UNIT I

INTRODUCTION & GRAPHICS FOR VR

CHAPTER 2

COMPUTER GRAPHICS

3.1 Computer graphics

- Computer graphics can be a series of images which is most often called a video or single image. Computer graphics is the technology that concerns with designs and pictures on computers. Hence computer graphics are visual representations of data shown on a monitor made on a computer.
- It is used to define, store, manipulate, interrogate, and represent the pictorial output." An image in computer graphics is made up of a number of pixels.
- Some **applications of CG** are – Computer Art, Presentation Graphics, Entertainment, Education & Training, Visualization, Image Processing, GUI, Computer-Aided Design for engineering and architectural systems etc.
- There are two **kinds of computer graphics –**
  - **Interactive CG:**
    - ✓ In this, users have some controls over the image, i.e., the user can make any changes to the image produced.
    - ✓ It involves computer-user two-way communication. E.g. Drawing on touch screen, Animating pictures or graphics in movies, Graphics animation in video games.
    - ✓ It is composed of 3 components - **Display controller or video controller** (it passes Frame Buffer's contents to the monitor), **Digital memory or frame buffer** (here images and pictures are stored as an array (matrix of 0 & 1, 0 represents darkness, and 1 represents image or picture)), **TV monitor** (to view the display).
    - ✓ **Advantages:** Superior quality, More accurate outcomes or products, increased productivity, low cost of development.
  - **Non-Interactive CG / Passive CG:**
    - ✓ In this, the user has no control over the image.
    - ✓ The photo is completely controlled by the instructions of the program, not by the user.
    - ✓ E.g. Screen Saver, Map representation of data, Business graphics are used as brochures, business cards, menu of the hotel etc.

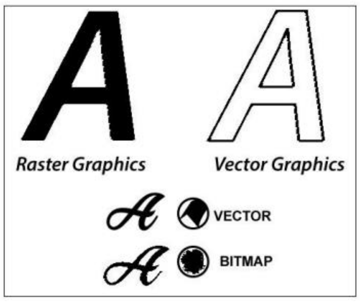
Representation of graphics

Graphics can be represented in two ways:

1. **Raster (Bitmap) Graphics**
   - In raster graphics, the image is **presented as a rectangular grid of coloured squares.**

- Raster images are also called bitmap images. Bitmap images are **stored as the collection of small individual dots called pixels**.
- Bitmap images **require high resolution and anti-aliasing** for a smooth appearance.
- For example– Paint, Photoshop, etc

2. **Vector Graphics**
    - In vector graphics, the image is represented in the form **of continuous geometric objects**: line, curve, etc.
    - Vector images are **not based on pixel** pattern. They **use mathematical formulas to draw line and curves**. The lines and curves can be combined to create an image.
    - For Example– PowerPoint, Corel Draw, etc.



Difference between Raster and Vector Graphics

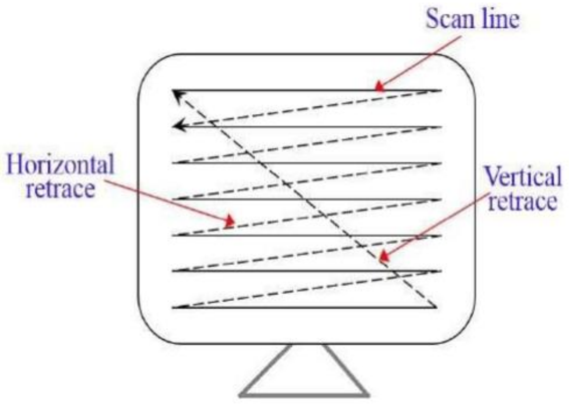
Raster Graphics	Vector Graphics
Raster images are the collection of the pixel.	The Vector images are composed of paths.
Scan conversion is required.	Scan Conversion is not required.
Raster Graphics are less costly.	Vector Graphics are more costly compared to raster graphics.
Raster image takes less space to store.	Vector image takes more space.
Raster graphics can draw mathematical curves, polygons, and boundaries.	Vector graphics can only draw continuous and smooth lines.
File Extension: .BMP, .TIF, .JPG etc.	**File Extension:** .SVG, .PDF, .AI etc.

Object Display on Screen

There are two ways by which we can display an object on the screen –

1. **Raster Scan:**
    a. In a raster scan system, the electron beam is swept across the screen, one row at a time from top to bottom.

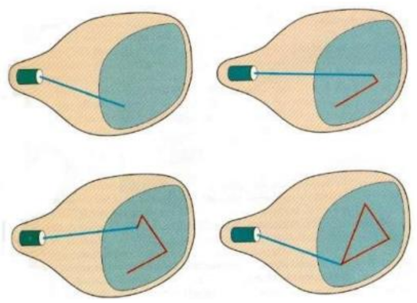
b. As the electron beam moves across each row, the beam intensity is turned on and off to create a pattern of illuminated spots.
c. Picture definition is stored in memory area called the Refresh Buffer or Frame Buffer.
d. This memory area holds the set of intensity values for all the screen points.
e. Stored intensity values are then retrieved from the refresh buffer and “painted” on the screen one row scan line at a time as shown in the following illustration.



f. Each screen point is referred to as a pixel picture element or pel. At the end of each scan line, the electron beam returns to the left side of the screen to begin displaying the next scan line.

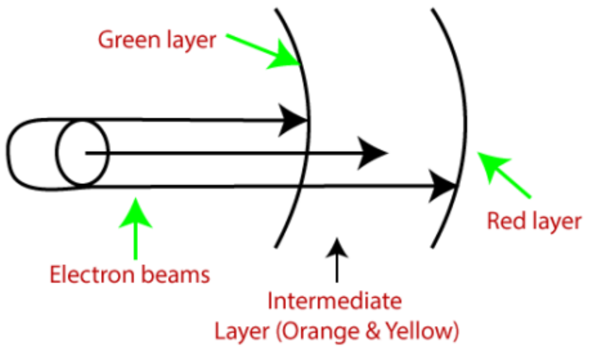
2. **Random Scan / Vector Scan:**

a. In this technique, the electron beam is directed only to the part of the screen where the picture is to be drawn rather than scanning from left to right and top to bottom as in raster scan.
b. It is also called vector display, stroke-writing display, or calligraphic display.
c. Picture definition is stored as a set of line-drawing commands in an area of memory referred to as the **refresh display file**.
d. To display a specified picture, the system cycles through the set of commands in the display file, drawing each component line in turn.
e. After all the line-drawing commands are processed, the system cycles back to the first line command in the list.
f. Random-scan displays are designed to draw all the component lines of a picture 30 to 60 times each second.



Colour Display Producing Technique

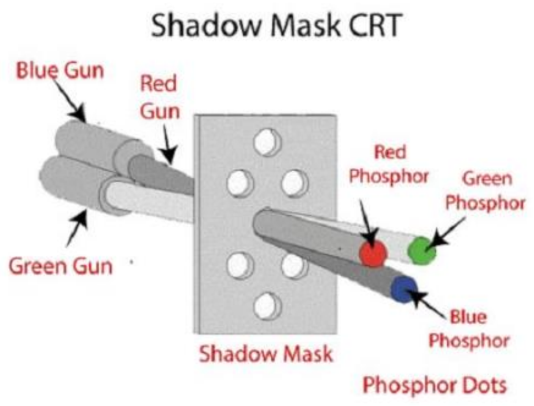
1. **Beam–Penetration Method:**
    a. It is used with a random scan monitor for displaying pictures. There are two phosphorus layers- Red and Green are coated inside the screen.
    b. The colour shown depends on how far the electron beam penetrates the phosphorus surface.
    c. A powerful electron beam penetrates the CRT, it passes through the red layer and excites the green layer within.
    d. A beam with slow electrons excites only the red layer. A beam with the medium speed of electrons, a mixture of red and green light is emitted to display two more colours- orange and yellow.



    e. Some advantages of Beam-Penetration method are better resolution, half cost, inexpensive. And some disadvantages are only four possible colours, time consuming.

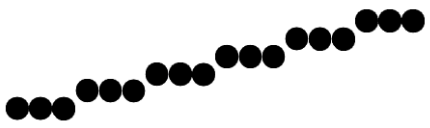
2. **Shadow-Mask Method:**
    a. It is used with a raster scan monitor for displaying pictures. It has more range of colour than the beam penetration method. It is used in television sets and monitors.
    b. It has three phosphorus colour dots at each position of the pixel.
        i. First Dot: Red colour
        ii. Second Dot: Green colour
        iii. Third Dot: Blue colour
    c. It has three different guns. Each for one colour.
    d. It has a metal screen or plate just before the phosphorus screen, named "Shadow-Mask."
    e. It also has a shadow grid just behind the phosphorus coated screen with tiny holes in a triangular shape.
    f. **Working:** A Shadow Mask is a metal plate with tiny holes present inside a colour monitor.
    g. A Shadow Mask directs the beam by consuming the electrons so that the beam hits only the desired point and displays a resulting picture.
    h. It has three different guns. These guns direct their beams to shadow mask, which allows them to pass. It is a task of a shadow mask to direct the beam on its particular dot on the screen and produce a picture on the screen.

i. A Shadow Mask can display a wider range of pictures than beam penetration.
j. Some advantages are Display a wider range picture, display realistic images, In-line arrangement of RGB colour. Some disadvantages are – Difficult to cover all three beams on the same hole, Poor Resolution.
k.

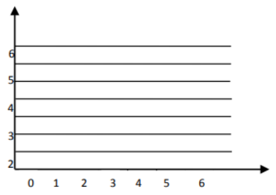


3.2 Line & Circle Drawing Algorithm

- Point plotting is done by converting a single coordinate position furnished by an application program into appropriate operations for the output device in use.
- Line drawing is done by calculating intermediate positions along the line path between two specified endpoint positions.
- The output device is then directed to fill in those positions between the end points with some colour.
- For some device such as a pen plotter or random scan display, a straight line can be drawn smoothly from one end point to other.
- Digital devices display a straight-line segment by plotting discrete points between the two endpoints.
- Discrete coordinate positions along the line path are calculated from the equation of the line.
- For a raster video display, the line intensity is loaded in frame buffer at the corresponding pixel positions.
- Reading from the frame buffer, the video controller then plots the screen pixels.
- Screen locations are referenced with integer values, so plotted positions may only approximate actual line positions between two specified endpoints.
- For example, line position of (12.36, 23.87) would be converted to pixel position (12, 24).
- This rounding of coordinate values to integers causes lines to be displayed with a stair step appearance as represented in following figure.



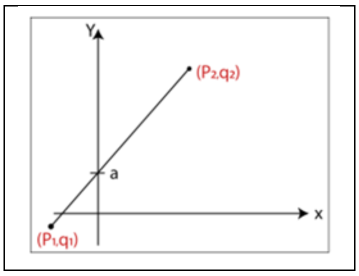
- The stair step shape is noticeable in low resolution system, and we can improve their appearance somewhat by displaying them on high resolution system.
- More effective techniques for smoothing raster lines are based on adjusting pixel intensities along the line paths.
- For raster graphics device-level algorithms discuss here, object positions are specified directly in integer device coordinates.
- Pixel position will be referenced according to scan-line number and column number which is illustrated by following figure –



- To load the specified colour into the frame buffer at a particular position, we will assume we have available low-level procedure of the form *setpixel* (x, y)
- Similarly, to retrieve the current frame buffer intensity we assume to have procedure *getpix* (x, y).

Line Drawing Algorithms

We can define a straight line with the help of the following equation.
Equation of the straight line –
y= mx + a
Where,
(x, y) = Axis of the line, m = Slope of the line, a = Interception point
Let us assume we have two points of the line (p_1, q_1) and (p_2, q_2).
Now, we will put values of the two points in straight line equation, and we get



y = mx + a
$q_2 = mp_2 + a$ ... (1)
$q_1 = mp_1 + a$ ... (2)
We have from equation (1) and (2)
$q_2 – q_1 = mp_2 – mp_1$
$q_2 – q_1 = m (p_2 –p_1)$
The value of $m = (q_2 – q_1)/ (p_2 –p_1)$
m = ▲q / ▲p

There are following algorithms used for drawing a line:

1. DDA (Digital Differential Analyzer) Line Drawing Algorithm

- The Digital Differential Analyzer helps us to interpolate the variables on an interval from one point to another point. We can use the digital Differential Analyzer Algorithm to perform rasterization on polygons, lines, and triangles.

- Digital Differential Analyzer Algorithm is also known as an incremental method of scan conversion. In this algorithm., we can perform the calculation in a step-by-step manner. We use the previous step result in the next step.

  As we know the general equation of the straight line is:
  $y = mx + c$
  Here, m is the slope of (x_1, y_1) and (x_2, y_2).
  $m = (y_2 - y_1)/ (x_2 - x_1)$
  Now, we consider one point (x_k, y_k) and (x_{k+1}, y_{k+1}) as the next point.
  Then the slope $m = (y_{k+1} - y_k) / (x_{k+1} - x_k)$
  Now, we have to find the slope between the starting point and ending point. There can be following three cases to discuss:
  Case 1: If **$m < 1$**
  Then **x** coordinate tends to the Unit interval.
  $\mathbf{x_{k+1} = x_k + 1}$
  $\mathbf{y_{k+1} = y_k + m}$
  Case 2: If **$m > 1$**
  Then **y** coordinate tends to the Unit interval.
  $\mathbf{y_{k+1} = y_k + 1}$
  $\mathbf{x_{k+1} = x_k + 1/m}$
  Case 3: If **$m = 1$**
  Then **x and y** coordinate tend to the Unit interval.
  $\mathbf{x_{k+1} = x_k + 1}$
  $\mathbf{y_{k+1} = y_k + 1}$
  We can calculate all intermediate points with the help of above three discussed cases.

DDA Algorithm

```
Step 1: Start.
Step 2: We consider Starting point as $(x_1, y_1)$, and ending point $(x_2, y_2)$.
Step 3: Now we calculate number of steps as
    If (absolute(▲x) > absolute(▲y))
    Steps = absolute(▲x)
    Else
    Steps = absolute(▲y)
Step 4: Now, we have to calculate ▲x and ▲y.
    $▲x = x_2 - x_1$
    $▲y = y_2 - y_1$
    $m = ▲y/▲x$
Step 4: Now, we calculate three cases.
    If $m < 1$
        Then x changes in Unit Interval
        y moves with deviation
        $(x_{k+1}, y_{k+1}) = (x_k+1, y_k+m)$
    If m > 1
        Then x moves with deviation
        y change in Unit Interval
        $(x_{k+1}, y_{k+1}) = (x_k+1/m, y_k+1)$
    If m = 1
        Then x moves in Unit Interval
        y moves in Unit Interval
        $(x_{k+1}, y_{k+1}) = (x_k+1, y_k+1)$
```

Step 5: We will repeat step 4 until we find the ending point of the line.

Step 6: Stop.

Examples 1: A line has a starting point (1,7) and ending point (11,17). Apply the Digital Differential Analyzer algorithm to plot a line.

Solution:

We have two coordinates,

Starting Point = (x_1, y_1) = (1,7)

Ending Point = (x_2, y_2) = (11,17)

Step 1: First, we calculate ▲x, ▲y and m.

▲x = $x_2 - x_1$ = 11-1 = 10

▲y = $y_2 - y_1$ = 17-7 = 10

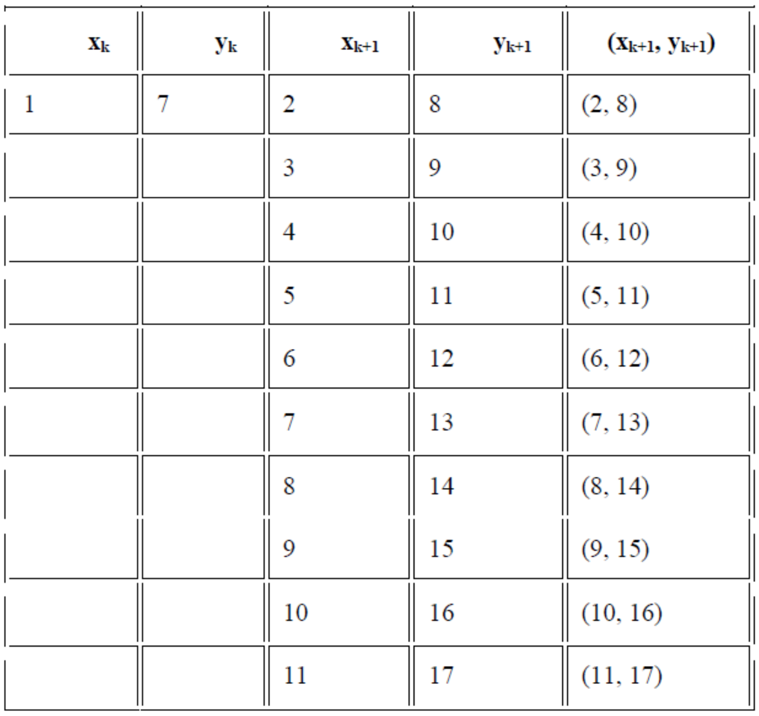
m = ▲y/▲x = 10/10 = 1

Step 2: Now, we calculate the number of steps.

▲x = ▲y = 10

Then, the number of steps = 10

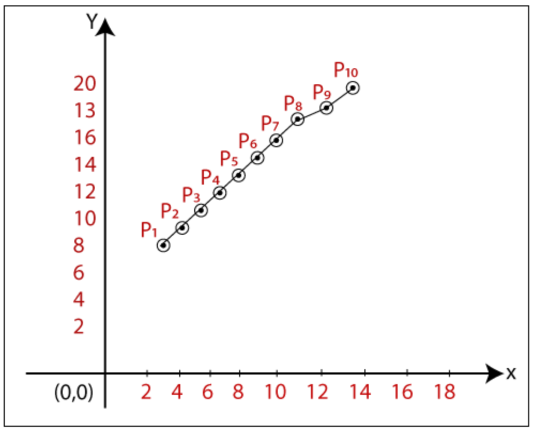
Step 3: We get m = 1, Third case is satisfied. Now move to next step.



x_k	y_k	x_{k+1}	y_{k+1}	(x_{k+1}, y_{k+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)

Step 4: We will repeat step 3 until we get the endpoints of the line.

Step 5: Stop.



The coordinates of drawn line are-
P1 = (2, 8)
P2 = (3, 9)
P3 = (4, 10)
P4 = (5, 11)
P5 = (6, 12)
P6 = (7, 13)
P7 = (8, 14)
P8 = (9, 15)
P9 = (10, 16)
P10 = (11, 17)

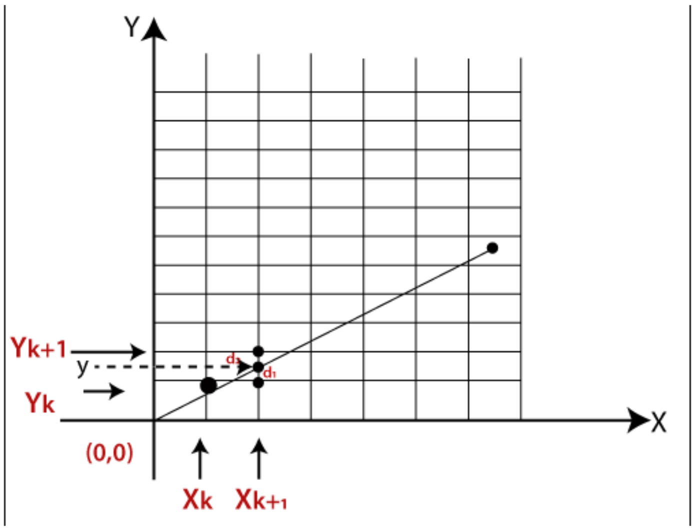
Examples 2: A line has a starting point (2,3) and ending point (6,15). Apply the Digital Differential Analyzer algorithm to plot a line.

Examples 3: A line has a starting point (5,6) and ending point (8,12). Apply the Digital Differential Analyzer algorithm to plot a line.

Examples 4: A line has a starting point (5,6) and ending point (13,10). Apply the Digital Differential Analyzer algorithm to plot a line.

2. Bresenham's Line Drawing Algorithm

- This algorithm helps us to perform scan conversion of a line. It is a powerful, useful, and accurate method. We use incremental integer calculations to draw a line.
- The integer calculations include addition, subtraction, and multiplication.
- In Bresenham's Line Drawing algorithm, we have to calculate the slope (m) between the starting point and the ending point.



- As shown in the above figure let, we have initial coordinates of a line = (x_k, y_k). The next coordinates of a line = (x_{k+1}, y_{k+1}). The intersection points between y_k and y_{k+1} = y. Let we assume that the distance between y and $y_k = d_1$. The distance between y and $y_{k+1} = d_2$.
- Now, we have to decide which point is nearest to the intersection point.

Algorithm of Bresenham's Line Drawing Algorithm

Step 1: Start.

Step 2: Now, we consider Starting point as (x_1, y_1) and ending point (x_2, y_2).

Step 3: Now, we have to calculate ▲x and ▲y.

$$▲x = x_2 - x_1$$

$$▲y = y_2 - y_1$$

$$m = ▲y/▲x$$

Step 4: Now, we will calculate the decision parameter p_k with following formula.

$$p_k = 2▲y - ▲x$$

Step 5: The initial coordinates of the line are (x_k, y_k), and the next coordinates are (x_{k+1}, y_{k+1}). Now, we are going to calculate next points depending on the value of decision parameter pk

Case 1: If $p_k < 0$ Then

$$p_{k+1} = p_k + 2▲y$$

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

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

Case 2: If $p_k >= 0$ Then

$$p_{k+1} = p_k + 2▲y - 2▲x$$

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

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

Step 6: We will repeat step 5 until we found the ending point of the line and the total number of iterations =▲x-1.

Step 7: Stop.

Example1: A line has a starting point (9,18) and ending point (14,22). Apply the Bresenham's Line Drawing algorithm to plot a line.

Solution: We have two coordinates,

Starting Point = (x_1, y_1) = (9, 18)

Ending Point = (x_2, y_2) = (14, 22)

Step 1: First, we calculate ▲x, ▲y.

▲$x = x_2 - x_1 = 14 - 9 = 5$

▲$y = y_2 - y_1 = 22 - 18 = 4$

Step 2: Now, we are going to calculate the decision parameter (p_k)

$p_k = 2$▲y - ▲x

$= 2 \times 4 - 5 = 3$

The value of $p_k = 3$

Step 3: Now, we will check both the cases.

If pk >= 0 Then Case 2 is satisfied. Thus

$p_{k+1} = p_k + 2$▲$y - 2$▲$x = 3 + (2 \times 4) - (2 \times 5) = 1$

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

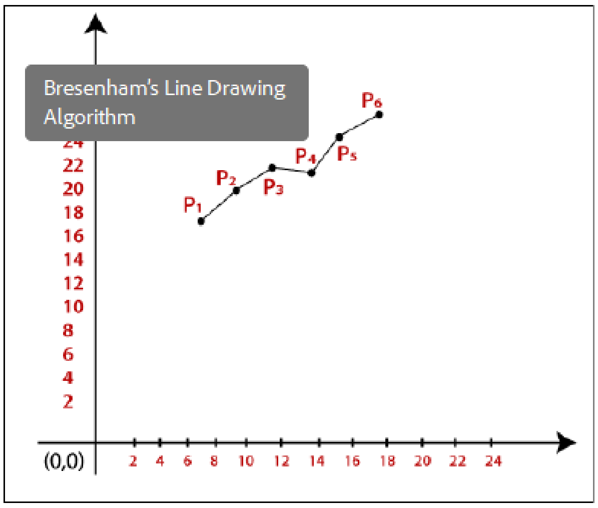
$y_{k+1} = y_k + 1 = 18 + 1 = 19$

Step 4: Now move to next step. We will calculate the coordinates until we reach the end point of the line.

▲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

Step 5: Stop



The Coordinates of drawn lines are-

P1 = (9, 18)

P2 = (10, 19)

P3 = (11, 20)

P4 = (12, 20)

P5 = (13, 21)

P6 = (14, 22)

Example2: A line has a starting point (20,10) and ending point (30,18). Apply the Bresenham's Line Drawing algorithm to plot a line.

3. Mid-Point Line Drawing Algorithm

Given: Starting Coordinates = (X_0, Y_0)

Ending Coordinates = (X_n, Y_n)

Step1: Calculate ΔX and ΔY from the given input.

These parameters are calculated as-

$\Delta X = X_n - X_0$

$\Delta Y = Y_n - Y_0$

Step 2: 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 3: The decision whether to increment X or Y coordinate depends upon the flowing values of $D_{initial}$. Follow the below two cases-

Case 1: If $D_{initial} < 0$ Then

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

$y_{k+1} = y_k$

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

Case 2: If $D_{initial} >= 0$ Then

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

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

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

Step 4: Keep repeating Step 3 until the end point is reached. For each D_{new} value, follow the above cases to find the next coordinates.

Example 1: A line has a starting point (20,10) and ending point (30,18). Apply the Midpoint algorithm to plot a line.

Solution:

Given- Starting coordinates = (X_0, Y_0) = (20, 10)

Ending coordinates = (X_n, Y_n) = (30, 18)

Step 1: Calculate ΔX and ΔY from the given input.

$\Delta X = X_n - X_0 = 30 - 20 = 10$

$\Delta Y = Y_n - Y_0 = 18 - 10 = 8$

Step 2: Calculate $D_{initial}$ and ΔD as-

$$D_{initial} = 2\Delta Y - \Delta X = 2 \times 8 - 10 = 6$$

$$\Delta D = 2(\Delta Y - \Delta X) = 2 \times (8 - 10) = -4$$

Step 3: As $D_{initial} >= 0$, so case-02 is satisfied. Thus,

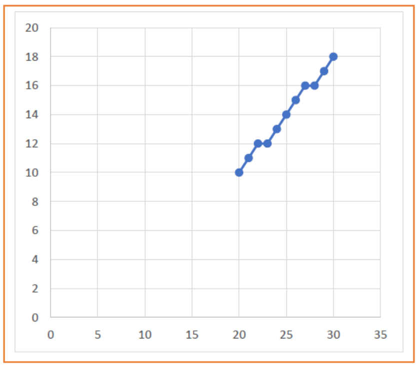
$$X_{k+1} = X_k + 1 = 20 + 1 = 21$$

$$Y_{k+1} = Y_k + 1 = 10 + 1 = 11$$

$$D_{new} = D_{initial} + \Delta D = 6 + (-4) = 2$$

Similarly, Step 3 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



Example 2: Calculate the points between the starting coordinates (5, 9) and ending coordinates (12, 16) using Midpoint Algorithm.

Answer: (Give complete solution. Following is final answer)

$D_{initial}$	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

Circle Drawing Algorithms

1. Bresenham's Circle Drawing Algorithm

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

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 1: Assign the starting point coordinates (X_0, Y_0) as –

$X_0 = 0$

$Y_0 = R$

Step 2: Calculate the value of initial decision parameter P_0 as –

$P_0 = 3 – 2R$

Step 3: 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 Pk. Follow the below two cases –

Case 1: If $p_k < 0$ Then

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

$y_{k+1} = y_k$

$p_{k+1} = p_k + 4\,(x_{k+1}) + 6$

Case 2: If $p_k >= 0$ Then

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

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

$p_{k+1} = p_k + 4\,(x_{k+1} - y_{k+1}) + 10$

Step 4: If the given centre point (X_0, Y_0) is not (0, 0), then do the following and plot the point-

$X_{plot} = X_c + X_0$

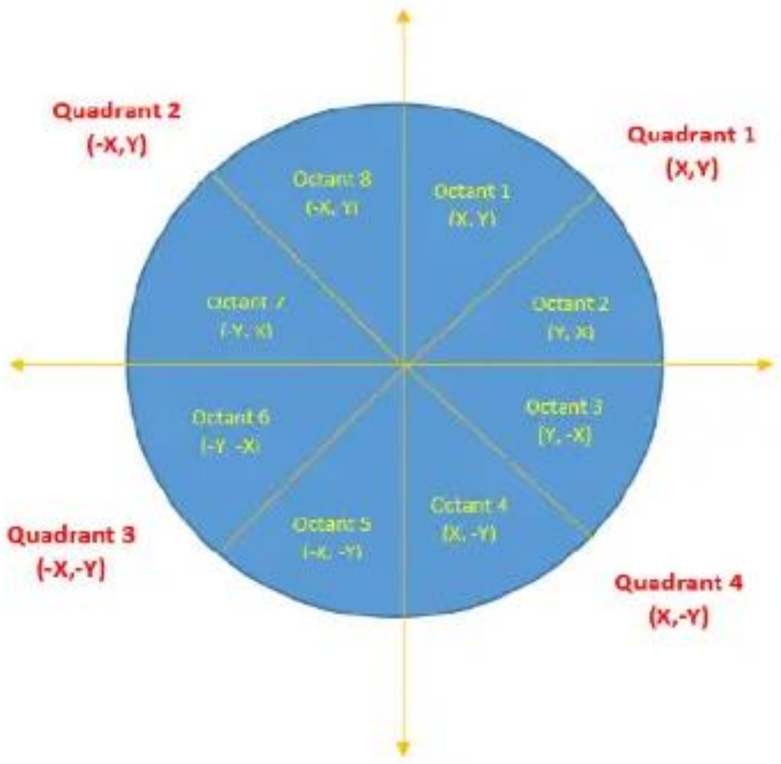
$Y_{plot} = Y_c + Y_0$

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

Step 5: Keep repeating Step 3 and Step 4 until $X_{plot} => Y_{plot}$.

Step 6: Step 5 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-



Example 1: 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 1: Assign the starting point coordinates (X_0, Y_0) as-

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

Step 2: Calculate the value of initial decision parameter P0 as-

$P_0 = 3 – 2R$
$P_0 = 3 – 2 \times 8$
$P_0 = -13$

Step 3: As $P_{initial} < 0$, so case 1 is satisfied.

Thus,

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

Step 4: This step is not applicable here as the given centre point coordinates is (0, 0).

Step 5: Step 3 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)
These all points are for Octant 1		

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, 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	

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)**
(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.			

Example 2: Given the centre point coordinates (10,10) and radius as 10, generate all the points to form as a circle. (**Give entire solution**)

Final Solution is as follows –

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.			

2. Midpoint Circle Drawing Algorithm

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

Given –

- Centre point of circle = (x_0, y_0)
- Radius of circle = R

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

Step 1: Assign the starting point coordinates (x_0, y_0) as –

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

Step 2: Calculate the value of initial decision parameter P_k as –

$P_k = 1 – R$

Step 3: 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 –

Case 1: If $p_k < 0$ Then

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

$y_{k+1} = y_k$

$p_{k+1} = p_k + 2\,(x_{k+1}) + 1$

Case 2: If $p_k >= 0$ Then

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

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

$p_{k+1} = p_k - 2\,(y_{k+1}) + 2\,(x_{k+1}) + 1$

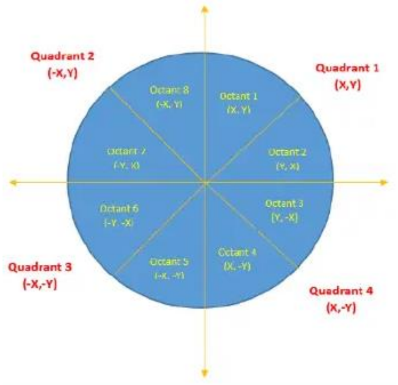
Step 4: If the given centre point $(x_0.y_0)$ is not (0, 0), then do the following and plot the point –

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

Here, (X_c, Y_c) denotes the current value of x & y coordinates.

Step 5: Keep repeating step 3 and step 4 until $x_{plot} >= y_{plot}$

Step 6: Step 5 generates all the points for one octant, follow the eight symmetries Property of circle. This is depicted by the following figure –



Example 1: 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 1: Assign the starting point coordinates (X_0, Y_0) as-

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

Step 2: Calculate the value of initial decision parameter P_0 as-

$P_0 = 1 – R$

$P_0 = 1 – 10$

$P_0 = -9$

Step 3: As $P_{initial} < 0$, so case 1 is satisfied.

Thus,

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

Step 4: This step is not applicable here as the given centre point coordinates is

(0, 0).

Step 5: Step-03 is executed similarly until $X_{k+1} >= Y_{k+1}$ as follows-

P_k	P_{k+1}	(X_{k+1}, Y_{k+1})
		(0, 10)
-9	-6	(1, 10)
-6	-1	(2, 10)
-1	6	(3, 10)
6	-3	(4, 9)
-3	8	(5, 9)
8	5	(6, 8)
Algorithm Terminates **These are all points for Octant-1.**		

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

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

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

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)
(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.			

Example 2: Given the centre point coordinates (4, -4) and radius as 10, generate all the points to form a circle. **(Give complete solution)**

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.			

References:

1. https://www.gatevidyalay.com/bresenham-circle-drawing-algorithm/

********************************** END **********************************