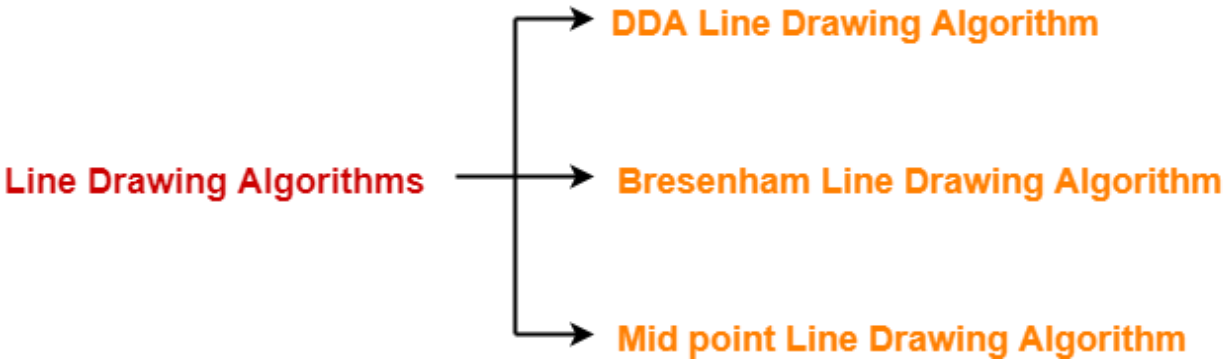
Line Drawing Algorithms

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

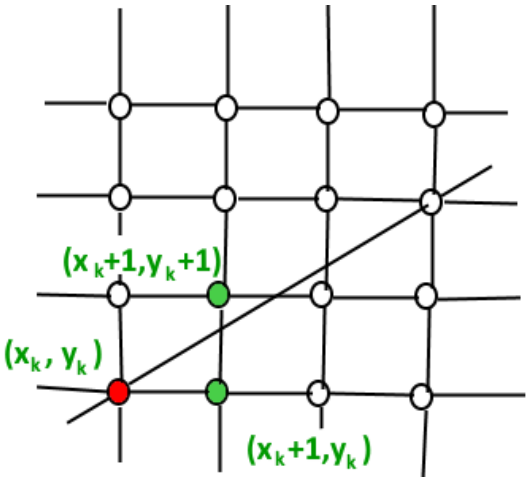


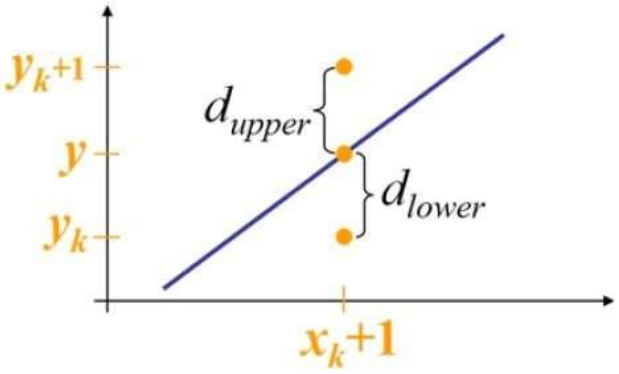
1. Digital Differential Analyzer (DDA) Line Drawing Algorithm
2. Bresenham Line Drawing Algorithm
3. Mid Point Line Drawing Algorithm

Bresenham Line Drawing algorithm

The Bresenham algorithm is another incremental scan conversion algorithm. The big advantage of this algorithm is that it uses only integer calculations.

Deriving The Bresenham Line Algorithm





- At sample position x_k+1 the vertical separations from the mathematical line are labelled d_{upper} and d_{lower}
- The y coordinate on the mathematical line at x_k+1 is:

$$y = m(x_k + 1) + b$$

- So, d_{upper} and d_{lower} are given as follows:

$$d_{upper} = (y_k + 1) - y = y_k + 1 - m(x_k + 1) - b$$

$$d_{lower} = y - y_k = m(x_k + 1) + b - y_k$$

- We can use d_{upper} and d_{lower} to make a simple decision about which pixel is closer to the mathematical line. This

simple decision is based on the difference between the two pixel positions:

$$d_{lower} - d_{upper} = 2m(x_k + 1) - 2y_k + 2b - 1$$

- Let's substitute m with Δy/Δx where Δx and Δy are the differences between the end-points:

$$\Delta x(d_{lower} - d_{upper}) = \Delta x(2\frac{\Delta y}{\Delta x}(x_k + 1) - 2y_k + 2b - 1)$$

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

- So, a decision parameter pk for the kth step along a line is given by:

$$p_k = \Delta x(d_{lower} - d_{upper})$$
$$= 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c$$

  - The sign of the decision parameter pk is the same as that of dlower – dupper
  - If p_k is negative, then we choose the lower pixel, otherwise we choose the upper pixel

- Remember coordinate changes occur along the x axis in unit steps so we can do everything with integer calculations. At step k+1 the decision parameter is given as:

$$p_{k+1} = 2\Delta y \cdot x_{k+1} - 2\Delta x \cdot y_{k+1} + c$$

Subtracting pk from this we get:

$$p_{k+1} - p_k = 2\Delta y(x_{k+1} - x_k) - 2\Delta x(y_{k+1} - y_k)$$

- But, x_{k+1} is the same as x_k+1 so:

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x(y_{k+1} - y_k)$$

where $y_{k+1} - y_k$ is either 0 or 1 depending on the sign of p_k
The first decision parameter p_0 is evaluated at (x_0, y_0) is given as:

$$p_0 = 2\Delta y - \Delta x$$

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

Given the starting and ending coordinates of a line,

Bresenham Line Drawing Algorithm attempts to generate the points between the starting and ending coordinates.

Procedure-

Given-

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

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

Step-01:

Calculate ΔX and ΔY from the given input.

These parameters are calculated as-

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

Step-02:

Calculate the decision parameter P_k.

It is calculated as-

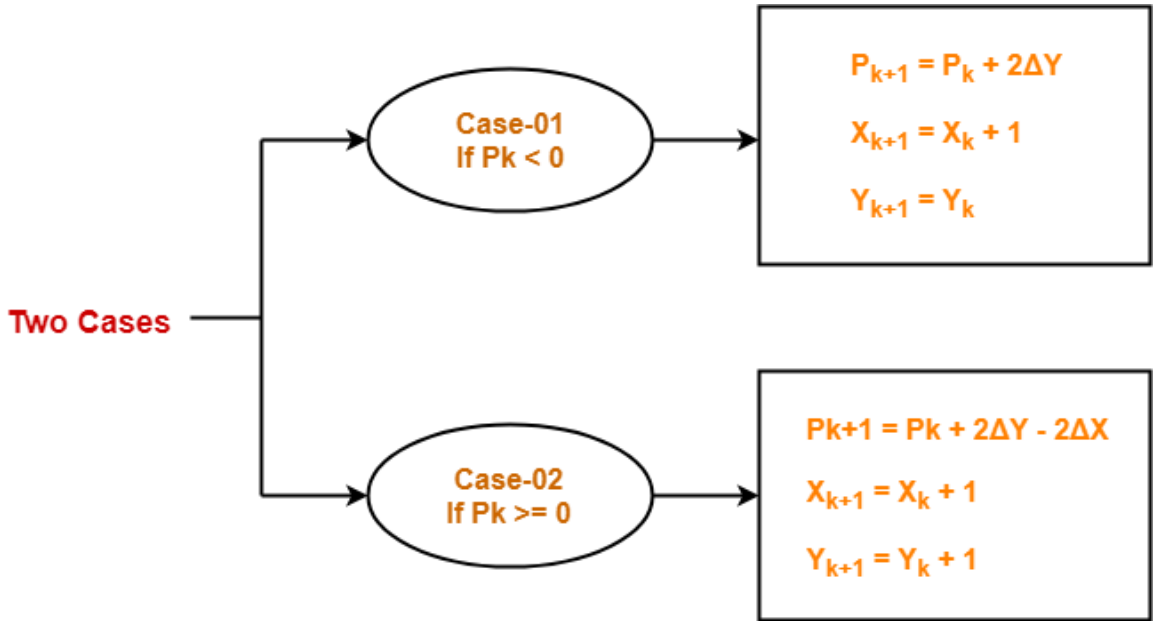
$$P_k = 2\Delta Y - \Delta X$$

Step-03:

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

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

Follow the below two cases-



Step-04:

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

Problem-01:

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,

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

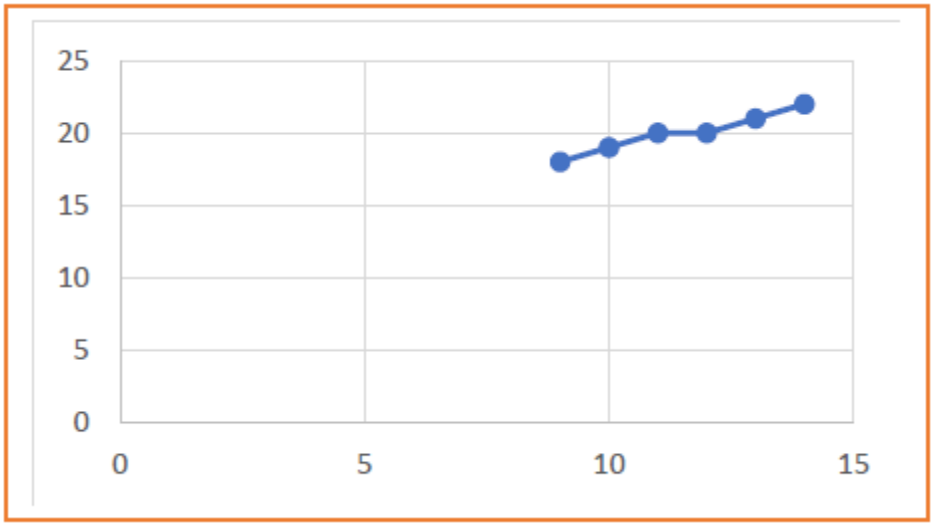
Similarly, Step-03 is executed until the end point is reached or the number of iterations

STEP	P = (2 * Dy) - Dx;	P < 0	Calculation Method	Calculation	Points
1	Starting Point				9 , 18
2	3 = (2 * 4) - 5	ELSE	$X_{k+1} = X_k + 1;$ $Y_{k+1} = Y_k + 1;$ $P_{k+1} = P_k + (2 * dy) - (2 * dx);$	10 = 9 + 1 19 = 18 + 1; 1 = 3 + (2 * 4)-(2 *5)	10 , 19
3	1	ELSE	$X_{k+1} = X_k + 1;$ $Y_{k+1} = Y_k + 1;$ $P_{k+1} = P_k + (2 * dy) - (2 * dx);$	11 = 10 + 1 20 = 19 + 1; -1 = 1 + (2 * 4)-(2*5)	11 , 20
4	-1	IF	$X_{k+1} = X_k + 1;$	12 = 11 + 1	12 , 20

			$P_{k+1} = P_k + (2 * Dy);$	7 = -1 + (2 * 4)	
5	7	ELSE	$X_{k+1} = X_k + 1;$ $Y_{k+1} = Y_k + 1;$ $P_{k+1} = P_k + (2 * dy) - (2 * dx);$	13 = 12 + 1 21 = 20 + 1; 5 = 7 + (2 * 4)-(2*5)	13 , 21
6	5	ELSE	$X_{k+1} = X_k + 1;$ $Y_{k+1} = Y_k + 1;$ $P_{k+1} = P_k + (2 * dy) - (2 * dx);$	14 = 13 + 1 22 = 21 + 1; 3 = 5 + (2 * 4) -(2*5)	14 , 22

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



Problem-02:

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

Solution-

Given-

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

Step-01:

Calculate ΔX and ΔY from the given input.

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

Step-02:

Calculate the decision parameter.

P_k

$= 2\Delta Y – \Delta X$

$= 2 \times 8 – 10$

$= 6$

So, decision parameter $P_k = 6$

Step-03:

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

Thus,

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

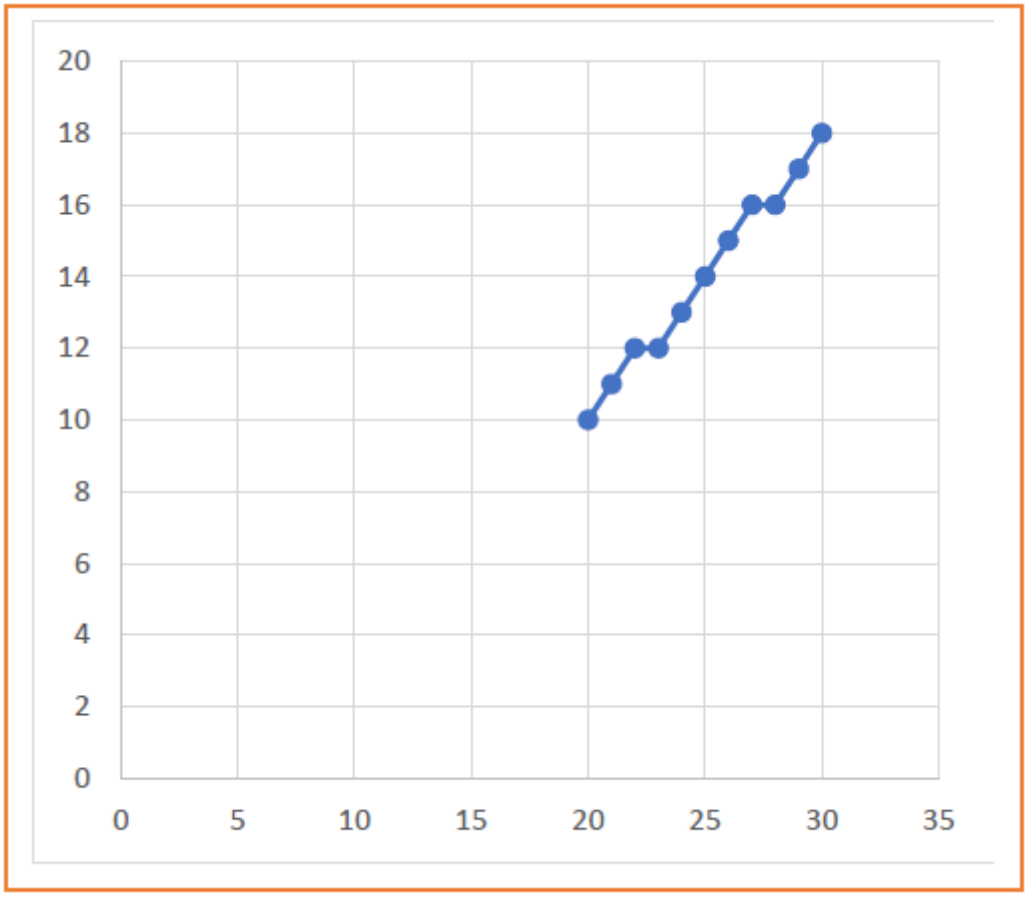
Similarly, Step-03 is executed until the end point is reached

STEP	P = (2 * Dy) - Dx;	P < 0	Calculation Method	Calculation	Points
1	Starting Point				20 , 10
2	6 = (2 * 8) - 10	ELSE	$X_{k+1} = X_k + 1;$ $Y_{k+1} = Y_k + 1;$ $P_{k+1} = P_k + (2 * dy) - (2 * dx);$	21 = 20 + 1 11 = 10 + 1; 2 = 6 + (2 * 8) - (2*10)	21 , 11
3	2	ELSE	$X_{k+1} = X_k + 1;$ $Y_{k+1} = Y_k + 1;$ $P_{k+1} = P_k + (2 * dy) - (2 * dx);$	22 = 21 + 1 12 = 11 + 1;	22 , 12

				-2 = 2 + (2 * 8) -(2*10)	
4	-2	IF	$X_{k+1} = X_k + 1;$ $P_{k+1} = P_k + (2 * Dy);$	23 = 22 + 1 14 = -2 + (2 * 8)	23 , 12
5	14	ELSE	$X_{k+1} = X_k + 1;$ $Y_{k+1} = Y_k + 1;$ $P_{k+1} = P_k + (2 * dy) - (2 * dx);$	24 = 23 + 1 13 = 12 + 1; 10 = 14 + (2 * 8)-(2*10)	24 , 13
6	10	ELSE	$X_{k+1} = X_k + 1;$ $Y_{k+1} = Y_k + 1;$ $P_{k+1} = P_k + (2 * dy) - (2 * dx);$	25 = 24 + 1 14 = 13 + 1; 6 = 10 + (2 * 8)-(2*10)	25 , 14
7	6	ELSE	$X_{k+1} = X_k + 1;$ $Y_{k+1} = Y_k + 1;$ $P_{k+1} = P_k + (2 * dy) - (2 * dx);$	26 = 25 + 1 15 = 14 + 1; 2 = 6 + (2 * 8) - (2*10)	26 , 15

8	2	ELSE	$X_{k+1} = X_k + 1;$ $Y_{k+1} = Y_k + 1;$ $P_{k+1} = P_k + (2 * dy) - (2 * dx);$	27 = 26 + 1 16 = 15 + 1; -2 = 2 + (2 * 8)-(2*10)	27 , 16
9	-2	IF	$X_{k+1} = X_k + 1;$ $P_{k+1} = P_k + (2 * Dy);$	28 = 27 + 1 14 = -2 + (2 * 8)	28 , 16
10	14	ELSE	$X_{k+1} = X_k + 1;$ $Y_{k+1} = Y_k + 1;$ $P_{k+1} = P_k + (2 * dy) - (2 * dx);$	29 = 28 + 1 17 = 16 + 1; 10 = 14 + (2 * 8)-(2*10)	29 , 17
11	10	ELSE	$X_{k+1} = X_k + 1;$ $Y_{k+1} = Y_k + 1;$ $P_{k+1} = P_k + (2 * dy) - (2 * dx);$	30 = 29 + 1 18 = 17 + 1; 6 = 10 + (2 * 8)-(2*10)	30 , 18

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



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-

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.

Difference Between DDA and Bresenham Line Drawing algorithm

DDA Algorithm	Bresenham's Algorithm
1. DDA Algorithm uses floating point, real arithmetic.	1. Bresenham's Algorithm uses fixed points, integer arithmetic.
2. DDA algorithm uses multiplication and division operations.	2. Bresenham's Algorithm uses addition and subtraction operations.
3. DDA algorithm is slower than Bresenham's Algorithm because it uses real arithmetic floating point operations.	3. Bresenham's Algorithm is faster than DDA algorithm because it uses integer arithmetic.

4. DDA algorithm can draw circles and curves with less accuracy.	4. Bresenham's Algorithm can draw circles and curves with much more accuracy.
5. DDA algorithm is less efficient than Bresenham's Algorithm.	5. Bresenham's Algorithm is more efficient than DDA Algorithm.
6. DDA algorithm round-off the co-ordinates to integer that is nearest to the line.	6. Bresenham's Algorithm doesn't round-off the co-ordinates.