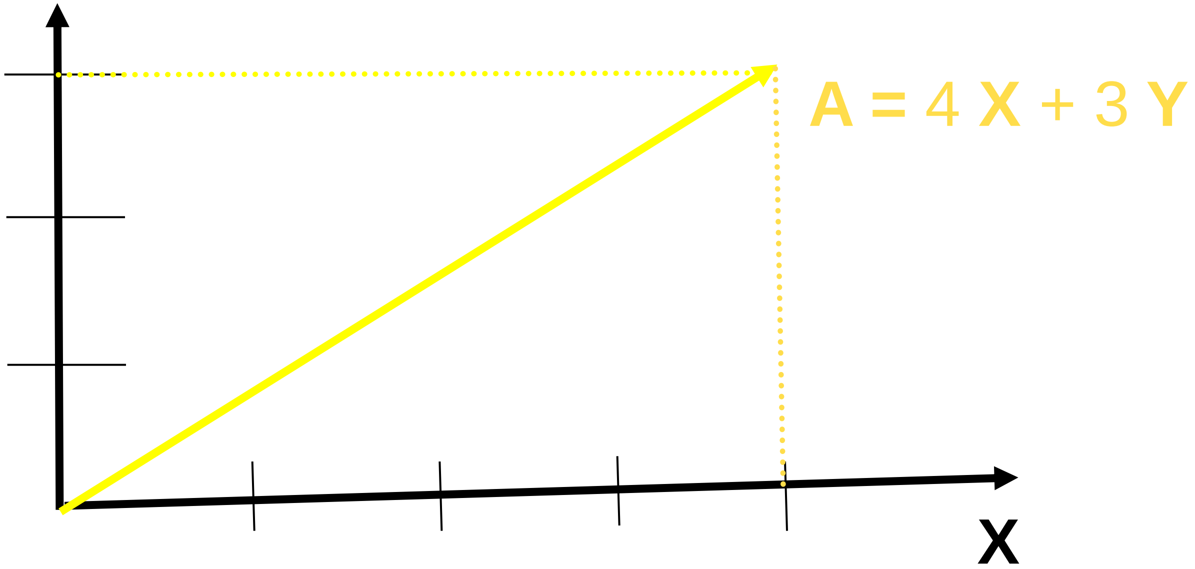
- Use to store offsets, displacements, locations
  - But strictly speaking, positions are not vectors and cannot be added: a location implicitly involves an origin, while an offset does not.

Vector Addition



- Geometrically: Parallelogram rule
- In cartesian coordinates (next), simply add coords

Cartesian Coordinates



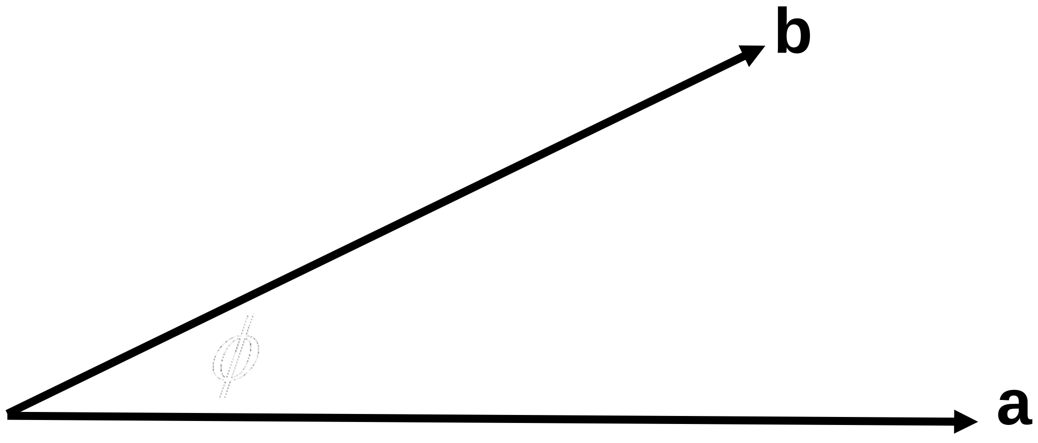
- X and Y can be any (usually orthogonal ***unit***) vectors

$$A = \begin{pmatrix} x \\ y \end{pmatrix} \qquad A^T = \begin{pmatrix} x & y \end{pmatrix} \qquad \|A\| = \sqrt{x^2 + y^2}$$

Vector Multiplication

- *Dot product*
- Cross product
- Orthonormal bases and coordinate frames

Dot (scalar) product



$$a \cdot b = b \cdot a = ?$$

$$a \cdot b = \|a\| \|b\| \cos\phi$$

$$a \cdot (b + c) = a \cdot b + a \cdot c$$

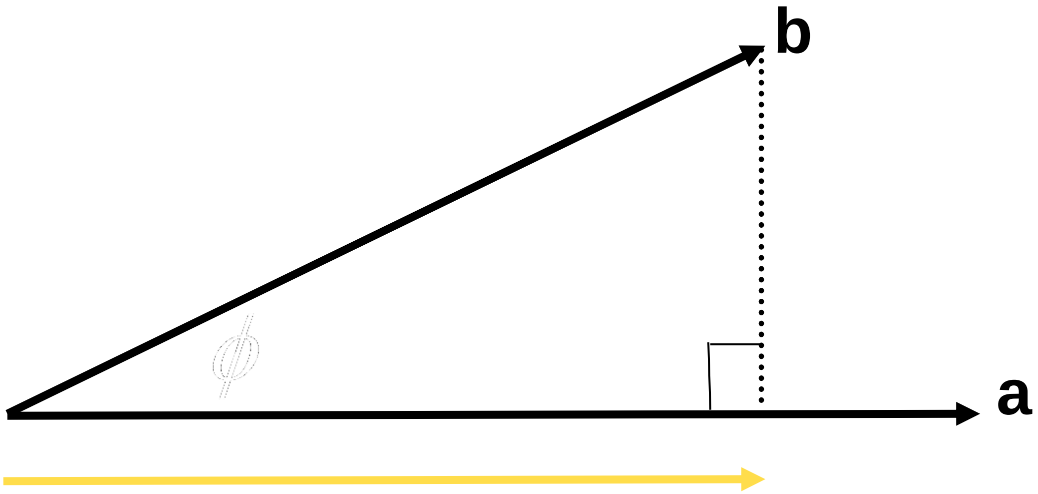
$$(ka) \cdot b = a \cdot (kb) = k(a \cdot b)$$

$$\phi = \cos^{-1}\left(\frac{a \cdot b}{\|a\| \|b\|}\right)$$

Dot product: some applications in CG

- Find angle between two vectors (e.g. cosine of angle between light source and surface for shading)
- Finding projection of one vector on another (e.g. coordinates of point in arbitrary coordinate system)
- Advantage: computed easily in cartesian components

Projections (of b on a)



$$\|b \to a\| = ?$$

$$b \to a \quad = ?$$

$$\|b \to a\| = \|b\| \cos\phi = \frac{a \cdot b}{\|a\|}$$

$$b \to a = \|b \to a\| \frac{a}{\|a\|} = \frac{a \cdot b}{\|a\|^2} a$$

Dot product in Cartesian components

$$a \bullet b = \begin{pmatrix} x_a \\ y_a \end{pmatrix} \bullet \begin{pmatrix} x_b \\ y_b \end{pmatrix} = ?$$

$$a \bullet b = \begin{pmatrix} x_a \\ y_a \end{pmatrix} \bullet \begin{pmatrix} x_b \\ y_b \end{pmatrix} = x_a x_b + y_a y_b$$

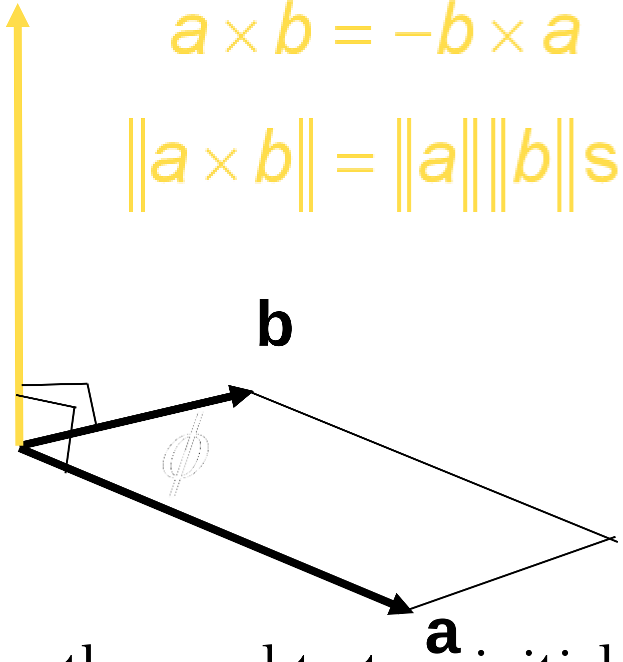
Vector Multiplication

- Dot product
- *Cross product*

Cross (vector) product

$$a \times b = -b \times a$$

$$\|a \times b\| = \|a\|\|b\| \sin \phi$$



- Cross product orthogonal to two initial vectors
- Direction determined by right-hand rule
- Useful in constructing coordinate systems

Cross product: Properties

$$x \times y = +z$$

$$y \times x = -z$$

$$y \times z = +x$$

$$z \times y = -x$$

$$z \times x = +y$$

$$x \times z = -y$$

$$a \times b = -b \times a$$

$$a \times a = 0$$

$$a \times (b + c) = a \times b + a \times c$$

$$a \times (kb) = k(a \times b)$$

Cross product: Cartesian formula?

$$a \times b = \begin{vmatrix} x & y & z \\ x_a & y_a & z_a \\ x_b & y_b & z_b \end{vmatrix} = \begin{pmatrix} y_a z_b - y_b z_a \\ z_a x_b - x_a z_b \\ x_a y_b - y_a x_b \end{pmatrix}$$

$$a \times b = A^* b = \begin{pmatrix} 0 & -z_a & y_a \\ z_a & 0 & -x_a \\ -y_a & x_a & 0 \end{pmatrix} \begin{pmatrix} x_b \\ y_b \\ z_b \end{pmatrix}$$

Dual matrix of vector a

Vector Multiplication

- Dot product
- Cross product

Matrices

- Can be used to transform points (vectors)
  - Translation, rotation, shear, scale.

What is a matrix

- Array of numbers (m×n = m rows, n columns)

$$\begin{pmatrix} 1 & 3 \\ 5 & 2 \\ 0 & 4 \end{pmatrix}$$

- Addition, multiplication by a scalar simple: element by element

Matrix-matrix multiplication

- Number of columns in first must = rows in second

$$\begin{pmatrix} 1 & 3 \\ 5 & 2 \\ 0 & 4 \end{pmatrix}\begin{pmatrix} 3 & 6 & 9 & 4 \\ 2 & 7 & 8 & 3 \end{pmatrix}$$

- Element (i,j) in product is dot product of row i of first matrix and column j of second matrix

Matrix-matrix multiplication

- Number of columns in first must = rows in second

$$\begin{pmatrix} 1 & 3 \\ 5 & 2 \\ 0 & 4 \end{pmatrix}\begin{pmatrix} 3 & 6 & 9 & 4 \\ 2 & 7 & 8 & 3 \end{pmatrix}=\begin{pmatrix} 9 & 27 & 33 & 13 \\ 19 & 44 & 61 & 26 \\ 8 & 28 & 32 & 12 \end{pmatrix}$$

- Element (i,j) in product is dot product of row i of first matrix and column j of second matrix

Matrix-matrix multiplication

- Number of columns in first must = rows in second

$$\begin{pmatrix} 3 & 6 & 9 & 4 \\ 2 & 7 & 8 & 3 \end{pmatrix}\begin{pmatrix} 1 & 3 \\ 5 & 2 \\ 0 & 4 \end{pmatrix}$$ NOT EVEN LEGAL!!

- Non-commutative (AB and BA are different in general)
- Associative and distributive
  - A(B+C) = AB + AC
  - (A+B)C = AC + BC

Matrix-Vector Multiplication

- Key for transforming points (next lecture)
- Treat vector as a column matrix (m×1)

- E.g. 2D reflection about y-axis (from textbook)

$$\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -x \\ y \end{pmatrix}$$

Transpose of a Matrix (or vector?)

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}^T = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$$

$$(AB)^T = B^T A^T$$

Identity Matrix and Inverses

$$I_{3\times 3} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$AA^{-1} = A^{-1}A = I$$

$$(AB)^{-1} = B^{-1}A^{-1}$$

Vector multiplication in Matrix form

- Dot product? $a \bullet b = a^T b$

$$\begin{pmatrix} x_a & y_a & z_a \end{pmatrix} \begin{pmatrix} x_b \\ y_b \\ z_b \end{pmatrix} = \left(x_a x_b + y_a y_b + z_a z_b \right)$$

- Cross product?

$$a \times b = A^* b = \begin{pmatrix} 0 & -z_a & y_a \\ z_a & 0 & -x_a \\ -y_a & x_a & 0 \end{pmatrix} \begin{pmatrix} x_b \\ y_b \\ z_b \end{pmatrix}$$

Scan Conversion Definition

- It is a process of representing graphics objects as 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 represent 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. For generating graphical object, many algorithms have been developed.

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

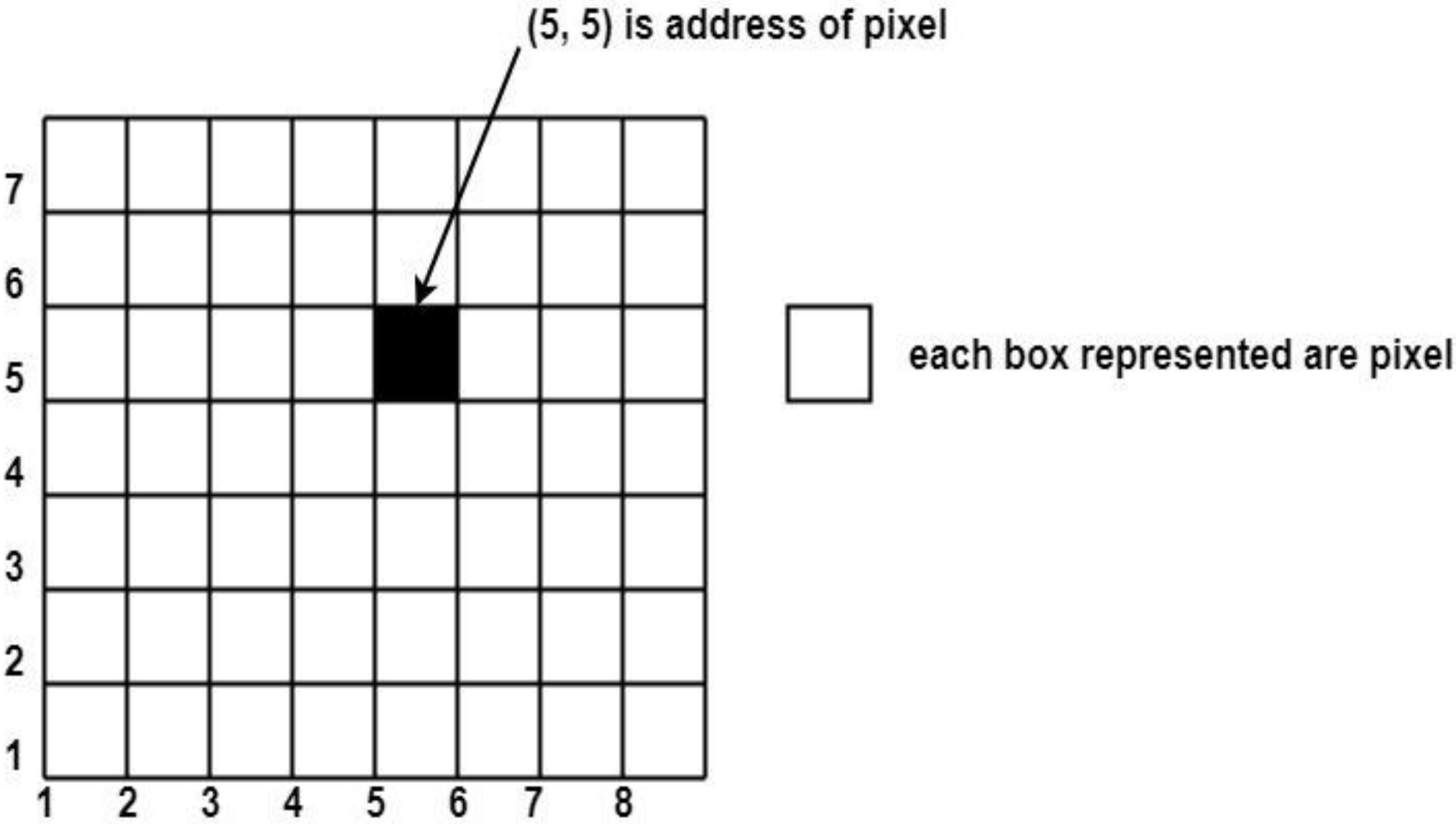
1. Point
2. Line
3. Sector
4. Arc
5. Ellipse
6. Rectangle
7. Polygon
8. Characters
9. Filled Regions

Therefore, 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. 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:



(5, 5) is address of pixel

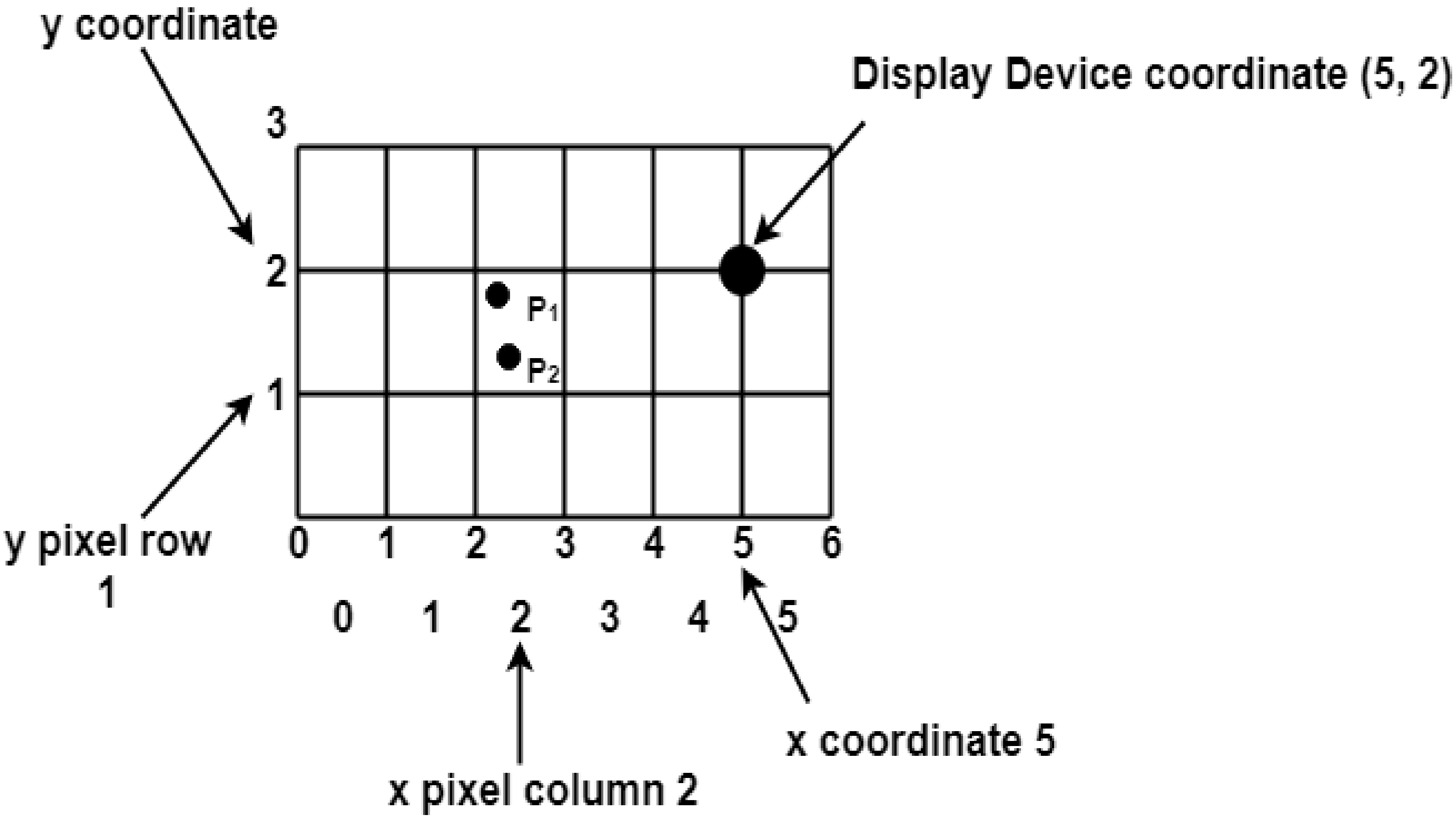
each box represented are pixel

- 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.

Scan Converting a Point

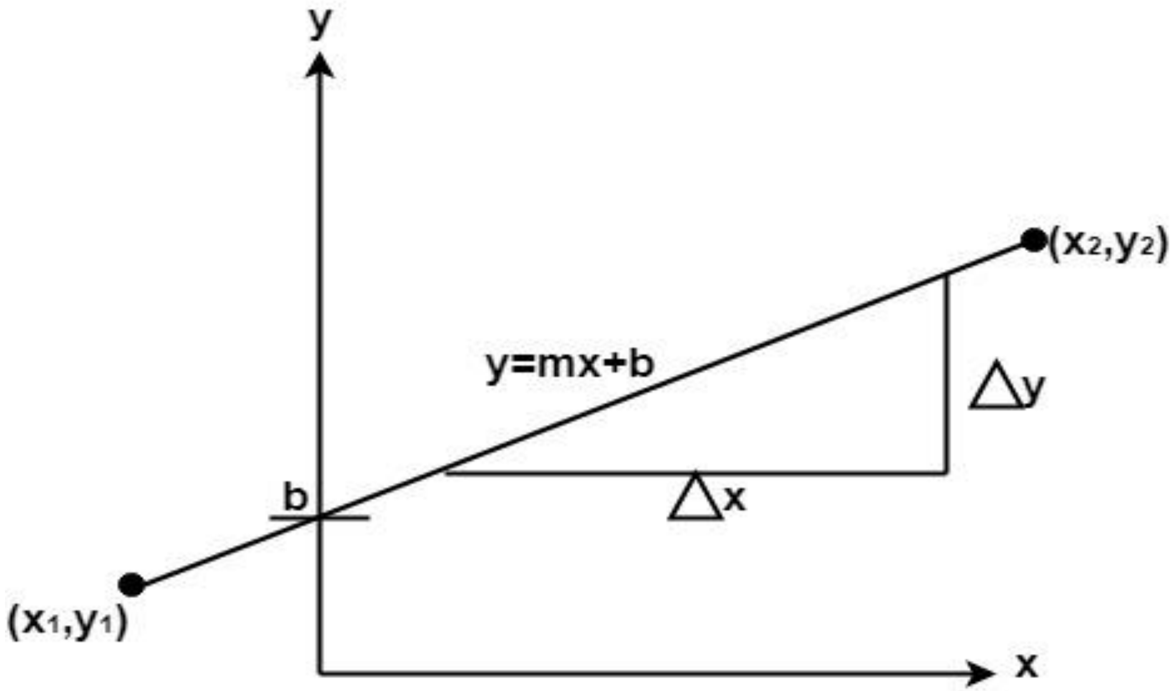
Each pixel on the graphics display does not represent a mathematical point. Instead, it means a region which theoretically can contain an infinite number of points. Scan-Converting a point involves illuminating the pixel that contains the point.

Example: Display coordinates points $P_1\ (2\tfrac{1}{4}, 1\tfrac{3}{4})$ & $P_2(2\tfrac{2}{3}, 1\tfrac{1}{4})$ 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).


y coordinate
Display Device coordinate (5, 2)
3
2
1
P1
P2
y pixel row
1
0
1
2
3
4
5
6
0
1
2
3
4
5
x coordinate 5
x pixel column 2

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 intercept (the point on the y-axis where the line crosses it), 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.

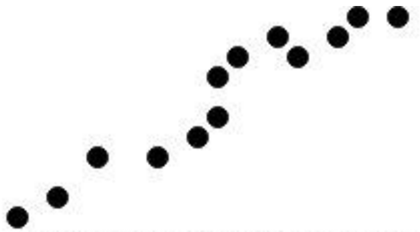


Fig: O/P from a poor line generating algorithm

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.

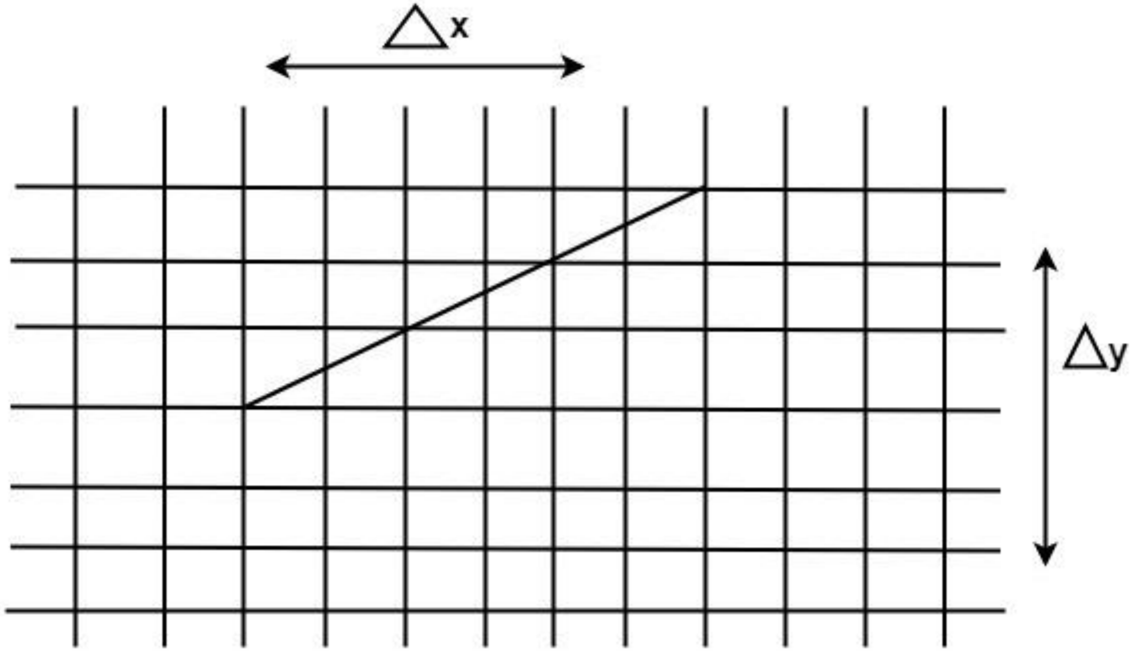


Fig: A straight line segment connecting 2 grid intersection may fail to pass through any other grid intersections.

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

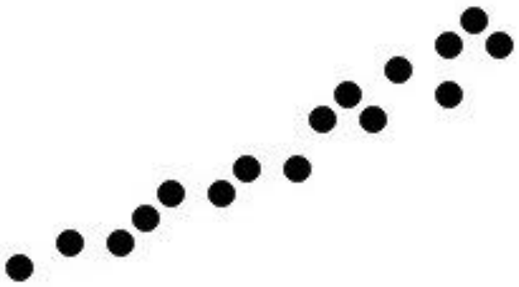


Fig: Uneven line density caused by bunching of dots.

3. Lines should have constant density: Line density is proportional to the no. of dots displayed divided by the length of the line.

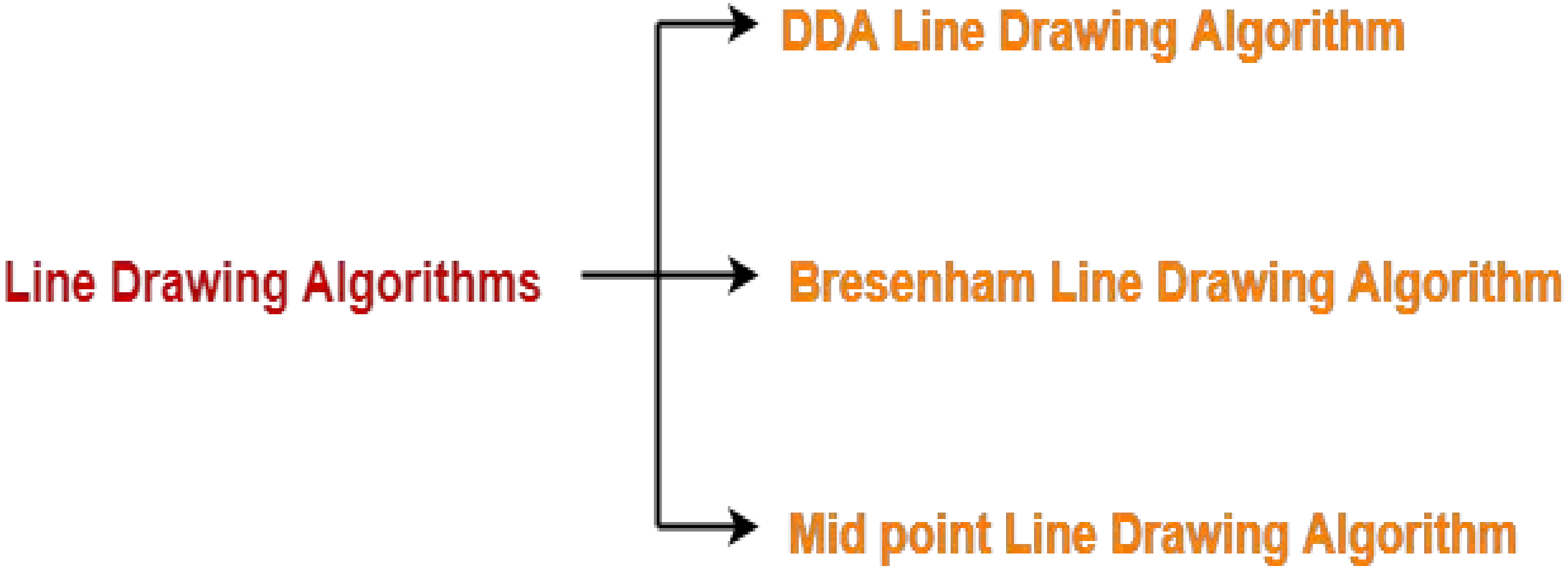
To maintain constant density, dots should be equally spaced.

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.

Line Drawing Algorithms-

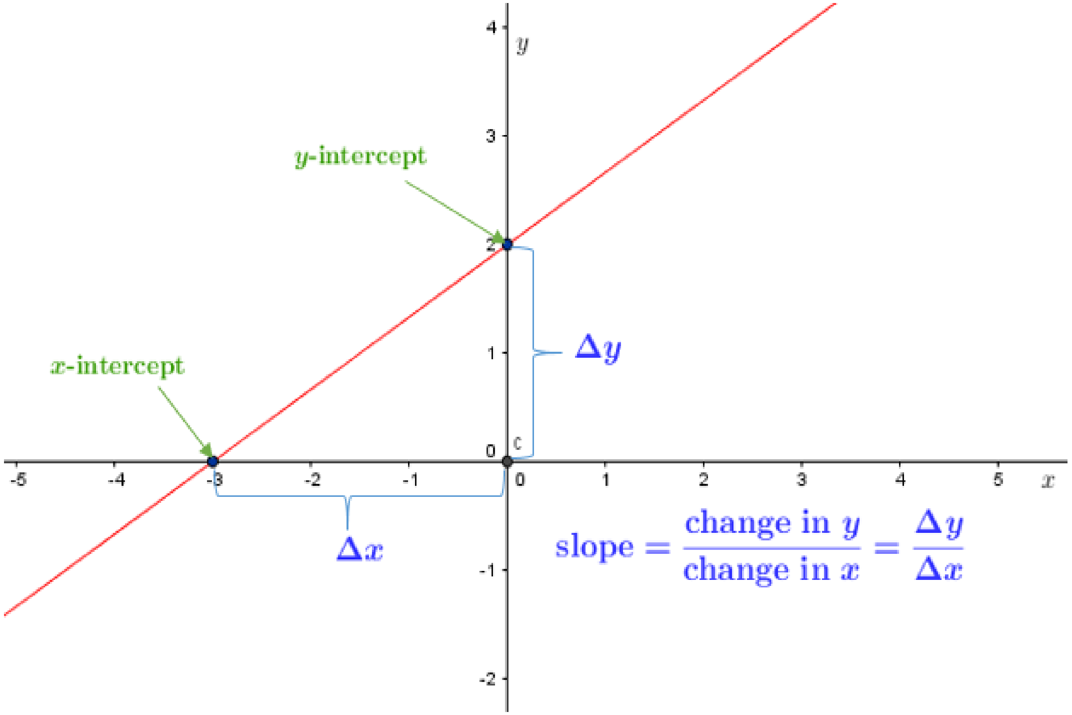
In computer graphics, popular algorithms used to generate lines are-


Line Drawing Algorithms
DDA Line Drawing Algorithm
Bresenham Line Drawing Algorithm
Mid point Line Drawing Algorithm

1. Digital Differential Analyzer (DDA) Line Drawing Algorithm
2. Bresenham Line Drawing Algorithm
3. Mid Point Line Drawing Algorithm

Recall the line equation below:

$$y = mx + c$$
$$m = slope$$
$$c = intercept$$



$$\text{slope} = \frac{\text{change in } y}{\text{change in } x} = \frac{\Delta y}{\Delta x}$$

DDA Algorithm-

DDA Algorithm is the simplest line drawing algorithm.

Procedure-

Given-

- Starting coordinates = (X_0, Y_0)
- Ending coordinates = (X_n, Y_n)

The points generation using DDA Algorithm involves the following steps-

Step-01:

Calculate ΔX, ΔY and M from the given input.

These parameters are calculated as-

- $\Delta X = X_n – X_0$
- $\Delta Y = Y_n – Y_0$
- $M = \Delta Y / \Delta X$

Step-02:

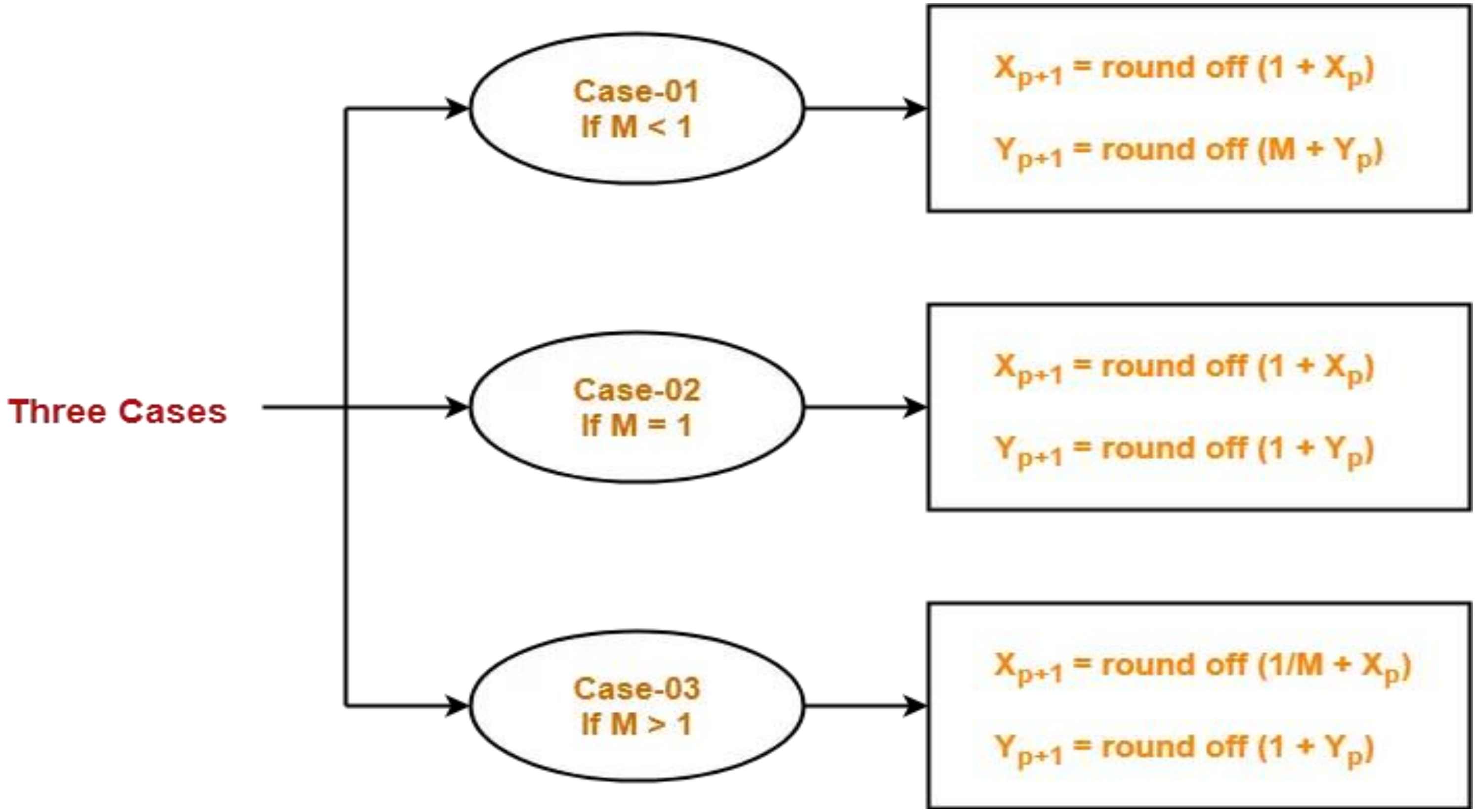
Find the number of steps or points in between the starting and ending coordinates.

```
if (absolute (ΔX) > absolute (ΔY))
Steps = absolute (ΔX);
else
Steps = absolute (ΔY);
```

Step-03:

Suppose the current point is (X_p, Y_p) and the next point is (X_{p+1}, Y_{p+1}).

Find the next point by following the below three cases-



Three Cases
Case-01
If M < 1
X_{p+1} = round off (1 + X_p)
Y_{p+1} = round off (M + Y_p)
Case-02
If M = 1
X_{p+1} = round off (1 + X_p)
Y_{p+1} = round off (1 + Y_p)
Case-03
If M > 1
X_{p+1} = round off (1/M + X_p)
Y_{p+1} = round off (1 + Y_p)

Step-04:

Keep repeating Step-03 until the end point is reached or the number of generated new points (including the starting and ending points) equals to the steps count.

Example1:

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.

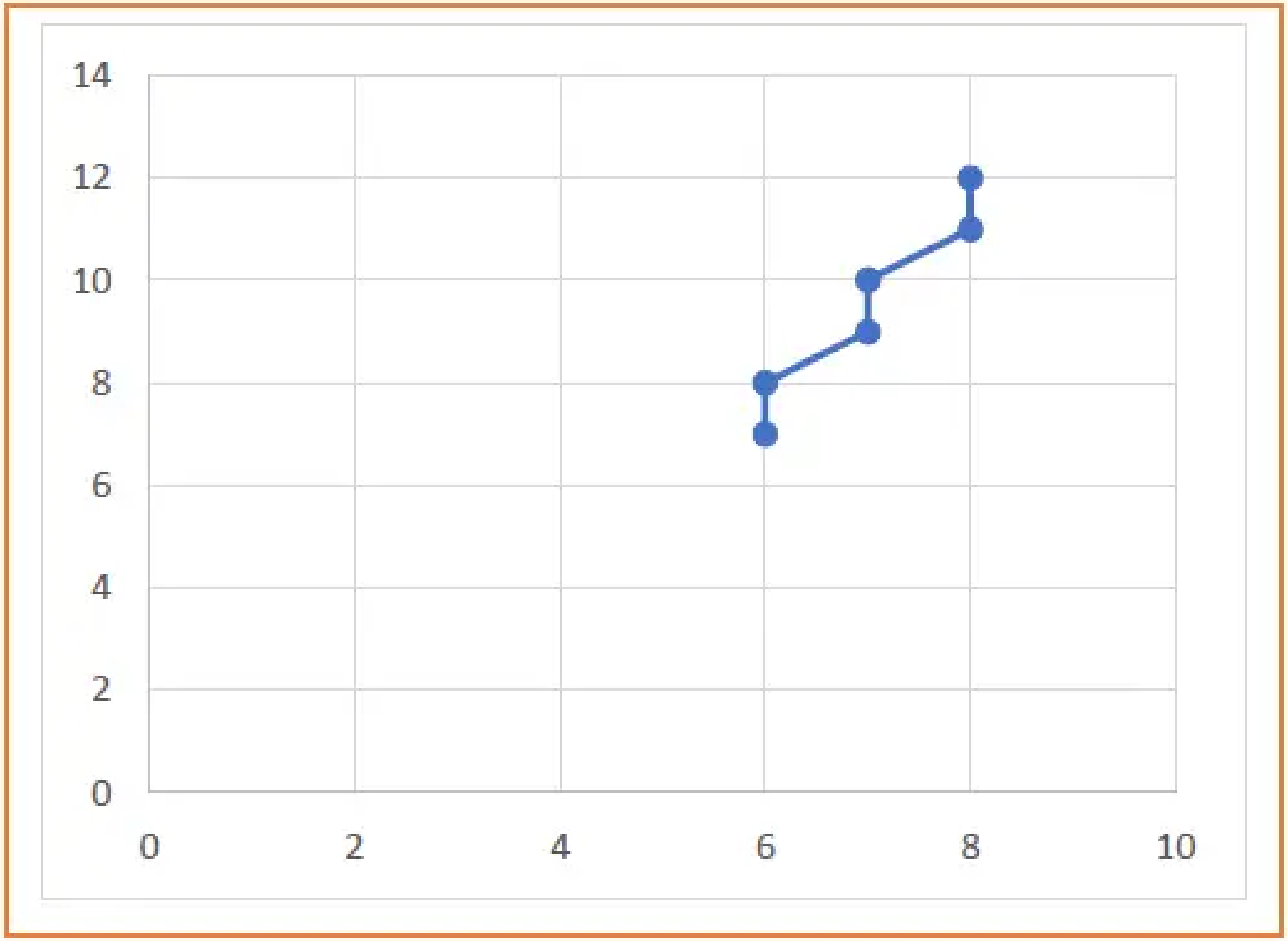
As $|\Delta X| < |\Delta Y| = 3 < 6$, so number of steps = ΔY = 6

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}	Round 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)


14
12
10
8
6
4
2
0
0
2
4
6
8
10

Example2:

Calculate the points between the starting point (1, 7) and ending point (11, 17).

Solution-

Given-

- Starting coordinates = (X_0, Y_0) = (1, 7)
- Ending coordinates = (X_n, Y_n) = (11, 17)

Step-01:

Calculate ΔX, ΔY and M from the given input.

- $\Delta X = X_n - X_0 = 11 - 1 = 10$
- $\Delta Y = Y_n - Y_0 = 17 - 7 = 10$
- $M = \Delta Y / \Delta X = 10 / 10 = 1$

Step-02:

Calculate the number of steps.

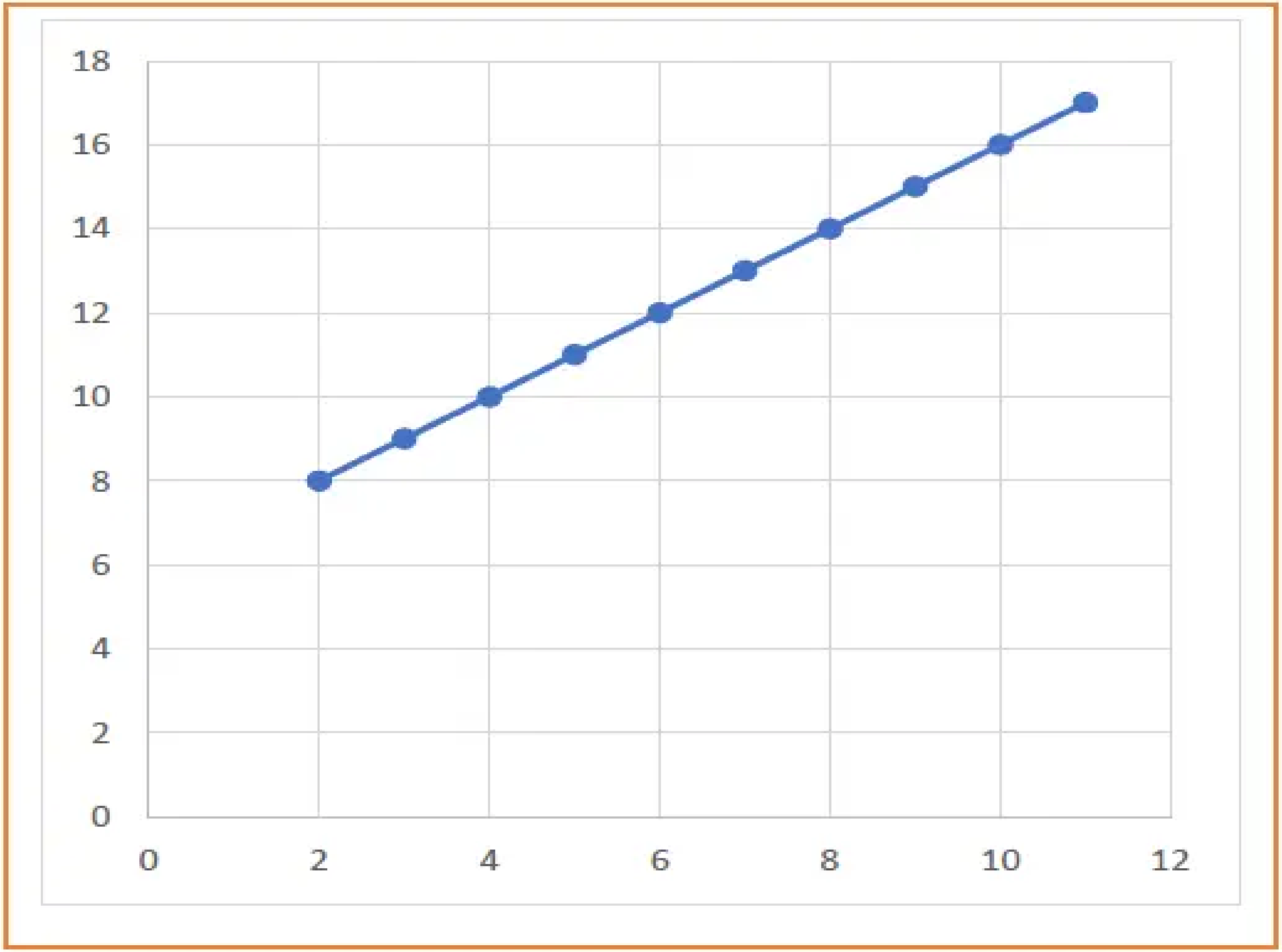
As |ΔX| = |ΔY| = 10 = 10, so number of steps = ΔX = ΔY = 10

Step-03:

As M = 1, so case-02 is satisfied.

Now, Step-03 is executed until Step-04 is satisfied.

X_p	Y_p	X_{p+1}	Y_{p+1}	Round off (X_{p+1}, Y_{p+1})
1	7	2	8	(2, 8)
		3	9	(3, 9)
		4	10	(4, 10)
		5	11	(5, 11)
		6	12	(6, 12)
		7	13	(7, 13)
		8	14	(8, 14)
		9	15	(9, 15)
		10	16	(10, 16)
		11	17	(11, 17)


18
16
14
12
10
8
6
4
2
0
0
2
4
6
8
10
12

The advantages of DDA Algorithm are-

- It is a simple algorithm.
- It is easy to implement.
- It avoids using the multiplication operation which is costly in terms of time complexity.

Disadvantages of DDA Algorithm-

The disadvantages of DDA Algorithm are-

- There is an extra overhead of using round off( ) function.
- Using round off( ) function increases time complexity of the algorithm.
- Resulted lines are not smooth because of round off( ) function.
- The points generated by this algorithm are not accurate.

DDA 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

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
```

Assignment 2

Given the initial values as: x0 = 100 , y0 = 200, x1 = 500, y1 = 300:

a. In tabular form, calculate the point between the pts (x0, y0) and (x1, y1).

b. Hence, write a program to DDA Line Drawing Algorithm.

Bresenham Line Drawing Algorithm-

This algorithm is used for scan converting a line.

It was developed by Bresenham. It is an efficient method because it involves only **integer addition, subtractions, and multiplication operations**.

These operations can be performed very rapidly so lines can be generated quickly.

In this method, next pixel selected is that one who has the least distance from true line.

Procedure-

Given-

- Starting coordinates = (X_0, Y_0)
- Ending coordinates = (X_n, Y_n)
- The points generation using Bresenham Line Drawing Algorithm involves the following steps-
- Step-01:
- Calculate ΔX and ΔY from the given input.

These parameters are calculated as-

- $\Delta X = X_n – X_0$
- $\Delta Y = Y_n – Y_0$

Step-02:

Calculate the decision parameter P_k.

It is calculated as-

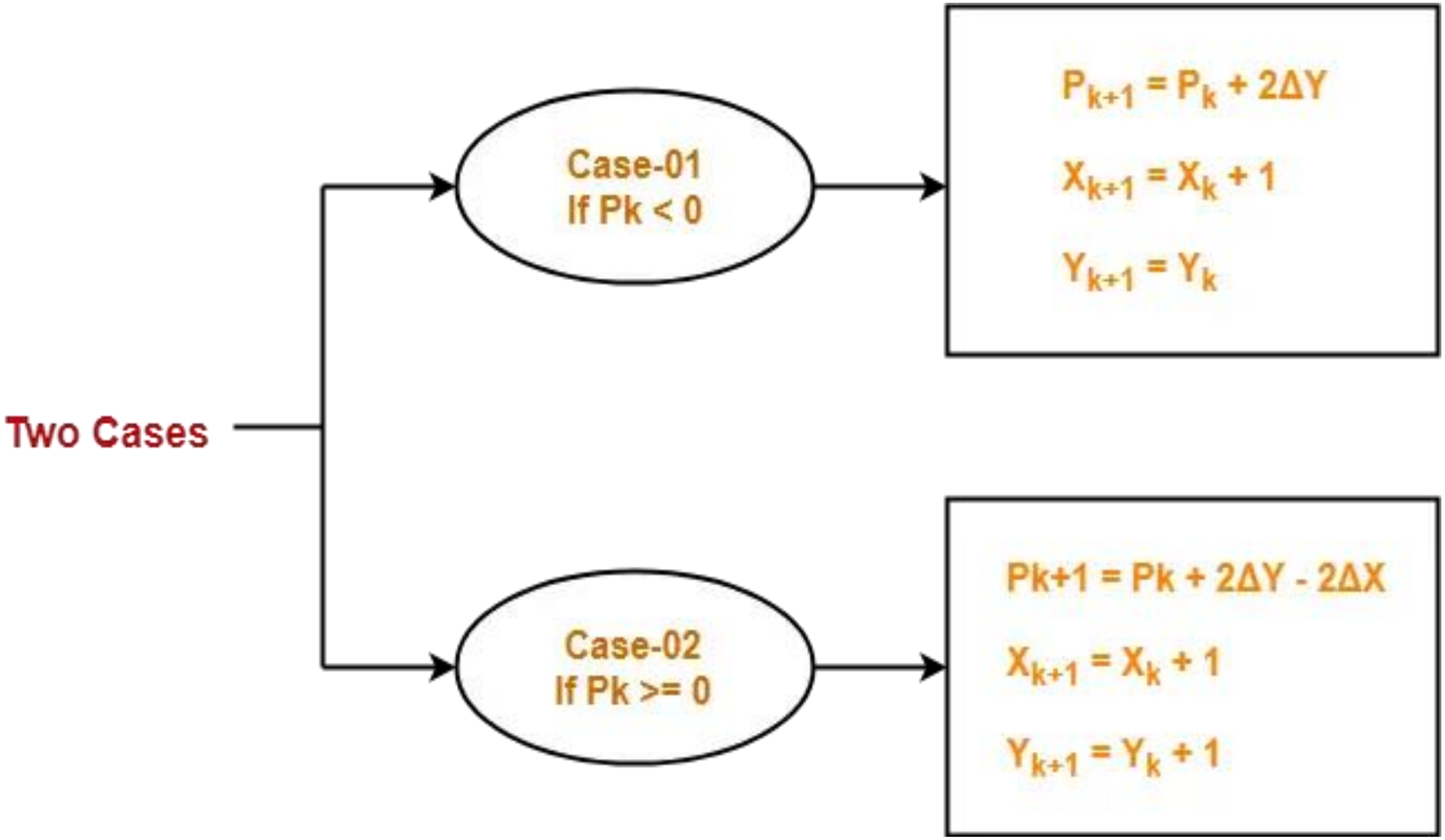
$$P_k = 2\Delta Y – \Delta X$$

Step-03:

Suppose the current point is (X_k, Y_k) and the next point is (X_{k+1}, Y_{k+1}).

Find the next point depending on the value of decision parameter P_k.

Follow the below two cases-



Step-04:

Keep repeating Step-03 until the end point is reached or number of iterations equals to (ΔX-1) times.

Example1:

Calculate the points between the starting coordinates (9, 18) and ending coordinates (14, 22).

Solution-

Given-

- Starting coordinates = (X_0, Y_0) = (9, 18)
- Ending coordinates = (X_n, Y_n) = (14, 22)

Step-01:

Calculate ΔX and ΔY from the given input.

- $\Delta X = X_n – X_0 = 14 – 9 = 5$
- $\Delta Y = Y_n – Y_0 = 22 – 18 = 4$

Step-02:

Calculate the decision parameter.

$P_k = 2\Delta Y – \Delta X$

$= 2 \times 4 – 5$

$= 3$

So, decision parameter $P_k = 3$

Step-03:

As P_k >= 0, so case-02 is satisfied.

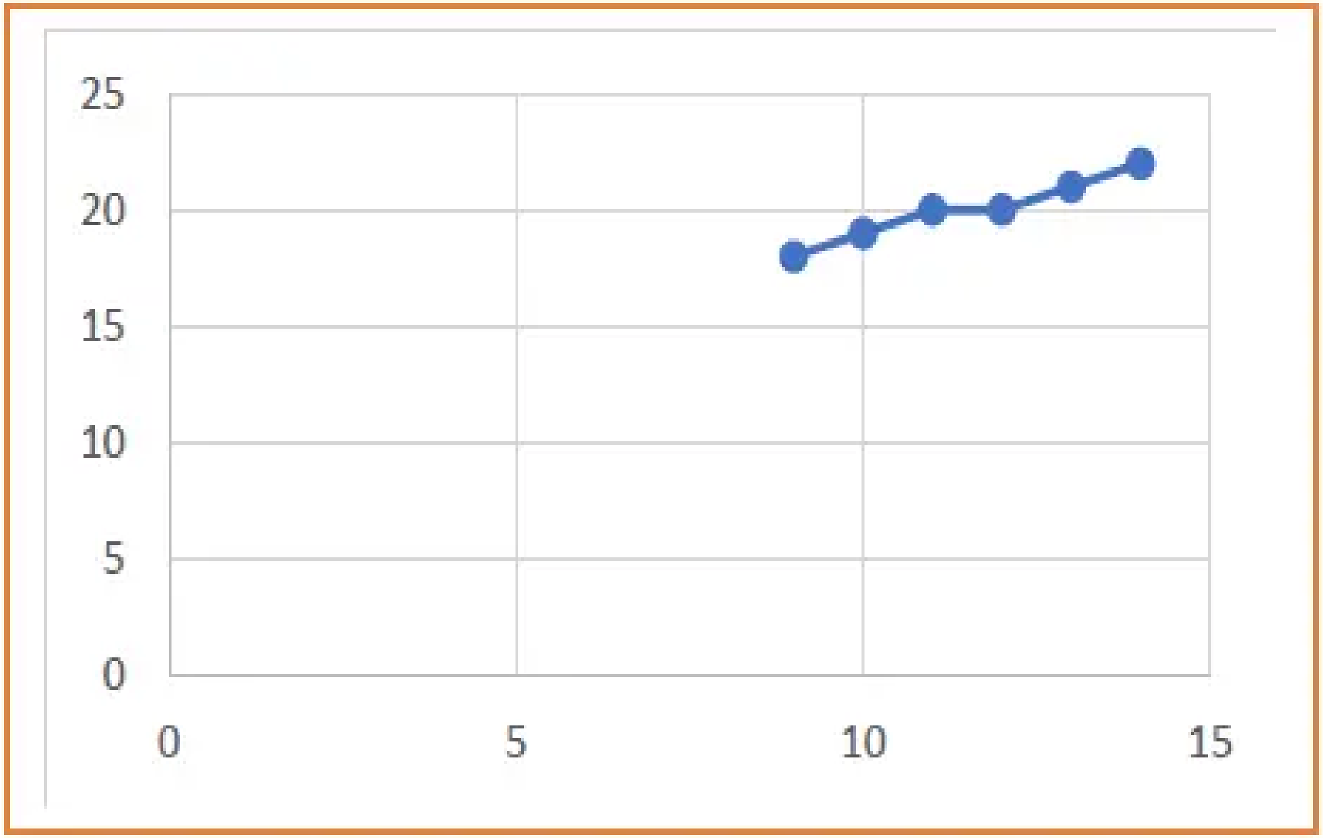
Thus,

- $P_{k+1} = P_k + 2\Delta Y – 2\Delta X = 3 + (2 \times 4) – (2 \times 5) = 1$
- $X_{k+1} = X_k + 1 = 9 + 1 = 10$
- $Y_{k+1} = Y_k + 1 = 18 + 1 = 19$

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

(Number 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	12	20
7	5	13	21
5	3	14	22


25
20
15
10
5
0
0
5
10
15

Example2:

Calculate the points between the starting coordinates (20, 10) and ending coordinates (30, 18).

Solution-

Given-

- Starting coordinates = (X_0, Y_0) = (20, 10)
- Ending coordinates = (X_n, Y_n) = (30, 18)

Step-01:

Calculate ΔX and ΔY from the given input.

- $\Delta X = X_n - X_0 = 30 - 20 = 10$
- $\Delta Y = Y_n - Y_0 = 18 - 10 = 8$

Step-02:

Calculate the decision parameter.

$P_k = 2\Delta Y – \Delta X$

$= 2 \times 8 – 10$

$= 6$

So, decision parameter $P_k = 6$

Step-03:

As $P_k >= 0$, so case-02 is satisfied.

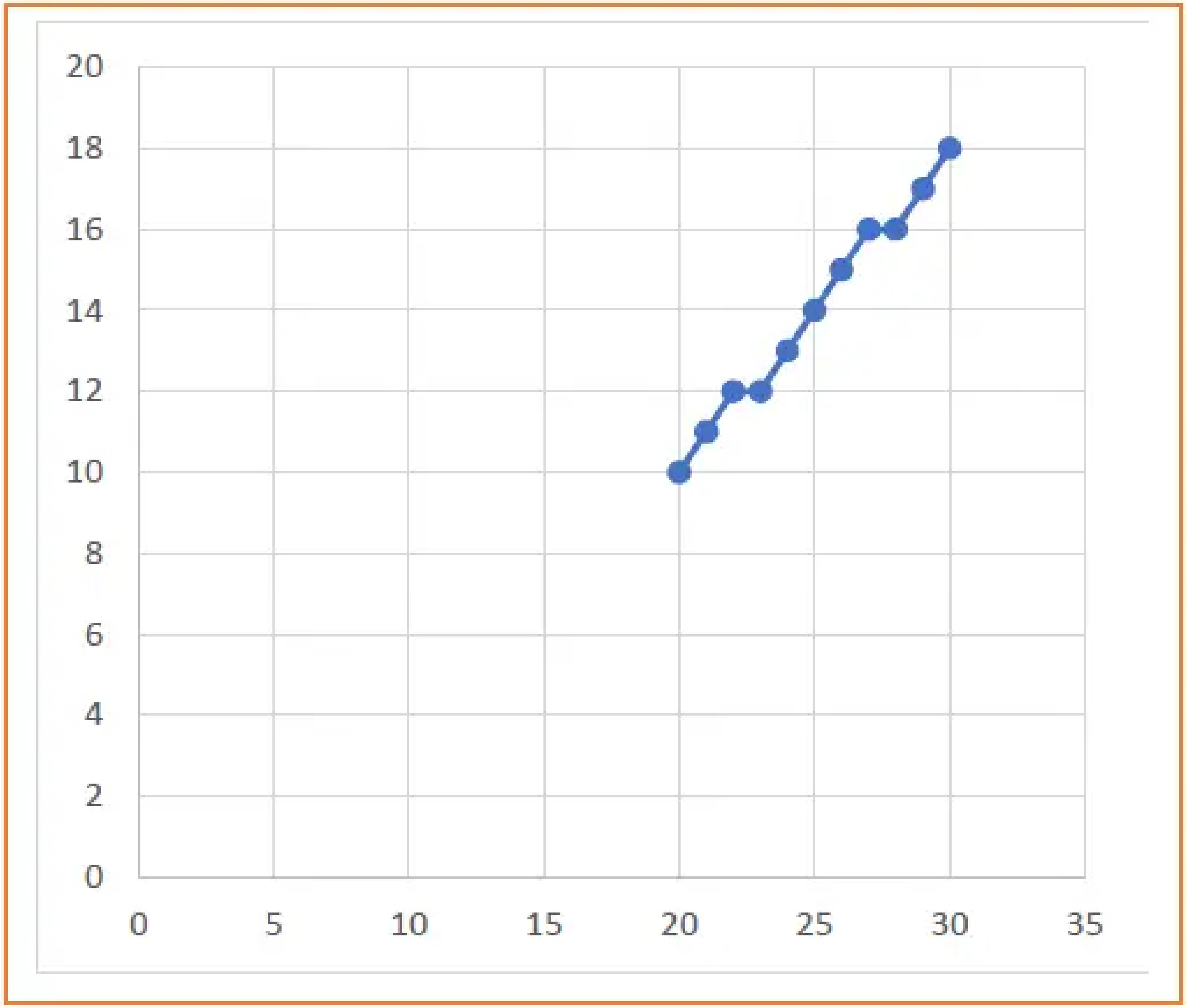
Thus,

- $P_{k+1} = P_k + 2\Delta Y – 2\Delta X = 6 + (2 \times 8) – (2 \times 10) = 2$
- $X_{k+1} = X_k + 1 = 20 + 1 = 21$
- $Y_{k+1} = Y_k + 1 = 10 + 1 = 11$

Similarly, Step-03 is executed until the end point is reached or number of iterations equals to 9 times.

(Number of iterations = ΔX – 1 = 10 – 1 = 9)

P_k	P_{k+1}	X_{k+1}	Y_{k+1}
		20	10
6	2	21	11
2	-2	22	12
-2	14	23	12
14	10	24	13
10	6	25	14
6	2	26	15
2	-2	27	16
-2	14	28	16
14	10	29	17
10	6	30	18


20
18
16
14
12
10
8
6
4
2
0
0
5
10
15
20
25
30
35

Advantages of Bresenham Line Drawing Algorithm-

The advantages of Bresenham Line Drawing Algorithm are-

- It is easy to implement. It is fast and incremental.
- It executes fast but less faster than DDA Algorithm.
- The points generated by this algorithm are more accurate than DDA Algorithm.
- It uses fixed points only.

- **Disadvantages of Bresenham Line Drawing Algorithm-**
- Though it improves the accuracy of generated points but still the resulted line is not smooth.
- This algorithm is for the basic line drawing.
- It can not handle diminishing jaggies.

Differentiate between DDA Algorithm and Bresenham's Line Algorithm

DDA Algorithm	Bresenham's Line Algorithm
1. DDA Algorithm use floating point, i.e., Real Arithmetic.	1. Bresenham's Line Algorithm use fixed point, i.e., Integer Arithmetic
2. DDA Algorithms uses multiplication & division its operation	2.Bresenham's Line Algorithm uses only subtraction and addition its operation
3. DDA Algorithm is slowly than Bresenham's Line Algorithm in line drawing because it uses real arithmetic (Floating Point operation)	3. Bresenham's Algorithm is faster than DDA Algorithm in line because it involves only addition & subtraction in its calculation and uses only integer arithmetic.
4. DDA Algorithm is not accurate and efficient as Bresenham's Line Algorithm.	4. Bresenham's Line Algorithm is more accurate and efficient at DDA Algorithm.
5.DDA Algorithm can draw circle and curves but are not accurate as Bresenham's Line Algorithm	5. Bresenham's Line Algorithm can draw circle and curves with more accurate than DDA Algorithm.

Assignment 3

Write a program to implement Bresenham's Line Drawing Algorithm. The coordinate of the points are: (x1, y1), (x2, y2) = (100, 100), (200, 200) respectively.

Also, display the generated pts in tabular form.

Mid Point Line Drawing Algorithm-

Procedure-

Given-

- Starting coordinates = (X_0, Y_0)
- Ending coordinates = (X_n, Y_n)

The points generation using Mid Point Line Drawing Algorithm involves the following steps-

Step-01:

Calculate ΔX and ΔY from the given input.

These parameters are calculated as-

- $\Delta X = X_n – X_0$

$\Delta Y = Y_n – Y_0$

Step-02:

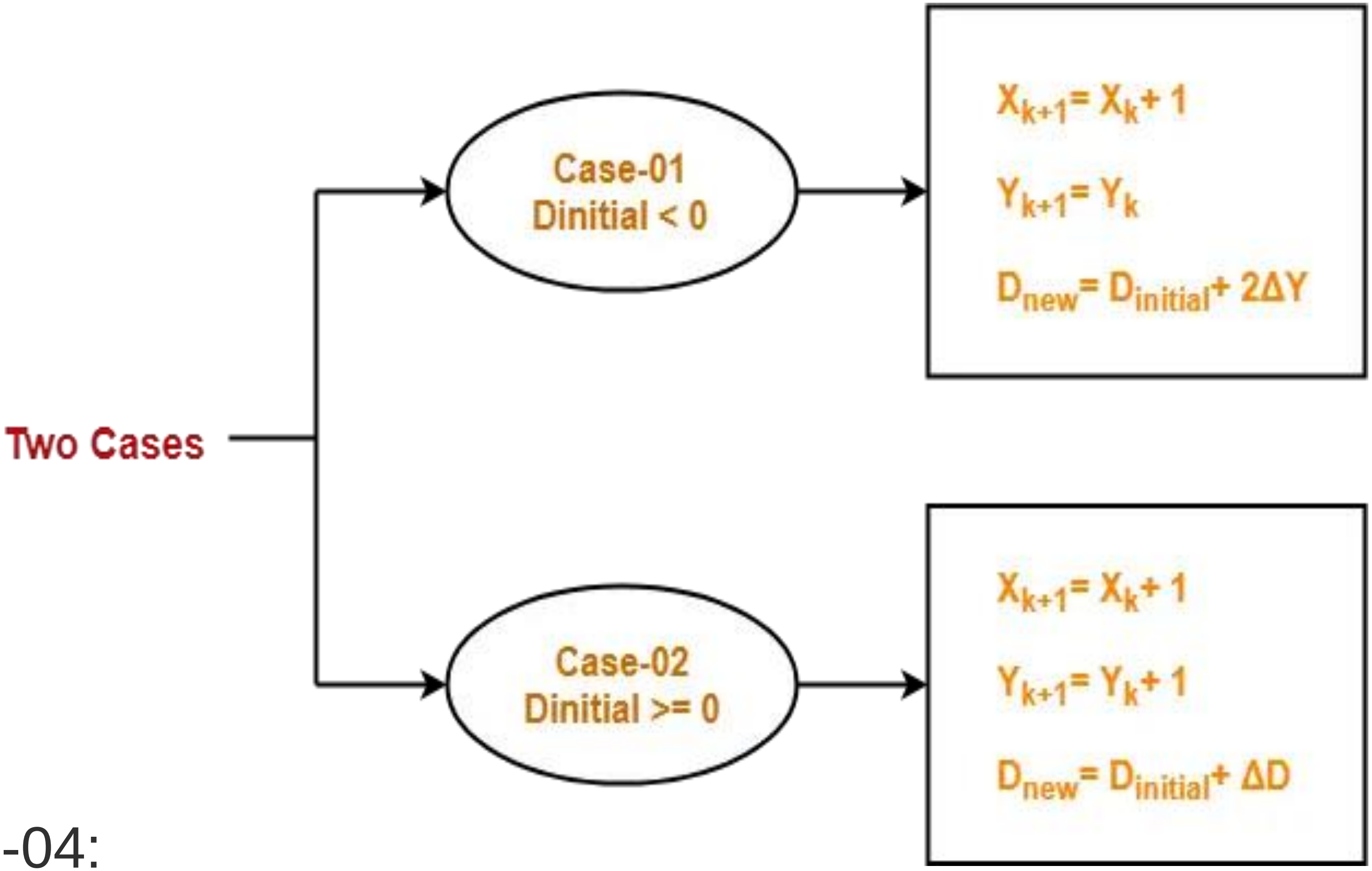
Calculate the value of initial decision parameter and ΔD.

These parameters are calculated as-

- $D_{initial} = 2\Delta Y – \Delta X$
- $\Delta D = 2(\Delta Y – \Delta X)$

Step-03: **The decision whether to increment X or Y coordinate depends upon the flowing values of $D_{initial}$.**

Follow the below two cases-



Two Cases

Case-01
Dinitial < 0

$X_{k+1} = X_k + 1$

$Y_{k+1} = Y_k$

$D_{new} = D_{initial} + 2\Delta Y$

Case-02
Dinitial >= 0

$X_{k+1} = X_k + 1$

$Y_{k+1} = Y_k + 1$

$D_{new} = D_{initial} + \Delta D$

Step-04:

Keep repeating Step-03 until the end point is reached.

For each D_{new} value, follow the above cases to find the next coordinates.

Example1:

Calculate the points between the starting coordinates (20, 10) and ending coordinates (30, 18).

Solution-

Given-

- **Starting coordinates = (X_0, Y_0) = (20, 10)**
- **Ending coordinates = (X_n, Y_n) = (30, 18)**

Step-01:

Calculate ΔX and ΔY from the given input.

- $\Delta X = X_n – X_0 = 30 – 20 = 10$
- $\Delta Y = Y_n – Y_0 = 18 – 10 = 8$

Step-02:

Calculate $D_{initial}$ and ΔD as-

- $D_{initial} = 2\Delta Y – \Delta X = 2 \times 8 – 10 = 6$
- $\Delta D = 2(\Delta Y – \Delta X) = 2 \times (8 – 10) = -4$

Step-03:

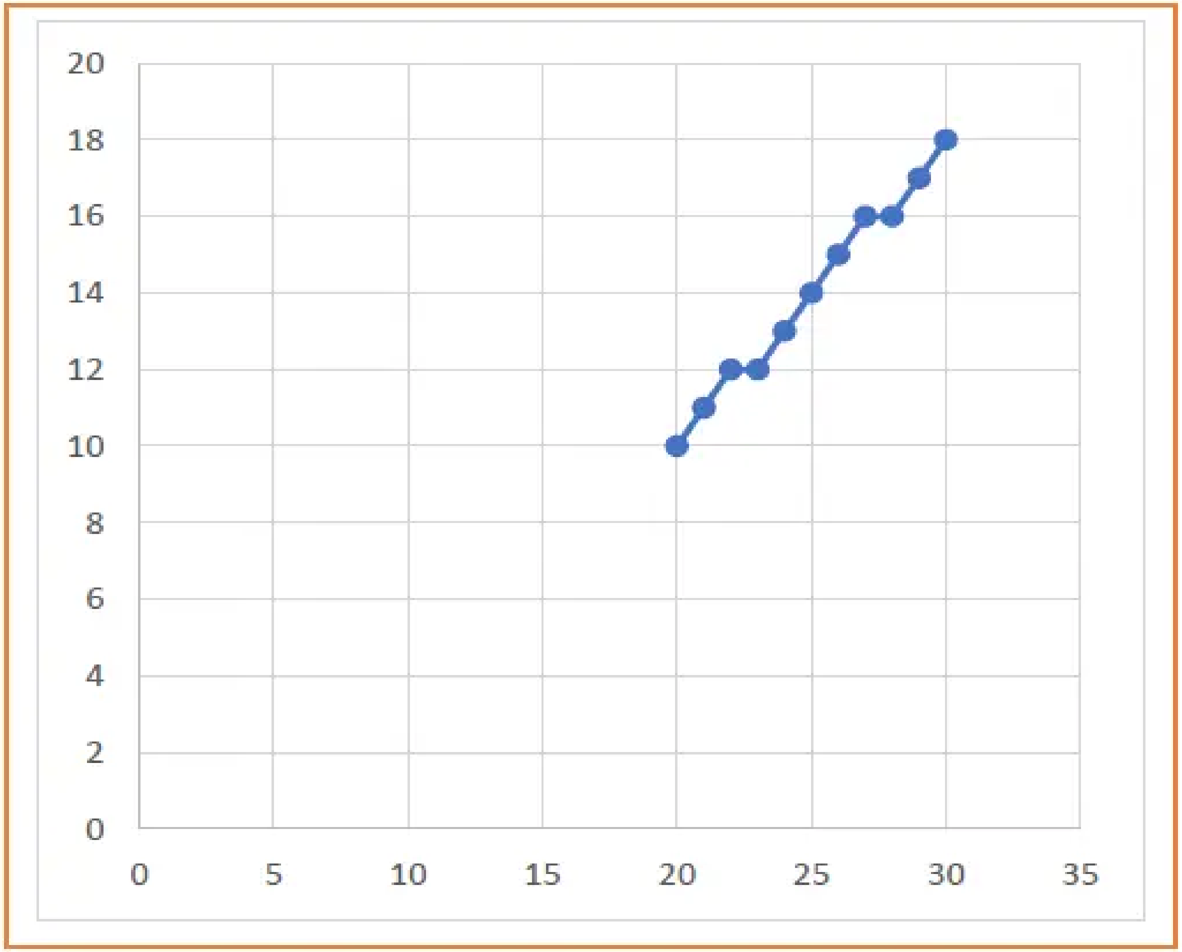
As $D_{initial}$ >= 0, so case-02 is satisfied.

Thus,

- $\mathbf{X_{k+1} = X_k + 1 = 20 + 1 = 21}$
- $\mathbf{Y_{k+1} = Y_k + 1 = 10 + 1 = 11}$
- $\mathbf{D_{new} = D_{initial} + \Delta D = 6 + (-4) = 2}$

Similarly, Step-03 is executed until the end point is reached.

$D_{initial}$	D_{new}	X_{k+1}	Y_{k+1}
		20	10
6	2	21	11
2	-2	22	12
-2	14	23	12
14	10	24	13
10	6	25	14
6	2	26	15
2	-2	27	16
-2	14	28	16
14	10	29	17
10		30	18


20
18
16
14
12
10
8
6
4
2
0
0
5
10
15
20
25
30
35

Example2:

Calculate the points between the starting coordinates (5, 9) and ending coordinates (12, 16).

Solution-

Given-

- Starting coordinates = (X_0, Y_0) = (5, 9)
- Ending coordinates = (X_n, Y_n) = (12, 16)

Step-01:

Calculate ΔX and ΔY from the given input.

- $\Delta X = X_n - X_0 = 12 - 5 = 7$
- $\Delta Y = Y_n - Y_0 = 16 - 9 = 7$

Step-02:

Calculate $D_{initial}$ and ΔD as-

- $D_{initial} = 2\Delta Y – \Delta X = 2 \times 7 – 7 = 7$
- $\Delta D = 2(\Delta Y – \Delta X) = 2 \times (7 – 7) = 0$

Step-03:

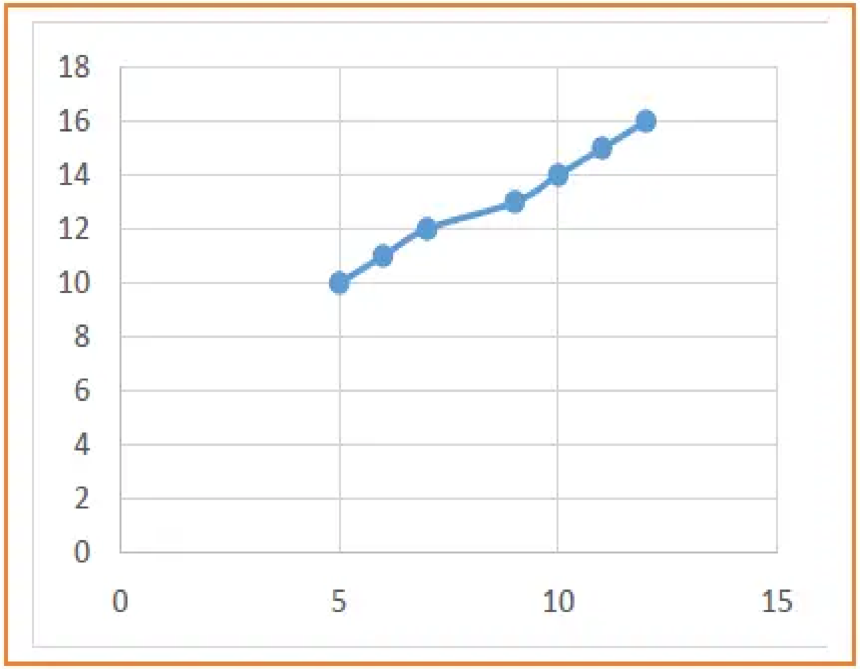
As $D_{initial}$ >= 0, so case-02 is satisfied.

Thus,

- $X_{k+1} = X_k + 1 = 5 + 1 = 6$
- $Y_{k+1} = Y_k + 1 = 9 + 1 = 10$
- $D_{new} = D_{initial} + \Delta D = 7 + 0 = 7$

Similarly, Step-03 is executed until the end point is reached.

$D_{initial}$	D_{new}	X_{k+1}	Y_{k+1}
		5	9
7	7	6	10
7	7	7	11
7	7	8	12
7	7	9	13
7	7	10	14
7	7	11	15
7		12	16



Advantages of Mid Point Line Drawing Algorithm-

The advantages of Mid Point Line Drawing Algorithm are-

- Accuracy of finding points is a key feature of this algorithm.
- It is simple to implement.
- It uses basic arithmetic operations.
- It takes less time for computation.
- The resulted line is smooth as compared to other line drawing algorithms.

Disadvantages of Mid Point Line Drawing Algorithm-

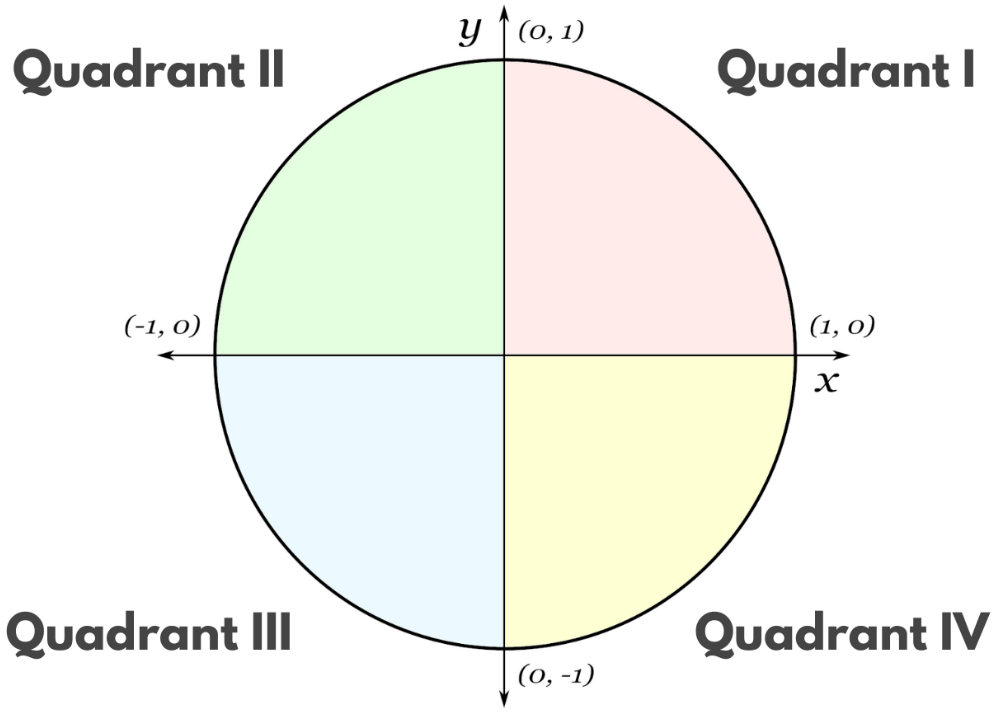
The disadvantages of Mid Point Line Drawing Algorithm are-

- This algorithm may not be an ideal choice for complex graphics and images.
- In terms of accuracy of finding points, improvement is still needed.
- There is no any remarkable improvement made by this algorithm.

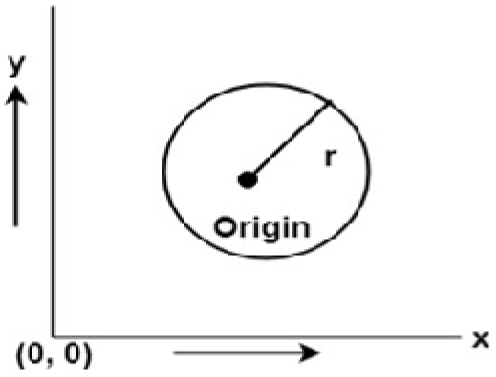
Defining a Circle:

- 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.

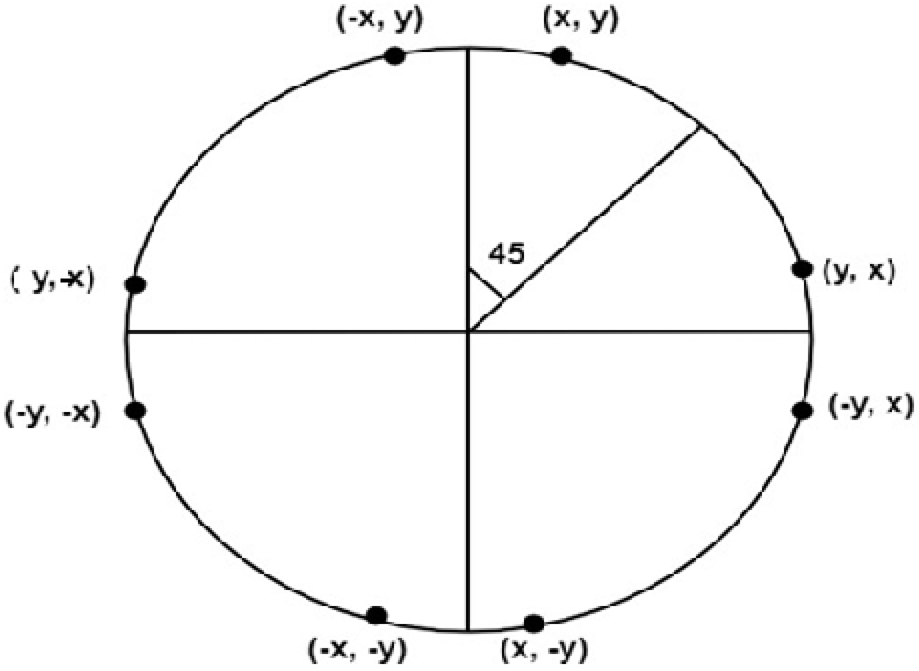
- Quadrant defines as the **four quarters** in the coordinate plane system.
- Each of the four sections is called a quadrant.
- The quarter of a circle is called a quadrant, which is a sector of 90 degrees.



- 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.

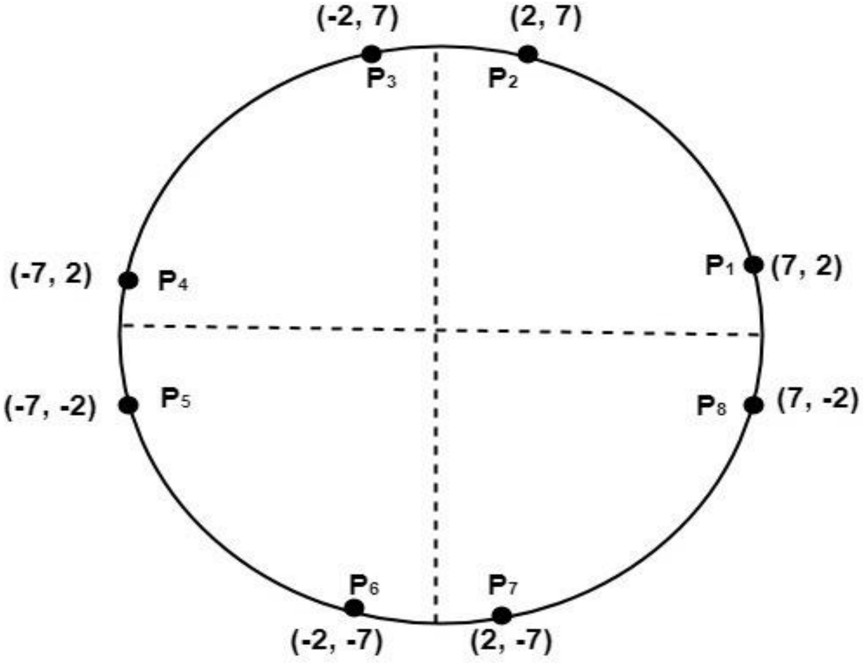
If we want to display circle on screen then the putpixel function is used for eight points as shown below:

```
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:

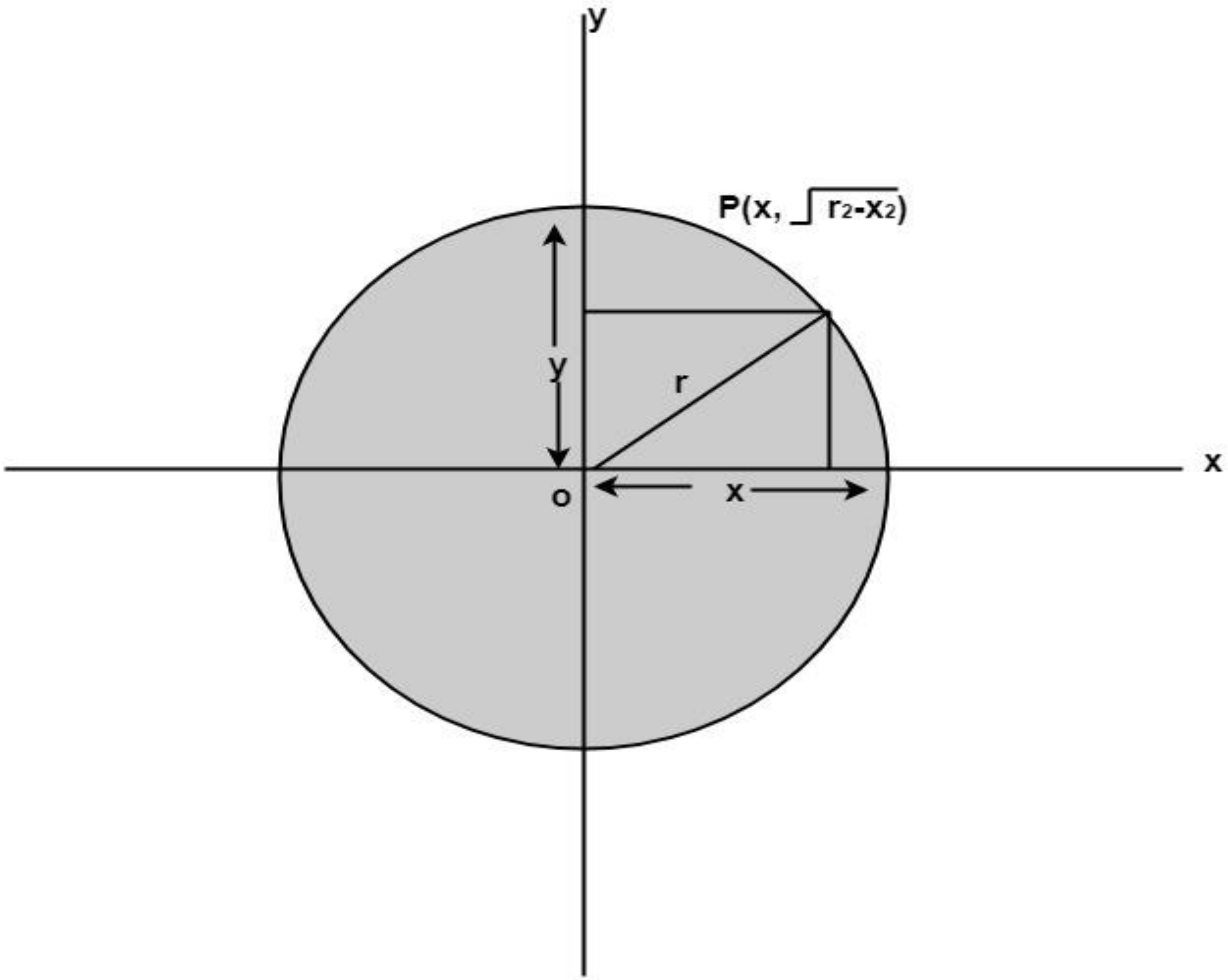
$$y^2=r^2-x^2$$

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.


y
P(x, √r2-x2)
y
r
x
o
x

Algorithm:

Step1: Set the initial variables

r = circle radius

(h, k) = coordinates of circle center

x=o

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) 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 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:

$$x=r\cos\theta \qquad y = r\sin\theta$$

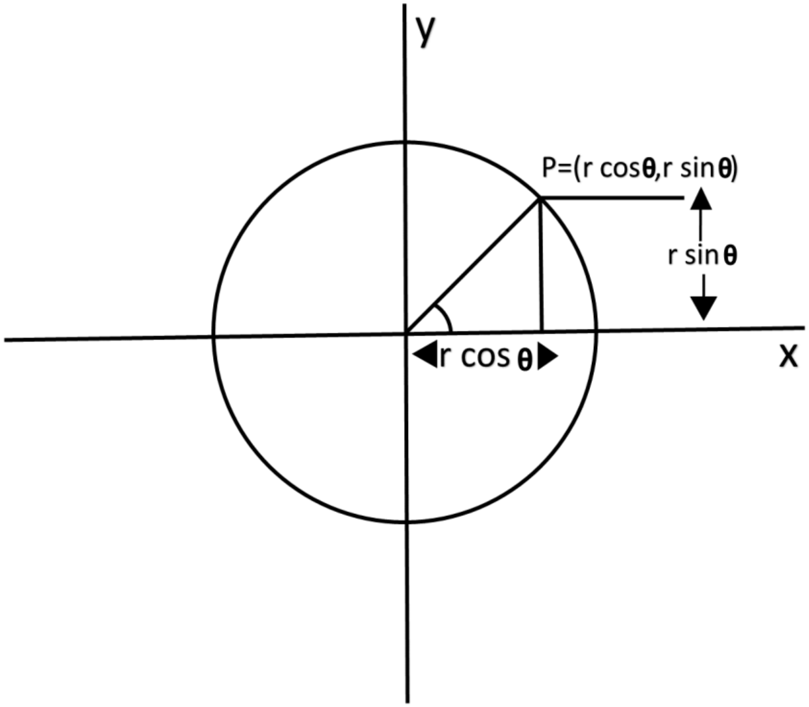
Where θ=current angle

r = circle radius

x = x coordinate

y = y coordinate

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


y
P=(r cosθ,r sinθ)
r sinθ
r cos θ
x

Algorithm:

Step1: Set the initial variables:

r = circle radius
(h, k) = coordinates of the circle center
i = step size
θ_end= $(\frac{22}{7})/4$

θ=0

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

Step3: Compute

$x = r * \cos\theta$ $y = r * \sin\theta$

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

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

Step5: Increment θ=θ+i

Step6: Go to step (ii).

Mid Point Circle Drawing Algorithm

Given the centre point and radius of circle,

Mid Point Circle Drawing Algorithm attempts to generate the points of one octant.

The points for other octacts are generated using the eight symmetry property.

Procedure-

Given-

- Centre point of Circle = (X_0, Y_0)
- Radius of Circle = R

The points generation using Mid Point Circle Drawing Algorithm involves the following steps-

Step-01:

- Assign the starting point coordinates (X_0, Y_0) as-

$X_0 = 0$

- $Y_0 = R$

Step-02:

Calculate the value of initial decision parameter P_0 as-

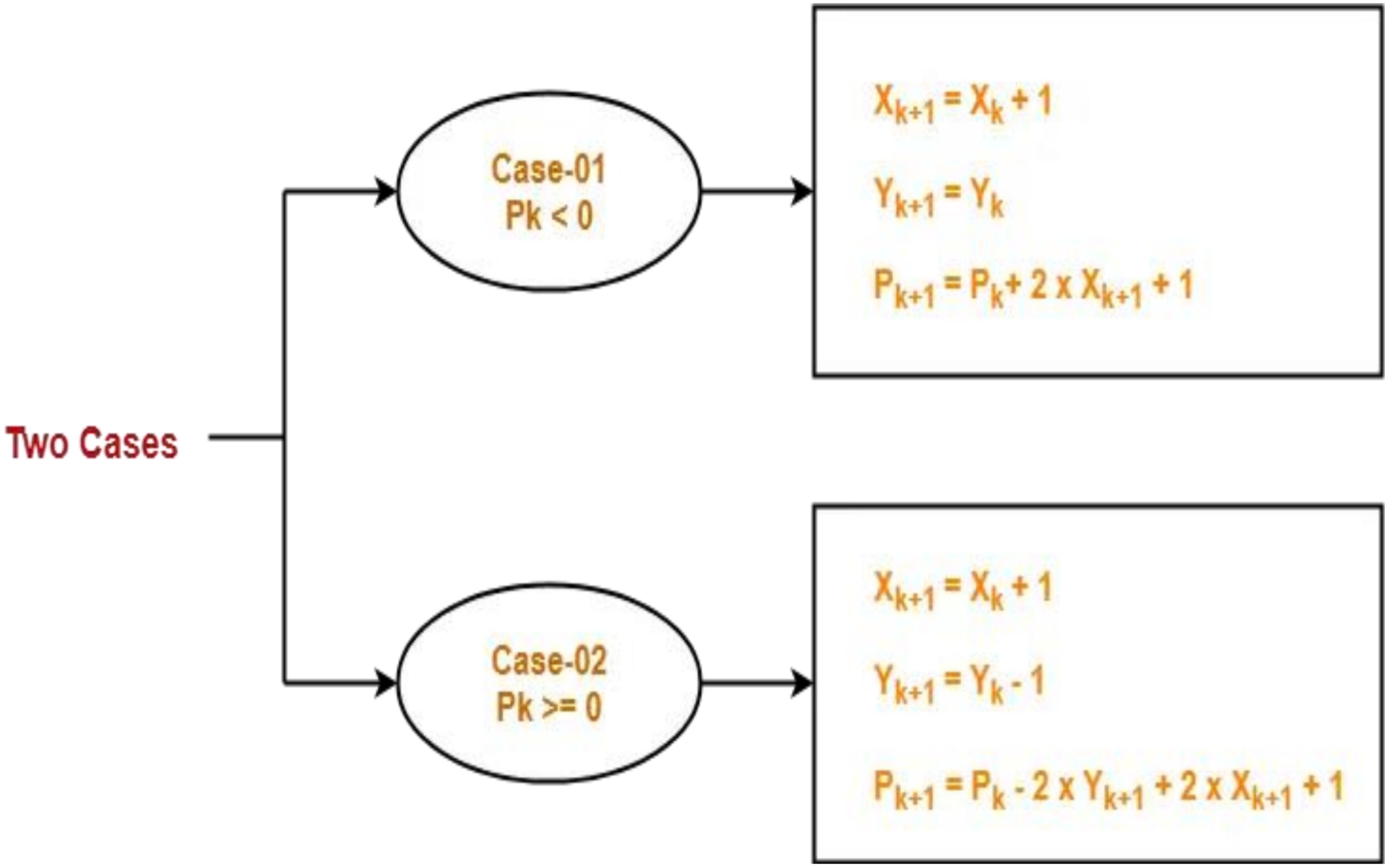
$$P_0 = 1 – R$$

Step-03:

Suppose the current point is (X_k, Y_k) and the next point is (X_{k+1}, Y_{k+1}).

Find the next point of the first octant depending on the value of decision parameter P_k.

Follow the below two cases-



Two Cases

Case-01
Pk < 0

$X_{k+1} = X_k + 1$

$Y_{k+1} = Y_k$

$P_{k+1} = P_k + 2 \text{ x } X_{k+1} + 1$

Case-02
Pk >= 0

$X_{k+1} = X_k + 1$

$Y_{k+1} = Y_k - 1$

$P_{k+1} = P_k - 2 \text{ x } Y_{k+1} + 2 \text{ x } X_{k+1} + 1$

Step-04:

If the given centre point (X_0, Y_0) is not (0, 0), then do the following and plot the point-

- $X_{plot} = X_c + X_0$
- $Y_{plot} = Y_c + Y_0$

Here, (X_c, Y_c) denotes the current value of X and Y coordinates.

Step-05:

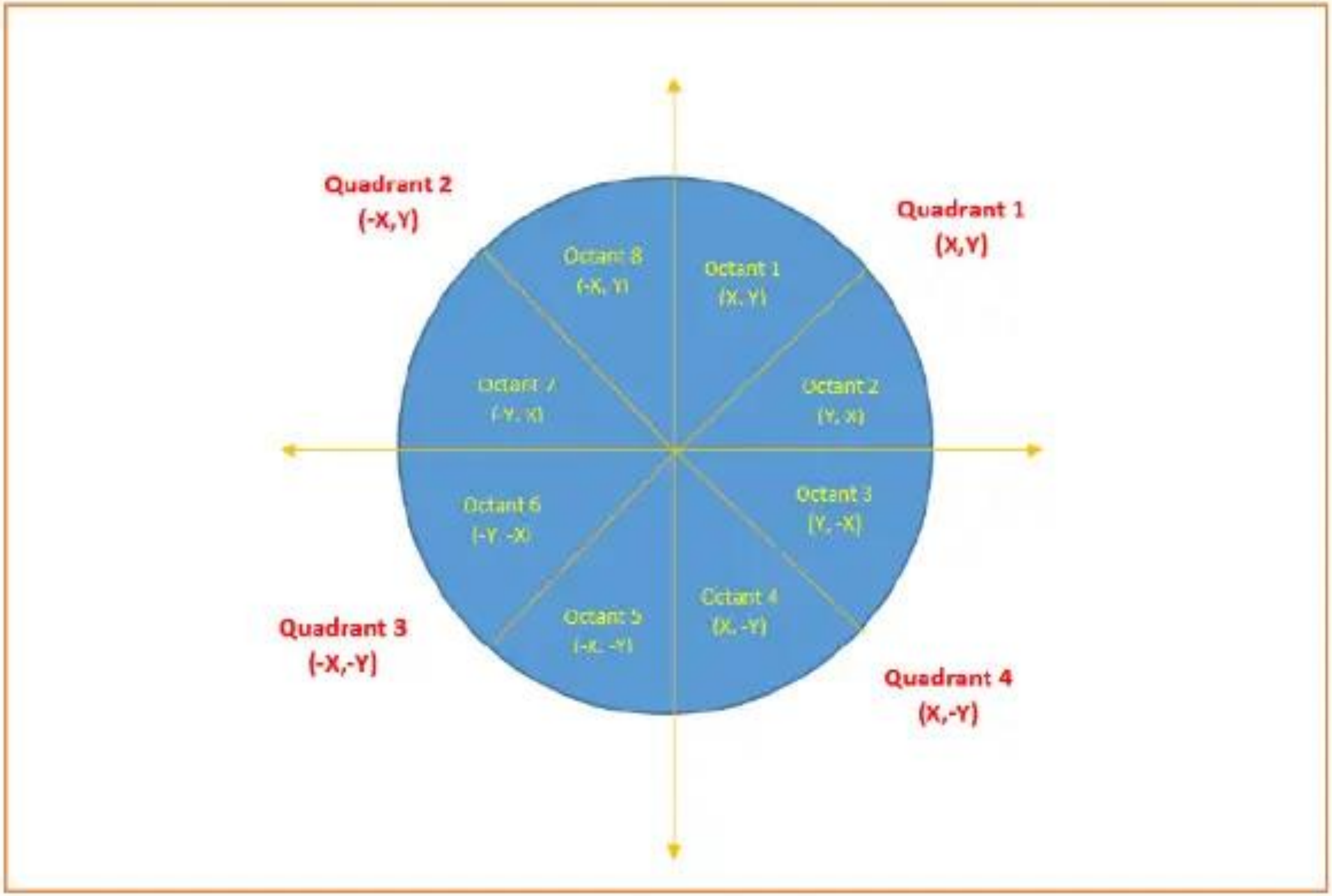
Keep repeating Step-03 and Step-04 until $X_{plot} >= Y_{plot}$.

Step-06:

Step-05 generates all the points for one octant.

To find the points for other seven octants, follow the eight symmetry property of circle.

This is depicted by the following figure-


Quadrant 2
(-X,Y)
Quadrant 1
(X,Y)
Octant 8
(-X, Y)
Octant 1
(X, Y)
Octant 7
(-Y, X)
Octant 2
(Y, X)
Octant 6
(-Y, -X)
Octant 3
(Y, -X)
Octant 5
(-X, -Y)
Octant 4
(X, -Y)
Quadrant 3
(-X,-Y)
Quadrant 4
(X,-Y)

Example1:

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

Solution

Given-

- Centre Coordinates of Circle $(X_0, Y_0) = (0, 0)$
- Radius of Circle = 10

Step-01:

Assign the starting point coordinates (X_0, Y_0) as-

- $X_0 = 0$
- $Y_0 = R = 10$

Step-02:

Calculate the value of initial decision parameter P_0 as-

$P_0 = 1 – R$

$P_0 = 1 – 10$

$P_0 = -9$

Step-03:

As $P_{initial} < 0$, so case-01 is satisfied.

Therefore,

- $X_{k+1} = X_k + 1 = 0 + 1 = 1$
- $Y_{k+1} = Y_k = 10$
- $P_{k+1} = P_k + 2 \text{ x } X_{k+1} + 1 = -9 + (2 \text{ x } 1) + 1 = -6$

Step-04:

This step is not applicable here as the given centre point coordinates is (0, 0).

Step-05:

Step-03 is executed similarly until $X_{k+1} >= Y_{k+1}$ 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.		

Algorithm calculates all the points of octant-1 and terminates.

Now, the points of octant-2 are obtained using the mirror effect by swapping X and Y coordinates.

Octant-1 Points	Octant-2 Points
(0, 10)	(8, 6)
(1, 10)	(9, 5)
(2, 10)	(9, 4)
(3, 10)	(10, 3)
(4, 9)	(10, 2)
(5, 9)	(10, 1)
(6, 8)	(10, 0)
These are all points for Quadrant-1.	

- The points for rest of the part are generated by following the signs of other quadrants.
- The other points can also be generated by calculating each octant separately.

Therefore, all the points have been generated with respect to quadrant-1-

Quadrant-1 (X,Y)	Quadrant-2 (-X,Y)	Quadrant-3 (-X,-Y)	Quadrant-4 (X,-Y)
(0, 10)	(0, 10)	(0, -10)	(0, -10)
(1, 10)	(-1, 10)	(-1, -10)	(1, -10)
(2, 10)	(-2, 10)	(-2, -10)	(2, -10)
(3, 10)	(-3, 10)	(-3, -10)	(3, -10)
(4, 9)	(-4, 9)	(-4, -9)	(4, -9)
(5, 9)	(-5, 9)	(-5, -9)	(5, -9)
(6, 8)	(-6, 8)	(-6, -8)	(6, -8)
(8, 6)	(-8, 6)	(-8, -6)	(8, -6)
(9, 5)	(-9, 5)	(-9, -5)	(9, -5)
(9, 4)	(-9, 4)	(-9, -4)	(9, -4)
(10, 3)	(-10, 3)	(-10, -3)	(10, -3)
(10, 2)	(-10, 2)	(-10, -2)	(10, -2)
(10, 1)	(-10, 1)	(-10, -1)	(10, -1)
(10, 0)	(-10, 0)	(-10, 0)	(10, 0)
These are all points of the Circle.			

Example2:

Given the centre point coordinates (4, 4) and radius as 10, generate all the points to form a circle.

Solution

Given-

- Centre Coordinates of Circle (X_0, Y_0) = (4, 4)
- Radius of Circle = 10

As stated in the algorithm,

- We first calculate the points assuming the centre coordinates is (0, 0).
- At the end, we translate the circle.

Step-01, Step-02 and Step-03 are already completed in Problem-01.

Now, we find the values of X_{plot} and Y_{plot} using the formula given in Step-04 of the main algorithm.

The following table shows the generation of points for Quadrant-1-

- $X_{plot} = X_c + X_0 = 4 + X_0$
- $Y_{plot} = Y_c + Y_0 = 4 + Y_0$

(X_{k+1}, Y_{k+1})	(X_{plot}, Y_{plot})
(0, 10)	(4, 14)
(1, 10)	(5, 14)
(2, 10)	(6, 14)
(3, 10)	(7, 14)
(4, 9)	(8, 13)
(5, 9)	(9, 13)
(6, 8)	(10, 12)
(8, 6)	(12, 10)
(9, 5)	(13, 9)
(9, 4)	(13, 8)
(10, 3)	(14, 7)
(10, 2)	(14, 6)
(10, 1)	(14, 5)
(10, 0)	(14, 4)
These are all points for Quadrant-1.	

The following table shows the points for all the quadrants-

Quadrant-1 (X,Y)	Quadrant-2 (-X,Y)	Quadrant-3 (-X,-Y)	Quadrant-4 (X,-Y)
(4, 14)	(4, 14)	(4, -6)	(4, -6)
(5, 14)	(3, 14)	(3, -6)	(5, -6)
(6, 14)	(2, 14)	(2, -6)	(6, -6)
(7, 14)	(1, 14)	(1, -6)	(7, -6)
(8, 13)	(0, 13)	(0, -5)	(8, -5)
(9, 13)	(-1, 13)	(-1, -5)	(9, -5)
(10, 12)	(-2, 12)	(-2, -4)	(10, -4)
(12, 10)	(-4, 10)	(-4, -2)	(12, -2)
(13, 9)	(-5, 9)	(-5, -1)	(13, -1)
(13, 8)	(-5, 8)	(-5, 0)	(13, 0)
(14, 7)	(-6, 7)	(-6, 1)	(14, 1)
(14, 6)	(-6, 6)	(-6, 2)	(14, 2)
(14, 5)	(-6, 5)	(-6, 3)	(14, 3)
(14, 4)	(-6, 4)	(-6, 4)	(14, 4)
These are all points of the Circle.			

Advantages of Mid Point Circle Drawing Algorithm

The advantages of Mid Point Circle Drawing Algorithm are-

- It is a powerful and efficient algorithm.
- The entire algorithm is based on the simple equation of circle $X^2 + Y^2 = R^2$.
- It is easy to implement from the programmer's perspective.
- This algorithm is used to generate curves on raster displays.

Disadvantages of Mid Point Circle Drawing Algorithm

The disadvantages of Mid Point Circle Drawing Algorithm are-

- Accuracy of the generating points is an issue in this algorithm.
- The circle generated by this algorithm is not smooth.
- This algorithm is time consuming.

Important Points

Circle drawing algorithms take the advantage of 8 symmetry property of circle.

- Every circle has 8 octants and the circle drawing algorithm generates all the points for one octant.
- The points for other 7 octants are generated by changing the sign towards X and Y coordinates.
- To take the advantage of 8 symmetry property, the circle must be formed assuming that the centre point coordinates is (0, 0).
- If the centre coordinates are other than (0, 0), then we add the X and Y coordinate values with each point of circle with the coordinate values generated by assuming (0, 0) as centre point.

Bresenham Circle Drawing Algorithm

Given the centre point and radius of circle,

Bresenham Circle Drawing Algorithm attempts to generate the points of one octant.

The points for other octants are generated using the eight symmetry property.

Procedure-

Given-

- Centre point of Circle = (X_0, Y_0)
- Radius of Circle = R

The points generation using Bresenham Circle Drawing Algorithm involves the following steps-

Step-01:

Assign the starting point coordinates (X_0, Y_0) as-

- $X_0 = 0$
- $Y_0 = R$

Step-02:

Calculate the value of initial decision parameter P_0 as-

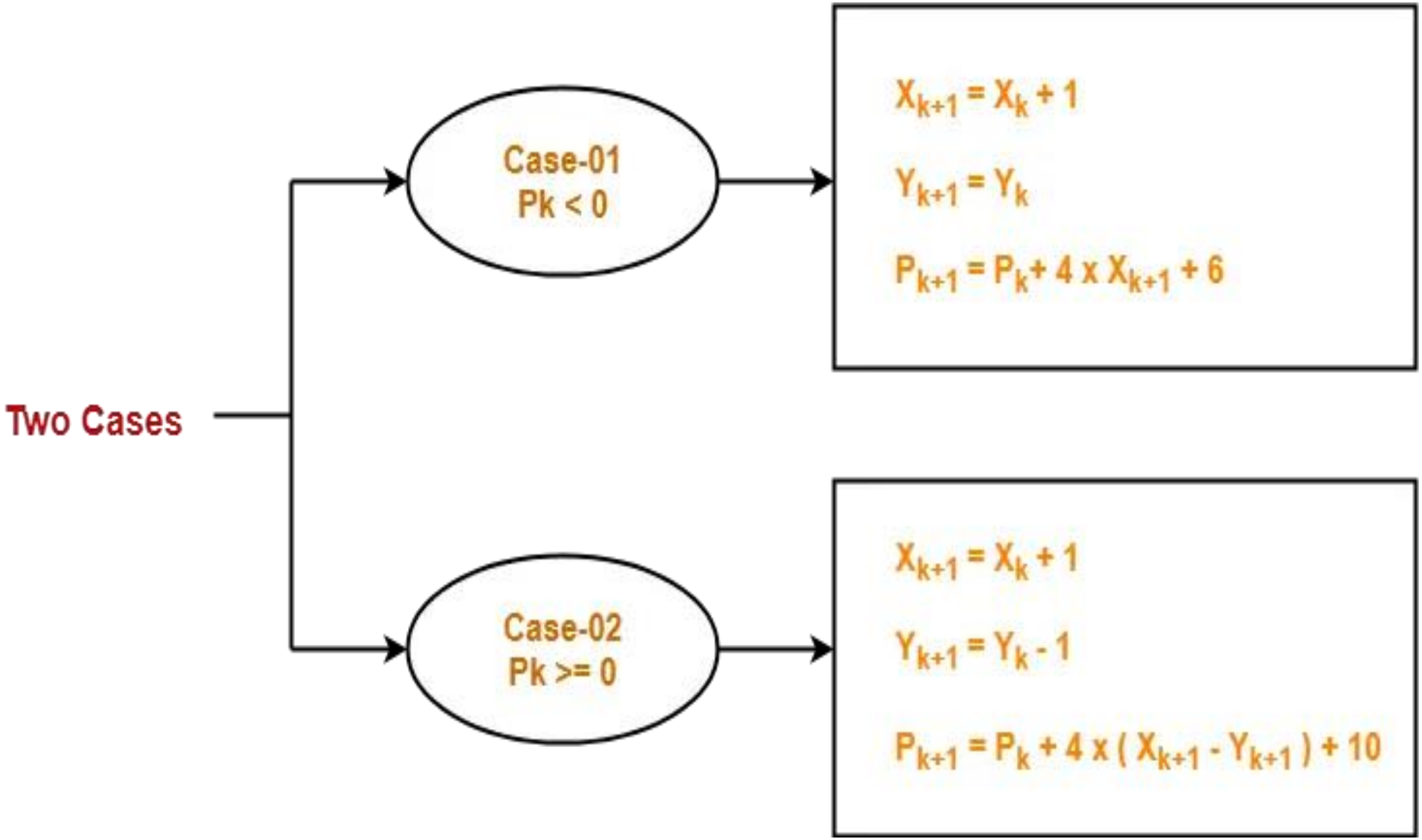
$$P_0 = 3 – 2 x R$$

Step-03:

Suppose the current point is (X_k, Y_k) and the next point is (X_{k+1}, Y_{k+1}).

Find the next point of the first octant depending on the value of decision parameter P_k.

Follow the below two cases:



Two Cases

Case-01
Pk < 0

$X_{k+1} = X_k + 1$

$Y_{k+1} = Y_k$

$P_{k+1} = P_k + 4 \times X_{k+1} + 6$

Case-02
Pk >= 0

$X_{k+1} = X_k + 1$

$Y_{k+1} = Y_k - 1$

$P_{k+1} = P_k + 4 \times (X_{k+1} - Y_{k+1}) + 10$

Step-04:

If the given centre point (X_0, Y_0) is not (0, 0), then do the following and plot the point-

- $X_{plot} = X_c + X_0$
- $Y_{plot} = Y_c + Y_0$

Here, (X_c, Y_c) denotes the current value of X and Y coordinates.

Step-05:

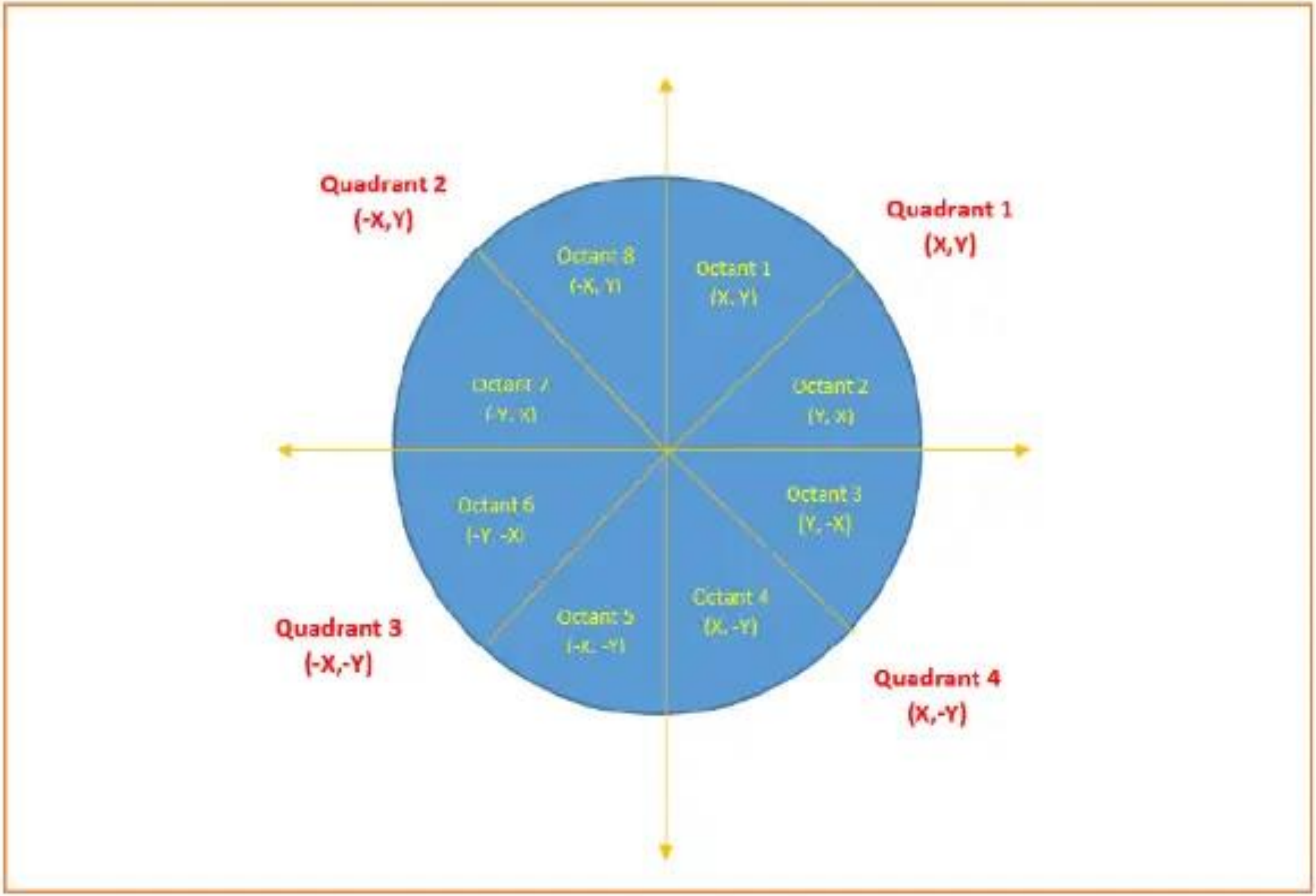
Keep repeating Step-03 and Step-04 until X_{plot} => Y_{plot}.

Step-06:

Step-05 generates all the points for one octant.

To find the points for other seven octants, follow the eight symmetry property of circle.

This is depicted by the following figure-


Quadrant 2
(-X,Y)
Quadrant 1
(X,Y)
Octant 8
(-X, Y)
Octant 1
(X, Y)
Octant 7
(-Y, X)
Octant 2
(Y, X)
Octant 6
(-Y, -X)
Octant 3
(Y, -X)
Octant 5
(-X, -Y)
Octant 4
(X, -Y)
Quadrant 3
(-X,-Y)
Quadrant 4
(X,-Y)

Example1:

Given the centre point coordinates (0, 0) and radius as 8, generate all the points to form a circle.

Solution

Given-

- Centre Coordinates of Circle (X_0, Y_0) = (0, 0)
- Radius of Circle = 8

Step-01:

Assign the starting point coordinates (X_0, Y_0) as-

- $X_0 = 0$
- $Y_0 = R = 8$

Step-02:

Calculate the value of initial decision parameter P_0 as-

$P_0 = 3 – 2 \text{ x } R$

$P_0 = 3 – 2 \text{ x } 8$

$P_0 = -13$

Step-03:

As $P_{initial} < 0$, so case-01 is satisfied.

Thus,

- $X_{k+1} = X_k + 1 = 0 + 1 = 1$
- $Y_{k+1} = Y_k = 8$
- $P_{k+1} = P_k + 4 \text{ x } X_{k+1} + 6 = -13 + (4 \text{ x } 1) + 6 = -3$

Step-04:

This step is not applicable here as the given centre point coordinates is (0, 0).

Step-05:

Step-03 is executed similarly until $X_{k+1} >= Y_{k+1}$ as follows-

P_k	P_{k+1}	(X_{k+1}, Y_{k+1})
		(0, 8)
-13	-3	(1, 8)
-3	11	(2, 8)
11	5	(3, 7)
5	7	(4, 6)
7		(5, 5)

Algorithm Terminates

These are all points for Octant-1.

Algorithm calculates all the points of octant-1 and terminates.

The points of octant-2 are obtained using the mirror effect by swapping X and Y coordinates.

Octant-1 Points	Octant-2 Points
(0, 8)	(5, 5)
(1, 8)	(6, 4)
(2, 8)	(7, 3)
(3, 7)	(8, 2)
(4, 6)	(8, 1)
(5, 5)	(8, 0)
These are all points for Quadrant-1.	

- The points for rest of the part are generated by following the signs of other quadrants.
- The other points can also be generated by calculating each octant separately.

All the points have been generated with respect to quadrant-1 in the table below:

Quadrant-1 (X,Y)	Quadrant-2 (-X,Y)	Quadrant-3 (-X,-Y)	Quadrant-4 (X,-Y)
(0, 8)	(0, 8)	(0, -8)	(0, -8)
(1, 8)	(-1, 8)	(-1, -8)	(1, -8)
(2, 8)	(-2, 8)	(-2, -8)	(2, -8)
(3, 7)	(-3, 7)	(-3, -7)	(3, -7)
(4, 6)	(-4, 6)	(-4, -6)	(4, -6)
(5, 5)	(-5, 5)	(-5, -5)	(5, -5)
(6, 4)	(-6, 4)	(-6, -4)	(6, -4)
(7, 3)	(-7, 3)	(-7, -3)	(7, -3)
(8, 2)	(-8, 2)	(-8, -2)	(8, -2)
(8, 1)	(-8, 1)	(-8, -1)	(8, -1)
(8, 0)	(-8, 0)	(-8, 0)	(8, 0)

These are all points of the Circle.

Example2:

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

Solution

Given-

- Centre Coordinates of Circle $(X_0, Y_0) = (10, 10)$
- Radius of Circle = 10

Step-01:

Assign the starting point coordinates (X_0, Y_0) as-

- $X_0 = 0$
- $Y_0 = R = 10$

Step-02:

Calculate the value of initial decision parameter P_0 as-

$P_0 = 3 – 2 \text{ x } R$

$P_0 = 3 – 2 \text{ x } 10$

$P_0 = -17$

Step-03:

As $P_{initial} < 0$, so case-01 is satisfied.

Thus,

- $X_{k+1} = X_k + 1 = 0 + 1 = 1$
- $Y_{k+1} = Y_k = 10$
- $P_{k+1} = P_k + 4 \text{ x } X_{k+1} + 6 = -17 + (4 \text{ x } 1) + 6 = -7$

Step-04:

This step is applicable here as the given centre point coordinates is (10, 10).

$X_{plot} = X_c + X_0 = 1 + 10 = 11$

$Y_{plot} = Y_c + Y_0 = 10 + 10 = 20$

Step-05:

Step-03 and Step-04 are executed similarly until $X_{plot} => Y_{plot}$ as follows-

P_k	P_{k+1}	(X_{k+1}, Y_{k+1})	(X_{plot}, Y_{plot})
		(0, 10)	(10, 20)
-17	-7	(1, 10)	(11, 20)
-7	7	(2, 10)	(12, 20)
7	-7	(3, 9)	(13, 19)
-7	15	(4, 9)	(14, 19)
15	13	(5, 8)	(15, 18)
13	19	(6, 7)	(16, 17)
Algorithm Terminates **These are all points for Octant-1.**			

Algorithm calculates all the points of octant-1 and terminates.

Now, the points of octant-2 are obtained using the mirror effect by swapping X and Y coordinates.

Octant-1 Points	Octant-2 Points
(10, 20)	(17, 16)
(11, 20)	(18, 15)
(12, 20)	(19, 14)
(13, 19)	(19, 13)
(14, 19)	(20, 12)
(15, 18)	(20, 11)
(16, 17)	(20, 10)
These are all points for Quadrant-1.	

Now, the points for rest of the part are generated by following the signs of other quadrants.

The other points can also be generated by calculating each octant separately.

Here, all the points have been generated with respect to quadrant-1-

Quadrant-1 (X,Y)	Quadrant-2 (-X,Y)	Quadrant-3 (-X,-Y)	Quadrant-4 (X,-Y)
(10, 20)	(10, 20)	(10, 0)	(10, 0)
(11, 20)	(9, 20)	(9, 0)	(11, 0)
(12, 20)	(8, 20)	(8, 0)	(12, 0)
(13, 19)	(7, 19)	(7, 1)	(13, 1)
(14, 19)	(6, 19)	(6, 1)	(14, 1)
(15, 18)	(5, 18)	(5, 2)	(15, 2)
(16, 17)	(4, 17)	(4, 3)	(16, 3)
(17, 16)	(3, 16)	(3, 4)	(17, 4)
(18, 15)	(2, 15)	(2, 5)	(18, 5)
(19, 14)	(1, 14)	(1, 6)	(19, 6)
(19, 13)	(1, 13)	(1, 7)	(19, 7)
(20, 12)	(0, 12)	(0, 8)	(20, 8)
(20, 11)	(0, 11)	(0, 9)	(20, 9)
(20, 10)	(0, 10)	(0, 10)	(20, 10)
These are all points of the Circle.			

Advantages of Bresenham Circle Drawing Algorithm

The advantages of Bresenham Circle Drawing Algorithm are-

The entire algorithm is based on the simple equation of circle $X^2 + Y^2 = R^2$.

- It is easy to implement.

Disadvantages of Bresenham Circle Drawing Algorithm

The disadvantages of Bresenham Circle Drawing Algorithm are-

- Like Mid Point Algorithm, accuracy of the generating points is an issue in this algorithm.
- This algorithm suffers when used to generate complex and high graphical images.
- There is no significant enhancement with respect to performance.