Parul® University

05201330 / 05202182 – Computer Graphics

Dr. Ghanshyam Rathod, Assistant Professor
Parul Institute of Computer Application - BCA

Parul®
University

DIGITAL LEARNING CONTENT

CHAPTER-1

Introduction and Basic Drawing Algorithms

Computer Graphics

- A field of study that deals with the pictures generated by a computer and the tools used to make and present them.
- Computer graphics deals with generating images and art with the aid of computers. Computer graphics is a core technology in digital photography, film, video games, digital art, cell phone and computer displays, and many specialized applications.

History of Computer Graphics

1950 – Ben Laposky created the first graphics image

1987 – VGA (Video Graphics Array) was introduced

(A standard connector used for computer video output)

1995 – 3D animated motion picture was developed.

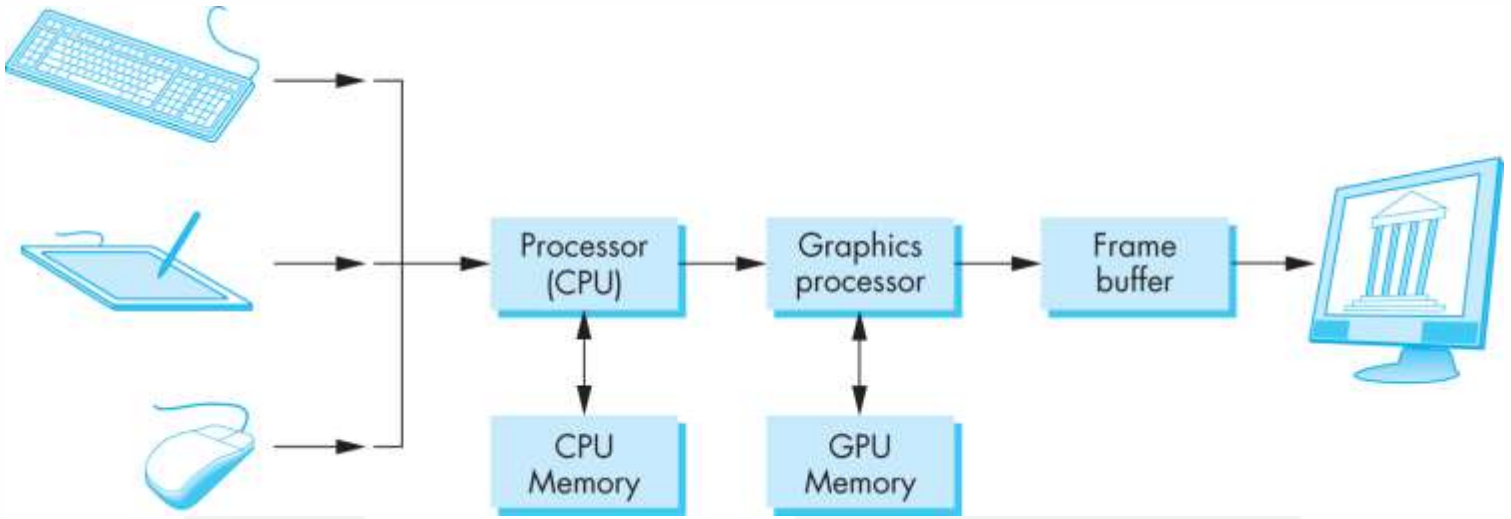
Introduction

- Computer graphics involves display, manipulation of data for proper visualization using computer.
- Generates 2D images of a 3D world represented in a computer.
- Main tasks:
  - ***Modelling*:** Creating and representing the geometry of objects in the 3D world
  - ***Rendering*:** Generating 2D images of the objects
  - ***Animation*:** Describing how objects change in time

Advantages of Computer Graphics

1. Information becomes more communicative when its represented graphically
2. A huge number of numbers may be replaced by a simple graph or chart diagram
3. Visual form of data may be more relevant to realize their characteristics
4. Creation of engineering designs and architectural blue-print may be fully automated
5. Mathematical models of hyperspace can be easily realized using computers
6. Using motion dynamics simulation experiments of moving objects can be performed

Basic Graphic System



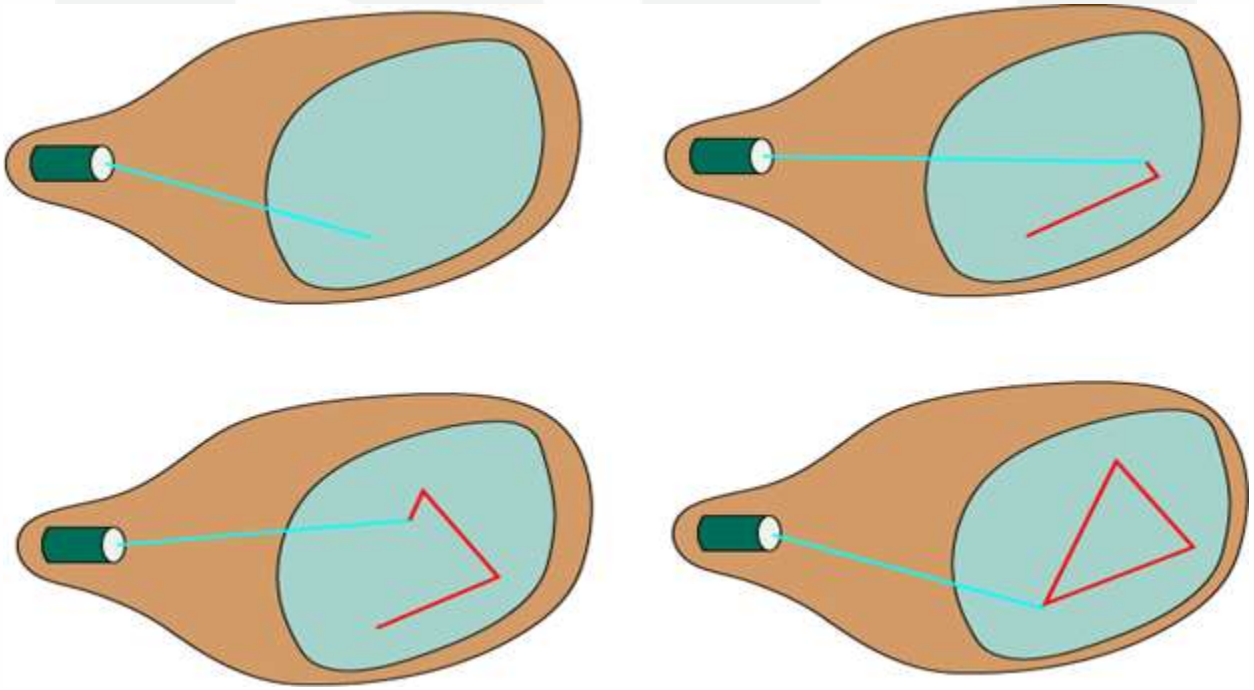
Input devices

Output device

Image formed in frame buffer

Random Scan (Vector) Displays

Vector stands for line. A set of line drawing instructions are stored in memory. The electron beam is directed only to parts of the screen where the picture is to be drawn.

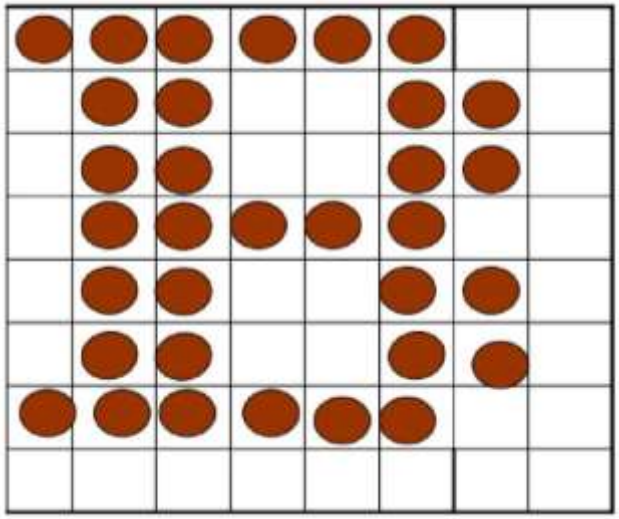


Random Scan Display

- It is also known as **Stroke writing** or C**alligraphic display** or **Vector display**.
- In random scan display, an electron beam is directed only to the parts of the screen where a picture is to be drawn.
- Random scan monitor draw only one line at a time, that's why it is known as vector display or/writing or calligraphic display.
- Refresh rate of random scam system depends on the number of lines to be displayed.

Raster Scan Displays

- Developed in the early seventies.
- It is today's dominant hardware technology. Almost all graphics systems are raster-based.
- A picture is produced as an array – the raster – of picture elements.
- This elements are called Pixels or Pels (Picture Elements).
- A pixel corresponds to a location, or small area, in the image.
- Collectively, the pixels are stored in a part of memory called the refresh buffer or frame buffer.



Raster Scan display

- This technology is similar to television technology here the electron beam continuously move up line by line from top to bottom.
- As the electron beam moves across each row, the beam intensity is turned on or off according to the requirement.
- Picture definition is stored in the memory area known as refresh buffer or frame buffer.

Parul®
University

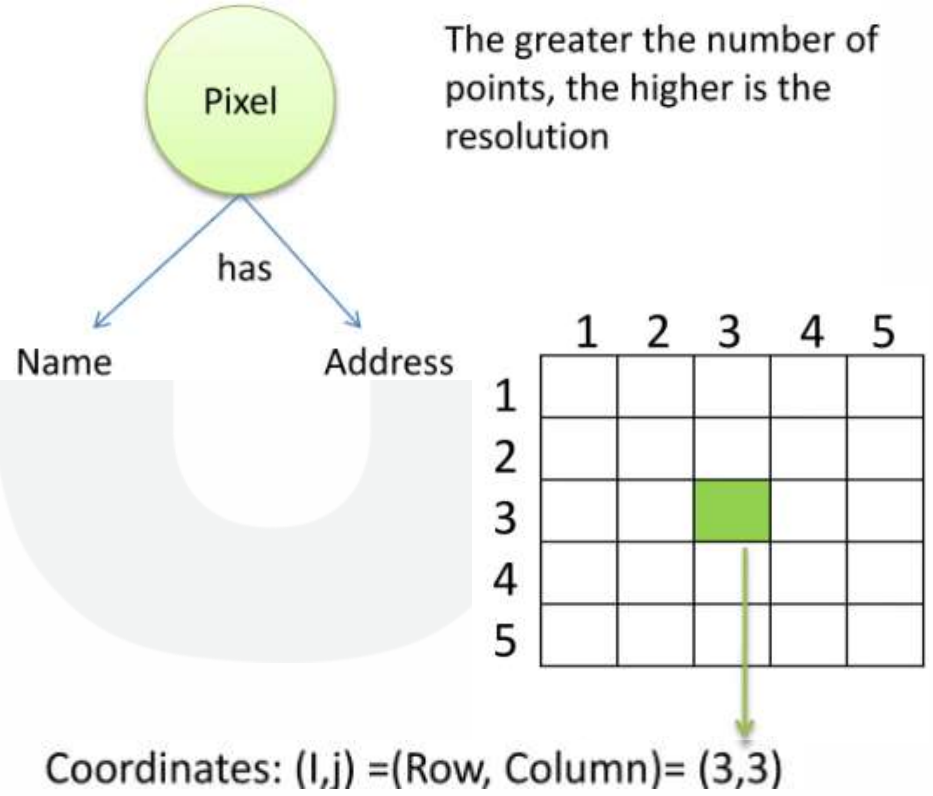
DIGITAL LEARNING CONTENT

Differentiate between Random and Raster Scan

Random Scan	Raster Scan
1. It has high Resolution	1. Its resolution is low.
2. It is more expensive	2. It is less expensive
3. Any modification if needed is easy	3.Modification is tough
4. Solid pattern is tough to fill	4.Solid pattern is easy to fill
5. Refresh rate depends or resolution	5. Refresh rate does not depend on the picture.
6. Only screen with view on an area is displayed.	6. Whole screen is scanned.
7. Beam Penetration technology come under it.	7. Shadow mark technology came under this.
8. It does not use interlacing method.	8. It uses interlacing
9. It is restricted to line drawing applications	9. It is suitable for realistic display.

Graphics Primitives: Pixel

Pixel: In computer graphics, pictures or graphics objects or each screen point is represented as a collection of discrete picture elements which are called as pixels. It is also referred to as a pel, as it is a smallest addressable screen element. It is the smallest piece of screen, which we can control by setting the intensity and color.



Graphics Primitives: Resolution

Resolution: It refers to the maximum number of dots or points that can be displayed on the screen without overlap on the Cathode Ray Tube (CRT). It is expressed as the number of points per centimeter that can be plotted horizontally and vertically. Resolution depends on the type of phosphor, the intensity to be displayed and the focusing and deflection systems used in the CRT. The most commonly used four resolutions today are: 640 × 480, 1024 × 768, 800 × 600, 1280 × 1024, 1920 × 1080.

Graphics Primitives: Frame Buffer, Aspect Ratio

Frame Buffer: A special area of memory is dedicated to graphics only, and picture definition is stored in this memory area which is called frame buffer or refresh buffer. This memory area holds the set of intensity values for all the screen points and the stored intensity value are then retrieved from this frame buffer and painted on the screen one row at a time to display picture.

Aspect Ratio: This number gives the ratio of vertical points to horizontal points necessary to produce equal-length line in both directions on the screen. Sometimes it is also referred to as the ratio of horizontal to vertical points. An aspect ratio of 5/6 means that a vertical line plotted with 5 points has the same length as a horizontal line plotted with 6 points. e.g. aspect ratio of 12” × 16” display is 12/16 = 3/4.

Bit depth

16 bits per pixel (high color)

5 bits for red, 5/6 bits for green, 5 bits for blue

potential of 32 reds, 32/64 green, 32 blues

total colors: 65536

32 bits per pixel (true color)

8 bits for red, green, blue, and alpha

potential for 256 reds, greens, and blues

total colors: 16777216 (more than the eye can distinguish)

Refresh Rate

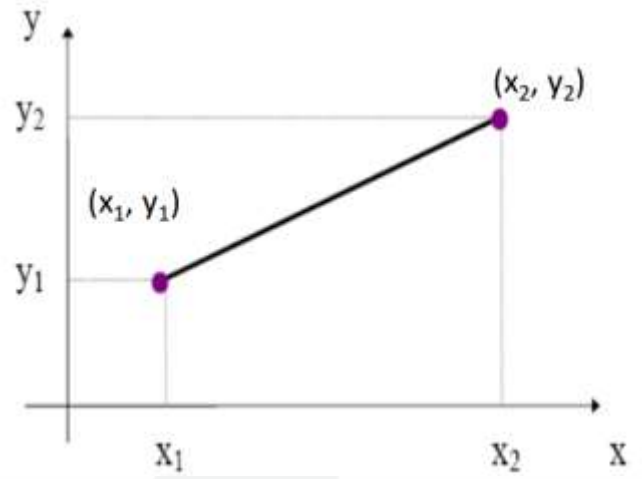
Definition: The number of times per second the image is redrawn.

The entire contents of the frame buffer are displayed on the CRT at a rate high enough to avoid **flicker.** This rate is called the refresh rate.

For a human to see a steady image on most CRT displays, the same path must be retraced, or refreshed, by the beam at least 60 times per second.

Line Drawing Algorithms:

- The Cartesian slope-intercept equation for a straight line is: $y = m \cdot x + b$ ------ (1)
- Where x , y are co-ordinate points, m is the slope of line, and b is known as y intercept.
- Thus, value of slope for the line two endpoints as (x1, y1) and (x2 , y2) as shown in Figure, we can determine values for the slope m and y intercept b with the following calculations:



$$m = \frac{y_2 - y_1}{x_2 - x_1} \quad \text{-------(2)}$$

Line Drawing Algorithms:

$$b = y1 - m \cdot x1 \qquad \text{----- (3)}$$

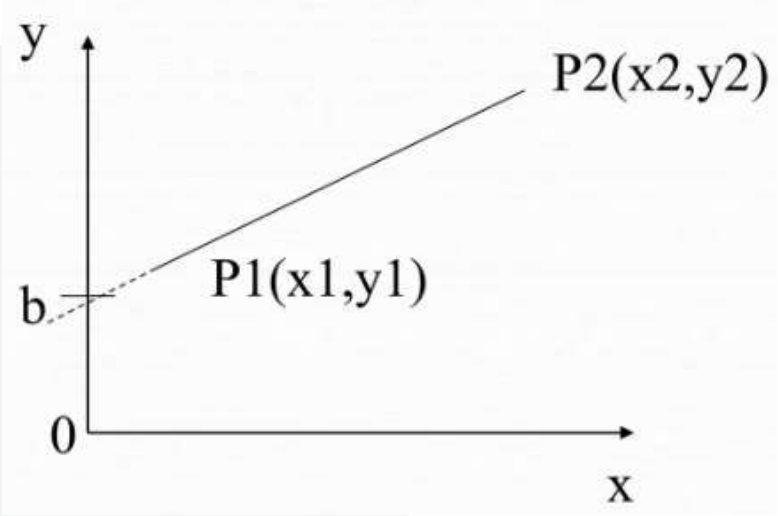
- Algorithms for displaying straight lines are based on the line equation (1) and the calculations given in Eq. (2) and (3).
- For any given x interval Δx along a line, we can compute the corresponding y interval from Eq. (2) as

$$\Delta y = m\,\Delta x \qquad \text{----- (4)}$$

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

$$\Delta x = \frac{\Delta y}{m} \qquad \text{----- (5)}$$

- These equations form the basis for determining deflection voltages in analog devices.



Line Drawing Algorithms:

- For lines with slope have absolute of m means | m | < 1, then Δx can be set proportional to a small horizontal deflection voltage and the corresponding vertical deflection is then set proportional to Δy as calculated from Eq. (4).
- For lines whose slopes have absolute of m means | m | > 1, Δy can be set proportional to a small vertical deflection voltage with the corresponding horizontal deflection voltage set proportional to Δx, calculated from Eq. (5).
- For lines with m = 1, Δx = Δy and the horizontal and vertical deflections voltages are equal.

DDA Algorithm

- The Digital Differential Analyzer (DDA) is a scan-conversion line algorithm based on calculating either Δy or Δx, using Eq. (4) or Eq. (5). We sample the line at unit intervals in one coordinate and determine corresponding integer values nearest the line path for the other coordinate.
- Consider first a line with positive slope, as shown in Fig. (3). If the slope is less than or equal to 1, we sample at unit x intervals (Δx = 1) and compute each successive y value as

$$y_{k+1} = y_k + m \quad \text{----- (6)}$$

DDA Algorithm

- Subscript k takes integer values starting from 1, for the first point, and increases by 1 until the final endpoint is reached. Since m can be any real number between 0 and 1, the calculated y values must be rounded to the nearest integer.
- For lines with a positive slope greater than 1, we reverse the roles of x and y. That is, we sample at unit y intervals ($\Delta y = 1$) and calculate each succeeding x value as

$$x_{k+1} = x_k + \frac{1}{m} \quad \text{----- (7)}$$

DDA Algorithm

- Equations (6) and (7) are based on the assumption that lines are to be processed from the left endpoint to the right endpoint (Fig. 3). If this processing is reversed, so that the starting endpoint is at the right, then either we have $\Delta x = -1$ and

$$y_{k+1} = y_k - m \qquad \text{----- (8)}$$

- or (when the slope is greater than 1) we have $\Delta y = -1$ with

$$x_{k+1} = x_k - \frac{1}{m} \qquad \text{----- (9)}$$

Equations (6) through (9) can also be used to calculate pixel positions along a line with negative slope.

DDA Algorithm

- If the absolute value of the slope is less than 1 and the start endpoint is at the left, we set Δx = 1 and calculate y values with Eq. 6. When the start endpoint is at the right (for the same slope), we set Δx = -1 and Output Primitives obtain y positions from Eq. (8).
- Similarly, when the absolute value of a negative slope is greater than 1,
- we use Δy = -1 and Eq. (9) or we use Δy = 1 and Eq. (7).

DDA Algorithm

- From the above rules, we can summaries the algorithm as below:
- Line endpoints are (xa , ya) and (xb , yb) …

1. Δx = xb - xa ;
   Δy = yb - ya ; [Find the difference of endpoints]

2. if abs(Δx) > abs(Δy) then
   steps = abs(Δx)
   else
   steps = abs(Δy)

3. xincrement = Δx / steps ;
   yincrement = Δy / steps ; [Find total intermediate points]

DDA Algorithm

4. setPixel(round(x_a),round(y_a), 1) [Plot the first point]

5. Repeat step-6 steps times

6. x = x + xincrement ;
   y = y + yincrement;
   setPixel(round(x), round(y), 1) [Plot all points]

DDA Algorithm (Example)

- End points are (x1, y1) = (5, 6) and (x2, y2) = (13, 10)
- **Step-1**

  $\Delta x = x2 - x1 = 13 - 5 = 8$

  $\Delta y = y2 - y1 = 10 - 6 = 4$

  $M = \Delta y / \Delta x = 4 / 8 = 0.5$

- **Step-2**

```
if abs(Δx) > abs(Δy) then
        steps = abs(Δx)
else
        steps = abs(Δy)
```

```
if abs(8) > abs(4) then
        steps = 8
else
        steps = 4
```

DDA Algorithm (Example)

- **Step-3** xincrement = Δx / steps xincrement = 8 / 8 = 1

  yincrement = Δy / steps yincrement = 4 / 8 = 0.5

- **Step-4**

  setPixel(round(xa),round(ya), 1) [Plot the first point]

- **Step-5**

  Repeat step-6 steps times (8 times)

- **Step-6** $x_{i+1} = x_i + xincrement$ $x_{i+1} = 5 + 1 = 6$

  $y_{i+1} = y_i + yincrement$ $y_{i+1} = 6 + 0.5 = 6.5$

  $setPixel(round(x_{i+1}), round(y_{i+1}), 1)$ setPixel(round(6), round(6.5), 1)

  Here, 1 means line width setPixel(6, 7, 1)

DDA Algorithm (Example)

- End points are $(x_1, y_1) = (5, 6)$ and $(x_2, y_2) = (13, 10)$

i	x	y	Result	Plotted Point
0	5	6	6, 6.5	6, 7
1	6	6.5	7, 7	7, 7
2	7	7	8, 7.5	8, 8
3	8	7.5	9, 8	9, 8
4	9	8	10, 8.5	10, 9
5	10	8.5	11, 9	11, 9
6	11	9	12, 9.5	12, 10
7	12	9.5	13, 10	13, 10

Advantages of DDA Algorithm

- 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

- 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.

Bresenhems Line Drawing Algorithm

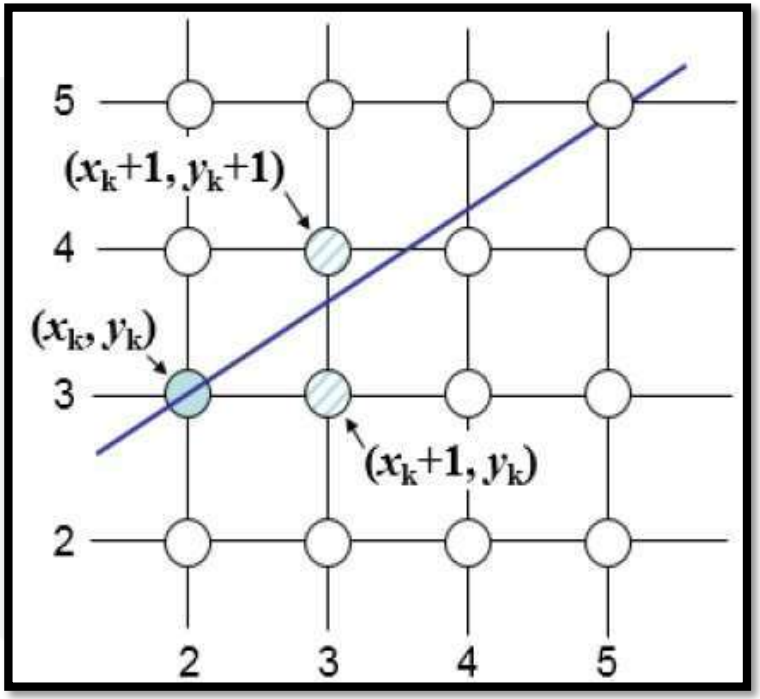
- It was developed by Bresenham, and it is accurate and efficient algorithm to display not only lines, but also the circles and other curves.
- The integer values of the co-ordinates of next point in line are obtained by the decision parameters, as per following steps, and thus the line is created.

Bresenham Line Drawing Algorithm

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-



Bresenham Line Drawing Algorithm for |m| < 1

1. Input the two endpoints of a line and store the left endpoint (X_0, Y_0).
2. Load (X_0,Y_0) into frame buffer, that is, plot the point.
3. Calculate the constants Δx, Δy, $2\Delta y$ and $2\Delta y - \Delta x$, and obtain the first decision parameter as

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

4. At each X_k along the line, starting at k =0, perform the following test:

   If $P_k < 0$, the next point to plot is ($X_k + 1$, Y_k) and next parameter

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

   Otherwise, the next point to plot is ($X_k + 1$, $Y_k + 1$), and next parameter

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

5. Repeat step-4, ΔX no. of times.

Bradenham's Line Drawing Algorithm (Example)

- To illustrate the algorithm, we digitize the line with endpoints (5, 6) and (13, 10). This line has a slope of 0.8, with

  $\Delta x = 13 - 5 = 8, \qquad \Delta y = 10 - 6 = 4$

- The initial decision parameter has the value

  $P_0 = 2\Delta y - \Delta x = 8 - 8 = 0$

- and the increments for calculating successive decision parameters are

  $2\Delta y = 8, \qquad 2\Delta y - 2\Delta x = 8 - 16 = -8$

Bresenham Line Drawing Algorithm (Example)

- We plot the initial point $(x_0, y_o) = (5, 6)$, and determine successive pixel positions along the line path from the decision parameter as
- $P_{k+1} = P_k + 2\Delta y \ (P_k < 0)$ or $P_{k+1} = P_k + 2\Delta y - 2\Delta x \ (P_k >= 0)$

k	P_k	(X_{k+1}, Y_{k+1})	k	P_k	(X_{k+1}, Y_{k+1})
0	0	(6, 7)	4	0	(10, 9)
1	-8	(7, 7)	5	-8	(11, 9)
2	0	(8, 8)	6	0	(12, 10)
3	-8	(9, 8)	7	-8	(13, 10)

Advantages of Bresenhems 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.

Dis-advantages of Bresenhems Line Drawing

The disadvantages of Bresenham Line Drawing Algorithm are-

- Though it improves the accuracy of generated points but still the resulted line is not smooth.
- This algorithm is for the basic line drawing.

Circle Drawing Algorithm

- Drawing a circle on the screen is a little complex than drawing a line.
- There are two popular algorithms for generating a circle – **Bresenham's Algorithm** and **Midpoint Circle Algorithm**.
- These algorithms are based on the idea of determining the subsequent points required to draw the circle
- The equation of circle is $X^2 + Y^2 = r^2$, where r is radius.

Bresenham Circle Drawing Algorithm

- To begin, note that only one octant of the circle need be generated. The other parts can be obtained by successive reflections.
- If the first octant (90 to 45 CCW) is generated, the second octant can be obtained by reflection through the line y=x to yield the first quadrant.
- The results in the first quadrant are reflected in the second quadrant, and so on, the circle will be generated.

Bresenham Circle Drawing Algorithm

1. Input radius r and circle center (x_c, y_c), and obtain the first point in circumference of a circle centered on the origin as

$$(x_0, y_0) = (0, r)$$

2. Calculate the initial value of the decision parameter as

$$P_0 = 3 - 2r$$

3. At each x_k position, starting at k = 0, perform the following test:

   If $P_k < 0$, the next point along circle centered on (0, 0) is (x_k+1, y_k) and

$$P_{k+1} = P_k + 4X_k + 6$$

   Otherwise, the next point is (x_k + 1 , y_k – 1) and

$$P_{k+1} = P_k + 4(x_k - y_k) + 10.$$

Bresenham Circle Drawing Algorithm

4. Determine the symmetry points in other seven octant

5. Move each pixel position (X,Y) into circular path

$$x = x + x_c \text{ and } y = y + y_c$$

6. Repeat step 3 to 5 until x >= y

Bresenham Circle Drawing Algorithm (Example)

- Given a circle radius r=10, we demonstrate the Bresenham circle drawing algorithm by determining position along the circle octant in the first quadrant from X=0 to X=Y.
- The initial decision parameter P_0
- $P_0 = 3 - 2r = 3 - 20 = -17$ therefore $P_0 < 0$, so, $(X_1, Y_1) = (1, 10)$
- For the circle centered on the coordinator origin, the initial point is $(X_0, Y_0) = (0, 10)$ and initial increment terms for calculating the decision parameter are

Bresenham Circle Drawing Algorithm (Example)

1) $P_1 = P_0 + 4X_0 + 6$

$= -17 + 0 + 6$

$= -11$

$\therefore P_1 < 0 \Longrightarrow (X_2, Y_2) = (2,10)$

2) $P_2 = P_1 + 4X_1 + 6$

$= -11 + 4 + 6$

$= -1$

$\therefore P_2 < 0 \Longrightarrow (X_3, Y_3) = (3,10)$

3) $P_3 = P_2 + 4X_2 + 6$

$= -1 + 8 + 6$

$= 13$

$\therefore P_3 > 0 \Longrightarrow (X_4, Y_4) = (4,9)$

Bresenham Circle Drawing Algorithm (Example)

4) $P_4 = P_3 + 4(X_3 - Y_3) + 10$

$= 13 - 28 + 10$

$= -5$

$\therefore P_4 < 0 \Longrightarrow (X_5, Y_5) = (5,9)$

5) $P_5 = P_4 + 4X_4 + 6$

$= -5 + 16 + 6$

$= 17$

$\therefore P_5 > 0 \Longrightarrow (X_6, Y_6) = (6,8)$

6) $P_6 = P_5 + 4(X_5 - Y_5) + 10$

$= 17 - 16 + 10$

$= 11$

$\therefore P_6 > 0 \Longrightarrow (X_7, Y_7) = (7,7)$

Bresenham Circle Drawing Algorithm (Example)

k	P_k	X	Y	(X_{k+1}, Y_{k+1})
0	-17	1	10	(1, 10)
1	-11	2	10	(2, 10)
2	-1	3	10	(3, 10)
3	13	4	9	(4, 9)
4	-5	5	9	(5, 9)
5	17	6	8	(6, 8)
6	11	7	7	(7, 7)

Mid Point Circle Drawing Algorithm

1. Input radius r and circle center (x_c, y_c), and obtain the first point in circumference of a circle centered on the origin as

$$(x_0, y_0) = (0, r)$$

2. Calculate the initial value of the decision parameter as

$$P_0 = 5/4 - r$$

3. At each x_k position, starting at k = 0, perform the following test:

   If $P_k < 0$, the next point along circle centered on (0, 0) is (x_k+1, y_k) and

$$P_{k+1} = P_k + 2X_{k+1} + 1$$

   Otherwise, the next point is ($x_k + 1$, $y_k - 1$) and

$$P_{k+1} = P_k + 2x_{k+1} + 1 - 2y_{k+1}.$$

   Where, $2x_{k+1} = 2x_k + 2$ and $2y_{k+1} = 2y_k - 2$.

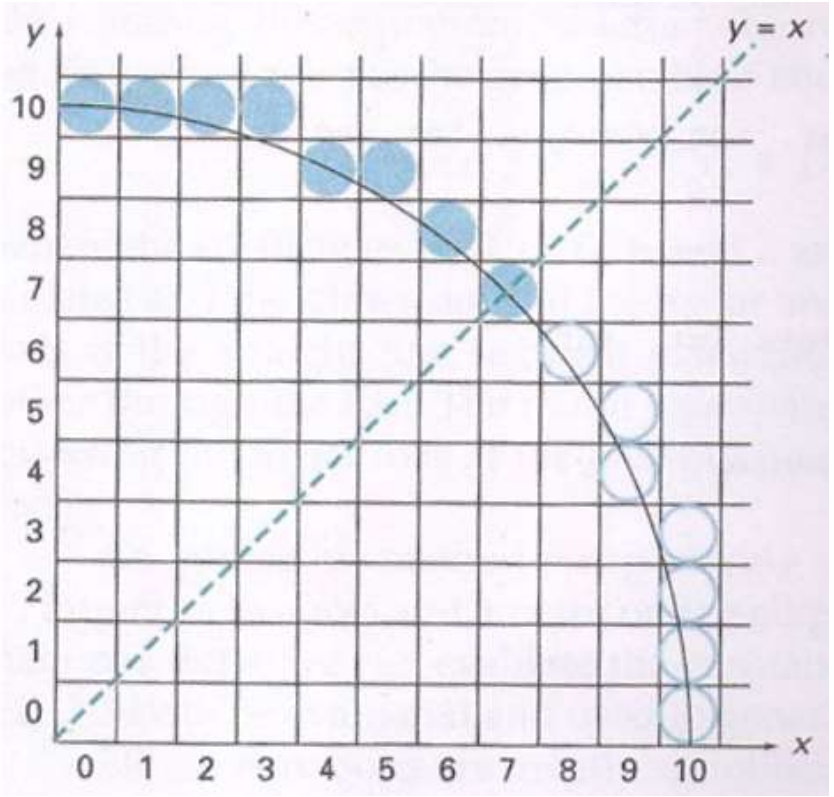
Mid Point Circle Drawing Algorithm

4. Determine the symmetry points in the other seven octants.
5. Move each pixel position (x, y) onto the circular path centered on (x_c, y_c) and plot the coordinate values:

$$x = x + x_c$$

$$y = y + y_c$$

6. Repeat steps 3 through 5 until x >= y.



Mid Point Circle Drawing Algorithm (Example)

Given a circle radius r = 10, we demonstrate the midpoint circle algorithm by determining positions along the circle octant in the first quadrant

from x = 0 to x = y.

The initial value of the decision parameter is

$$P_0 = 1 - r = -9$$

For the circle centered on the coordinate origin, the initial point is $(x_0, y_0) = (0, 10)$, and initial increment terms for calculating the decision parameters are

$$2x_0 = 0, \text{ and } 2y_0 = 20$$

Mid Point Circle Drawing Algorithm (Example)

Successive decision parameter values and positions along the circle path are calculated using the midpoint method as

k	P_k	(x_{k+1}, y_{k+1})	$2x_{k+1}$	$2y_{k+1}$
0	-9	(1 , 10)	2	**20**
1	-6	(2 , 10)	4	**20**
2	-1	(3 , 10)	6	**20**
3	6	(4 , 9)	8	**18**
4	-3	(5 , 9)	10	**18**
5	8	(6 , 8)	12	**16**
6	**5**	**(7 , 7)**	**14**	**14**

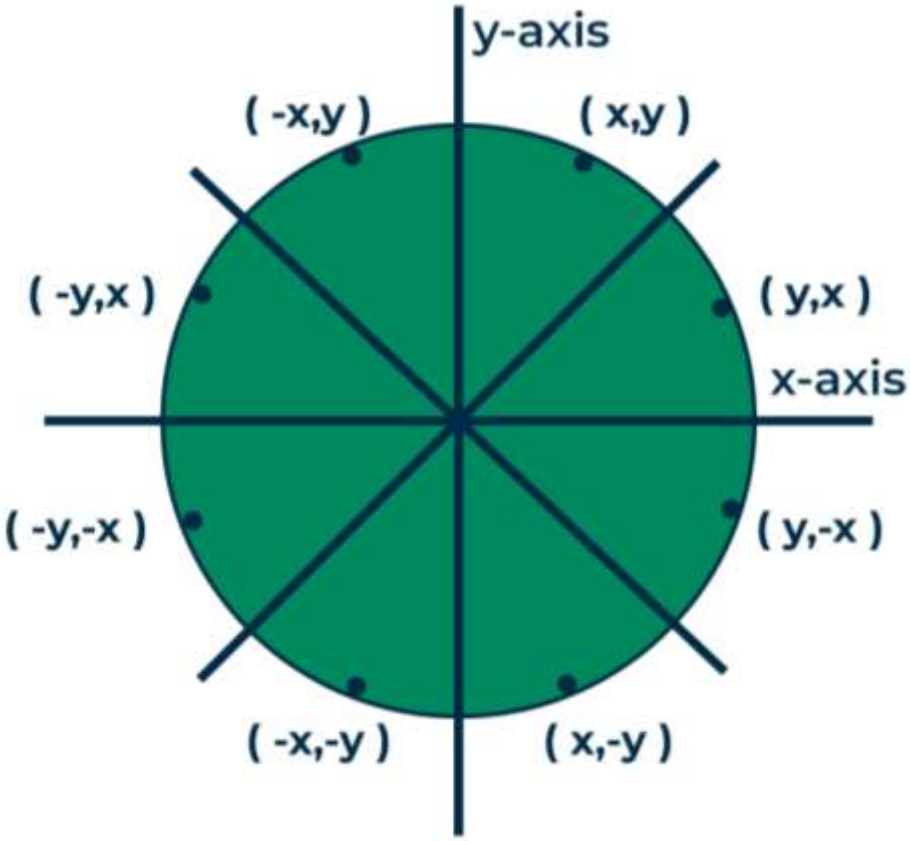
A plot of the generated pixel position in the first quadrant is shown in fig above.

Mid Point Circle Drawing Algorithm

This 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-



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.

Mid Point Ellipse Drawing Algorithm

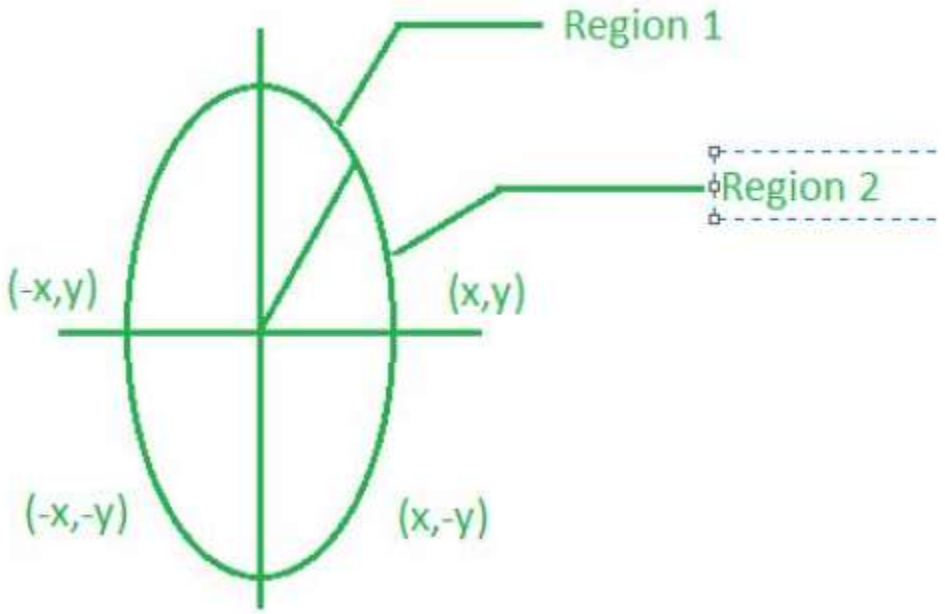
In computer graphics, the mid-point ellipse algorithm is an incremental method of drawing an ellipse. It is very similar to the mid-point algorithm used in the generation of a circle.

The mid-point ellipse drawing algorithm is used to calculate all the perimeter points of an ellipse. In this algorithm, the mid-point between the two pixels is calculated which helps in calculating the decision parameter. The value of the decision parameter determines whether the mid-point lies inside, outside, or on the ellipse boundary and the then position of the mid-point helps in drawing the ellipse.

Mid Point Ellipse Drawing Algorithm

Midpoint ellipse algorithm plots(finds) points of an ellipse on the first quadrant by dividing the quadrant into two regions.

Each point(x, y) is then projected into other three quadrants (-x, y), (x, -y), (-x, -y) i.e. it uses 4-way symmetry.



Mid Point Ellipse Drawing Algorithm

$$fellipse(x, y) = r_y^2 x^2 + r_x^2 y^2 - r_x^2 r_y^2$$

Which has the following properties:

If fellipse(x, y) < 0 means (x, y) is inside the ellipse boundary

If fellipse(x, y) = 0 means (x, y) is on the ellipse boundary

If fellipse(x, y) > 0 means (x, y) is outside the ellipse boundary

Thus, the ellipse function *fellipse(x, y)* serves as the decision parameter in the mid-point algorithm.

Mid Point Ellipse Drawing Algorithm

1. Input r_x, r_y, and ellipse center (x_c, y_c) and obtain the first point on an ellipse centered on the origin as $(x_0, y_0) = (0, r_y)$
2. Calculate the initial value of the decision parameter in region 1 as

$$P1_0 = r_y^2 - r_x^2 r_y + \tfrac{1}{4} r_x^2$$

3. At each x_k, position in region 1, starting at k = 0, perform the following test: If $p1_k < 0$, the next point along the ellipse centered on (0, 0) is $(x_k + 1, y_k)$ and

$$P1_{k+1} = P1_k + 2r_y^2 x_{k+1} + r_y^2$$

Otherwise, the next point along the circle is $(x_k + 1, y_k - 1)$ and

$$P1_{k+1} = P1_k + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_y^2$$

Mid Point Ellipse Drawing Algorithm

4. Calculate the initial value of the decision parameter in region 2 using the last point (x_0, y_0) calculated in region 1 as

$$P2_0 = r_y^2 (x_0 + ½)^2 + r_x^2(y_0 - 1) - r_x^2 r_y^2$$

5. At each y_k, position in region 2, starting at k = 0, perform the following test: If $p2_k > 0$, the next point along the ellipse centered on (0, 0) is $(x_k, y_k - 1)$ and

$$P2_{k+1} = P2_k - 2r_x^2 y_{k+1} + r_x^2$$

Otherwise, the next point along the circle is $(x_k + 1, y_k - 1)$ and

$$P2_{k+1} = P2_k + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_x^2$$

Mid Point Ellipse Drawing Algorithm

6. Determine symmetry points in the other three quadrants.
7. Move each calculated pixel position (x, y) onto the elliptical path centered on (x_c, y_c) and plot the coordinate values:

$$x = x + x_c, y = y + y_c$$

8. Repeat the steps for region 1 until $2r_y^2x >= 2r_x^2y$

Mid Point Ellipse Drawing Algorithm (Example)

Parameters: $r_x = 8$, and $r_y = 6$

For region 1: The initial point for the ellipse centered on the origin is

$(x_0, y_0) = (0, 6)$, and the initial decision parameter value is

$$P1_0 = r_y^2 - r_x^2 r_y + \tfrac{1}{4} r_x^2$$

$$= 36 - (64 * 6) + (\tfrac{1}{4} * 64)$$

$$= 36 - 384 + 16 = -332$$

Mid Point Ellipse Drawing Algorithm (Example)

Successive decision parameter values and positions along the ellipse path are calculated using the midpoint method as

K	$P1_k$	(x_{k+1}, y_{k+1})	$2r_y^2x_{k+1}$	$2r_x^2y_{k+1}$
0	-332	(1, 6)	72	768
1	-224	(2, 6)	144	768
2	-44	(3, 6)	216	768
3	208	(4, 5)	288	640
4	-108	(5, 5)	360	640
5	288	(6, 4)	432	512
6	244	(7, 3)	504	384

Now, we move out of region 1, since $r_y^2x > r_x^2y$

Mid Point Ellipse Drawing Algorithm (Example)

For region 2, the initial point is (x0, y0) = (7, 3) and the initial decision parameter is

$$P2_0 = r_y^2 (x_0 + ½)^2 + r_x^2(y_0 - 1) - r_x^2 r_y^2 = 6^2(7+½)^2 + 8^2(3-1) - (8^2*6^2)$$

$$= (36*56.25) + (64*2) - (64*36) = 2025+128-2304 = -151$$

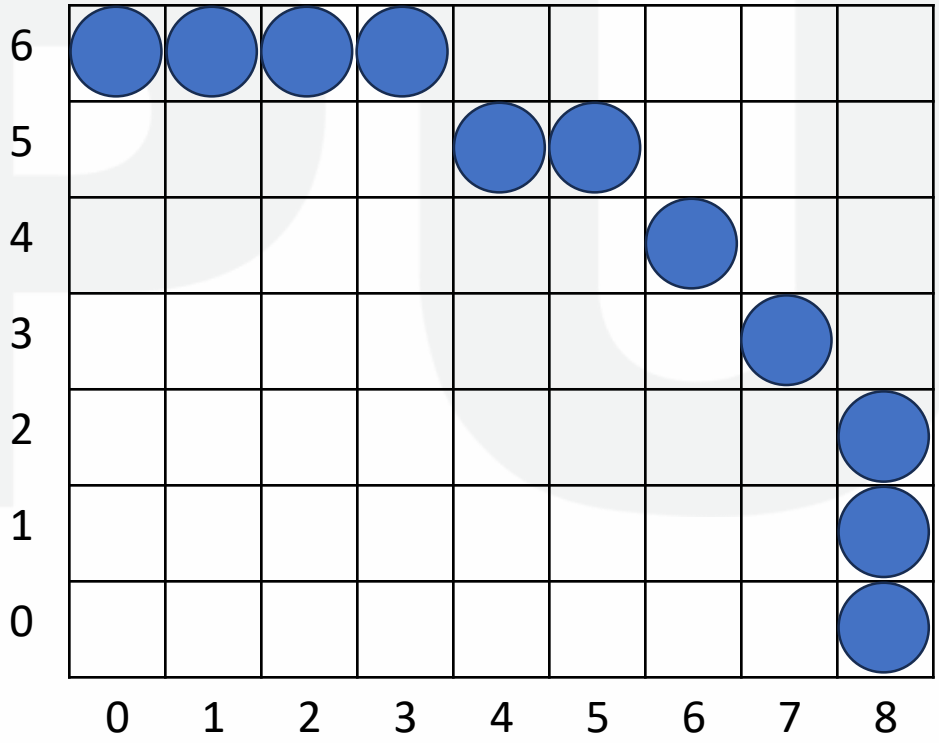
$$P2_0 = f(7 + ½, 2) = -151$$

The remaining positions along the ellipse path in the first quadrant are then calculated as.

K	$P2_k$	(x_{k+1}, y_{k+1})	$2r_y^2x_{k+1}$	$2r_x^2y_{k+1}$
0	-151	(8, 2)	576	256
1	233	(8, 1)	576	128
2	745	(8, 0)	–	–

Mid Point Ellipse Drawing Algorithm (Example)

A plot of the selected positions around the ellipse boundary within the first quadrant is shown in Fig. After this do, this process for other three quadrants.



DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in