BCT 2405 Computer Graphical systems

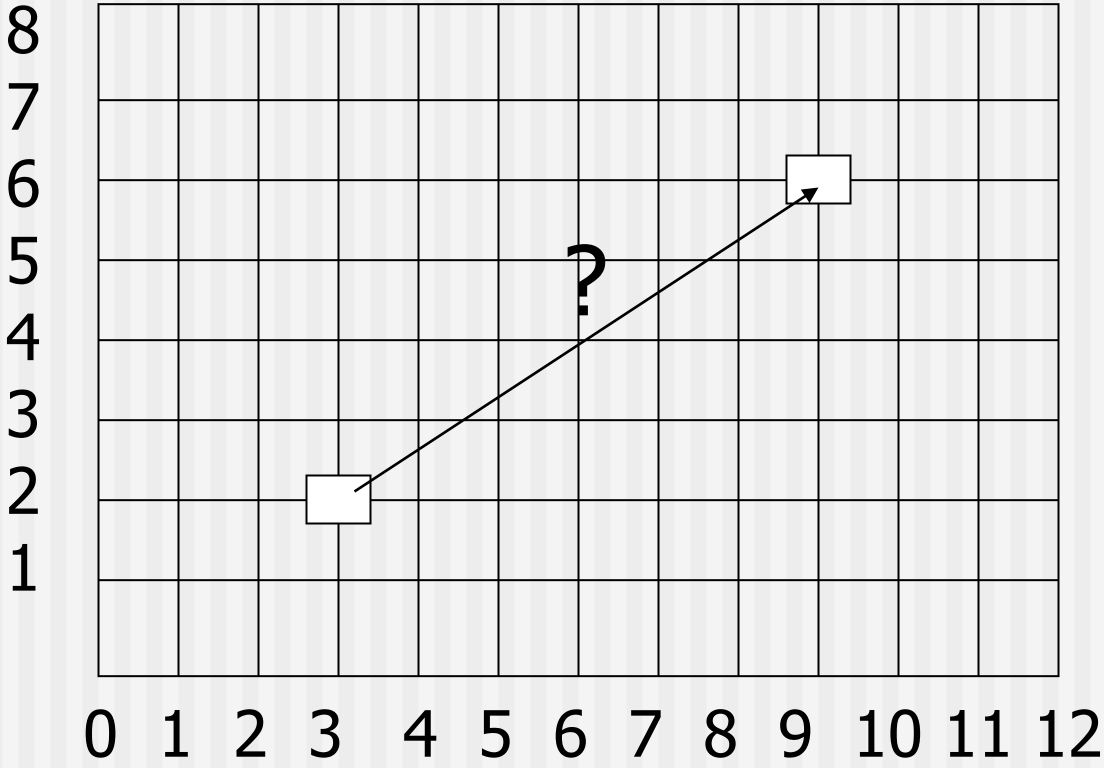
Karanja Mwangi

- Line Drawing Algorithms

Concept: Line drawing algorithm

- Programmer specifies (x,y) values of end pixels
- Need algorithm to figure out which intermediate pixels are on line path
- Pixel (x,y) values constrained to integer values
- Actual computed intermediate line values may be floats
- Rounding may be required. E.g. computed point (10.48, 20.51) rounded to (10, 21)
- Rounded pixel value is off actual line path (jaggy!!)
  - Jaggies are stair-like lines that appear where there should be "smooth" straight lines or curves
- Sloped lines end up having jaggies
- Vertical, horizontal lines, no jaggies

Concept: Line Drawing Algorithm



Line: (3,2) -> (9,6)

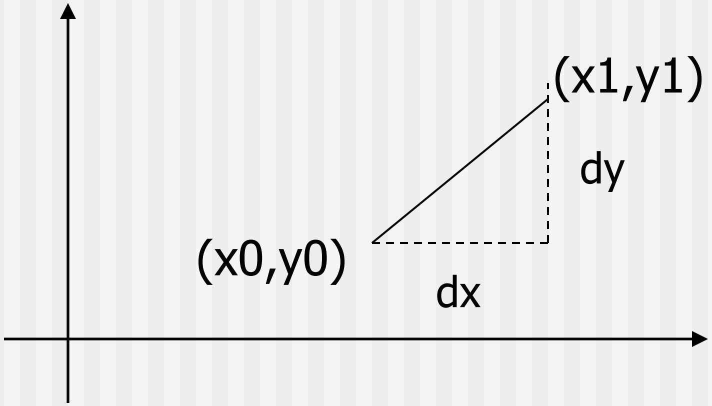
Which intermediate pixels to turn on?

Concept: Line Drawing Algorithm

- Slope-intercept line equation
  - y = mx + b
  - Given two end points (x0,y0), (x1, y1), how to compute m and b?

$$m = \frac{dy}{dx} = \frac{y1 - y0}{x1 - x0}$$

$$b = y0 - m * x0$$



Line Drawing Algorithm -Gradient

- Numerical example of finding slope m:
- (Ax, Ay) = (23, 41), (Bx, By) = (125, 96)

$$m = \frac{By - Ay}{Bx - Ax} = \frac{96 - 41}{125 - 23} = \frac{55}{102} = 0.5392$$

The various Line drawing Algorithms

- **DDA(**Digital Differential Analyzer**) Line Drawing Algorithm**

- **Mid Point Line Drawing Algorithm**

- **Bresenham Line Drawing Algorithm**

Read about (Important to do so)

- Xiaolin Wu's line algorithm
- Gupta-Sproull algorithm

There are many others but we limit ourselves to these

Line Drawing Algorithm: Digital Differential Analyzer (DDA): The Notion -1

- Given the starting and ending coordinates of a line,
- DDA Algorithm attempts to generate the points between the starting and ending coordinates.

DDA Procedure-

If we have the

Starting coordinates = (X_0, Y_0)

Ending coordinates = (X_n, Y_n)

The points generation using DDA Algorithm is as follows

Line Drawing Algorithm: Digital Differential Analyzer (DDA): The Notion -2

Step-1:

Calculate ΔX, ΔY and M from the given input.

These parameters are calculated as-

$\Delta X = X_n - X_0$

$\Delta Y = Y_n - Y_0$

$M = \Delta Y / \Delta X$

For example to **calculate the points between the starting point (5, 6) and ending point (8, 12).**

# Starting coordinates = (X_0, Y_0) = (5, 6)

Ending coordinates = (X_n, Y_n) = (8, 12)

#Calculate ΔX, ΔY and M from the given input.

$\Delta X = X_n - X_0 = 8 - 5 = 3$

$\Delta Y = Y_n - Y_0 = 12 - 6 = 6$

$M = \Delta Y / \Delta X = 6 / 3 = 2$

Line Drawing Algorithm: Digital Differential Analyzer (DDA): The Notion -3

Step-2:

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

```
if (absolute ($\Delta X$) > absolute ($\Delta Y$))
    Steps = absolute ($\Delta X$);
else
    Steps = absolute ($\Delta Y$);
```

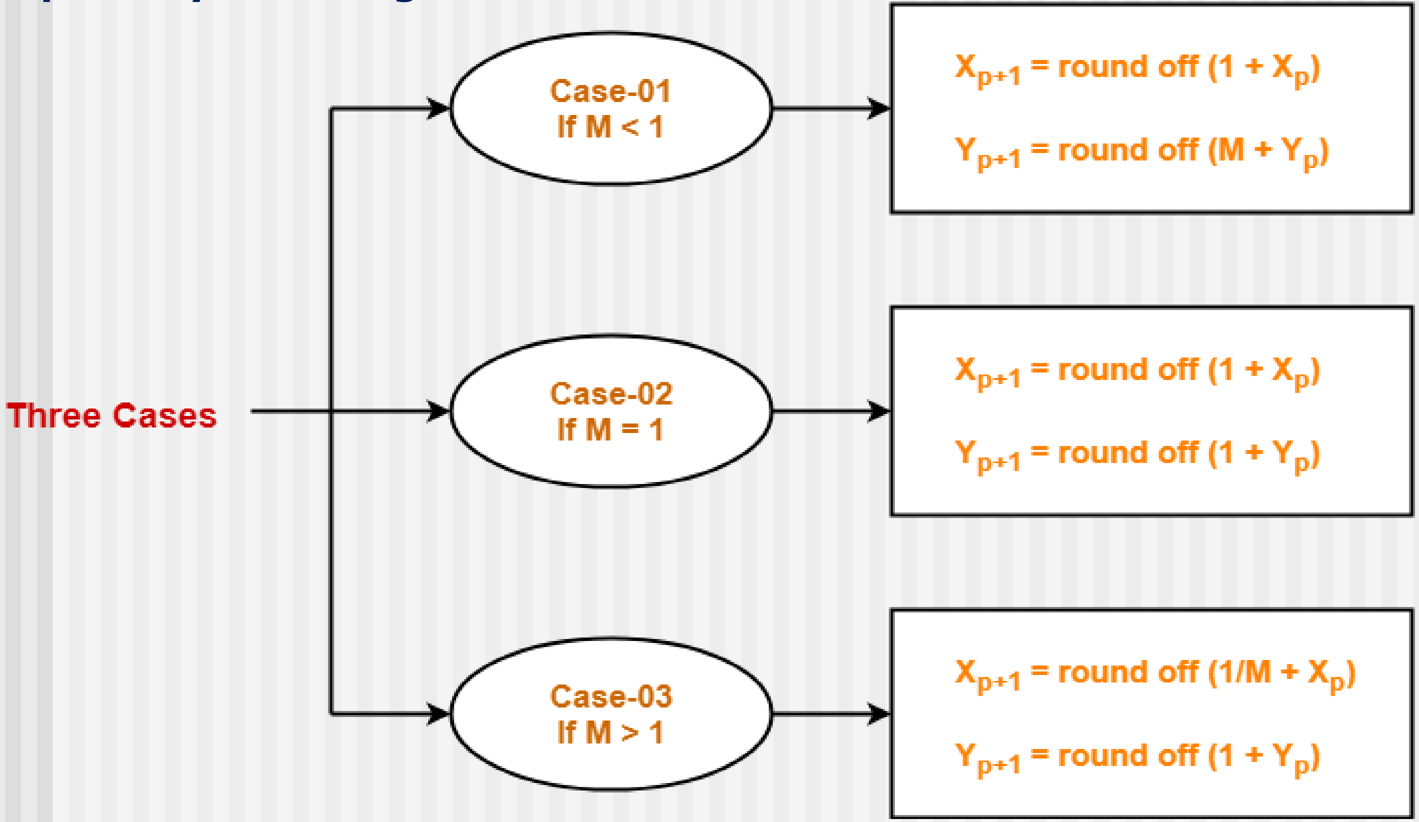
# in our case example of starting point (5, 6) and ending point (8, 12). The number of steps are

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

Line Drawing Algorithm: Digital Differential Analyzer (DDA): The Notion -4

Step-3:

If the current point is (X_p, Y_p) and the next point is (X_{p+1}, Y_{p+1}). Find the next point by following the below three cases-



Three Cases

Case-01
If M < 1

X_{p+1} = round off (1 + X_p)

Y_{p+1} = round off (M + Y_p)

Case-02
If M = 1

X_{p+1} = round off (1 + X_p)

Y_{p+1} = round off (1 + Y_p)

Case-03
If M > 1

X_{p+1} = round off (1/M + X_p)

Y_{p+1} = round off (1 + Y_p)

Line Drawing Algorithm: Digital Differential Analyzer (DDA): The Notion -5

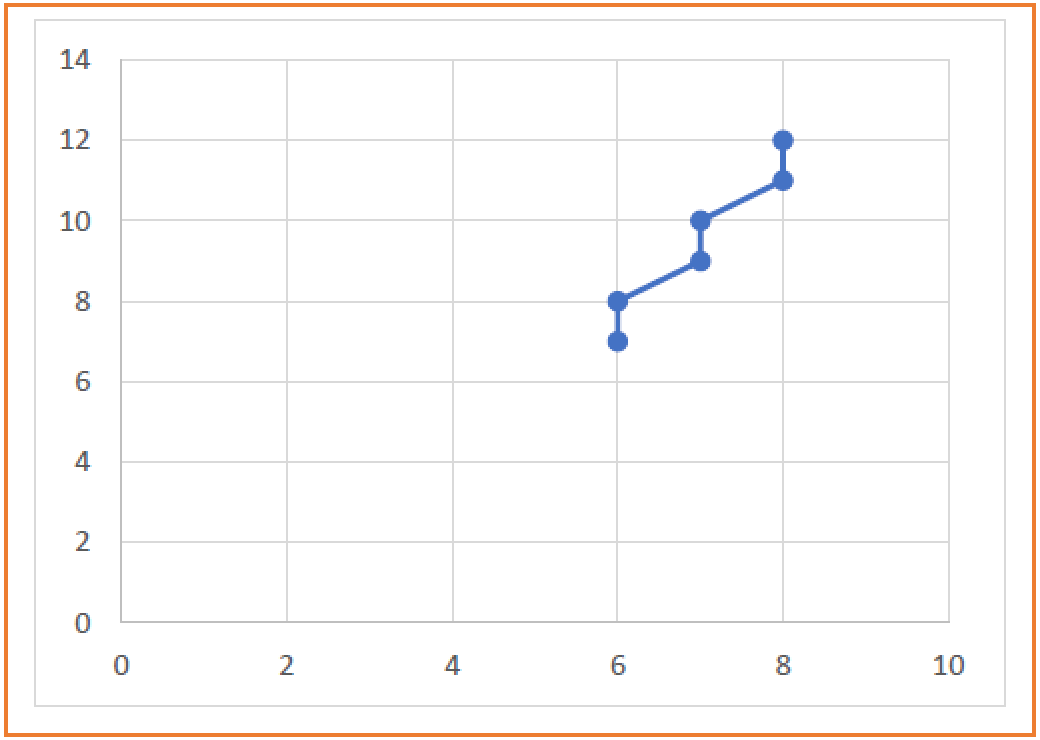
Step-3: continued

# As M > 1, so case-03 is satisfied in our example
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)

Line Drawing Algorithm: Digital Differential Analyzer (DDA): The Notion -5

Step-3: The plot



DDA Line Drawing Algorithm advantages

- DDA is the simplest line drawing algorithm
- It is easy to implement.
- It avoids using the multiplication operation which is costly in terms of time complexity.

DDA Line Drawing Algorithm Drawbacks

- DDA is the simplest line drawing algorithm
  - Not very efficient
  - Round operation is expensive
  - 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.

Exercise Gauge yourself with the tasks given in Print

Line Drawing Algorithm: Mid Point Line Drawing Algorithm The Notion -1

Given the starting and ending coordinates of a line, Mid Point Line Drawing Algorithm attempts to generate the points between the starting and ending coordinates.

Midpoint Procedure-

If we have the

Starting coordinates = (X_0, Y_0)

Ending coordinates = (X_n, Y_n)

The points generation using midpoint Algorithm is as follows

Line Drawing Algorithm :Mid Point Line Drawing Algorithm :The Notion -2

Step-1:

Calculate ΔX and ΔY from the given input.

These parameters are calculated as-

$\Delta X = X_n - X_0$

$\Delta Y = Y_n - Y_0$

For example to **calculate the points between the starting point (20, 10) and ending coordinates (30, 18).**

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

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

#**Calculate ΔX and ΔY from the given input.**

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

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

Line Drawing Algorithm :Mid Point Line Drawing Algorithm :The Notion -3

Step-2:

Calculate the value of initial decision parameter and ΔD.

These parameters are calculated as-

$D_{initial} = 2\Delta Y - \Delta X$

$\Delta D = 2(\Delta Y - \Delta X$

Starting coordinates = $(X_0, Y_0) = (20, 10)$

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

#**Calculate ΔX and ΔY from the given input.**

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

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

Calculate $D_{initial}$ and ΔD as-

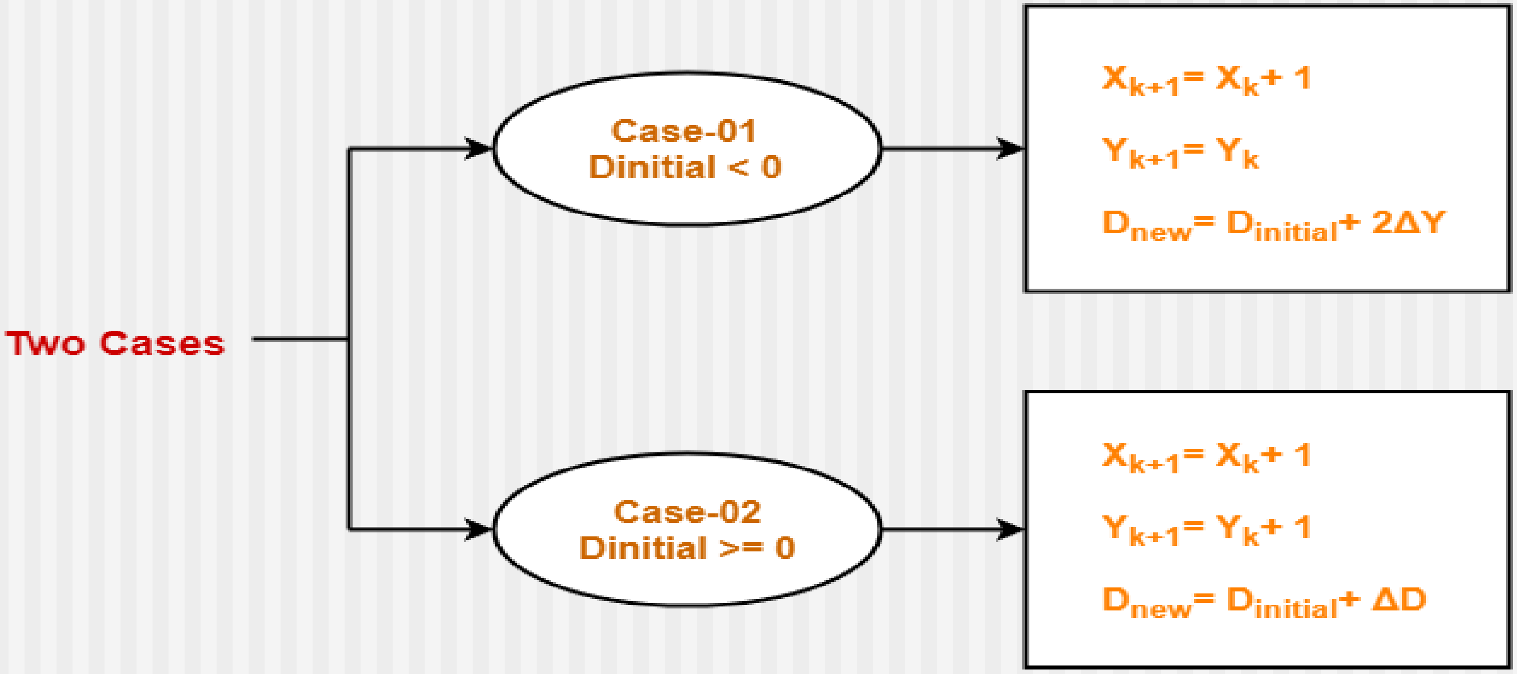
$D_{initial} = 2\Delta Y - \Delta X = 2 \times 8 - 10 = 6$

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

Line Drawing Algorithm :Mid Point Line Drawing Algorithm :The Notion -4

Step-3:

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



Two Cases

Case-01
Dinitial < 0

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

$Y_{k+1} = Y_k$

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

Case-02
Dinitial >= 0

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

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

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

Line Drawing Algorithm :Mid Point Line Drawing Algorithm :The Notion -4

Step-3: continued

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

Thus,

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

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

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

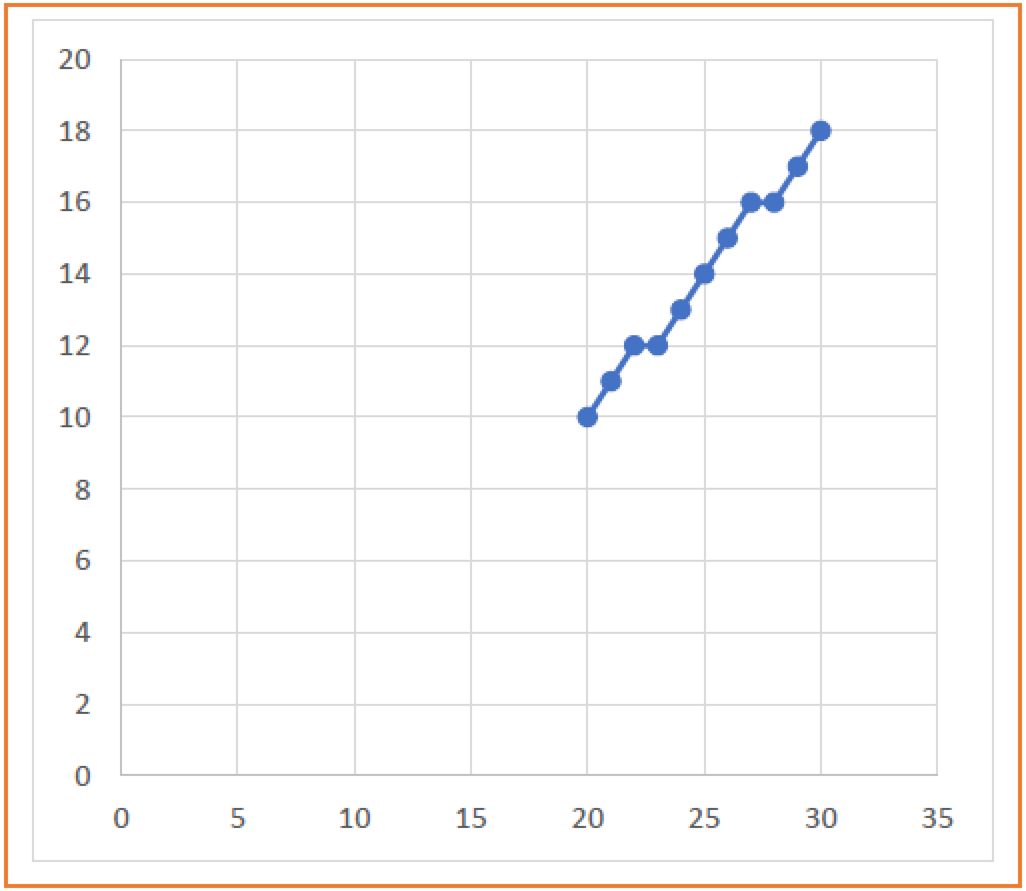
Similarly, Step 3 is executed until the end point is reached.

.

$D_{initial}$	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

Line Drawing Algorithm :Mid Point Line Drawing Algorithm :The Notion -5

Step-3: The plot



Mid Point Line Drawing Algorithm : Advantages

- 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

Mid Point Line Drawing Algorithm : Drawbacks

- 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.

Line Drawing Algorithm- Bresenham's Line-Drawing Algorithm :The Notion -1

- Given the starting and ending coordinates of a line, Bresenham's algorithm attempts to generate the points between the starting and ending coordinates.

Bresenham Procedure-

If we have the

- Starting coordinates = (X_0, Y_0)
- Ending coordinates = (X_n, Y_n)
- The points generation using Bresenham's algorithm is as follows

Line Drawing Algorithm- Bresenham's Line-Drawing Algorithm :The Notion -2

Step-1:

Calculate ΔX and ΔY from the given input.

These parameters are calculated as-

$\Delta X = X_n - X_0$

$\Delta Y = Y_n - Y_0$

For example to **calculate the points between the starting point (9, 18) and ending coordinates (14, 22).**

Starting coordinates = (X_0, Y_0) = (9, 18)

Ending coordinates = (X_n, Y_n) = (14, 22)

#Calculate ΔX, ΔY from the given input.

$\Delta X = X_n - X_0 = 14 - 9 = 5$

$\Delta Y = Y_n - Y_0 = 22 - 18 = 4$

Line Drawing Algorithm- Bresenham's Line-Drawing Algorithm :The Notion -3

Step-2:

Calculate the decision parameter P_k.

It is calculated as-

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

.

# in our case example of starting point (9, 18) and ending coordinates (14, 22). Where $\Delta X = X_n - X_0 = 14 - 9 = 5$ and $\Delta Y = Y_n - Y_0 = 22 - 18 = 4$

The decision parameter will be

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

$$= 2 \times 4 - 5$$

$$= 3$$

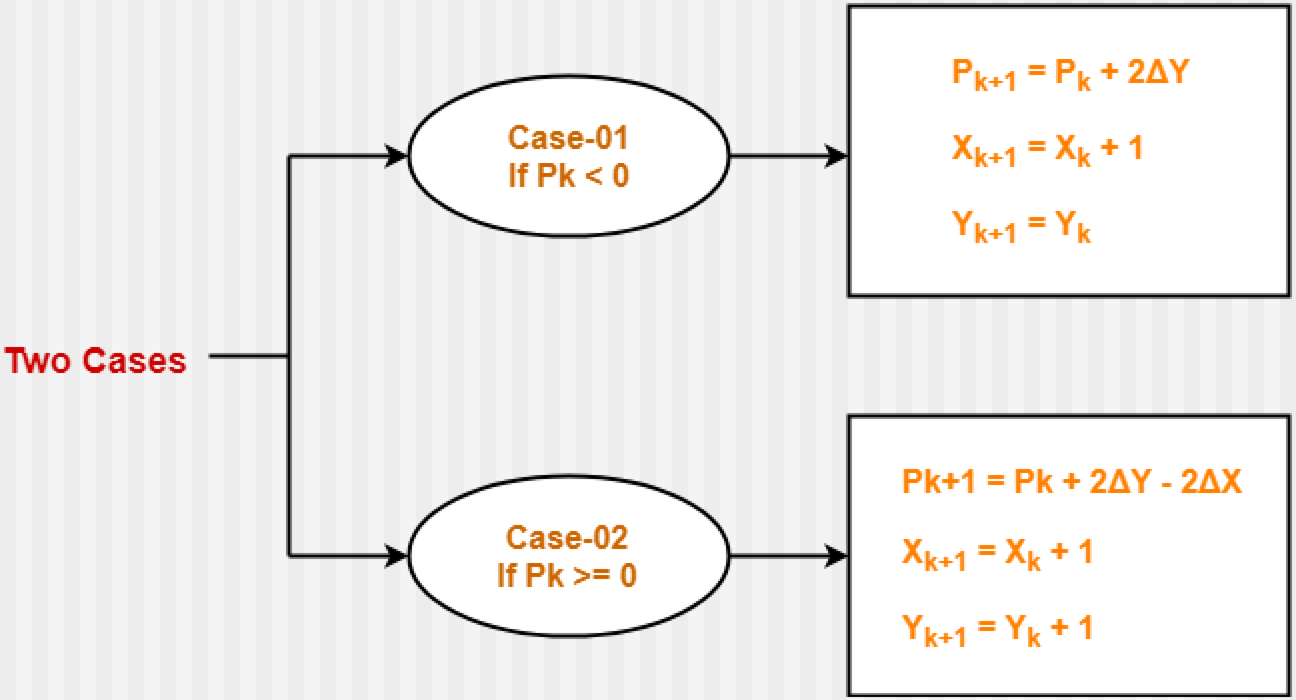
So, decision parameter $P_k = 3$

Line Drawing Algorithm- Bresenham's Line-Drawing Algorithm :The Notion -4

Step-3:

If the current point is (X_p, Y_p) and the next point is (X_{p+1}, Y_{p+1}). Find the next point the next point depending on the value of decision parameter P_k.

Follow the below two cases-



Two Cases

Case-01
If Pk < 0

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

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

$Y_{k+1} = Y_k$

Case-02
If Pk >= 0

Pk+1 = Pk + 2ΔY - 2ΔX

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

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

Line Drawing Algorithm- Bresenham's Line-Drawing Algorithm :The Notion -5

Step-3: continued

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

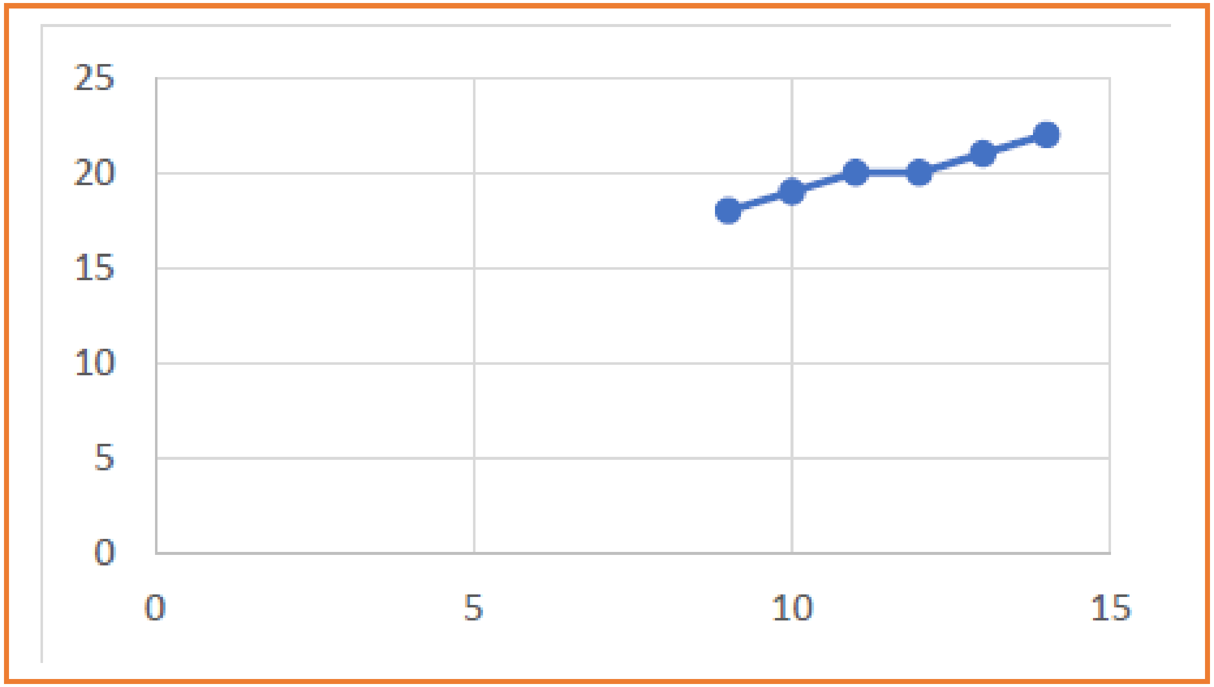
$P_{k+1} = P_k + 2\Delta Y – 2\Delta X = 3 + (2 \times 4) – (2 \times 5) = 1$
$X_{k+1} = X_k + 1 = 9 + 1 = 10$
$Y_{k+1} = Y_k + 1 = 18 + 1 = 19$

Similarly, **Step-3** is executed until the end point is reached or number of iterations equals to 4 times. (Number of iterations = ΔX – 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

Line Drawing Algorithm- Bresenham's Line-Drawing Algorithm :The Notion -5

Step-3: The plot



Bresenham’s Line-Drawing Algorithm: Advantages

- 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.
- This uses only integer calculations.

Bresenham’s Line-Drawing Algorithm: Drawbacks

- Cannot effectively handle “jaggies”
  - Remember: WHAT ARE jaggies - **Jaggies** are stair-like lines that appear where there should be "smooth" straight lines or curves
    - **We will deal with them when dealing with aliasing effect**
- Improves line accuracy in generation of points but not yet smooth

Assignment (Important to do so)

- 1. Read on various image formats such as Ai, wmf, Cmx, cgm,svg ,odg, eps , dxf , bmp, jpeg ,Gif ,Tiff,PICT and png
  - **Explain what the abbreviation stand for and some history on the format**
  - **State whether each of the graphic format above is raster or vector**
  - **Briefly explain a typical application or area of usage of each of the format**
- **2. Read about Xiaolin Wu's line algorithm and**
- **Gupta-Sproull algorithm**

  Can you implement all the Five algorithms in a programming language of your choice?OpenGL

3 .**How is Rasterization carried for other 2D primitives? Circles , Polygon, Ellipses , Triangles etc ?Read the textbook**

Assignment (Important to do so)

3. **Can you implement all the Five algorithms in a programming language of your choice? OpenGL**

4 .**How is Rasterization carried for other 2D primitives? Circles , Polygon, Ellipses , Triangles etc ?**

Read the textbook
https://piazza.com/class_profile/get_resource/kp2du2rm59l1rh/kp2e1yk9gkx4hy

Read page Graphics Output Primitives from page 45