Prof. Jelili O. Oyelade

# 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.

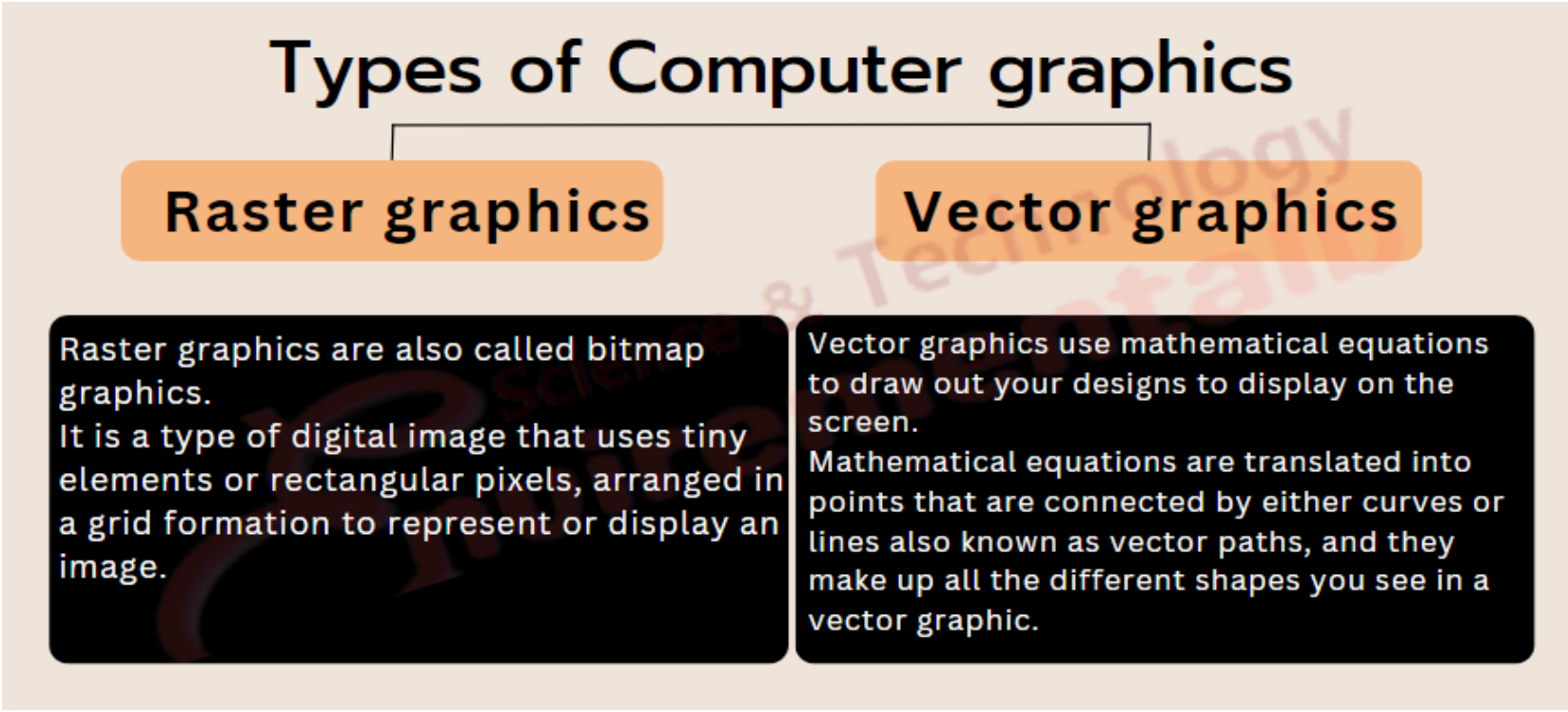
**It is used in digital photography, film and television, video games, and on electronic devices and is responsible for displaying images effectively to users. Think of computer graphics as the intersection of design and computer science, with the purpose of delighting and engaging audiences.**

- **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**

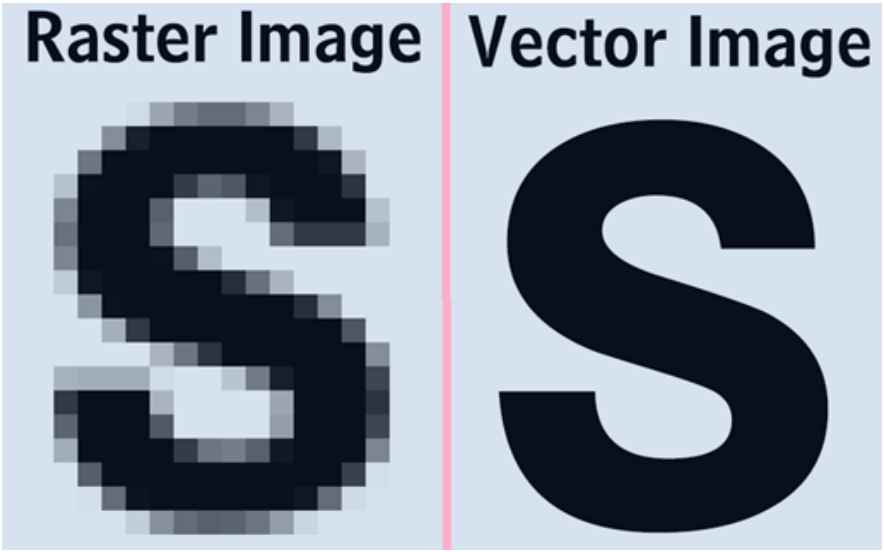


# Raster (Bitmap) Graphics

- **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.

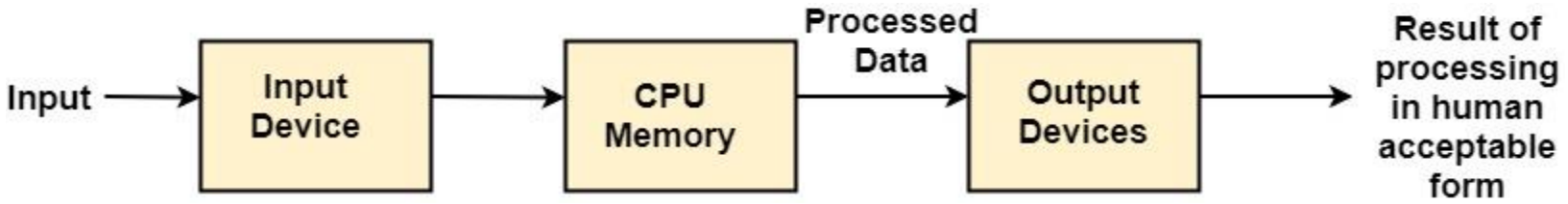
- Interactive Computer Graphics require two-way communication between the computer and the user. A User can see the image and make any change by sending his command with an input device.

# 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.

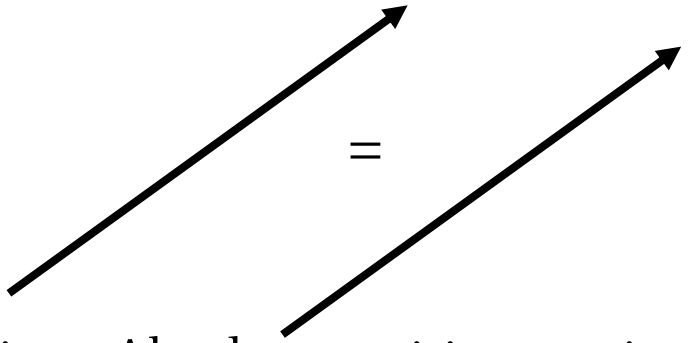
Following are Output Devices:

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

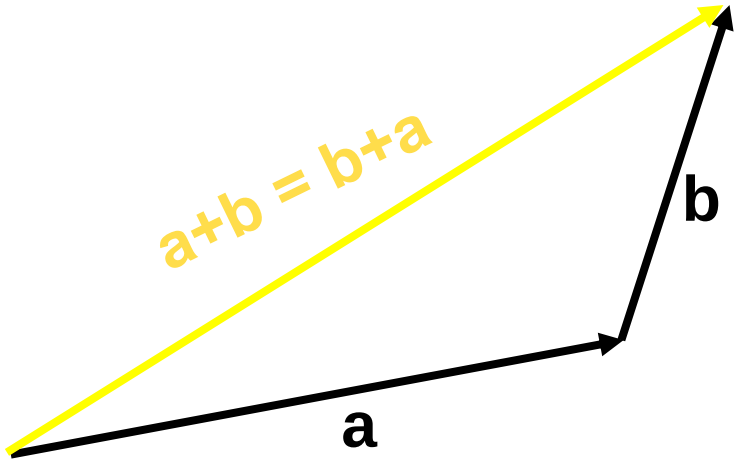



- Length and direction. Absolute position not important

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

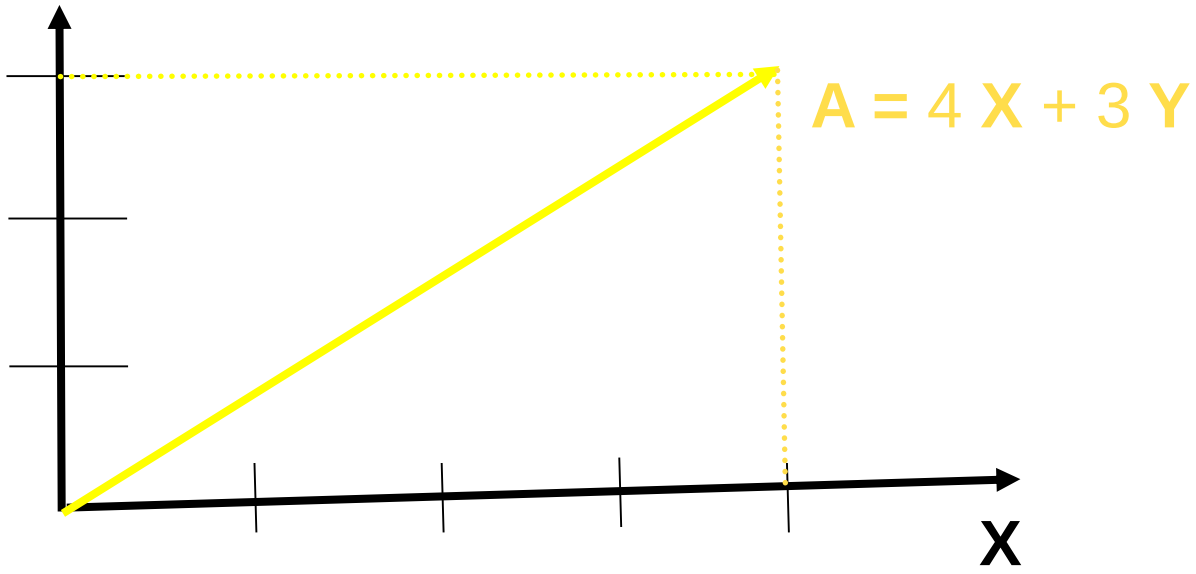
- Use to store offsets, displacements, locations
  - But strictly speaking, positions are not vectors and cannot be added: a location implicitly involves an origin, while an offset does not.

# Vector Addition




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

# Cartesian Coordinates



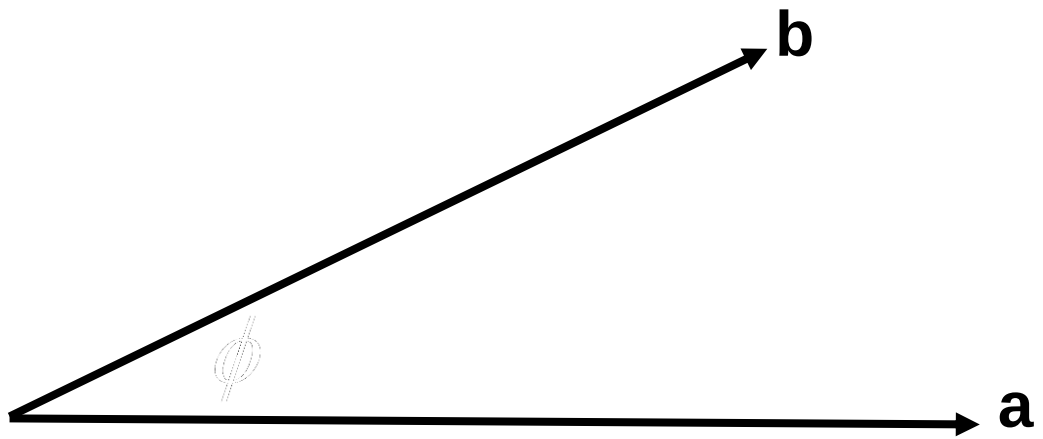

- X and Y can be any (usually orthogonal ***unit***) vectors

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

# Vector Multiplication

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

# Dot (scalar) product




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

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

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

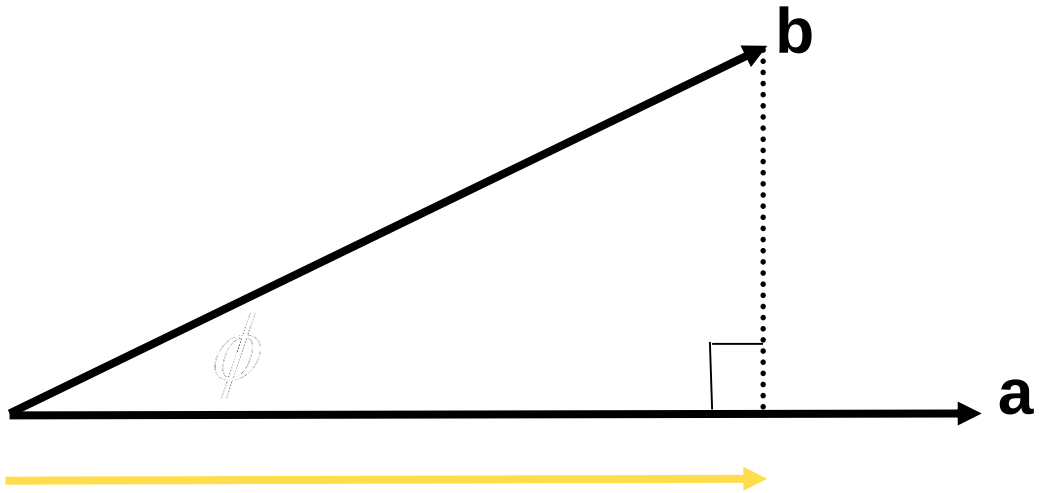
$$a \cdot b = \|a\| \|b\| \cos\phi$$

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

# 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)




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

$$b \to a = ?$$

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

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

# Dot product in Cartesian components

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

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

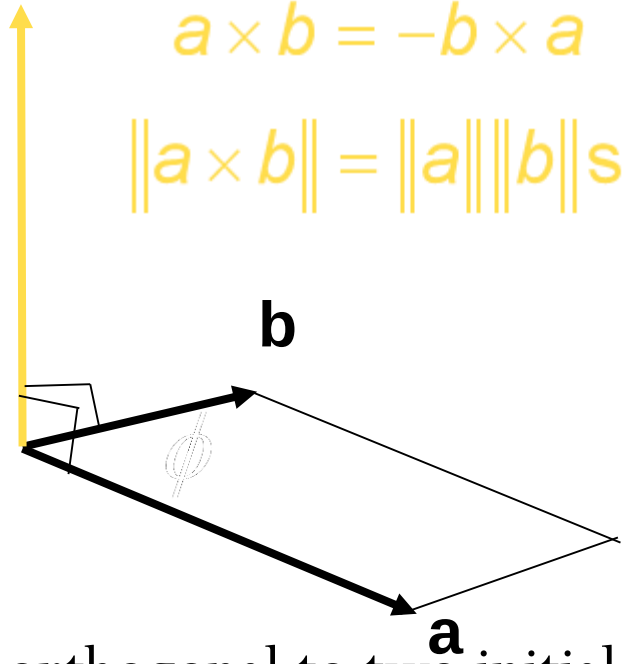
# Vector Multiplication

- 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 ($m \times n$ = 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?

$$a \bullet b = a^T b$$

$$a \circ b \quad a^T b$$

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

$$\begin{pmatrix} x_a & y_a & z_a \end{pmatrix} \begin{pmatrix} x_b \\ y_b \\ z_b \end{pmatrix} \quad \begin{pmatrix} x_a x_b + y_a y_b + z_a z_b \end{pmatrix}$$

- Cross product?

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

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

# 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

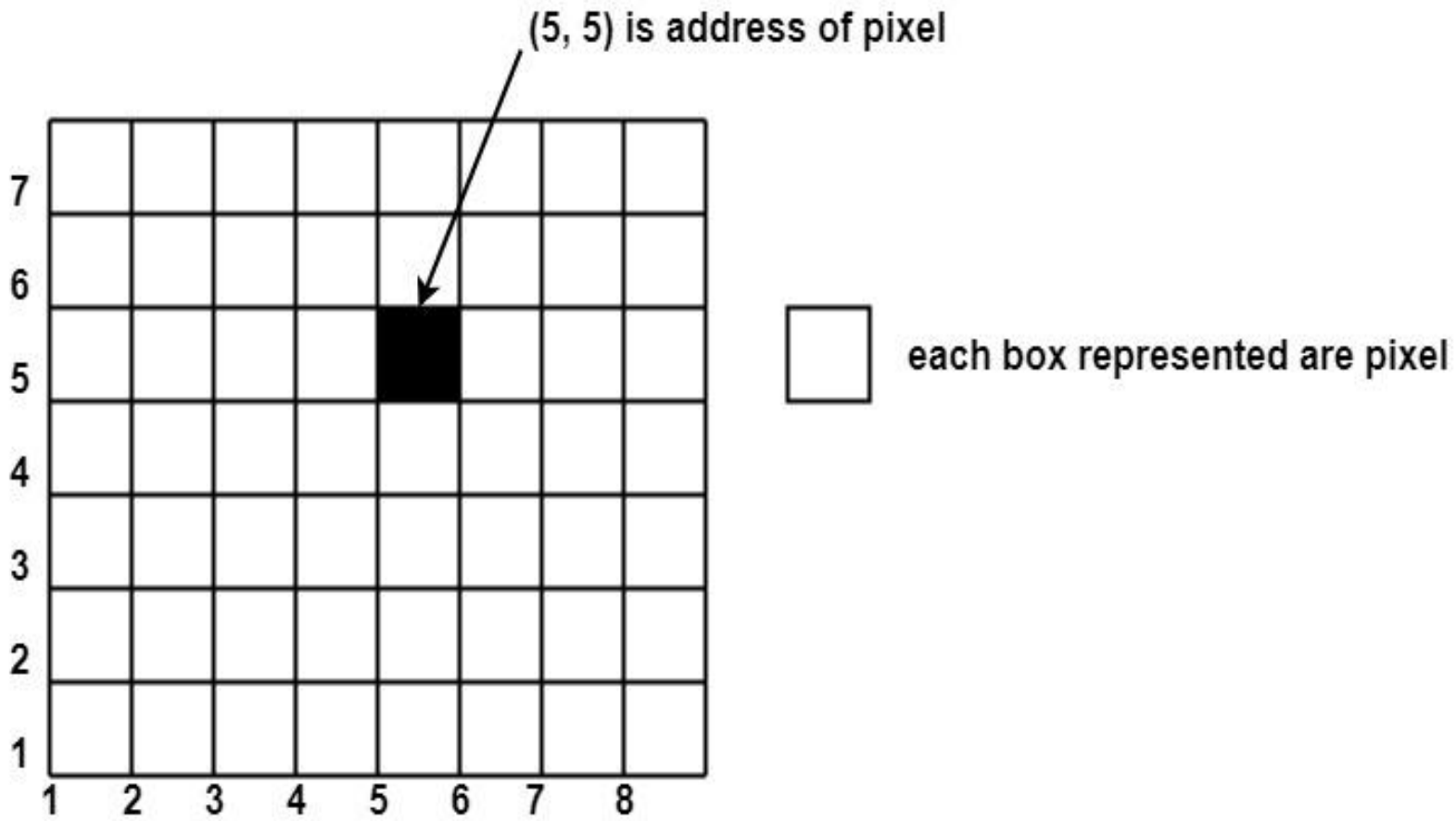
1. Point
2. Line
3. Sector
4. Arc
5. Ellipse
6. Rectangle
7. Polygon
8. Characters
9. Filled Regions

# 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:

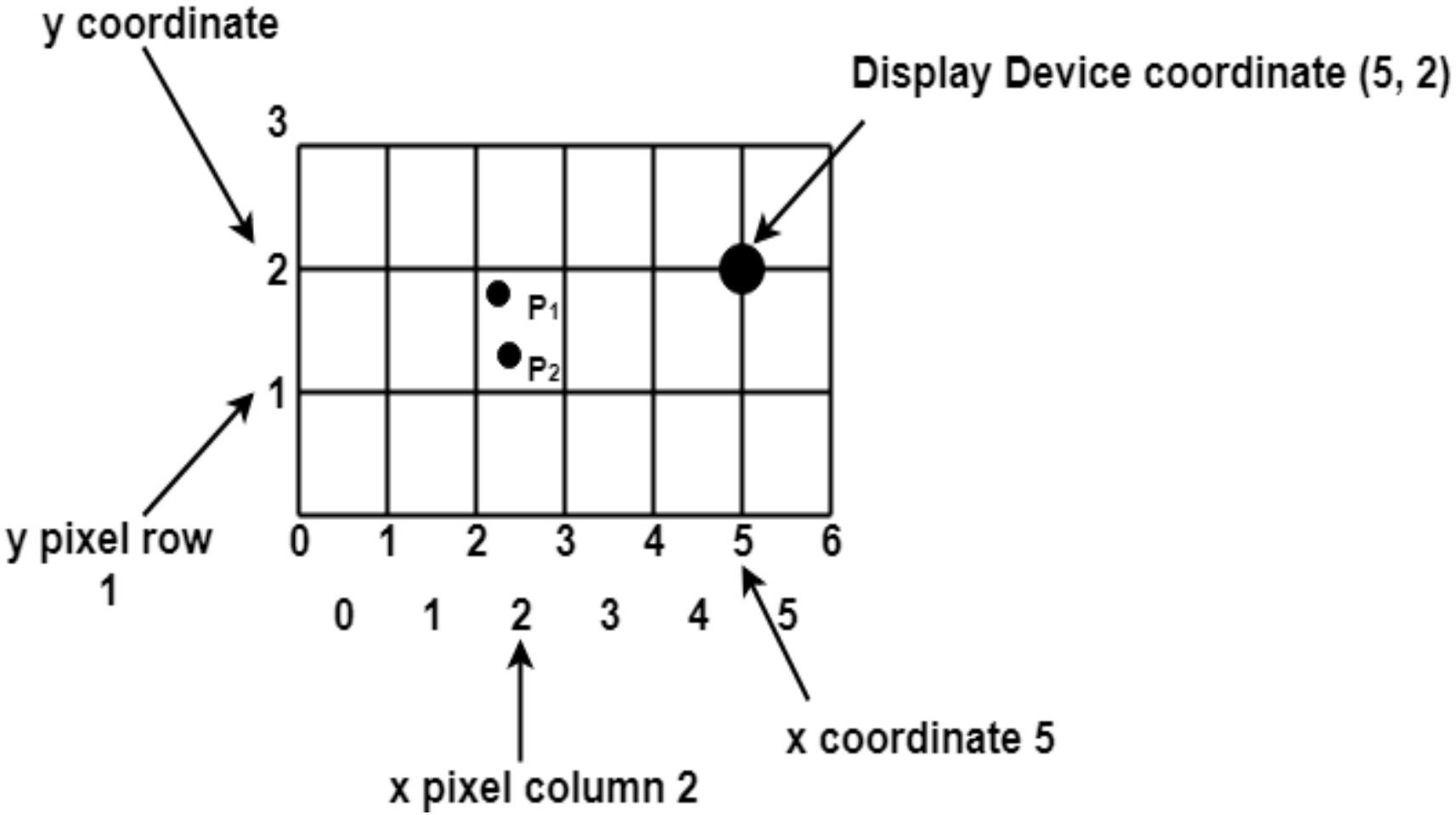

(5, 5) is address of pixel
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8
each box represented are pixel

- 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

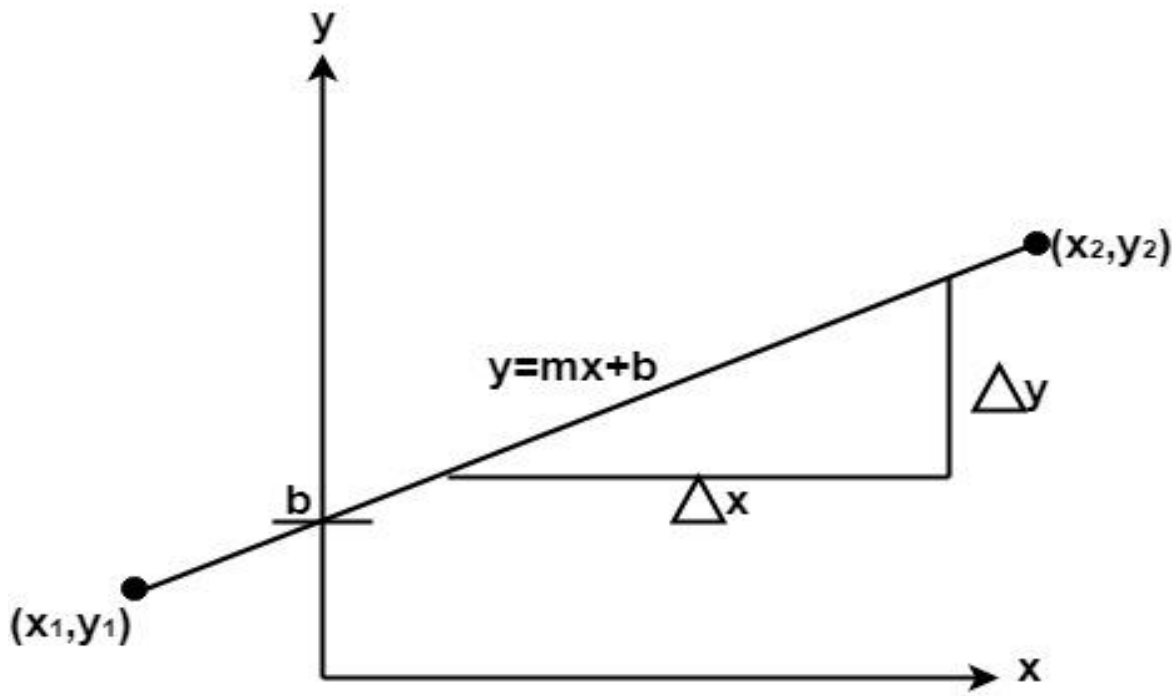
Each pixel on the graphics display does not represent a mathematical point. Instead, it means a region which theoretically can contain an infinite number of points. Scan-Converting a point involves illuminating the pixel that contains the point.

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


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 pixel column 2
x coordinate 5

# 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.

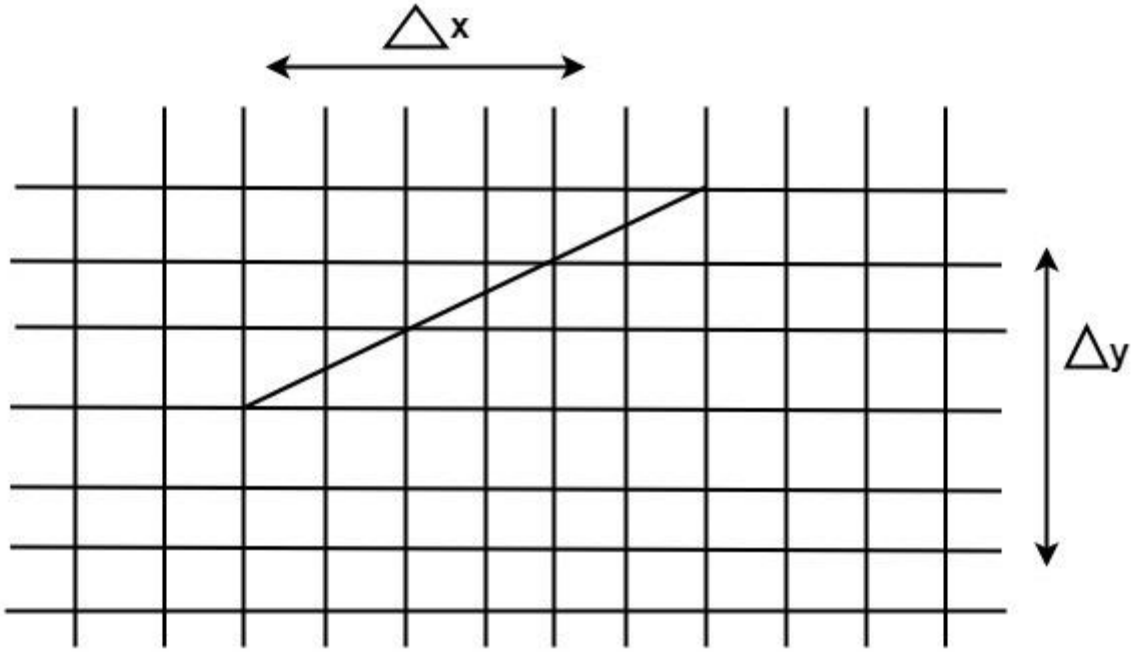



**Fig: A straight line segment connecting 2 grid intersection may fail to pass through any other grid intersections.**

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

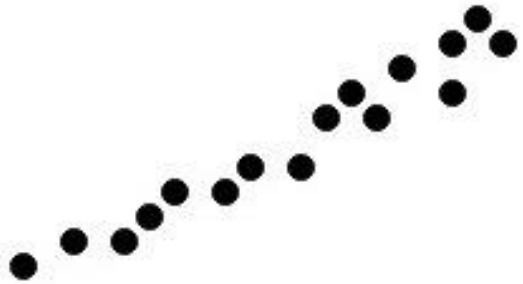


Fig: Uneven line density caused by bunching of dots.

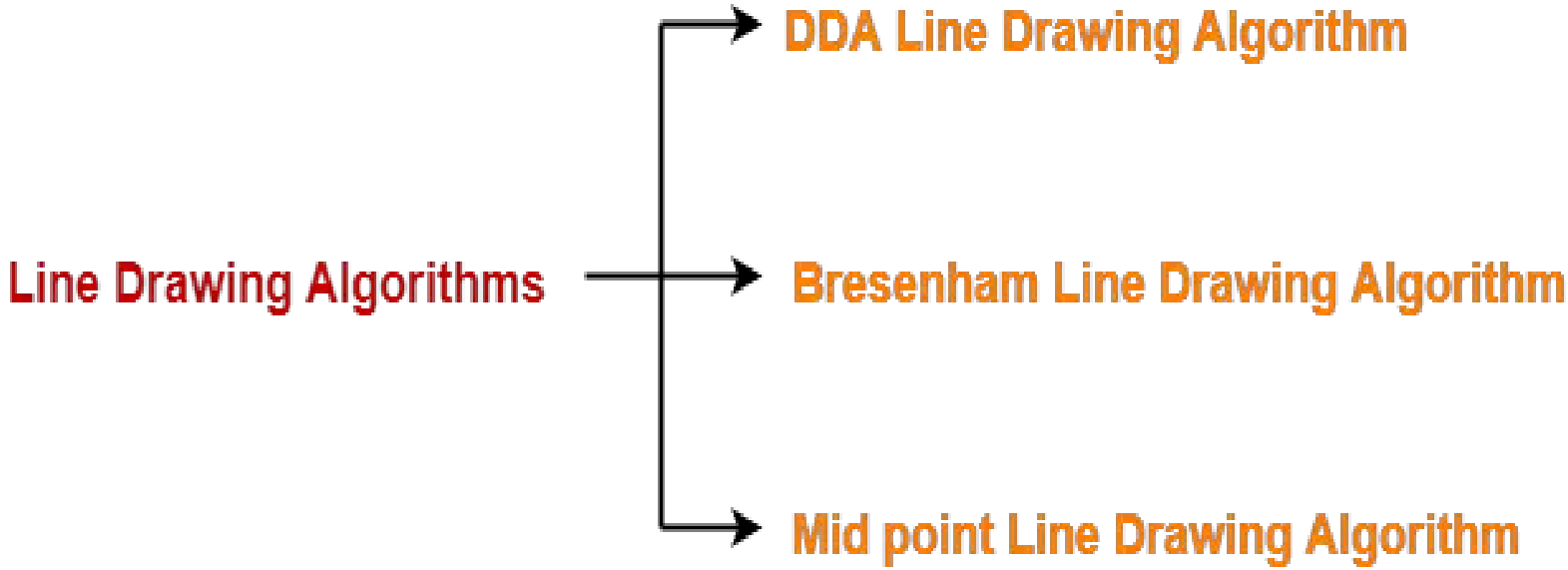
**3. Lines should have constant density:** Line density is proportional to the no. of dots displayed divided by the length of the line.
To maintain constant density, dots should be equally spaced.

**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-

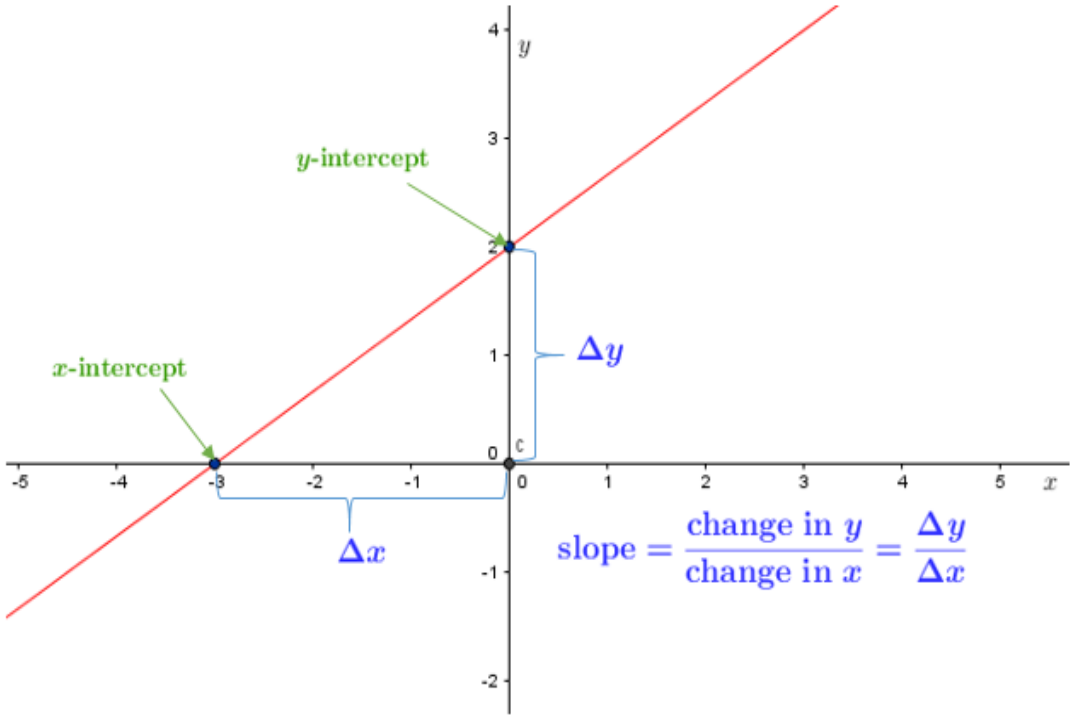
In computer graphics, popular algorithms used to generate lines are-




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

# Recall the line equation below:

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



# 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 $\Delta X$, $\Delta Y$ 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);

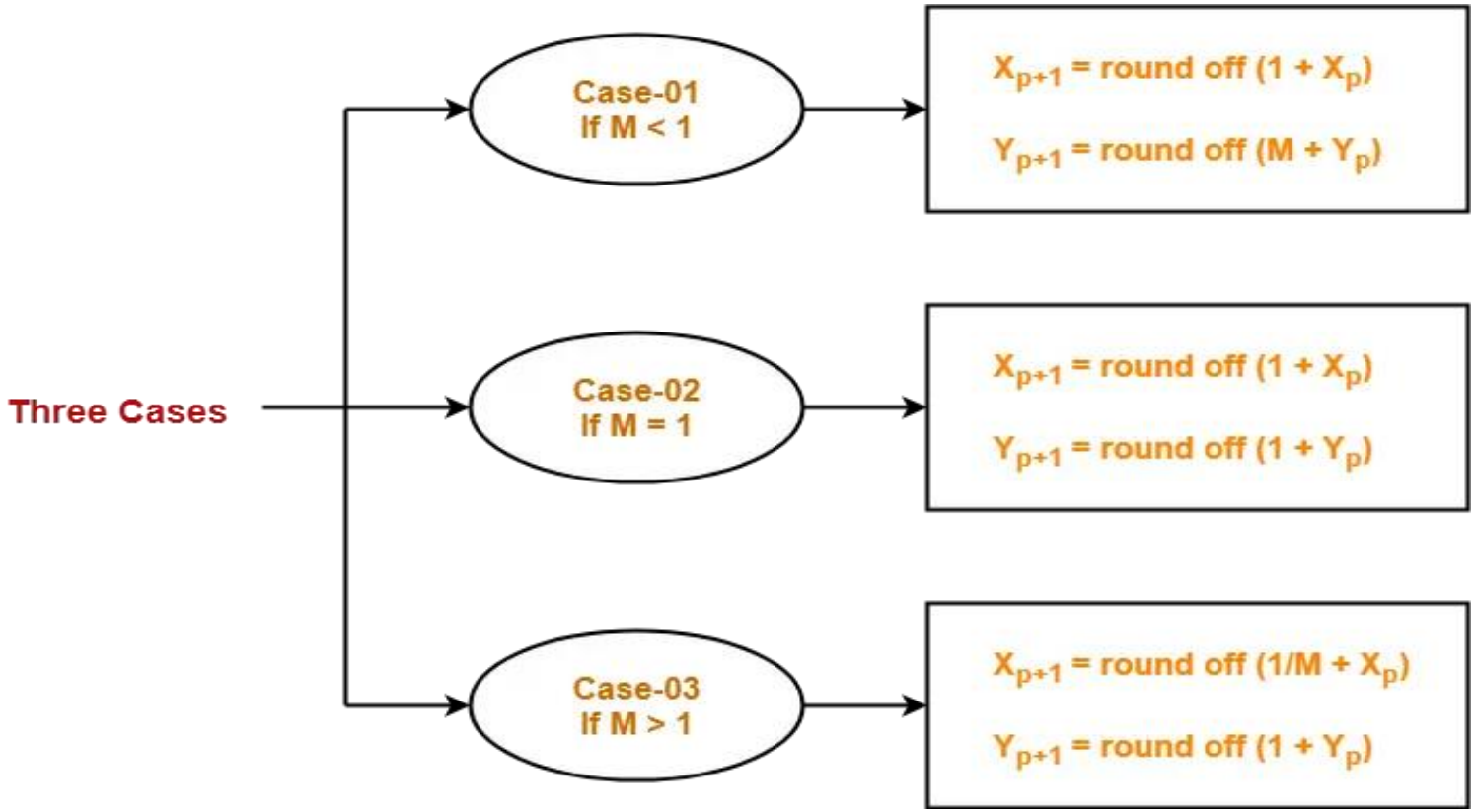
else

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-



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.

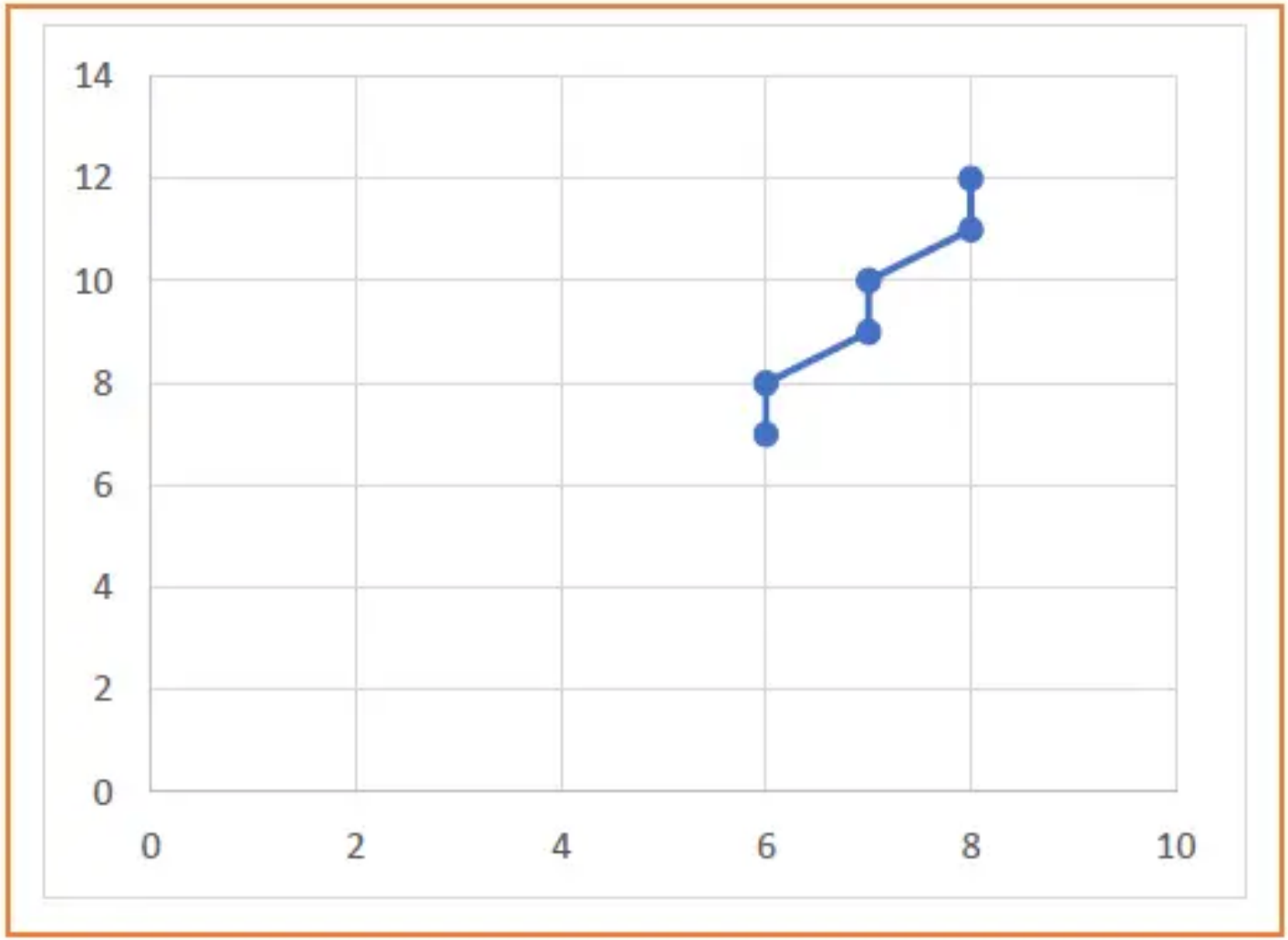
As $|\Delta X| < |\Delta Y| = 3 < 6$, so number of steps = $\Delta Y = 6$

Step-03:

As $M > 1$, so case-03 is satisfied.

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

| $X_p$ | $Y_p$ | $X_{p+1}$ | $Y_{p+1}$ | 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.

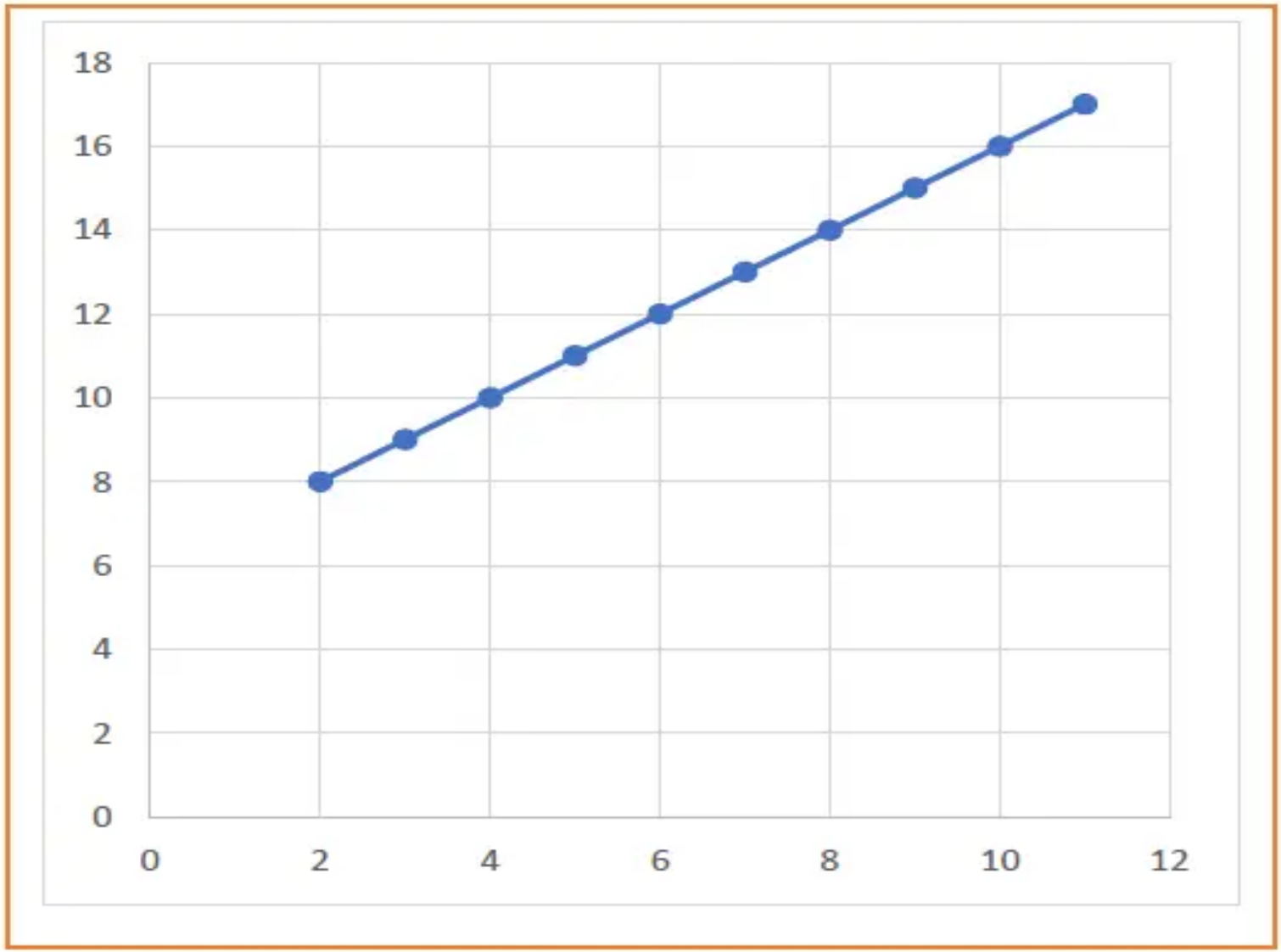
As $|\Delta X| = |\Delta Y| = 10 = 10$, so number of steps = $\Delta X = \Delta Y = 10$

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 $\Delta X$ and $\Delta Y$ 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-

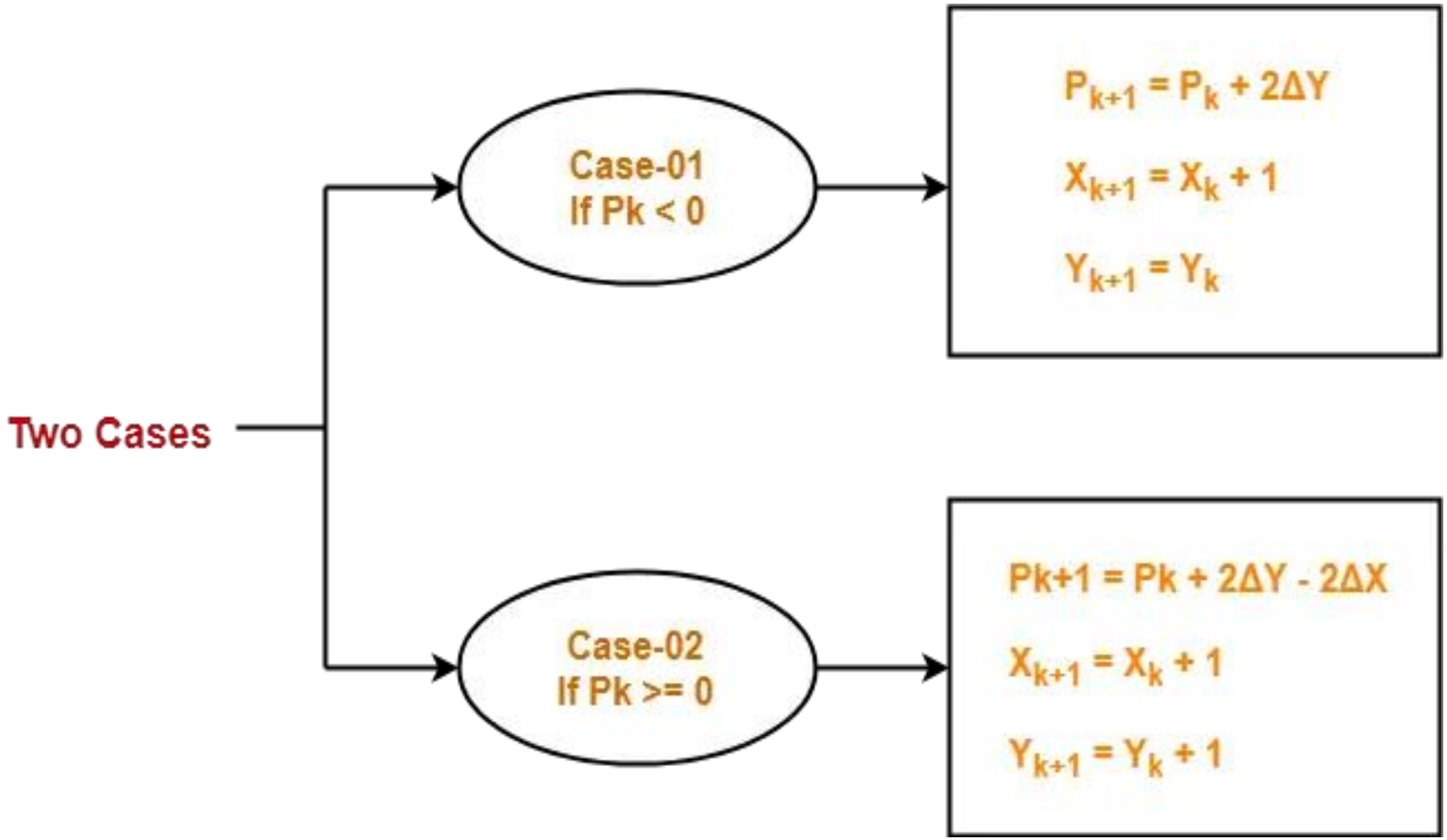
$$P_k = 2\Delta Y - \Delta X$$

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.

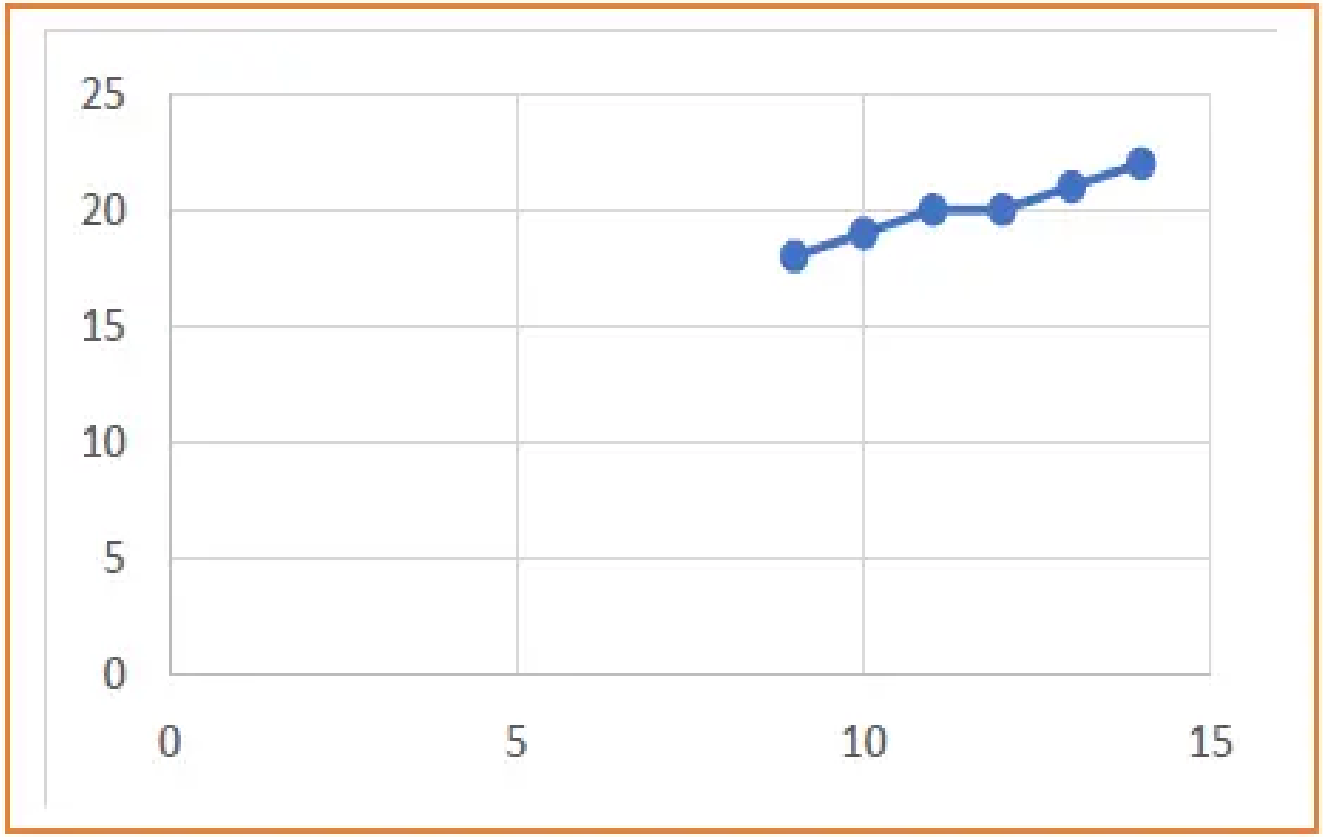
Thus,

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

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 $\Delta X$ and $\Delta Y$ 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.

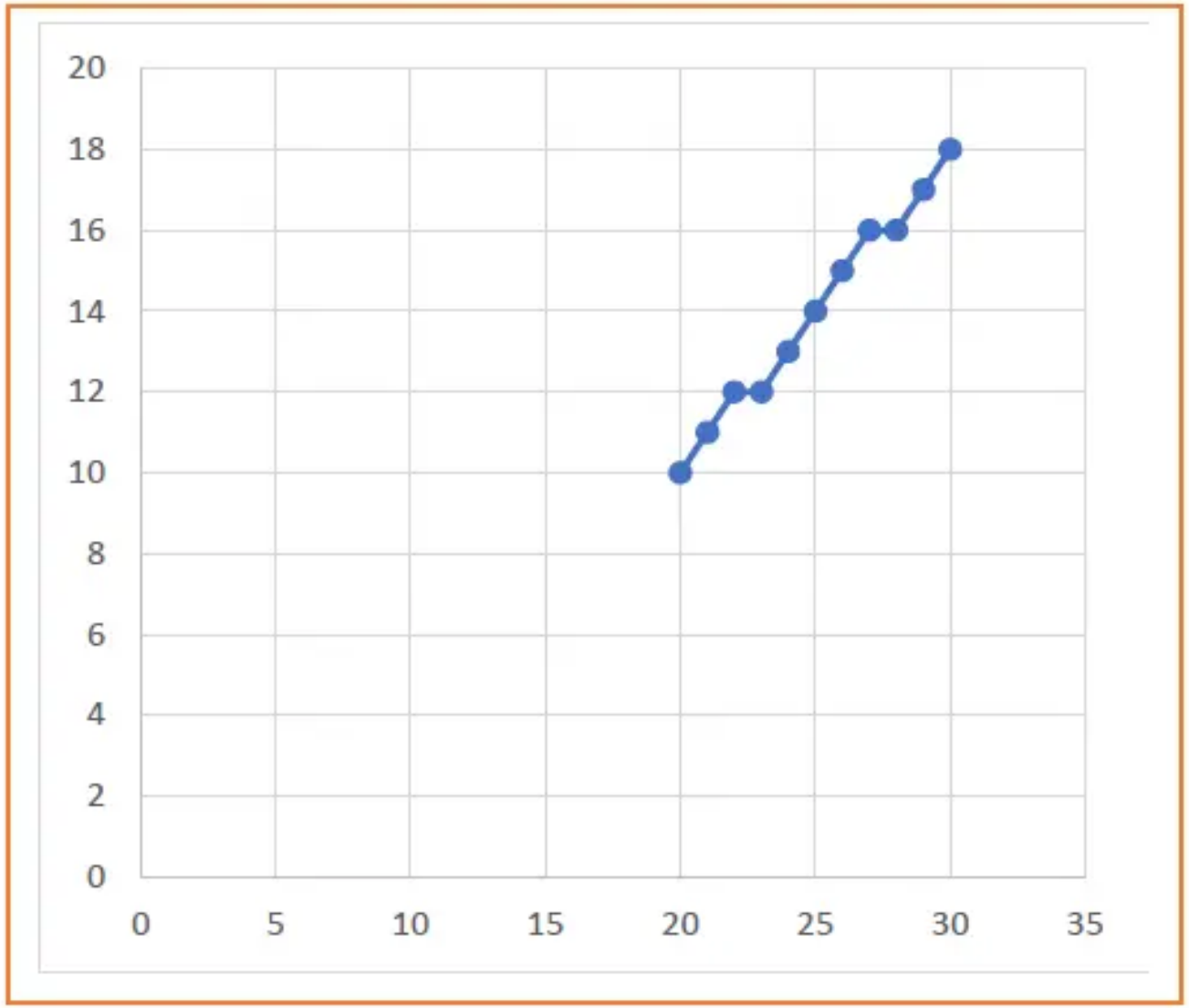
Thus,

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

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 $\Delta X$ and $\Delta Y$ from the given input.

These parameters are calculated as-

- $\Delta X = X_n - X_0$

$\Delta Y = Y_n - Y_0$

Step-02:

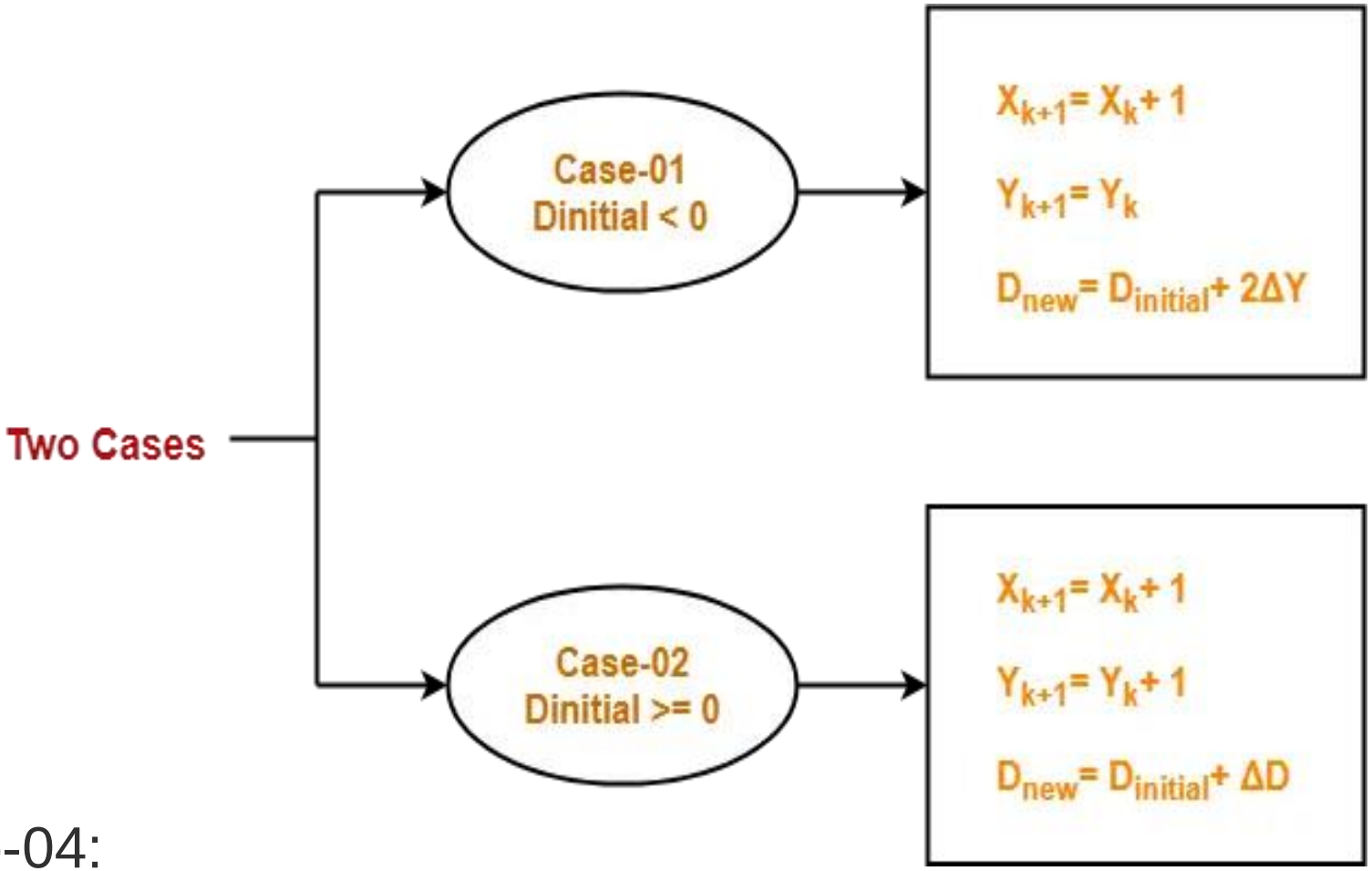
Calculate the value of initial decision parameter and ΔD.

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-




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 $\Delta X$ and $\Delta Y$ 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 $\Delta D$ 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:

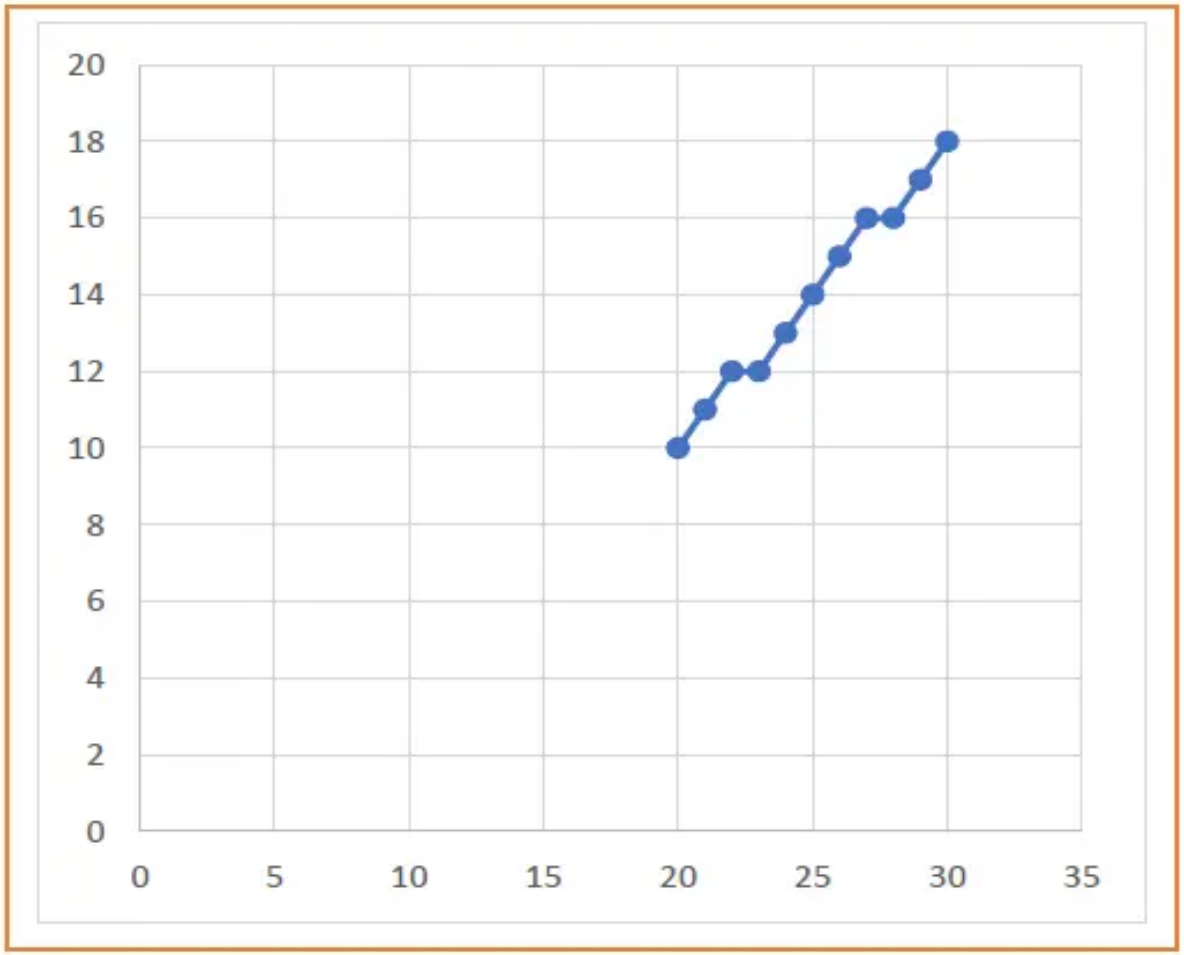
**As $D_{initial}$ >= 0, so case-02 is satisfied.**

**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 $\Delta X$ and $\Delta Y$ 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 $\Delta D$ 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:

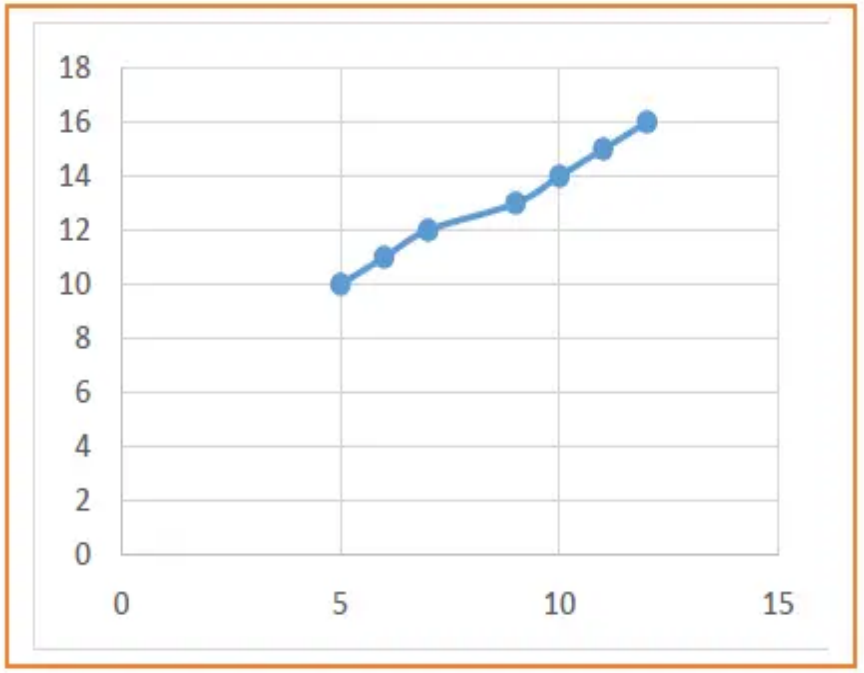
As $D_{initial} >= 0$, so case-02 is satisfied.

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-

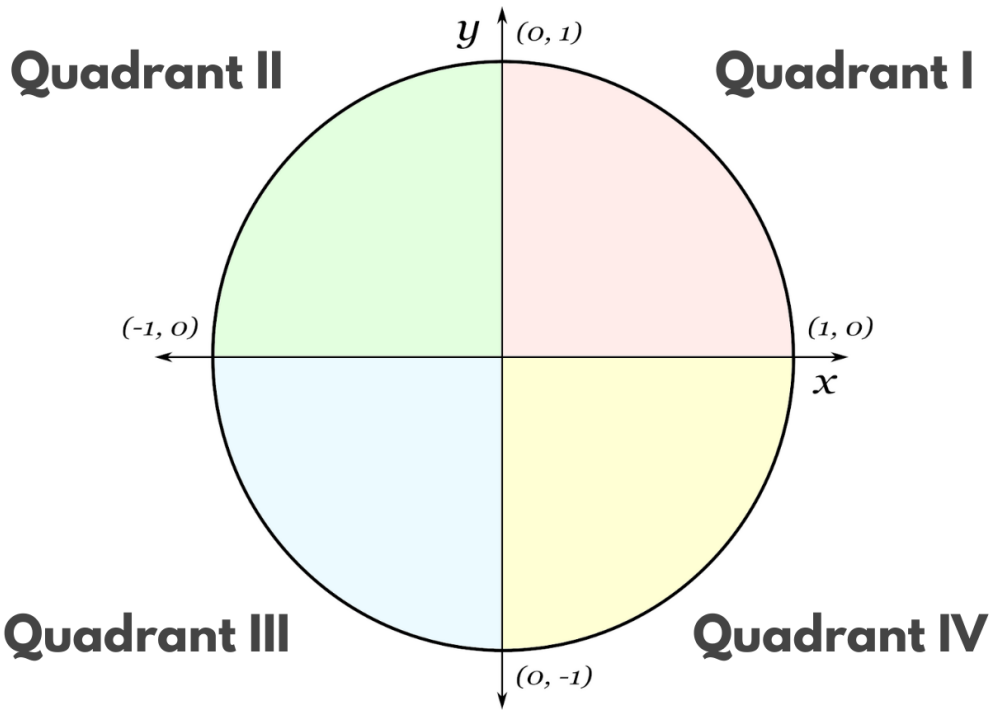
The disadvantages of Mid Point Line Drawing Algorithm are-

- 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.

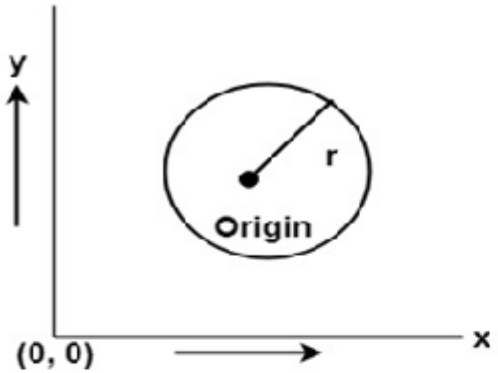
# Defining a Circle:

- 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.

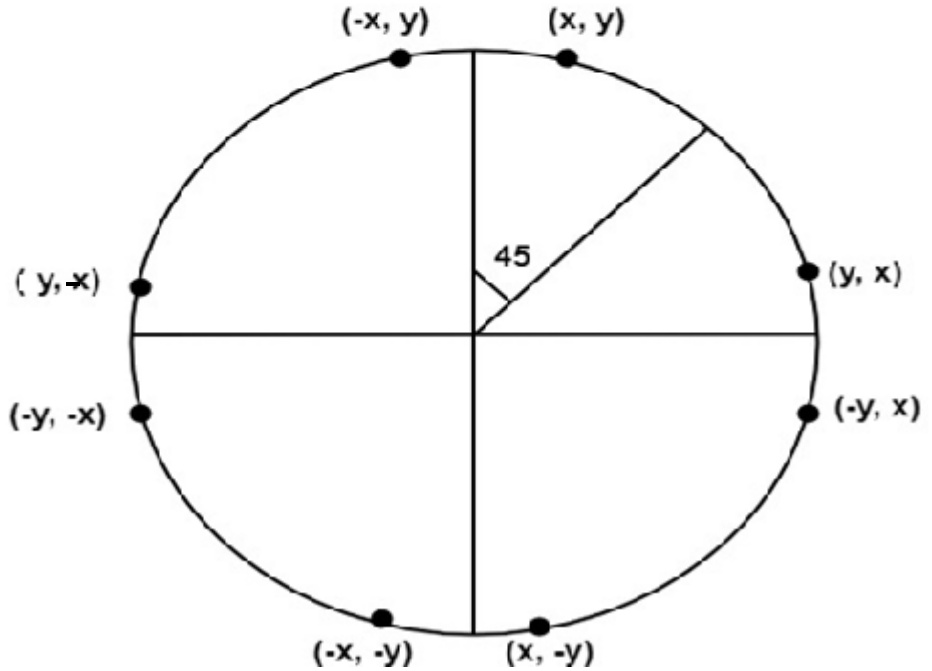
- Quadrant defines as the **four quarters** in the coordinate plane system.
- Each of the four sections is called a quadrant.
- The quarter of a circle is called a quadrant, which is a sector of 90 degrees.



- 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.

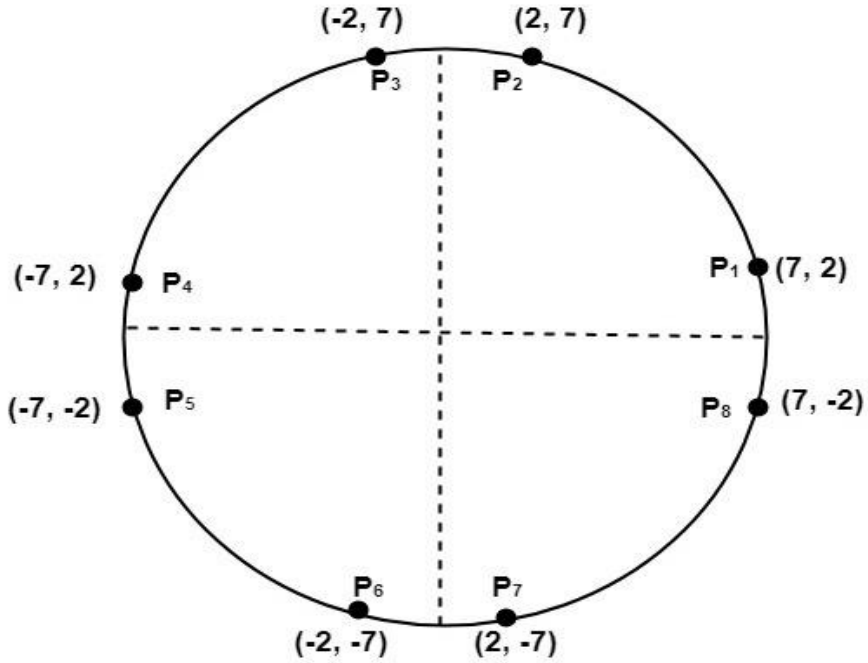
If we want to display circle on screen then the putpixel function is used for eight points as shown below:

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:

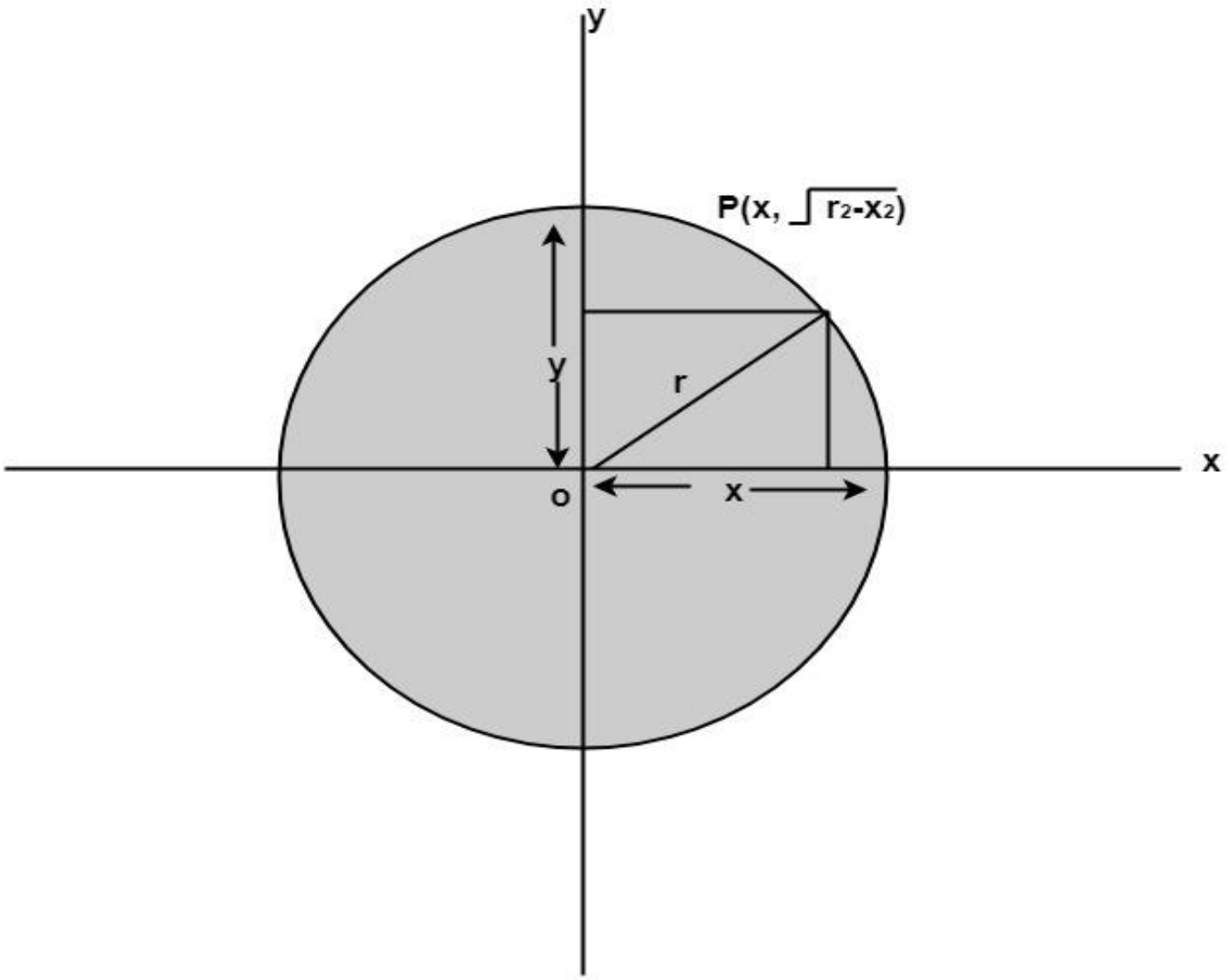
$$y^2=r^2-x^2$$

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

l = 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:

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

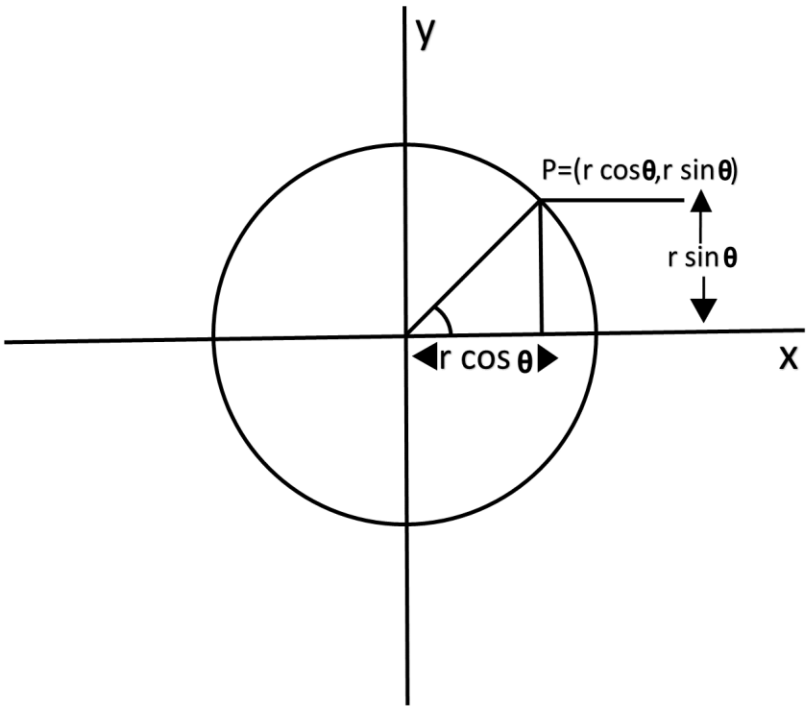
Where θ=current angle

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 cosθ
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 θ  y=r*sinθ

**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).

# 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-

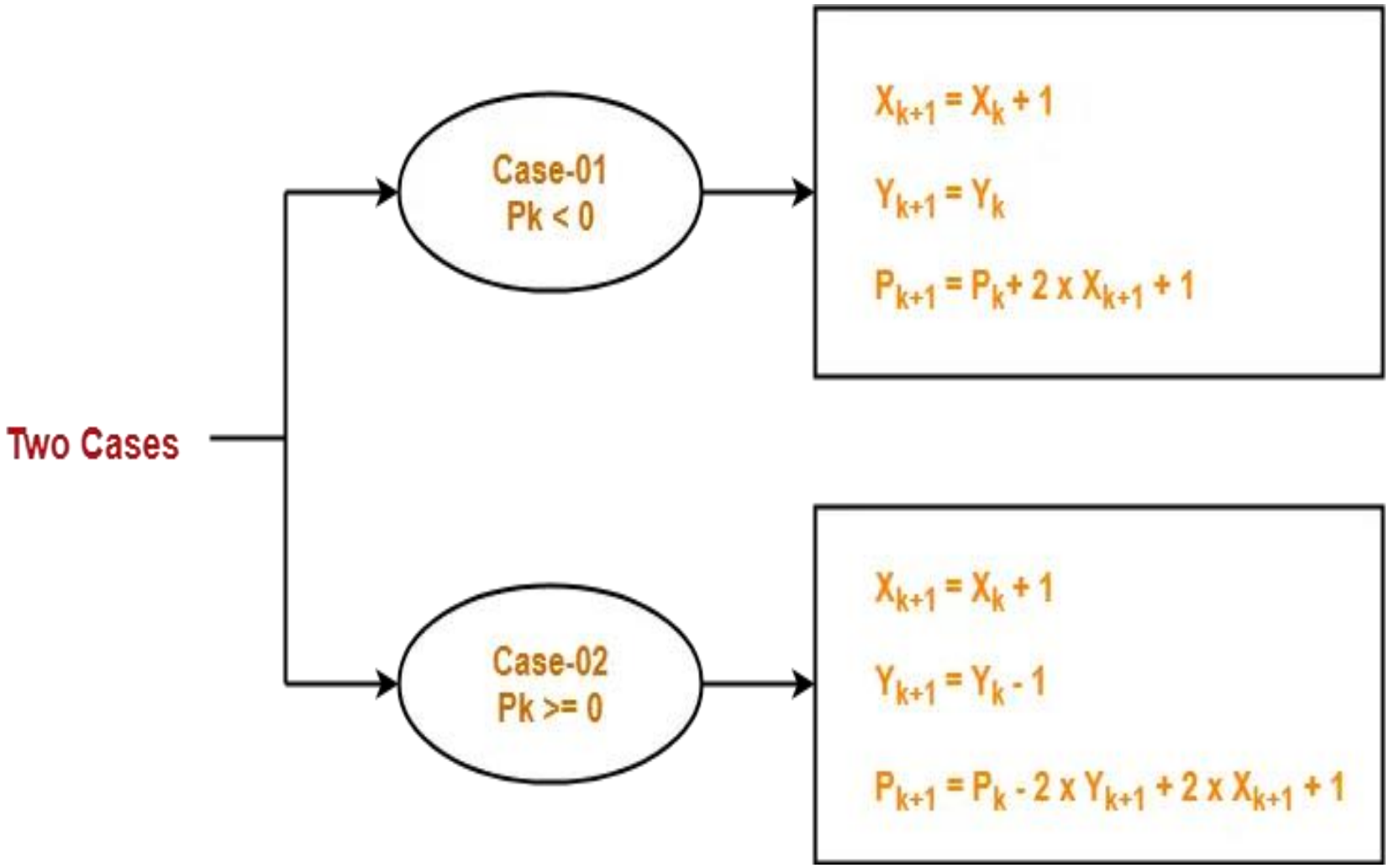
$$P_0 = 1 – R$$

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
$X_{k+1} = X_k + 1$
$Y_{k+1} = Y_k$
$P_{k+1} = P_k + 2 \times X_{k+1} + 1$
Case-02
Pk >= 0
$X_{k+1} = X_k + 1$
$Y_{k+1} = Y_k - 1$
$P_{k+1} = P_k - 2 \times Y_{k+1} + 2 \times X_{k+1} + 1$

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:

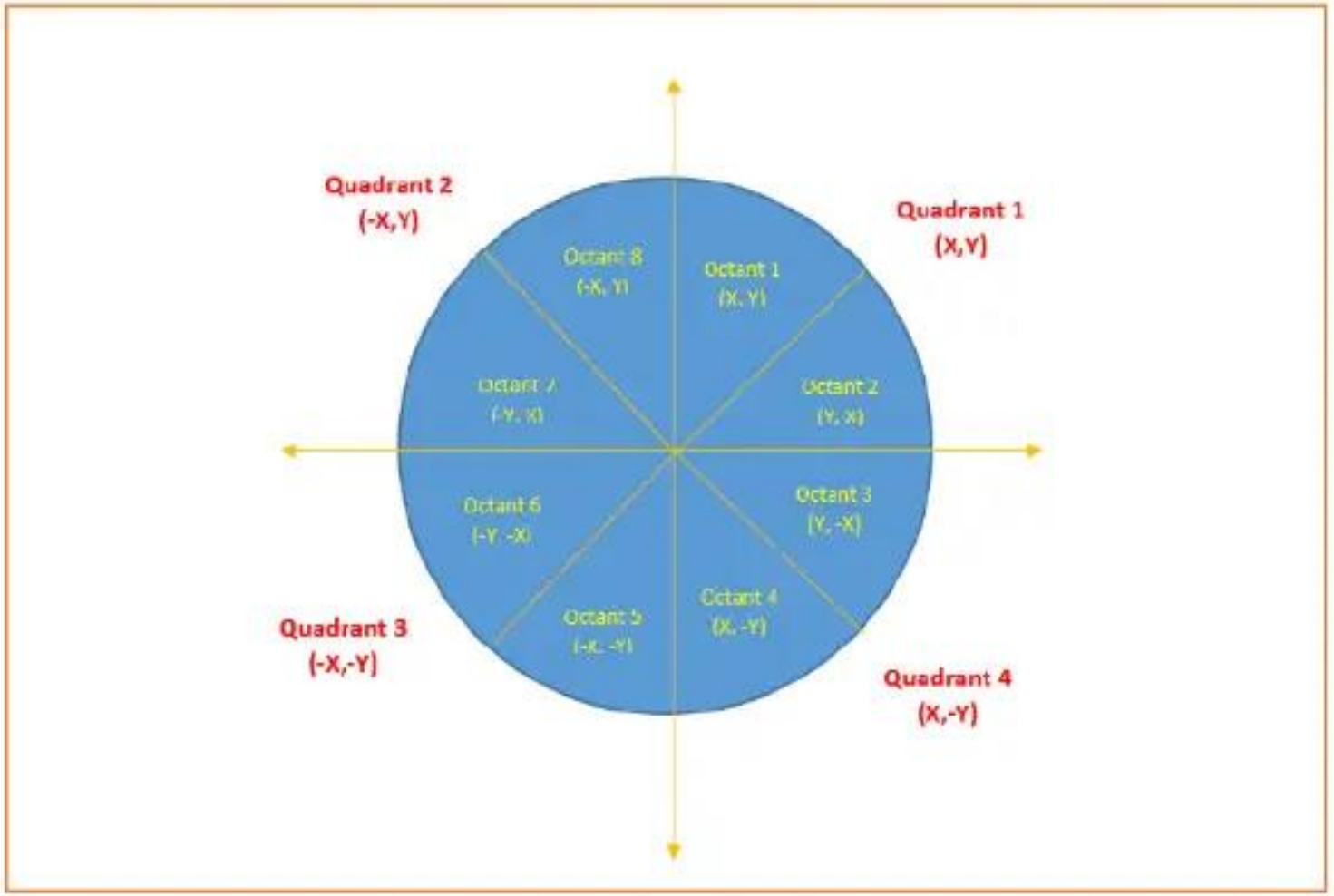
Keep repeating Step-03 and Step-04 until $X_{plot} >= Y_{plot}$.

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,

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

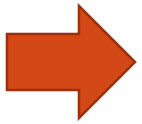
Step-04:

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<br>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-

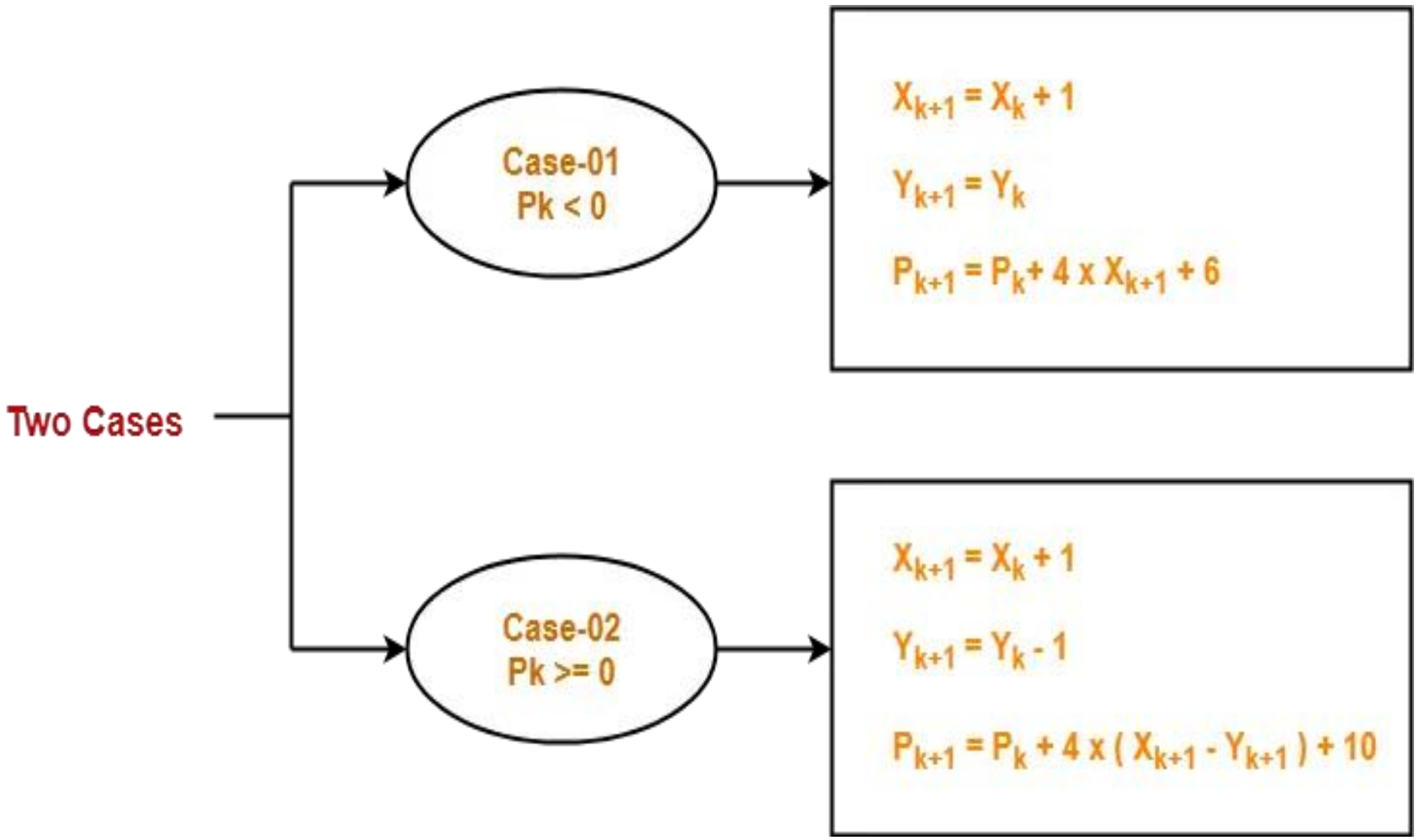
$$P_0 = 3 – 2 \times R$$

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:

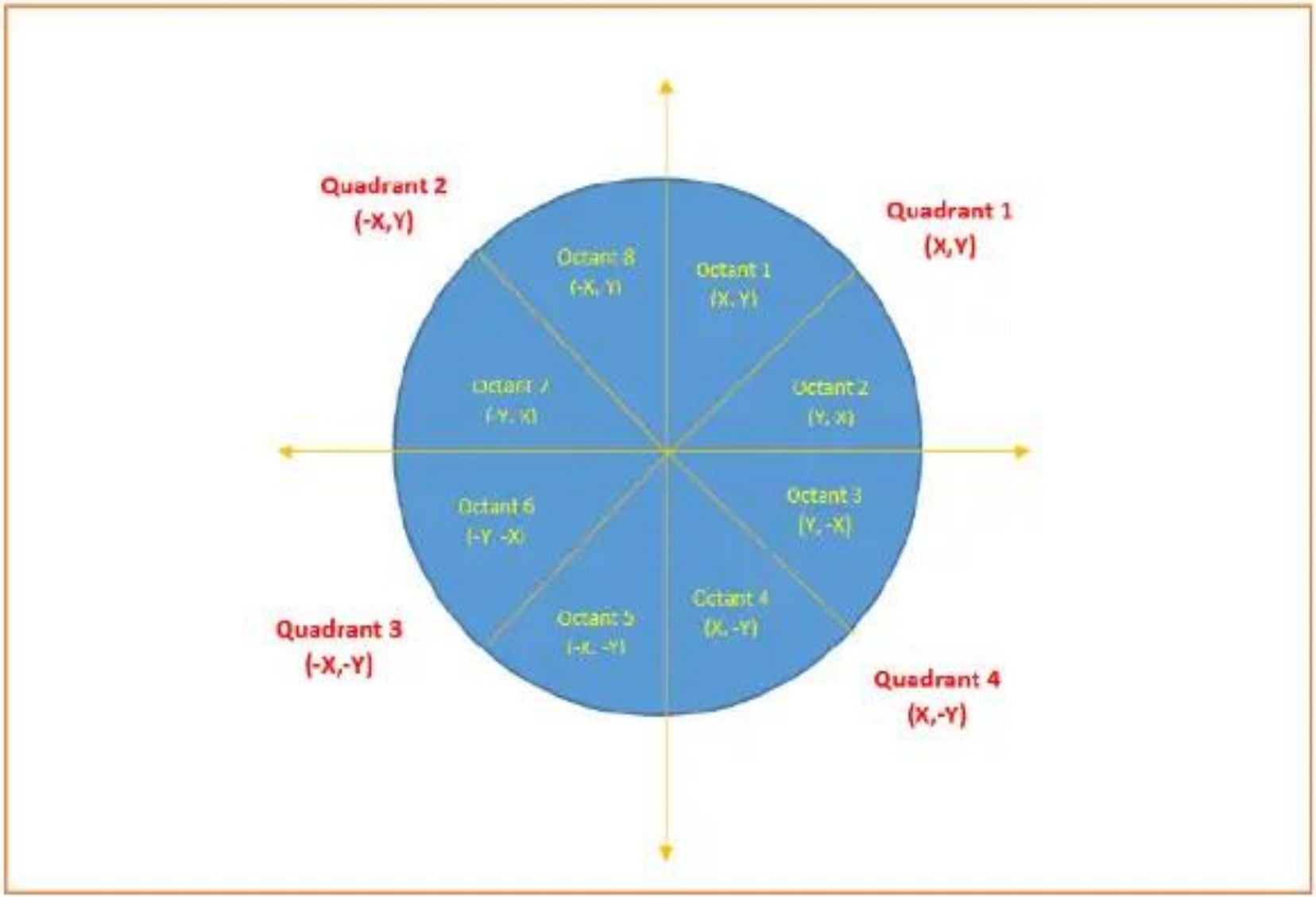
Keep repeating Step-03 and Step-04 until $X_{plot}$ => $Y_{plot}$.

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-

$P_0 = 3 – 2 \times R$

$P_0 = 3 – 2 \times 8$

$P_0 = -13$

Step-03:

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

Thus,

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

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-

$P_0 = 3 – 2 \times R$

$P_0 = 3 – 2 \times 10$

$P_0 = -17$

Step-03:

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

Thus,

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

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**<br>**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.

Prof. Jelili O. Oyelade

# Computer Graphics and Animation (CSC433)

# Introduction to Transformation

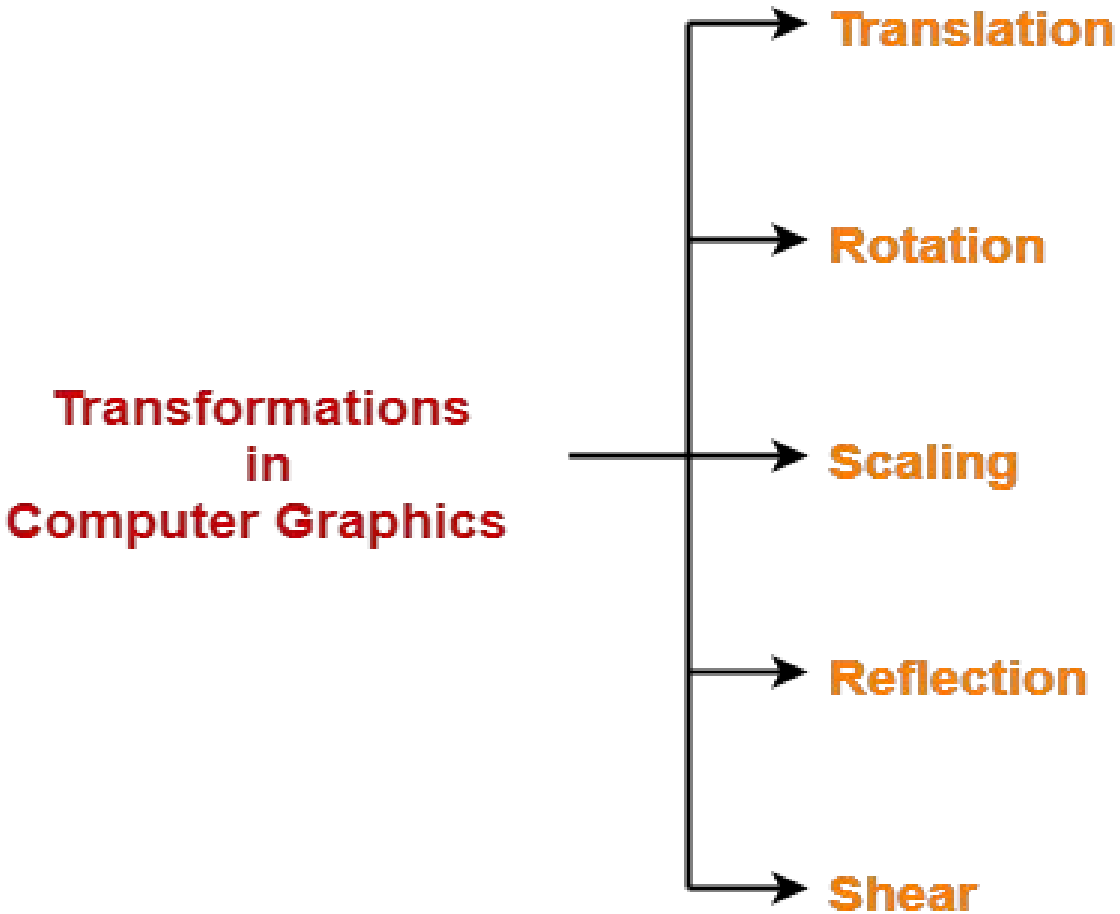
- In Computer graphics,Transformation is a process of modifying and re-positioning the existing graphics.

- The process of using computers for drawing by providing facility to user to view the object from different angles, enlarging or reducing the scale or shape of object is called Transformation

  - 2D Transformations take place in a two dimensional plane.
  - 3D Transformations take place in a three dimensional plane.
  - Transformations are helpful in changing the position, size, orientation, shape etc of the object.

# Transformation Techniques

- In computer graphics, various transformation techniques are:



# 2D Translation in Computer Graphics

**2D Translation is a process of moving an object from one position to another in a two dimensional plane.**

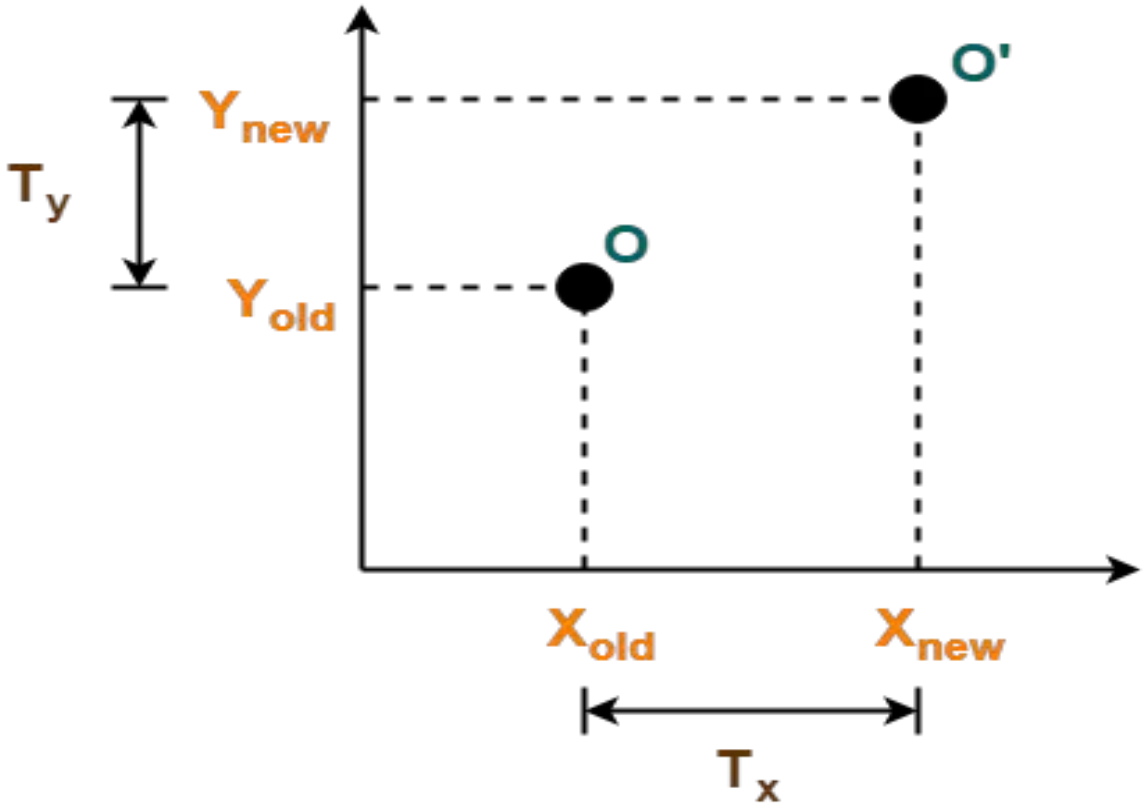
Let consider a point object O has to be moved from one position to another in a 2D plane.

Let:

- Initial coordinates of the object O = $(X_{old}, Y_{old})$
- New coordinates of the object O after translation = $(X_{new}, Y_{new})$
- Translation vector or Shift vector = $(T_x, T_y)$

Given a Translation vector $(T_x, T_y)$-

- $T_x$ defines the distance the $X_{old}$ coordinate has to be moved.
- $T_y$ defines the distance the $Y_{old}$ coordinate has to be moved.




2D Translation in Computer Graphics

This translation is achieved by adding the translation coordinates to the old coordinates of the object as-

- $X_{new} = X_{old} + T_x$ (This denotes translation towards X axis)
- $Y_{new} = Y_{old} + T_y$ (This denotes translation towards Y axis)

In Matrix form, the above translation equations may be represented as:

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$

**Translation Matrix**

- The homogeneous coordinates representation of (X, Y) is (X, Y, 1).
- Through this representation, all the transformations can be performed using matrix / vector multiplications.

The above translation matrix may be represented as a 3 x 3 matrix as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ 1 \end{bmatrix}$$

**Translation Matrix**

**(Homogeneous Coordinates Representation)**

# Example1:

Given a circle C with radius 10 and center coordinates (1, 4). Apply the translation with distance 5 towards X axis and 1 towards Y axis. Obtain the new coordinates of C without changing its radius.

# Solution:

Given-

- Old center coordinates of C = $(X_{old}, Y_{old}) = (1, 4)$
- Translation vector = $(T_x, T_y) = (5, 1)$

Let the new center coordinates of C = $(X_{new}, Y_{new})$.

Applying the translation equations, we have-

- $X_{new} = X_{old} + T_x = 1 + 5 = 6$
- $Y_{new} = Y_{old} + T_y = 4 + 1 = 5$
- Thus, New center coordinates of C = (6, 5).

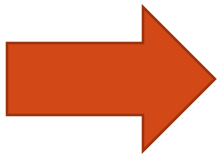
## Alternatively,

- In matrix form, the new center coordinates of C after translation may be obtained as-

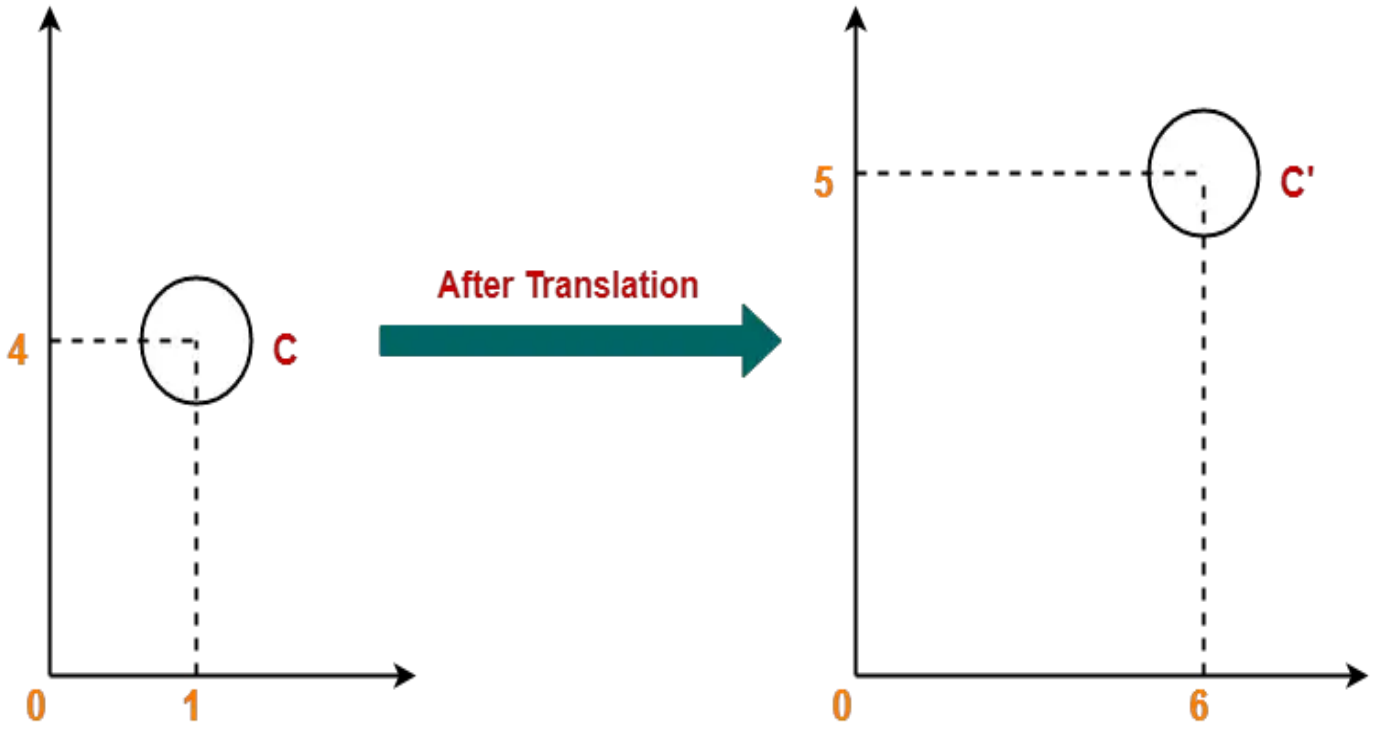
$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \end{bmatrix} + \begin{bmatrix} 5 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 6 \\ 5 \end{bmatrix}$$



Therefore, New center coordinates of C = (6, 5).


4
C
0
1
After Translation
5
C'
0
6

# Example2:

Given a square with coordinate points A(0, 3), B(3, 3), C(3, 0), D(0, 0). Apply the translation with distance 1 towards X axis and 1 towards Y axis. Obtain the new coordinates of the square.

# Solution:

Given-

- Old coordinates of the square = A (0, 3), B(3, 3), C(3, 0), D(0, 0)
- Translation vector = $(T_x, T_y) = (1, 1)$

For Coordinates A(0, 3)

Let the new coordinates of corner A = $(X_{new}, Y_{new})$.

Applying the translation equations, we have-

- $X_{new} = X_{old} + T_x = 0 + 1 = 1$
- $Y_{new} = Y_{old} + T_y = 3 + 1 = 4$

Thus, New coordinates of corner A = (1, 4).

For Coordinates B(3, 3)

Let the new coordinates of corner B = ($X_{new}$, $Y_{new}$).

Applying the translation equations, we have-

- $X_{new} = X_{old} + T_x = 3 + 1 = 4$
- $Y_{new} = Y_{old} + T_y = 3 + 1 = 4$

Thus, New coordinates of corner B = (4, 4).

For Coordinates C(3, 0)

Let the new coordinates of corner C = ($X_{new}$, $Y_{new}$).

Applying the translation equations, we have-

- $X_{new} = X_{old} + T_x = 3 + 1 = 4$
- $Y_{new} = Y_{old} + T_y = 0 + 1 = 1$

Thus, New coordinates of corner C = (4, 1).

For Coordinates D(0, 0)

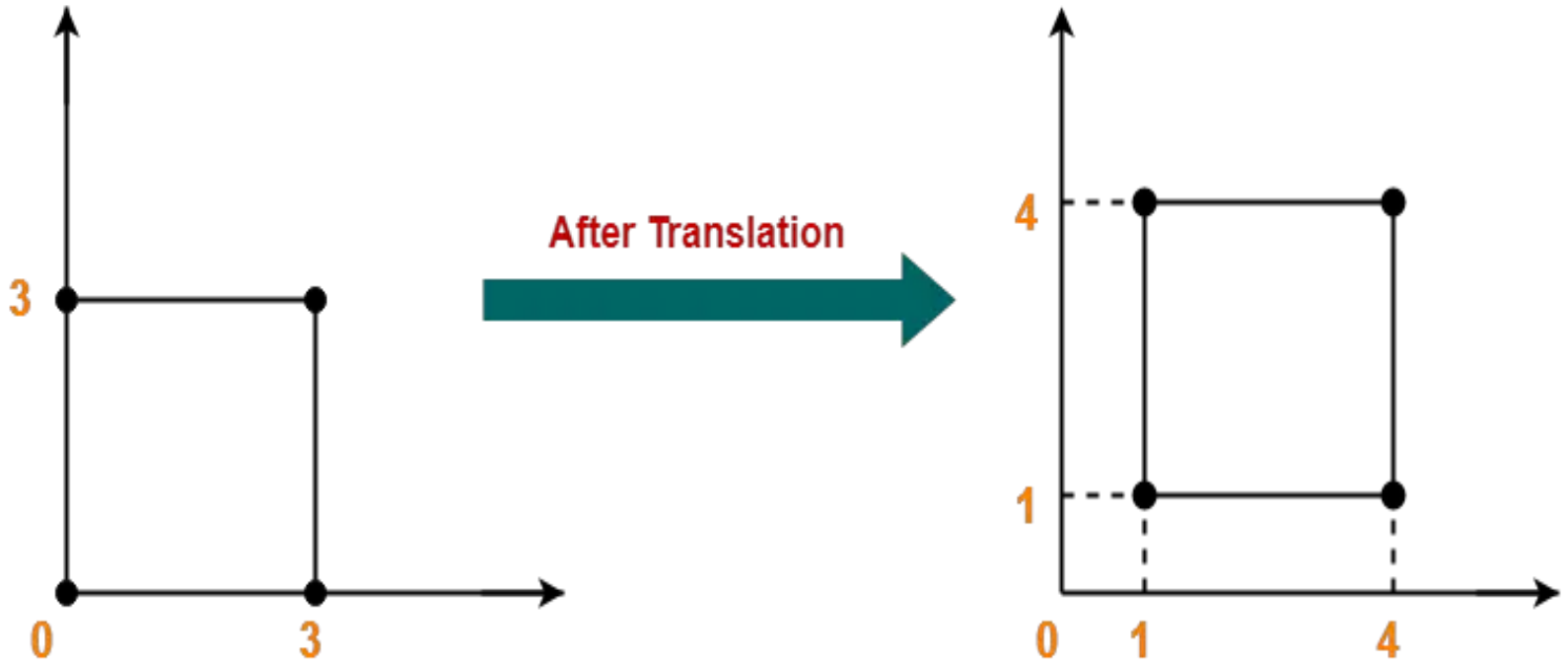
Let the new coordinates of corner D = ($X_{new}$, $Y_{new}$).

Applying the translation equations, we have-

- $X_{new} = X_{old} + T_x = 0 + 1 = 1$
- $Y_{new} = Y_{old} + T_y = 0 + 1 = 1$

Thus, New coordinates of corner D = (1, 1).

Therefore, New coordinates of the square = A (1, 4), B(4, 4), C(4, 1), D(1, 1).


3
0
3
After Translation
4
1
0
1
4

**Problem:** Given a Point with coordinates (2, 4). Apply the translation with distance 4 towards x-axis and 2 towards the y-axis. Find the new coordinates without changing the radius?

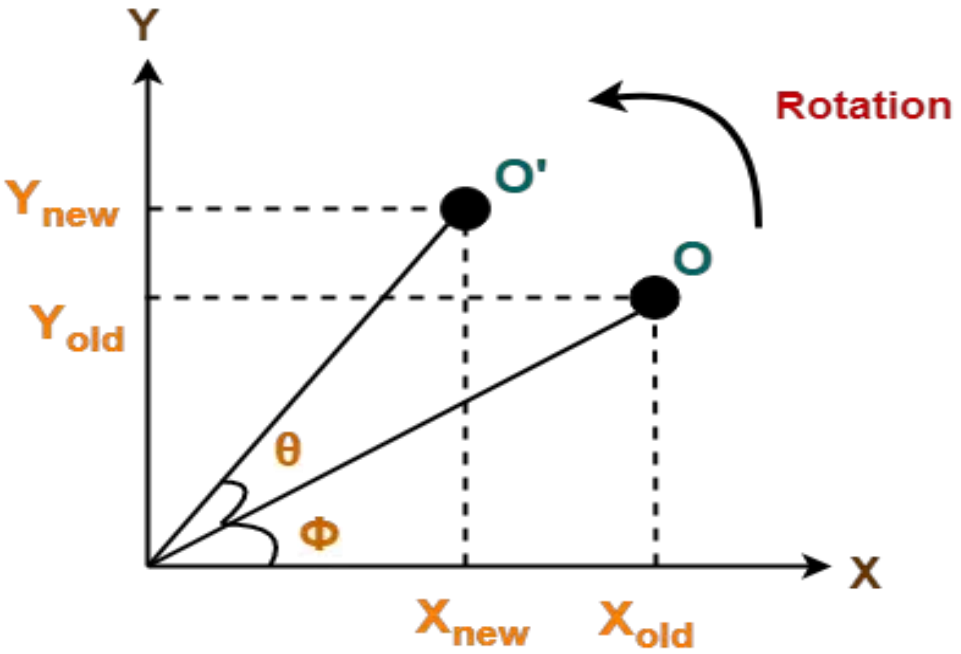
# 2D Rotation in Computer Graphics

2D Rotation is a process of rotating an object with respect to an angle in a two dimensional plane

Let consider a point object O has to be rotated from one angle to another in a 2D plane.

Let:

- Initial coordinates of the object O = ($X_{old}$, $Y_{old}$)
- Initial angle of the object O with respect to origin = Φ
- Rotation angle = θ
- New coordinates of the object O after rotation = ($X_{new}$, $Y_{new}$)




2D Rotation in Computer Graphics

This rotation is achieved by using the following rotation equations-

- $X_{new} = X_{old} \times \cos\theta - Y_{old} \times \sin\theta$
- $Y_{new} = X_{old} \times \sin\theta + Y_{old} \times \cos\theta$

In Matrix form, the above rotation equations may be represented as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

Rotation Matrix

For homogeneous coordinates, the above rotation matrix may be represented as a 3 x 3 matrix as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ 1 \end{bmatrix}$$

**Rotation Matrix**

**(Homogeneous Coordinates Representation)**

# Example1:

Given a line segment with starting point as (0, 0) and ending point as (4, 4). Apply 30 degree rotation anticlockwise direction on the line segment and find out the new coordinates of the line.

# Solution-

We rotate a straight line by its end points with the same angle. Then, we re-draw a line between the new end points.

Given-

- Old ending coordinates of the line = $(X_{old}, Y_{old}) = (4, 4)$
- Rotation angle = $\theta = 30^o$

Let new ending coordinates of the line after rotation = $(X_{new}, Y_{new})$.

Applying the rotation equations, we have:

$X_{new}$

$= X_{old} \times \cos\theta - Y_{old} \times \sin\theta$

$= 4 \times \cos 30^o - 4 \times \sin 30^o$

$= 4 \times (\sqrt{3} / 2) - 4 \times (1 / 2)$

$= 2\sqrt{3} - 2$

$= 2(\sqrt{3} - 1)$

$= 2(1.73 - 1)$

$= 1.46$

$Y_{new}$

$= X_{old} \times \sin\theta + Y_{old} \times \cos\theta$

$= 4 \times \sin 30^o + 4 \times \cos 30^o$

$= 4 \times (1 / 2) + 4 \times (\sqrt{3} / 2)$

$= 2 + 2\sqrt{3}$

$= 2(1 + \sqrt{3})$

$= 2(1 + 1.73)$

$= 5.46$

Therefore, New ending coordinates of the line after rotation

= (1.46, 5.46).

**Alternatively,**

In matrix form, the new ending coordinates of the line after rotation may be obtained as

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

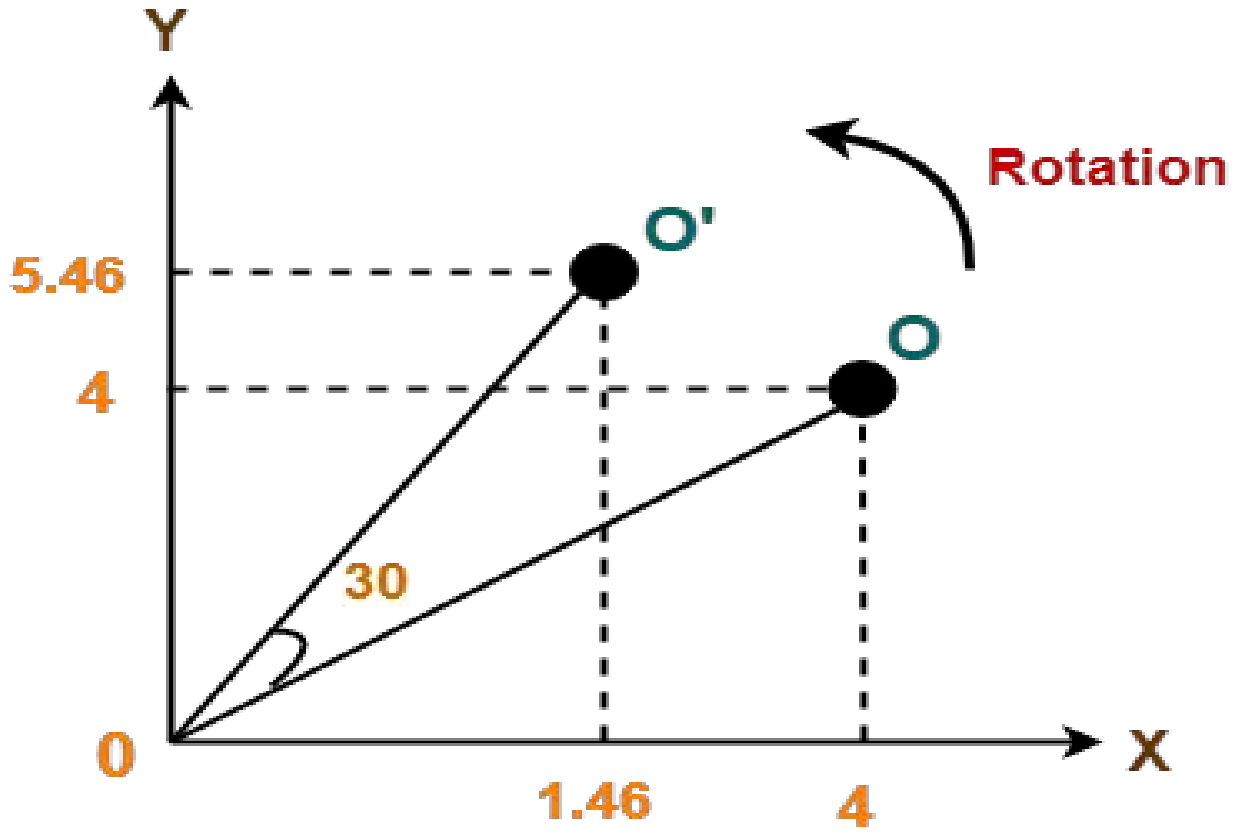
$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} \cos 30 & -\sin 30 \\ \sin 30 & \cos 30 \end{bmatrix} \times \begin{bmatrix} 4 \\ 4 \end{bmatrix}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 4 \times \cos 30 - 4 \times \sin 30 \\ 4 \times \sin 30 + 4 \times \cos 30 \end{bmatrix}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 4 \times \cos 30 - 4 \times \sin 30 \\ 4 \times \sin 30 + 4 \times \cos 30 \end{bmatrix}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 1.46 \\ 5.46 \end{bmatrix}$$

Therefore, New ending coordinates of the line after rotation = (1.46, 5.46).


Y
Rotation
O'
O
5.46
4
30
0
X
1.46
4

# Example2:

Given a triangle with corner coordinates (0, 0), (1, 0) and (1, 1). Rotate the triangle by 90 degree anticlockwise direction and find out the new coordinates.

# Solution:

We rotate a polygon by rotating each vertex of it with the same rotation angle.

Given-

- Old corner coordinates of the triangle = A (0, 0), B(1, 0), C(1, 1)
- Rotation angle = $\theta = 90^o$

For Coordinates A(0, 0)

Let the new coordinates of corner A after rotation = ($X_{new}$, $Y_{new}$).

Applying the rotation equations, we have-

$X_{new}$

$= X_{old} \times \cos\theta - Y_{old} \times \sin\theta$

$= 0 \times \cos 90^o - 0 \times \sin 90^o$

$= 0$

$Y_{new}$

$= X_{old} \times \sin\theta + Y_{old} \times \cos\theta$

$= 0 \times \sin 90^o + 0 \times \cos 90^o$

$= 0$

Therefore, New coordinates of corner A after rotation = (0, 0).

For Coordinates B(1, 0)

Let the new coordinates of corner B after rotation = $(X_{new}, Y_{new})$.

$X_{new}$

$= X_{old} \times \cos\theta - Y_{old} \times \sin\theta$

$= 1 \times \cos 90^{o} - 0 \times \sin 90^{o}$

$= 0$

$Y_{new}$

$= X_{old} \times \sin\theta + Y_{old} \times \cos\theta$

$= 1 \times \sin 90^{o} + 0 \times \cos 90^{o}$

$= 1 + 0$

$= 1$

Therefore, New coordinates of corner B after rotation = (0, 1).

For Coordinates C(1, 1)

Let the new coordinates of corner C after rotation = $(X_{new}, Y_{new})$.

$X_{new}$

$= X_{old} \times \cos\theta - Y_{old} \times \sin\theta$

$= 1 \times \cos 90^{o} - 1 \times \sin 90^{o}$

$= 0 - 1$

$= -1$

$Y_{new}$

$= X_{old} \times \sin\theta + Y_{old} \times \cos\theta$

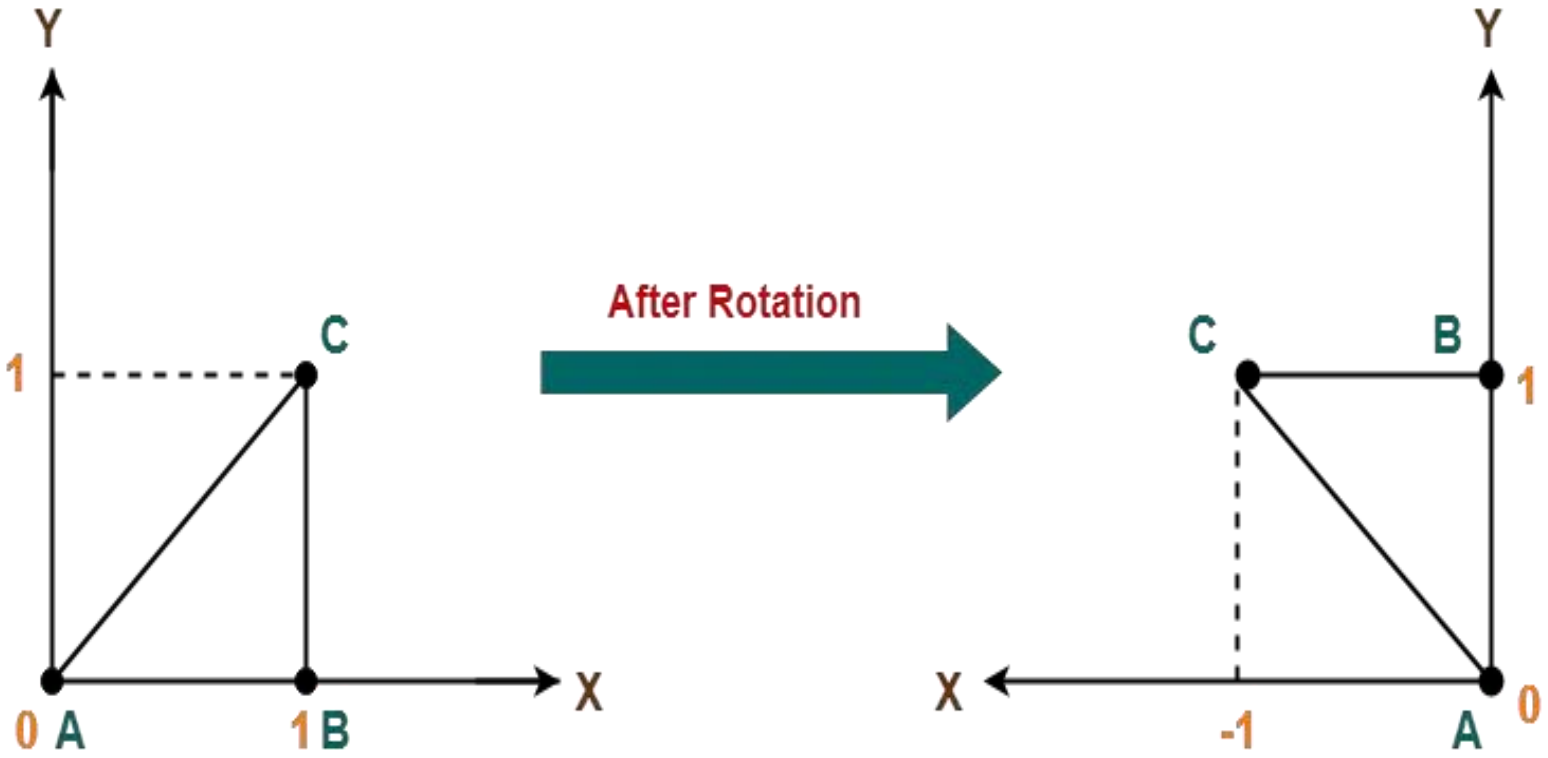
$= 1 \times \sin 90^{o} + 1 \times \cos 90^{o}$

$= 1 + 0$

$= 1$

Therefore, New coordinates of corner C after rotation = (-1, 1).

Therefore, New coordinates of the triangle after rotation = A (0, 0), B(0, 1), C(-1, 1).


Y
C
1
0 A
1 B
X
After Rotation
Y
C
B
1
X
-1
A
0

## 2D Scaling in Computer Graphics-

**In computer graphics, scaling is a process of modifying or altering the size of objects.**

- Scaling may be used to increase or reduce the size of object.
- Scaling subjects the coordinate points of the original object to change.
- Scaling factor determines whether the object size is to be increased or reduced.
- If scaling factor > 1, then the object size is increased.
- If scaling factor < 1, then the object size is reduced

Consider a point object O has to be scaled in a 2D plane.

Let-

- Initial coordinates of the object O = $(X_{old}, Y_{old})$
- Scaling factor for X-axis = $S_x$
- Scaling factor for Y-axis = $S_y$
- New coordinates of the object O after scaling = $(X_{new}, Y_{new})$

This scaling is achieved by using the following scaling equations-

- $X_{new} = X_{old} \times S_x$
- $Y_{new} = Y_{old} \times S_y$

In Matrix form, the above scaling equations may be represented as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

Scaling Matrix

For homogeneous coordinates, the above scaling matrix may be represented as a 3 x 3 matrix as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ 1 \end{bmatrix}$$

Scaling Matrix

(Homogeneous Coordinates Representation)

# PRACTICE EXAMPLES BASED ON 2D SCALING IN COMPUTER GRAPHICS

**Example01:**

Given a square object with coordinate points A(0, 3), B(3, 3), C(3, 0), D(0, 0). Apply the scaling parameter 2 towards X axis and 3 towards Y axis and obtain the new coordinates of the object.

**Solution-**

Given: Old corner coordinates of the square = A (0, 3), B(3, 3), C(3, 0), D(0, 0)

- Scaling factor along X axis = 2
- Scaling factor along Y axis = 3

**For Coordinates A(0, 3)**

Let the new coordinates of corner A after scaling = ($X_{new}$, $Y_{new}$).

Applying the scaling equations, we have-

- $X_{new} = X_{old} \times S_x = 0 \times 2 = 0$
- $Y_{new} = Y_{old} \times S_y = 3 \times 3 = 9$

Thus, New coordinates of corner A after scaling = (0, 9).

**For Coordinates B(3, 3)**

Let the new coordinates of corner B after scaling = ($X_{new}$, $Y_{new}$).

Applying the scaling equations, we have-

- $X_{new} = X_{old} \times S_x = 3 \times 2 = 6$
- $Y_{new} = Y_{old} \times S_y = 3 \times 3 = 9$

Thus, New coordinates of corner B after scaling = (6, 9).

**For Coordinates C(3, 0)**

Let the new coordinates of corner C after scaling = $(X_{new}, Y_{new})$.

Applying the scaling equations, we have-

- $X_{new} = X_{old} \times S_x = 3 \times 2 = 6$
- $Y_{new} = Y_{old} \times S_y = 0 \times 3 = 0$

Thus, New coordinates of corner C after scaling = (6, 0).

**For Coordinates D(0, 0)**

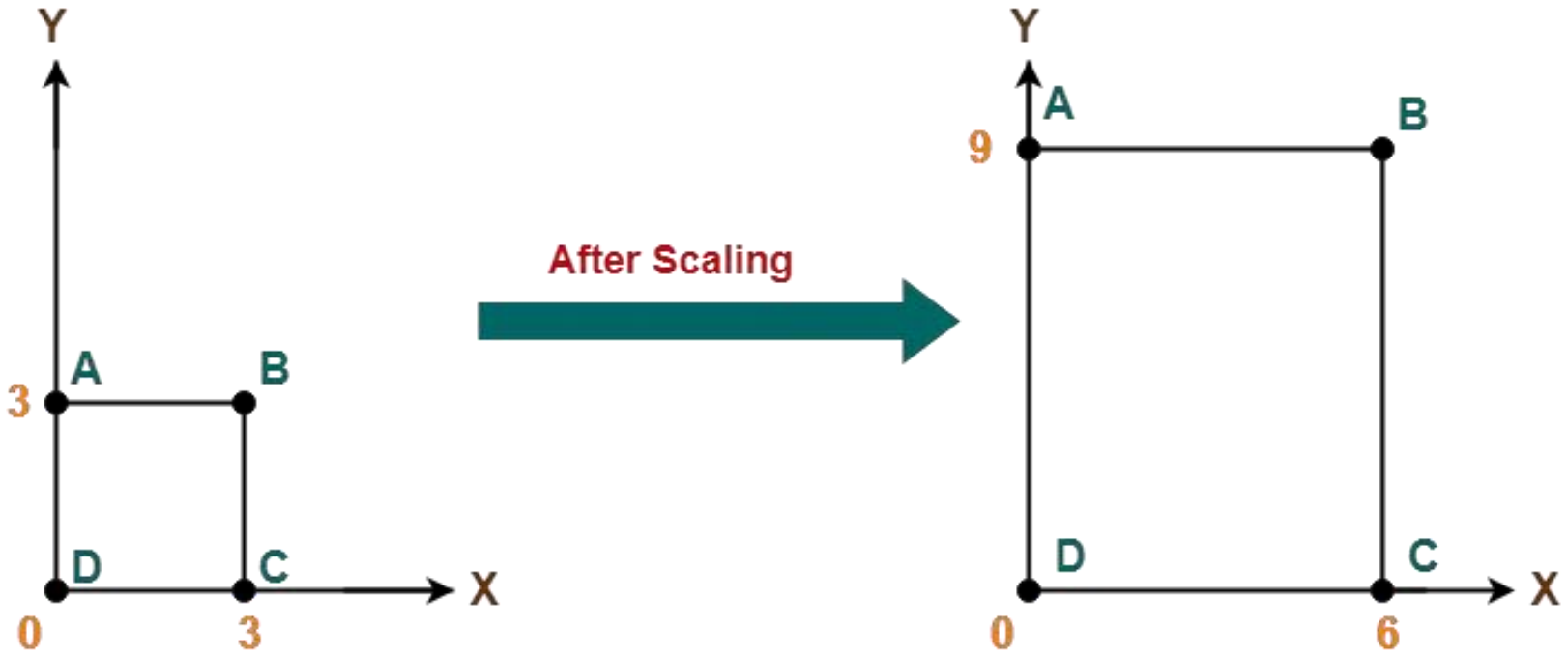
Let the new coordinates of corner D after scaling = $(X_{new}, Y_{new})$.

Applying the scaling equations, we have-

- $X_{new} = X_{old} \times S_x = 0 \times 2 = 0$
- $Y_{new} = Y_{old} \times S_y = 0 \times 3 = 0$

Thus, New coordinates of corner D after scaling = (0, 0).

Thus, New coordinates of the square after scaling = A (0, 9), B(6, 9), C(6, 0), D(0, 0).



# 2D Reflection in Computer Graphics

- Reflection is a kind of rotation where the angle of rotation is 180 degree.
- The reflected object is always formed on the other side of mirror.
- The size of reflected object is same as the size of original object.

Consider a point object O has to be reflected in a 2D plane.

Let-

- initial coordinates of the object O = $(X_{old}, Y_{old})$
- New coordinates of the reflected object O after reflection = $(X_{new}, Y_{new})$

## Reflection On X-Axis:

This reflection is achieved by using the following reflection equations-

- $X_{new} = X_{old}$
- $Y_{new} = -Y_{old}$

In Matrix form, the above reflection equations may be represented as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

**Reflection Matrix**

**(Reflection Along X Axis)**

For homogeneous coordinates, the above reflection matrix may be represented as a 3 x 3 matrix as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ 1 \end{bmatrix}$$

**Reflection Matrix**

**(Reflection Along X Axis)**

**(Homogeneous Coordinates Representation)**

## Reflection On Y-Axis:

This reflection is achieved by using the following reflection equations-

- $X_{new} = -X_{old}$
- $Y_{new} = Y_{old}$

In Matrix form, the above reflection equations may be represented as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

**Reflection Matrix**

**(Reflection Along Y Axis)**

For homogeneous coordinates, the above reflection matrix may be represented as a 3 x 3 matrix as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ 1 \end{bmatrix}$$

**Reflection Matrix**

**(Reflection Along Y Axis)**

**(Homogeneous Coordinates Representation)**

# PRACTICE EXAMPLES BASED ON 2D REFLECTION IN COMPUTER GRAPHICS

## Example1:

Given a triangle with coordinate points A(3, 4), B(6, 4), C(5, 6). Apply the reflection on the X axis and obtain the new coordinates of the object.

## Solution

Given:

- Old corner coordinates of the triangle = A (3, 4), B(6, 4), C(5, 6)
- Reflection has to be taken on the X axis

**For Coordinates A(3, 4)**

Let the new coordinates of corner A after reflection = $(X_{new}, Y_{new})$.

Applying the reflection equations, we have-

- $X_{new} = X_{old} = 3$
- $Y_{new} = -Y_{old} = -4$

Thus, New coordinates of corner A after reflection = (3, -4).

**For Coordinates B(6, 4)**

Let the new coordinates of corner B after reflection = $(X_{new}, Y_{new})$.

Applying the reflection equations, we have-

- $X_{new} = X_{old} = 6$
- $Y_{new} = -Y_{old} = -4$

Therefore, New coordinates of corner B after reflection = (6, -4).

**For Coordinates C(5, 6)**

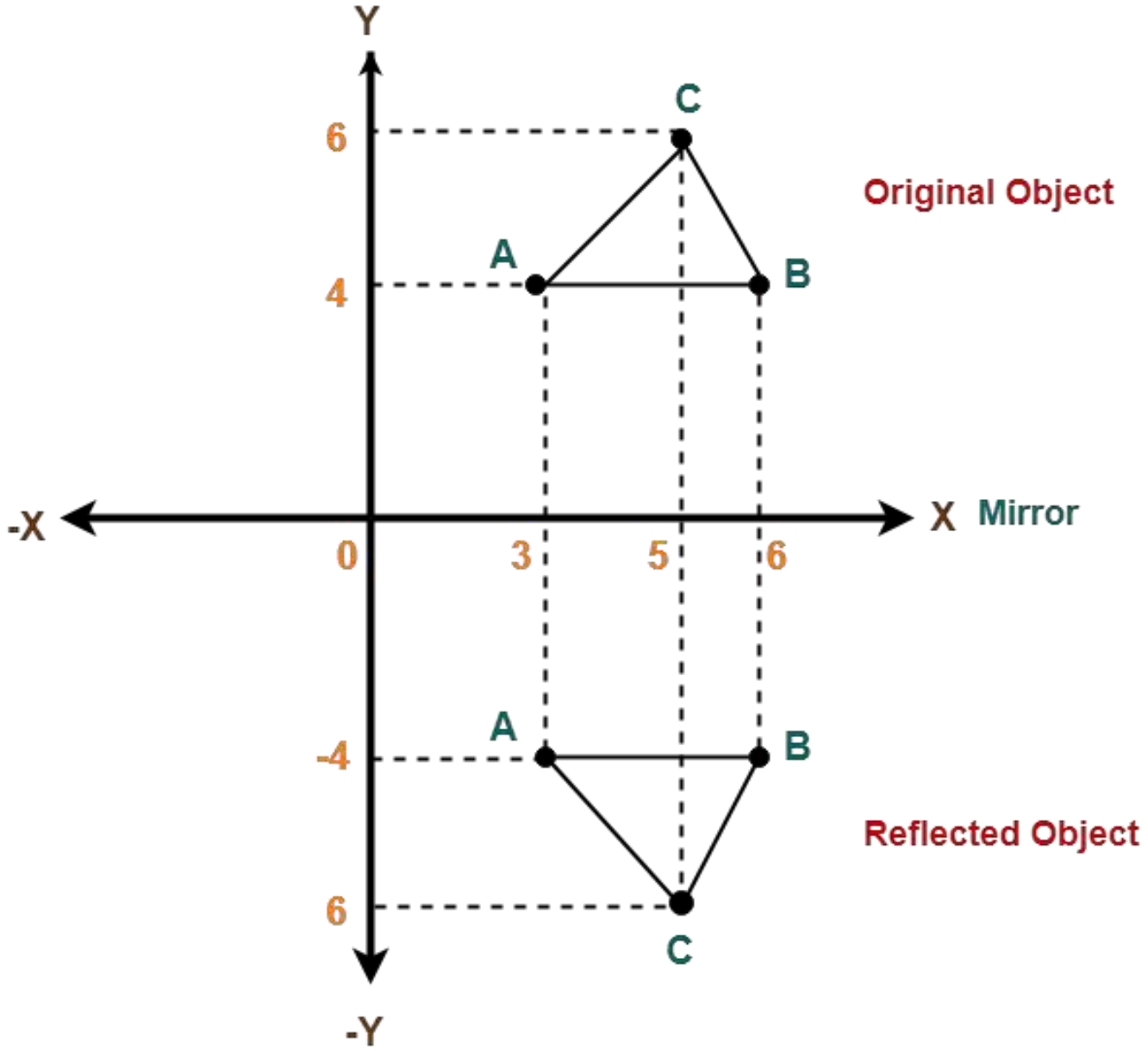
Let the new coordinates of corner C after reflection = ($X_{new}$, $Y_{new}$).

Applying the reflection equations, we have-

- $X_{new} = X_{old} = 5$
- $Y_{new} = -Y_{old} = -6$

The new coordinates of corner C after reflection = (5, -6).

Therefore, New coordinates of the triangle after reflection = A (3, -4), B(6, -4), C(5, -6).


Y
C
6
Original Object
A
B
4
-X
X
Mirror
0
3
5
6
A
B
-4
Reflected Object
6
C
-Y

**Example2:**

Given a triangle with coordinate points A(3, 4), B(6, 4), C(5, 6). Apply the reflection on the Y axis and obtain the new coordinates of the object.

**Solution**

Given:

- Old corner coordinates of the triangle = A (3, 4), B(6, 4), C(5, 6)
- Reflection has to be taken on the Y axis

**For Coordinates A(3, 4)**

Let the new coordinates of corner A after reflection = $(X_{new}, Y_{new})$.

Applying the reflection equations, we have-

- $X_{new} = -X_{old} = -3$
- $Y_{new} = Y_{old} = 4$

Therefore, New coordinates of corner A after reflection = (-3, 4).

## For Coordinates B(6, 4)

Let the new coordinates of corner B after reflection = $(X_{new}, Y_{new})$.

Applying the reflection equations, we have-

- $X_{new} = -X_{old} = -6$
- $Y_{new} = Y_{old} = 4$

Therefore, New coordinates of corner B after reflection = (-6, 4).

## For Coordinates C(5, 6)

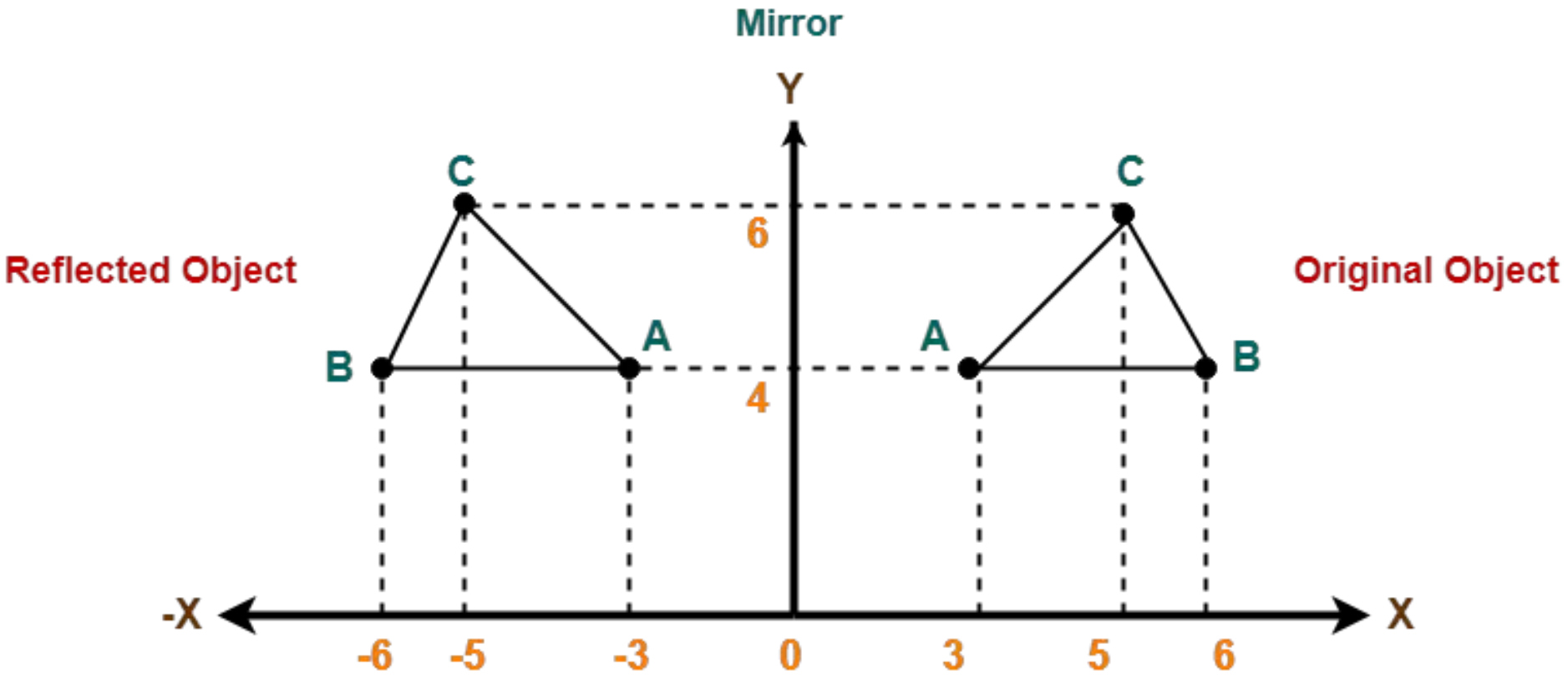
Let the new coordinates of corner C after reflection = $(X_{new}, Y_{new})$.

Applying the reflection equations, we have-

- $X_{new} = -X_{old} = -5$
- $Y_{new} = Y_{old} = 6$

Therefore, New coordinates of corner C after reflection = (-5, 6).

Therefore, New coordinates of the triangle after reflection = A (-3, 4), B(-6, 4), C(-5, 6).

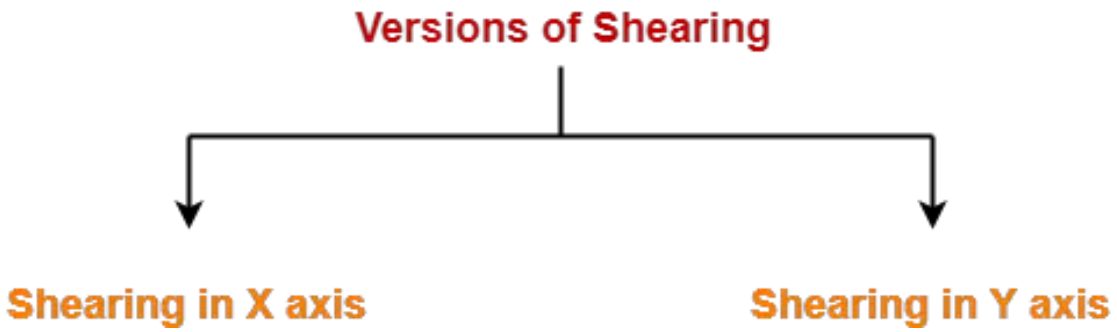


# 2D Shearing in Computer Graphics

In Computer graphics, 2D Shearing is an ideal technique to change the shape of an existing object in a two dimensional plane.

In a two dimensional plane, the object size can be changed along X direction as well as Y direction.

So, there are two versions of shearing-




1.Shearing in X direction
2.Shearing in Y direction

Consider a point object O has to be sheared in a 2D plane.

Let-

- Initial coordinates of the object O = $(X_{old}, Y_{old})$
- Shearing parameter towards X direction = $Sh_x$
- Shearing parameter towards Y direction = $Sh_y$
- New coordinates of the object O after shearing = $(X_{new}, Y_{new})$

## Shearing in X Axis

Shearing in X axis is achieved by using the following shearing equations-

- $X_{new} = X_{old} + Sh_x \times Y_{old}$
- $Y_{new} = Y_{old}$

In Matrix form, the above shearing equations may be represented as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 1 & Sh_x \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

**Shearing Matrix**

**(In X axis)**

For homogeneous coordinates, the above shearing matrix may be represented as a 3 x 3 matrix as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ Sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ 1 \end{bmatrix}$$

**Shearing Matrix**

**(In X axis)**

**(Homogeneous Coordinates Representation)**

## Shearing in Y Axis-

Shearing in Y axis is achieved by using the following shearing equations-

- $X_{new} = X_{old}$
- $Y_{new} = Y_{old} + Sh_y \times X_{old}$

In Matrix form, the above shearing equations may be represented as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ Sh_y & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

**Shearing Matrix**

**(In Y axis)**

For homogeneous coordinates, the above shearing matrix may be represented as a 3 x 3 matrix as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & Sh_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ 1 \end{bmatrix}$$

**Shearing Matrix**

**(In Y axis)**

**(Homogeneous Coordinates Representation)**

## PRACTICE EXAMPLE BASED ON 2D SHEARING IN COMPUTER GRAPHICS

**Example1:**

Given a triangle with points (1, 1), (0, 0) and (1, 0). Apply shear parameter 2 on X axis and 2 on Y axis and find out the new coordinates of the object.

**Solution**

Given:

- Old corner coordinates of the triangle = A (1, 1), B(0, 0), C(1, 0)
- Shearing parameter towards X direction ($Sh_x$) = 2
- Shearing parameter towards Y direction ($Sh_y$) = 2

## Shearing in X Axis-

### For Coordinates A(1, 1)

Let the new coordinates of corner A after shearing = $(X_{new}, Y_{new})$.

Applying the shearing equations, we have-

- $X_{new} = X_{old} + Sh_x \times Y_{old} = 1 + 2 \times 1 = 3$
- $Y_{new} = Y_{old} = 1$

Therefore, New coordinates of corner A after shearing = (3, 1).

### For Coordinates B(0, 0)

Let the new coordinates of corner B after shearing = $(X_{new}, Y_{new})$.

Applying the shearing equations, we have-

- $X_{new} = X_{old} + Sh_x \times Y_{old} = 0 + 2 \times 0 = 0$
- $Y_{new} = Y_{old} = 0$

Therefore, New coordinates of corner B after shearing = (0, 0).

## For Coordinates C(1, 0)

Let the new coordinates of corner C after shearing = ($X_{new}$, $Y_{new}$).

Applying the shearing equations, we have-

- $X_{new} = X_{old} + Sh_x \times Y_{old} = 1 + 2 \times 0 = 1$
- $Y_{new} = Y_{old} = 0$

Therefore, New coordinates of corner C after shearing = (1, 0).

Therefore, New coordinates of the triangle after shearing in X axis = A (3, 1), B(0, 0), C(1, 0).

# Shearing in Y Axis

## For Coordinates A(1, 1)

Let the new coordinates of corner A after shearing = $(X_{new}, Y_{new})$.

Applying the shearing equations, we have-

- $X_{new} = X_{old} = 1$
- $Y_{new} = Y_{old} + Sh_y \times X_{old} = 1 + 2 \times 1 = 3$

Thus, New coordinates of corner A after shearing = (1, 3).

## For Coordinates B(0, 0)

Let the new coordinates of corner B after shearing = $(X_{new}, Y_{new})$.

Applying the shearing equations, we have-

- $X_{new} = X_{old} = 0$
- $Y_{new} = Y_{old} + Sh_y \times X_{old} = 0 + 2 \times 0 = 0$

Therefore, New coordinates of corner B after shearing = (0, 0).

## For Coordinates C(1, 0)

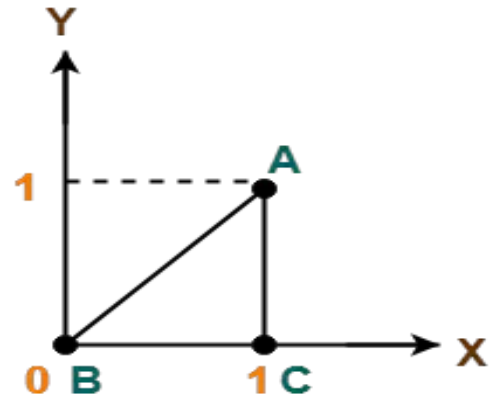
Let the new coordinates of corner C after shearing = $(X_{new}, Y_{new})$.

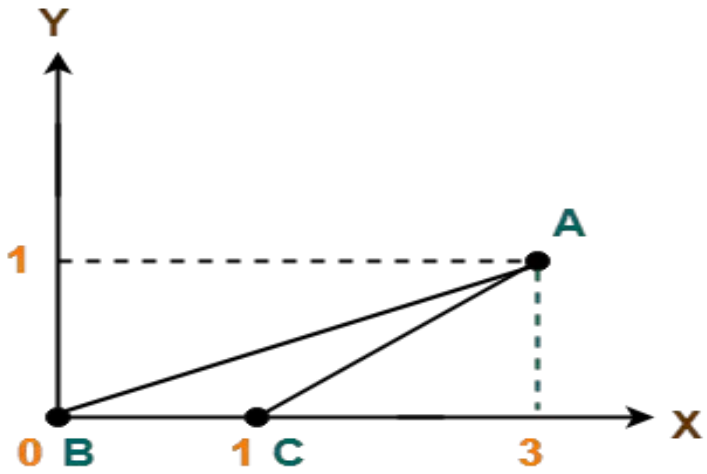
Applying the shearing equations, we have-

- $X_{new} = X_{old} = 1$
- $Y_{new} = Y_{old} + Sh_y \times X_{old} = 0 + 2 \times 1 = 2$

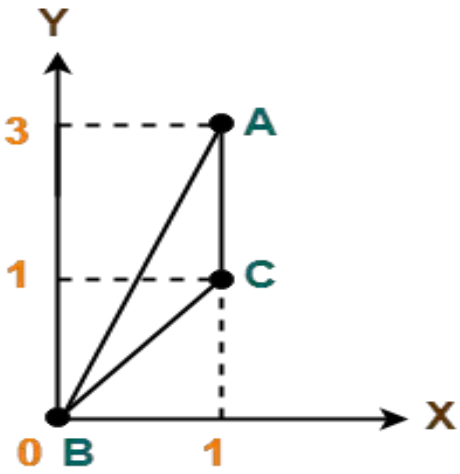
Therefore, New coordinates of corner C after shearing = (1, 2).

Therefore, New coordinates of the triangle after shearing in Y axis = A (1, 3), B(0, 0), C(1, 2).

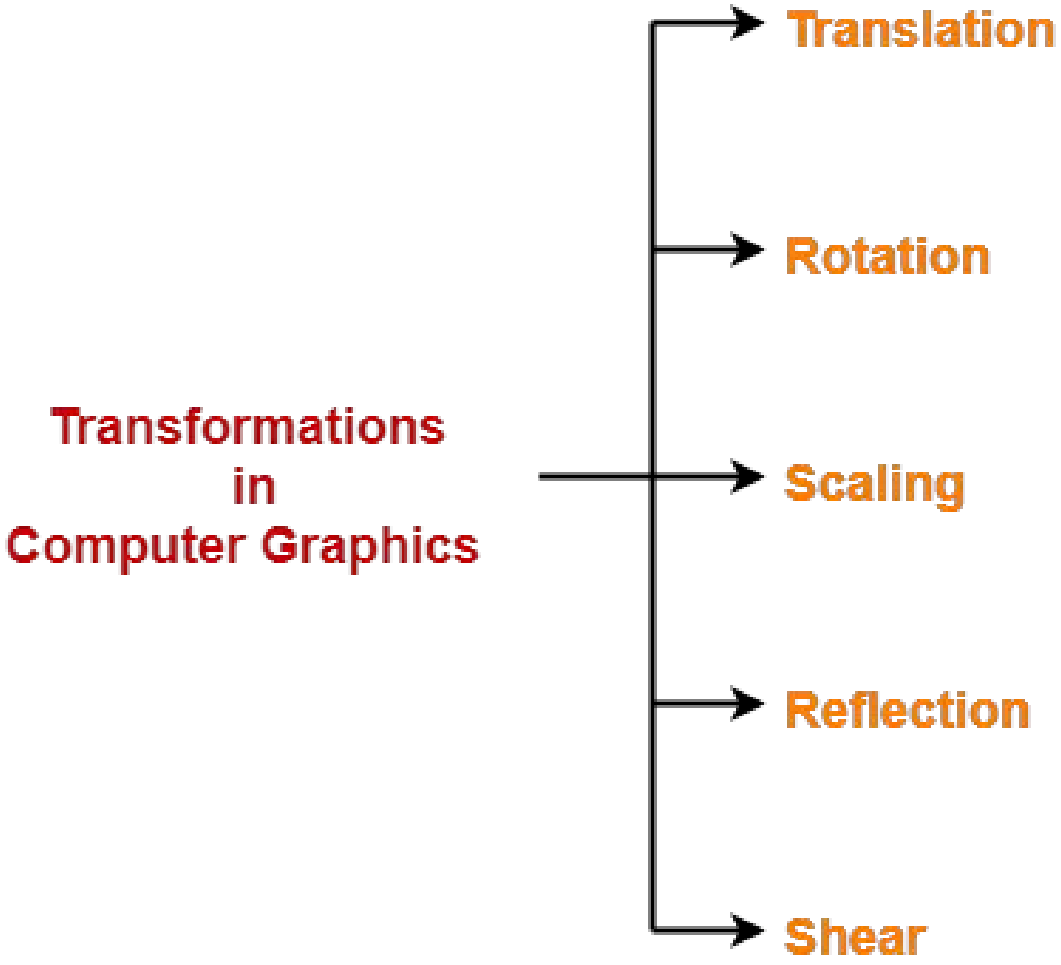



Shearing in X Axis




Shearing in Y Axis




## END OF 2D TRANSOFORMATIONS IN COMPUTER GRAPHICS

Prof. Jelili O. Oyelade

# Computer Graphics and Animation (CSC433)

# Introduction to Transformation

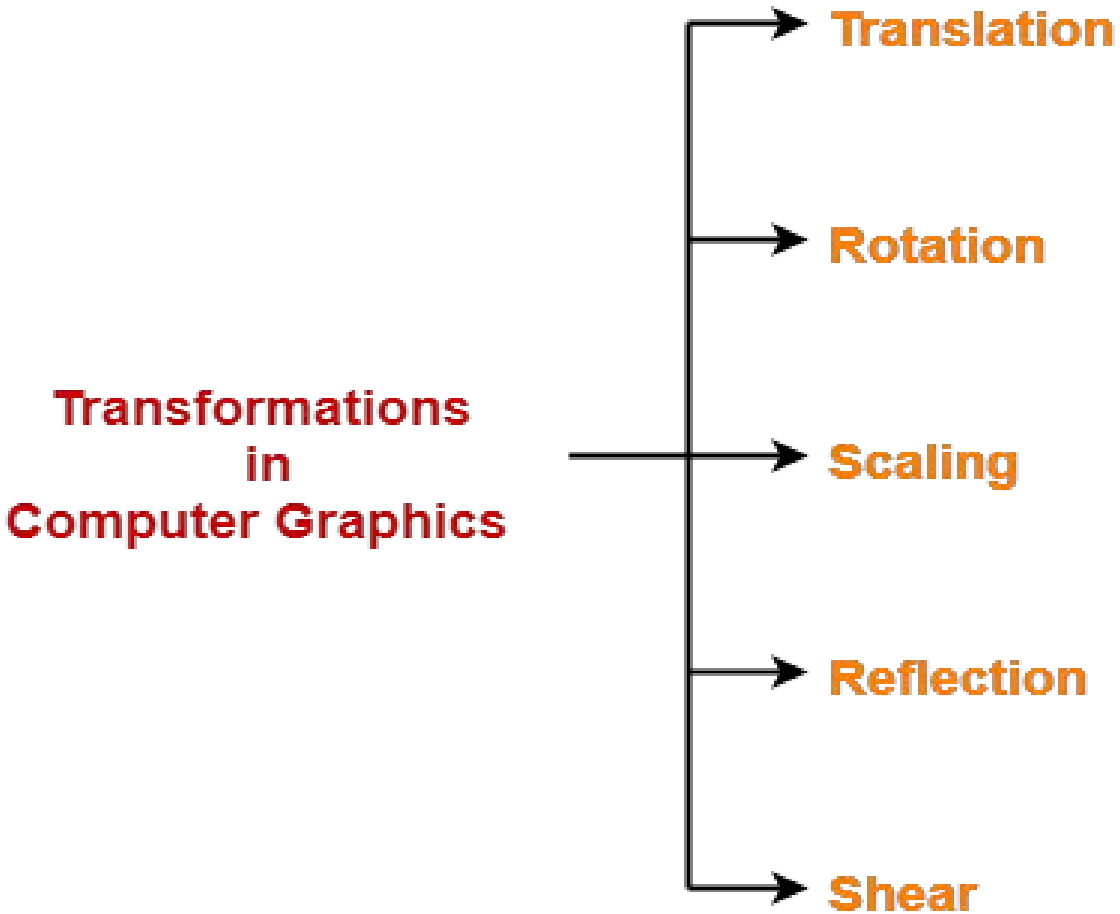
- In Computer graphics,Transformation is a process of modifying and re-positioning the existing graphics.

- The process of using computers for drawing by providing facility to user to view the object from different angles, enlarging or reducing the scale or shape of object is called Transformation

  - 2D Transformations take place in a two dimensional plane.
  - 3D Transformations take place in a three dimensional plane.
  - Transformations are helpful in changing the position, size, orientation, shape etc of the object.

# Transformation Techniques

- In computer graphics, various transformation techniques are:



# 2D Translation in Computer Graphics

**2D Translation is a process of moving an object from one position to another in a two dimensional plane.**

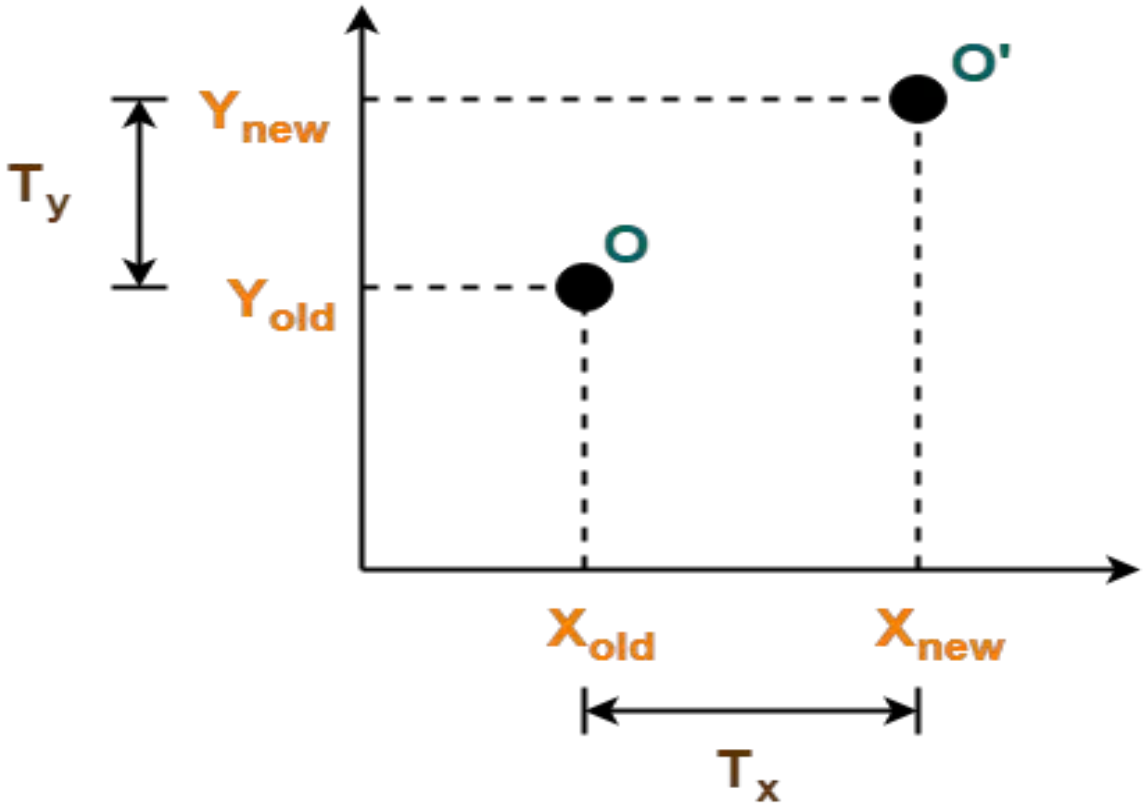
Let consider a point object O has to be moved from one position to another in a 2D plane.

Let:

- Initial coordinates of the object O = $(X_{old}, Y_{old})$
- New coordinates of the object O after translation = $(X_{new}, Y_{new})$
- Translation vector or Shift vector = $(T_x, T_y)$

Given a Translation vector $(T_x, T_y)$-

- $T_x$ defines the distance the $X_{old}$ coordinate has to be moved.
- $T_y$ defines the distance the $Y_{old}$ coordinate has to be moved.




2D Translation in Computer Graphics

This translation is achieved by adding the translation coordinates to the old coordinates of the object as-

- $X_{new} = X_{old} + T_x$ (This denotes translation towards X axis)
- $Y_{new} = Y_{old} + T_y$ (This denotes translation towards Y axis)

In Matrix form, the above translation equations may be represented as:

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$

**Translation Matrix**

- The homogeneous coordinates representation of (X, Y) is (X, Y, 1).
- Through this representation, all the transformations can be performed using matrix / vector multiplications.

The above translation matrix may be represented as a 3 x 3 matrix as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ 1 \end{bmatrix}$$

**Translation Matrix**

**(Homogeneous Coordinates Representation)**

# Example1:

Given a circle C with radius 10 and center coordinates (1, 4). Apply the translation with distance 5 towards X axis and 1 towards Y axis. Obtain the new coordinates of C without changing its radius.

# Solution:

Given-

- Old center coordinates of C = $(X_{old}, Y_{old}) = (1, 4)$
- Translation vector = $(T_x, T_y) = (5, 1)$

Let the new center coordinates of C = $(X_{new}, Y_{new})$.

Applying the translation equations, we have-

- $X_{new} = X_{old} + T_x = 1 + 5 = 6$
- $Y_{new} = Y_{old} + T_y = 4 + 1 = 5$
- Thus, New center coordinates of C = (6, 5).

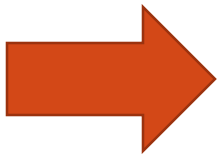
## Alternatively,

- In matrix form, the new center coordinates of C after translation may be obtained as-

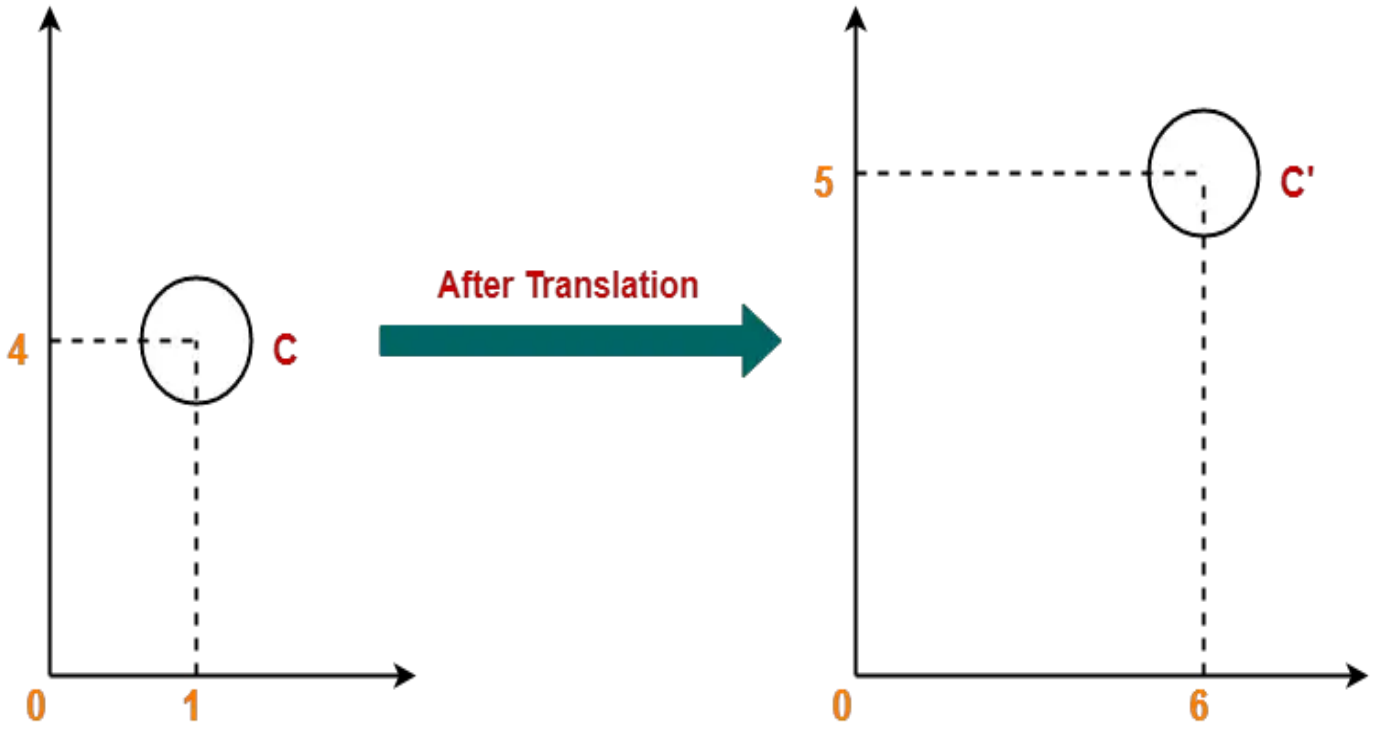
$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \end{bmatrix} + \begin{bmatrix} 5 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 6 \\ 5 \end{bmatrix}$$



Therefore, New center coordinates of C = (6, 5).


4
C
0
1
After Translation
5
C'
0
6

# Example2:

Given a square with coordinate points A(0, 3), B(3, 3), C(3, 0), D(0, 0). Apply the translation with distance 1 towards X axis and 1 towards Y axis. Obtain the new coordinates of the square.

# Solution:

Given-

- Old coordinates of the square = A (0, 3), B(3, 3), C(3, 0), D(0, 0)
- Translation vector = $(T_x, T_y) = (1, 1)$

For Coordinates A(0, 3)

Let the new coordinates of corner A = $(X_{new}, Y_{new})$.

Applying the translation equations, we have-

- $X_{new} = X_{old} + T_x = 0 + 1 = 1$
- $Y_{new} = Y_{old} + T_y = 3 + 1 = 4$

Thus, New coordinates of corner A = (1, 4).

For Coordinates B(3, 3)

Let the new coordinates of corner B = ($X_{new}$, $Y_{new}$).

Applying the translation equations, we have-

- $X_{new} = X_{old} + T_x = 3 + 1 = 4$
- $Y_{new} = Y_{old} + T_y = 3 + 1 = 4$

Thus, New coordinates of corner B = (4, 4).

For Coordinates C(3, 0)

Let the new coordinates of corner C = ($X_{new}$, $Y_{new}$).

Applying the translation equations, we have-

- $X_{new} = X_{old} + T_x = 3 + 1 = 4$
- $Y_{new} = Y_{old} + T_y = 0 + 1 = 1$

Thus, New coordinates of corner C = (4, 1).

For Coordinates D(0, 0)

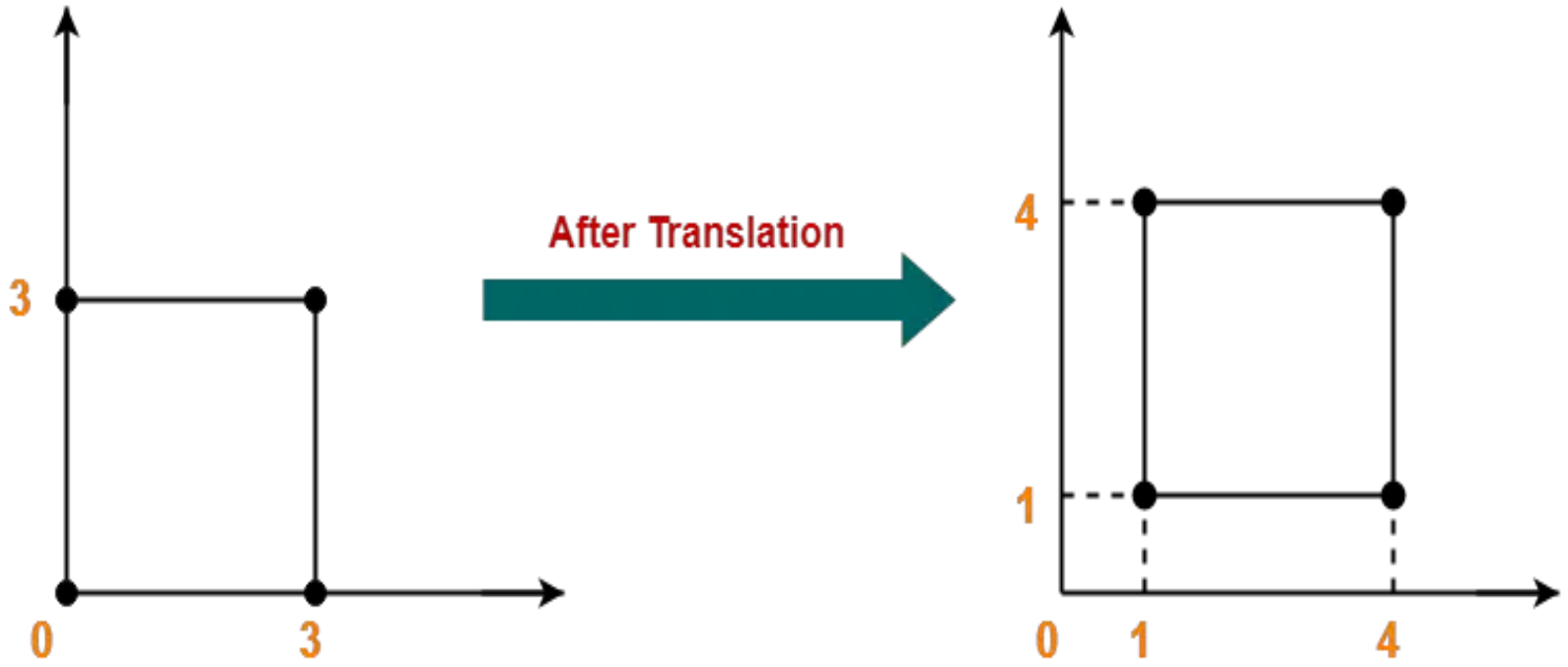
Let the new coordinates of corner D = ($X_{new}$, $Y_{new}$).

Applying the translation equations, we have-

- $X_{new} = X_{old} + T_x = 0 + 1 = 1$
- $Y_{new} = Y_{old} + T_y = 0 + 1 = 1$

Thus, New coordinates of corner D = (1, 1).

Therefore, New coordinates of the square = A (1, 4), B(4, 4), C(4, 1), D(1, 1).


3
0
3
After Translation
4
1
0
1
4

**Problem:** Given a Point with coordinates (2, 4). Apply the translation with distance 4 towards x-axis and 2 towards the y-axis. Find the new coordinates without changing the radius?

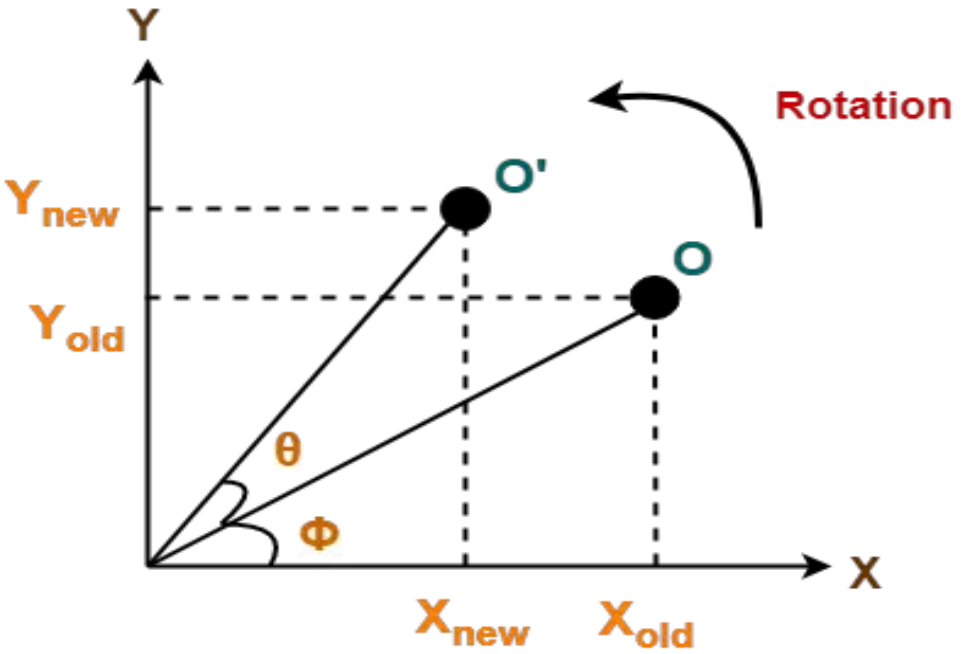
# 2D Rotation in Computer Graphics

2D Rotation is a process of rotating an object with respect to an angle in a two dimensional plane

Let consider a point object O has to be rotated from one angle to another in a 2D plane.

Let:

- Initial coordinates of the object O = ($X_{old}$, $Y_{old}$)
- Initial angle of the object O with respect to origin = Φ
- Rotation angle = θ
- New coordinates of the object O after rotation = ($X_{new}$, $Y_{new}$)




2D Rotation in Computer Graphics

This rotation is achieved by using the following rotation equations-

- $X_{new} = X_{old} \times \cos\theta - Y_{old} \times \sin\theta$
- $Y_{new} = X_{old} \times \sin\theta + Y_{old} \times \cos\theta$

In Matrix form, the above rotation equations may be represented as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

Rotation Matrix

For homogeneous coordinates, the above rotation matrix may be represented as a 3 x 3 matrix as-

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ 1 \end{bmatrix}$$

**Rotation Matrix**

**(Homogeneous Coordinates Representation)**

# Example1:

Given a line segment with starting point as (0, 0) and ending point as (4, 4). Apply 30 degree rotation anticlockwise direction on the line segment and find out the new coordinates of the line.

# Solution-

We rotate a straight line by its end points with the same angle. Then, we re-draw a line between the new end points.

Given-

- Old ending coordinates of the line = $(X_{old}, Y_{old}) = (4, 4)$
- Rotation angle = $\theta = 30^o$

Let new ending coordinates of the line after rotation = $(X_{new}, Y_{new})$.

Applying the rotation equations, we have:

$X_{new}$

$= X_{old} \times \cos\theta - Y_{old} \times \sin\theta$

$= 4 \times \cos 30^o - 4 \times \sin 30^o$

$= 4 \times (\sqrt{3} / 2) - 4 \times (1 / 2)$

$= 2\sqrt{3} - 2$

$= 2(\sqrt{3} - 1)$

$= 2(1.73 - 1)$

$= 1.46$

$Y_{new}$

$= X_{old} \times \sin\theta + Y_{old} \times \cos\theta$

$= 4 \times \sin 30^o + 4 \times \cos 30^o$

$= 4 \times (1 / 2) + 4 \times (\sqrt{3} / 2)$

$= 2 + 2\sqrt{3}$

$= 2(1 + \sqrt{3})$

$= 2(1 + 1.73)$

$= 5.46$

Therefore, New ending coordinates of the line after rotation

= (1.46, 5.46).

**Alternatively,**

In matrix form, the new ending coordinates of the line after rotation may be obtained as

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \end{bmatrix}$$

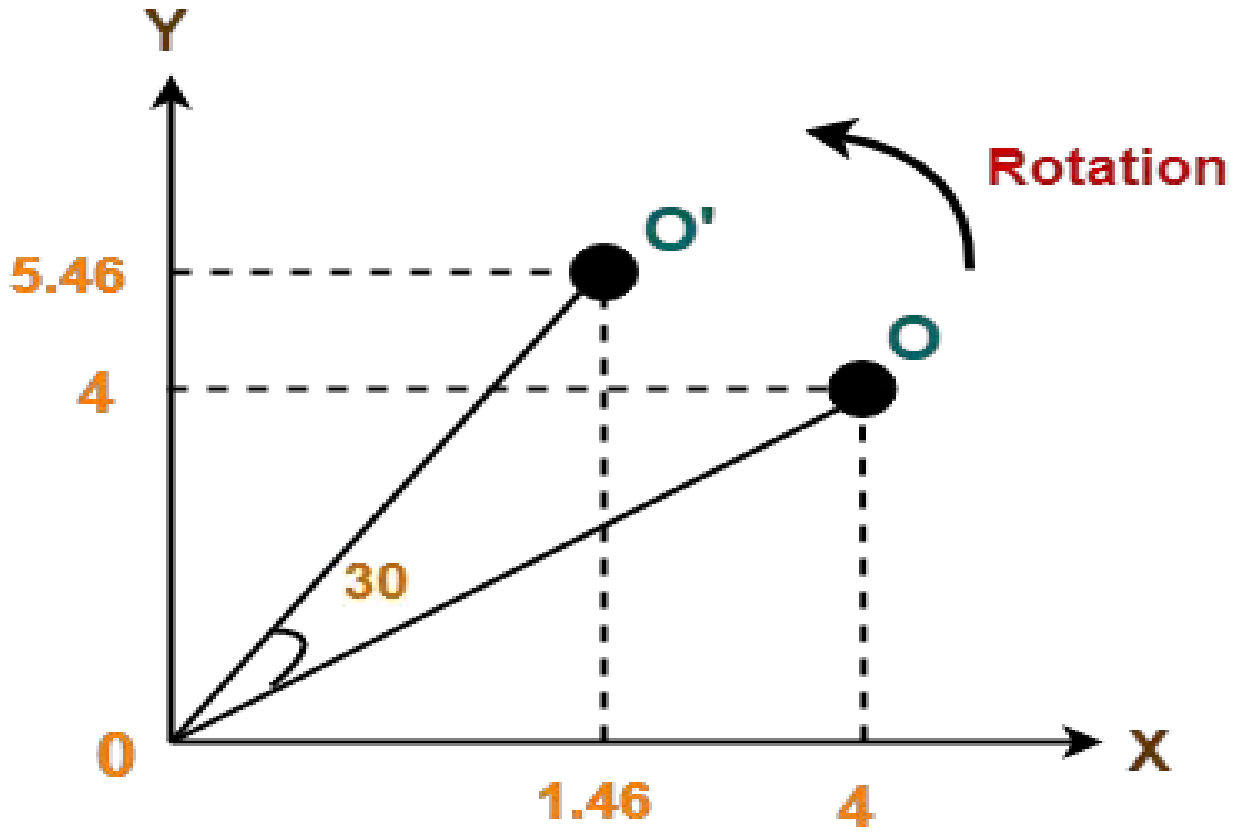
$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} \cos 30 & -\sin 30 \\ \sin 30 & \cos 30 \end{bmatrix} \times \begin{bmatrix} 4 \\ 4 \end{bmatrix}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 4 \times \cos 30 - 4 \times \sin 30 \\ 4 \times \sin 30 + 4 \times \cos 30 \end{bmatrix}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 4 \times \cos 30 - 4 \times \sin 30 \\ 4 \times \sin 30 + 4 \times \cos 30 \end{bmatrix}$$

$$\begin{bmatrix} X_{new} \\ Y_{new} \end{bmatrix} = \begin{bmatrix} 1.46 \\ 5.46 \end{bmatrix}$$

Therefore, New ending coordinates of the line after rotation = (1.46, 5.46).


Y
Rotation
O'
O
5.46
4
30
0
X
1.46
4

# Example2:

Given a triangle with corner coordinates (0, 0), (1, 0) and (1, 1). Rotate the triangle by 90 degree anticlockwise direction and find out the new coordinates.

# Solution:

We rotate a polygon by rotating each vertex of it with the same rotation angle.

Given-

- Old corner coordinates of the triangle = A (0, 0), B(1, 0), C(1, 1)
- Rotation angle = $\theta = 90^o$

For Coordinates A(0, 0)

Let the new coordinates of corner A after rotation = ($X_{new}$, $Y_{new}$).

Applying the rotation equations, we have-

$X_{new}$

$= X_{old} \times \cos\theta - Y_{old} \times \sin\theta$

$= 0 \times \cos 90^o - 0 \times \sin 90^o$

$= 0$

$Y_{new}$

$= X_{old} \times \sin\theta + Y_{old} \times \cos\theta$

$= 0 \times \sin 90^o + 0 \times \cos 90^o$

$= 0$

Therefore, New coordinates of corner A after rotation = (0, 0).

For Coordinates B(1, 0)

Let the new coordinates of corner B after rotation = $(X_{new}, Y_{new})$.

$X_{new}$

$= X_{old} \times \cos\theta - Y_{old} \times \sin\theta$

$= 1 \times \cos 90^{o} - 0 \times \sin 90^{o}$

$= 0$

$Y_{new}$

$= X_{old} \times \sin\theta + Y_{old} \times \cos\theta$

$= 1 \times \sin 90^{o} + 0 \times \cos 90^{o}$

$= 1 + 0$

$= 1$

Therefore, New coordinates of corner B after rotation = (0, 1).

For Coordinates C(1, 1)

Let the new coordinates of corner C after rotation = $(X_{new}, Y_{new})$.

$X_{new}$

$= X_{old} \times \cos\theta - Y_{old} \times \sin\theta$

$= 1 \times \cos90^{o} - 1 \times \sin90^{o}$

$= 0 - 1$

$= -1$

$Y_{new}$

$= X_{old} \times \sin\theta + Y_{old} \times \cos\theta$

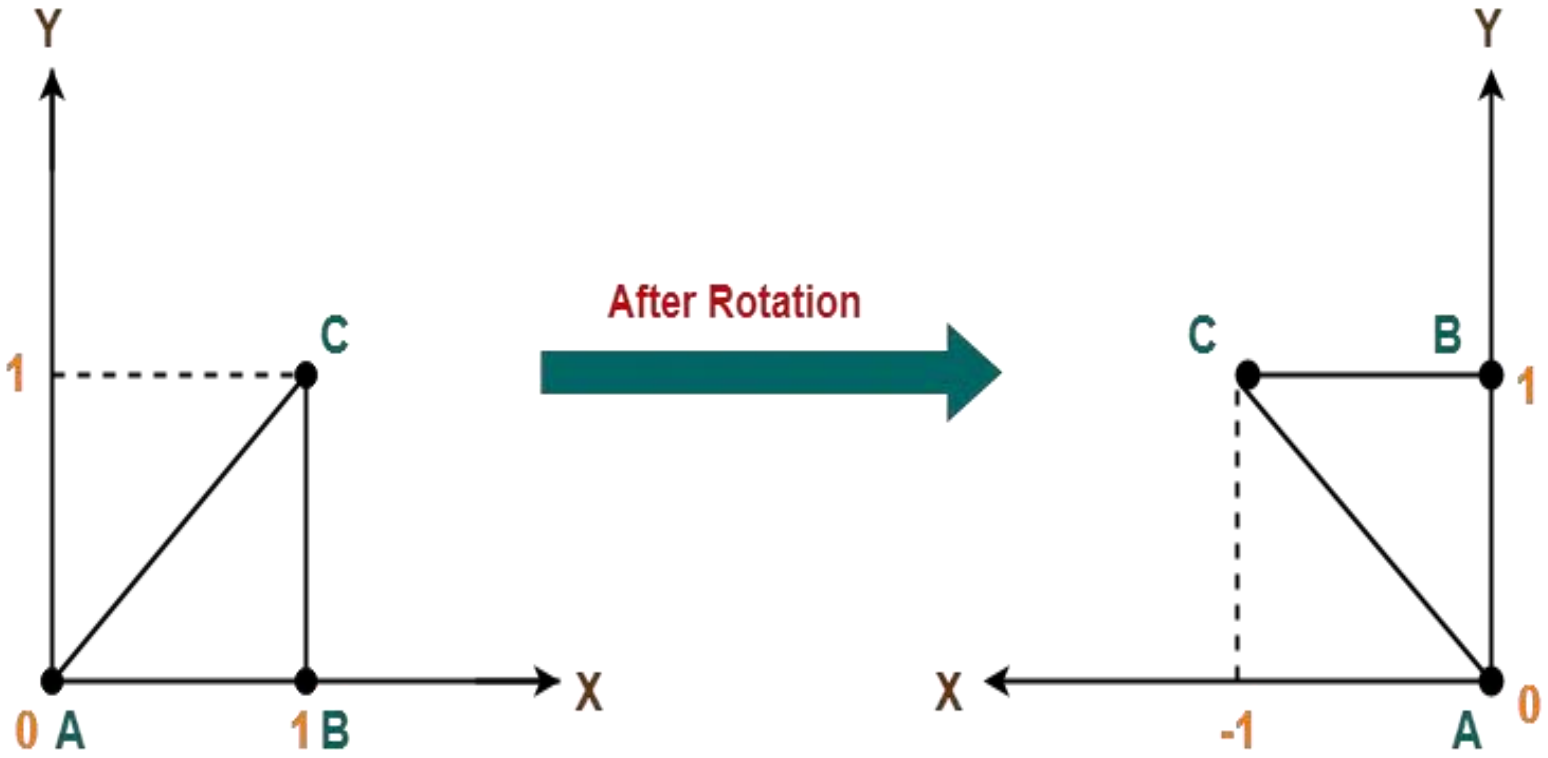
$= 1 \times \sin90^{o} + 1 \times \cos90^{o}$

$= 1 + 0$

$= 1$

Therefore, New coordinates of corner C after rotation = (-1, 1).

Therefore, New coordinates of the triangle after rotation = A (0, 0), B(0, 1), C(-1, 1).


Y
C
1
0 A
1 B
X
After Rotation
Y
C
B
1
X
-1
A
0

## GRAPHICS IMAGES

### BITMAP VERSES VECTOR

There are two main types of graphical images- Bitmap and Vector. A Bitmap (.bmp) is constructed out of many different blocks of colours, also known as pixels. The computer has to store information about every single pixel, which often results in the Bitmap file size being rather large. A bitmap is created when either a photograph or scanning in something onto the PC. A Vector is created by graphical packages out of shapes. A Vector graphic (.svp) has usually quite a small file size. They do not lose quality when zoomed in, as they are scalable.

Vector and bitmap images are both pictures on a screen, but they have different compositions and focuses. Bitmaps are made of pixels, while vector images are software-created and based on mathematical calculations Bitmaps are not only more common in everyday life but are easier to use. You can quickly convert one format of bitmap image into another, and you can't turn a bitmap into a vector without special software.

Vector images are generally smoother and more usable, and you can scale them freely without sacrificing quality. In general, vectors are for making scalable work files, while bitmaps are for making sharable final products.

### LOSSY AND LOSSLESS IMAGES

Lossy compression removes unnecessary information from the image in order to make the file size smaller; this therefore reduces the size of the file. However, once a file has been compressed, there is no way of getting the original file back. This is usually used when working with video and/or sound.

Lossless compression breaks the file into a smaller form in order to put it back together again. It will not remove any of the original files. It will recreate the file identically, except it will compress the file down in order to make it of a smaller file size. This is usually used when working with text or spreadsheet files.

### DOT PER INCH

This is the measurement of how many dots that will fit into linear inch, not square inch. This is used to measure the resolution of an image, whether it is printed out or on a computer screen. On a computer, dpi is usually used to measure how sharp the display on the screen is. The DPI can

vary depending on the size of the screen. When printing, DPI is the measurement of how many ink dots a printer can place within a square inch. Printers with low DPI may form jagged lined and pixelated, low quality images. DPI is used to measure the resolution of an image both on screen and in print. As the name suggests, the DPI measures how many dots fit into a linear inch. Therefore, the higher the DPI, the more detail can be shown in an image.

It should be noted that DPI is not dots per square inch. Since a 600 dpi printer can print 600 dots both horizontally and vertically per inch, it actually prints 360,000 (600 x 600) dots per square inch. Also, since most monitors have a native resolution of 72 or 96 pixels per inch, they cannot display a 300 dpi image in actual size. Instead, when viewed at 100%, the image will look much larger than the print version because the pixels on the screen take up more space than the dots on the paper.

**IMAGE FILE FORMAT**

1. **GIF- Graphics Interchange Formats:** The GIF format was created by CompuServe. It supports 256 colors. GIF format is the most popular on the Internet because of its compact size. It is ideal for small icons used for navigational purpose and simple diagrams. GIF creates a table of up to 256 colors from a pool of 16 million. If the image has less than 256 colors, GIF can easily render the image without any loss of quality. When the image contains more colors, GIF uses algorithms to match the colors of the image with the palette of optimum set of 256 colors available. Better algorithms search the image to find and the optimum set of 256 colors.

   Thus, GIF format is lossless only for the image with 256 colors or less. In case of a rich, true color image GIF may lose 99.998% of the colors. GIF files can be saved with a maximum of 256 colors. This makes it is a poor format for photographic images. GIFs can be animated, which is another reason they became so successful. Most animated banner ads are GIFs. GIFs allow single bit transparency that is when you are creating your image, you can specify which color is to be transparent. This provision allows the background colors of the web page to be shown through the image.
2. **JPEG- Joint Photographic Experts Group:** The JPEG format was developed by the Joint Photographic Experts Group. JPEG files are bitmapped images. It stores information as 24-bit color. This is the format of choice for nearly all photograph images on the internet. Digital

cameras save images in a JPEG format by default. It has become the main graphics file format for the World Wide Web and any browser can support it without plug-ins. In order to make the file small, JPEG uses lossy compression. It works well on photographs, artwork and similar materials but not so well on lettering, simple cartoons or line drawings. JPEG images work much better than GIFs. Though JPEG can be interlaced, still this format lacks many of the other special abilities of GIFs, like animations and transparency, but they really are only for photos.

3. **PNG- Portable Network Graphics:** PNG is the only lossless format that web browsers support. PNG supports 8 bit, 24 bits, 32 bits and 48 bits data types. One version of the format PNG-8 is similar to the GIF format. But PNG is the superior to the GIF. It produces smaller files and with more options for colors. It supports partial transparency also. PNG-24 is another flavor of PNG, with 24-bit color supports, allowing ranges of color akin to high color JPEG. PNG-24 is in no way a replacement format for JPEG because it is a lossless compression format. This means that file size can be rather big against a comparable JPEG. Also, PNG supports for up to 48 bits of color information.
4. **TIFF- Tagged Image File Format:** The TIFF format was developed by the Aldus Corporation in the 1980 and was later supported by Microsoft. TIFF file format is widely used bitmapped file format. It is supported by many image editing applications, software used by scanners and photo retouching programs. TIFF can store many different types of image ranging from 1 bit image, grayscale image, 8-bit color image, 24-bit RGB image etc. TIFF files originally use lossless compression. Today TIFF files also use lossy compression according to the requirement. Therefore, it is a very flexible format. This file format is suitable when the output is printed. Multi-page documents can be stored as a single TIFF file and that is way this file format is so popular. The TIFF format is now used and controlled by Adobe.
5. **BMP- Bitmap:** The bitmap file format (BMP) is a very basic format supported by most Windows applications. BMP can store many different types of image: 1 bit image, grayscale image, 8-bit color image, 24-bit RGB image etc. BMP files are uncompressed. Therefore, these are not suitable for the internet. BMP files can be compressed using lossless data compression algorithms.
6. **EPS- Encapsulated Postscript:** The EPS format is a vector based graphic. EPS is popular

for saving image files because it can be imported into nearly any kind of application. This file format is suitable for printed documents. Main disadvantage of this format is that it requires more storage as compare to other formats.

7. **PDF- Portable Document Format:** PDF format is vector graphics with embedded pixel graphics with many compression options. When your document is ready to be shared with others or for publication. This is only format that is platform independent. If you have Adobe Acrobat you can print from any document to a PDF file. From illustrator you can save as .PDF.
8. **EXIF- Exchange Image File:** Exif is an image format for digital cameras. A variety of tags are available to facilitate higher quality printing, since information about the camera and picture - taking condition can be stored and used by printers for possible color correction algorithms.it also includes specification of file format for audio that accompanies digital images.
9. **WMF- Windows Metafile:** WMF is the vector file format for the MS-Windows operating environment. It consists of a collection of graphics device interface function calls to the MS-Windows graphic drawing library. Metafiles are both small and flexible, these images can be displayed properly by their proprietary software only.
10. **PICT:** PICT images are useful in Macintosh software development, but you should avoid them in desktop publishing. Avoid using PICT format in electronic publishing-PICT images are prone to corruption.

**3D ROTATION**

Just like 2D rotation we have a given angle $\theta$. In 3D, basic rotation can be done in three ways: along the z-axis, along the y-axis and along the x-axis. For the first rotation (z-axis rotation), the x and y coordinates are modified the same way it is done in 2D. Transformation matrices for the three types of rotation can be derived from the cyclic permutation: $\boldsymbol{x \rightarrow y \rightarrow z \rightarrow x}$

**Z-AXIS ROTATIO**

Given as:

$$x' = xcos\theta - ysin\theta$$
$$y' = xsin\theta + ycos\theta$$
$$z' = z$$

$$P' = R_z(\theta) * P$$
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta & 0 & 0 \\ sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

**Y-AXIS ROTATION**

Given as:

$$x' = zsin\theta + xcos\theta$$
$$y' = y$$
$$z' = zcos\theta - xsin\theta$$

$$P' = R_y(\theta) * P$$
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} cos\theta & 0 & sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -sin\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

**X-AXIS ROTATION**

Given as:

$$x' = x$$

$$y' = ycos\theta - zsin\theta$$

$$z' = ysin\theta + zcos\theta$$

$$P' = R_x(\theta) * P$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & -sin\theta & 0 \\ 0 & sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$