Two Dimensional Algorithm

Output Primitives

- The *basic building blocks for pictures* are referred to as output primitives.
- Output primitives are the geometric structures that used to describe the shapes and colors of the objects.
- They include character strings, and geometric entities such as points, straight lines, curved lines, polygons, circles etc.
- Points and straight line segments are the most basic components of a picture.
- In raster scan systems, a picture is completely specified by the set of intensities for the pixel positions in the display. The process that converts picture definition into a set of pixel intensity values is called scan conversion. This is done by display processor.

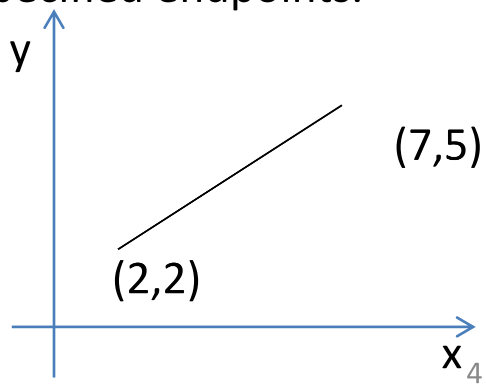
Lines and Points

- With raster-scan system, a point can be plotted, by simply turning on the electron beam at that point.
  - putpixel(20, 20, RED)
- And a random-scan system stores the point plotting instructions in the display list file.
  - LDXA 100 Load data value 100 into the X register.

  -LDYAP 450 Draw point at(100, 450)
- In raster scan systems line drawing is accomplished by calculating the intermediate positions along the line path between two specified endpoints.
- In random scan systems, line drawing is accomplished by retrieving line drawing commands from the display list.
- Scan lines are numbered consecutively from 0, starting at the bottom of the screen; and pixel columns are numbered from 0, from left to right across each scan line.

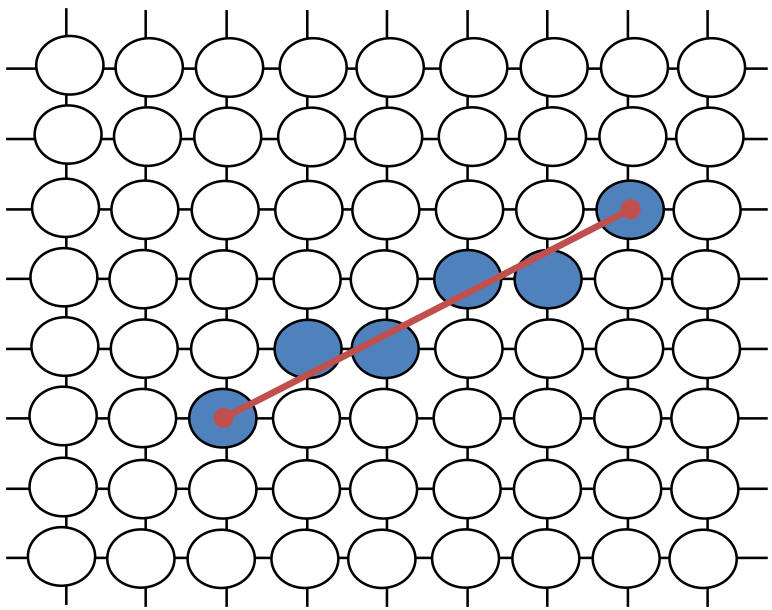
Points and Lines

- Points
  - Plotted by converting co-ordinate position to appropriate operations for the output device (e.g. : in CRT monitor, the electron beam is turned on to illuminate the screen phosphor at the selected location.)
- Line
  - A line segment in a scene is defined by the coordinate positions of the line end-points
  - Plotted by calculating intermediate positions along the line path between two specified endpoint positions.
  - Screen locations are referenced with integer values, so plotted positions may only approximate actual line positions between two specified endpoints.
  - E.g: position (10.48,20.51) → (10,21).



The Problem…..

- But what happens when we try to draw this on a pixel based display?



- How do we choose which pixels to turn on?

Consideration

- Considerations to keep in mind:
  - The line has to look good
    - Avoid *jaggies (i.e. the rounding of coordinates values to integers causes lines to be displayed with a stairstep appearance as represented in the figure)*
  - It has to be lightening fast!
    - How many lines need to be drawn in a typical scene?

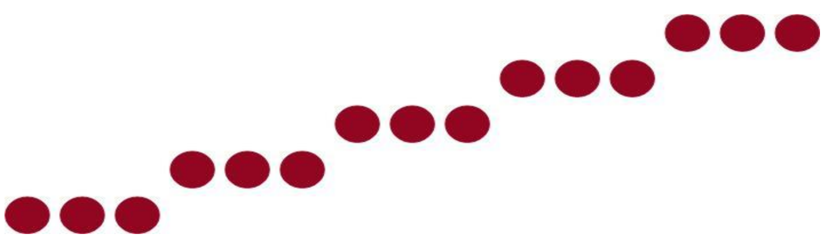


Figure: Stairstep effect (jaggies) produced when a line is generated as a series of pixel position.

Line Drawing Algorithms

- Direct use of Line Equation
- Digital Differential Analyzer Algorithm(DDA)
- Bresenham's Line Drawing Algorithm(BSA)

Direct Use of Line Equation

- The slope-intercept equation of a straight line is:

  $y = mx + b$

  where, m = slope of line and, b = y-intercept.
- For any two given points (x_1 , y_1) and (x_2 , y_2)
- slope (m) = $\dfrac{(y_2 - y_1)}{(x_2 - x_1)}$

$b = y_1 - m.x_1$ i.e. from above equation

- At any point (x_k, y_k)

  $y_k = mx_k + b$ ………… 1
- At (x_{k+1}, y_{k+1}),

  $y_{k+1} = mx_{k+1} + b$ ……….. 2
- Subtracting 1 from 2 we get,

  $y_{k+1} - y_k = m(x_{k+1} - x_k)$
- Here $(y_{k+1} - y_k)$ incremental in y as corresponding increment in x.

  therefore, $\Delta y = m.\Delta x$

  or $\Delta x = \dfrac{\Delta y}{m}$

- For incremental algorithm in line drawing ,
  - Increment *x* by 1
  - Computer corresponding *y* and display pixel at position *(x_i, round (y_i))*

For $|m| < 1$

- Set Δx proportional to horizontal deflection voltage. Then

$$\Delta y = m.\Delta x$$

For $|m| > 1$

- Set Δy set proportional to vertical deflection voltage. Then

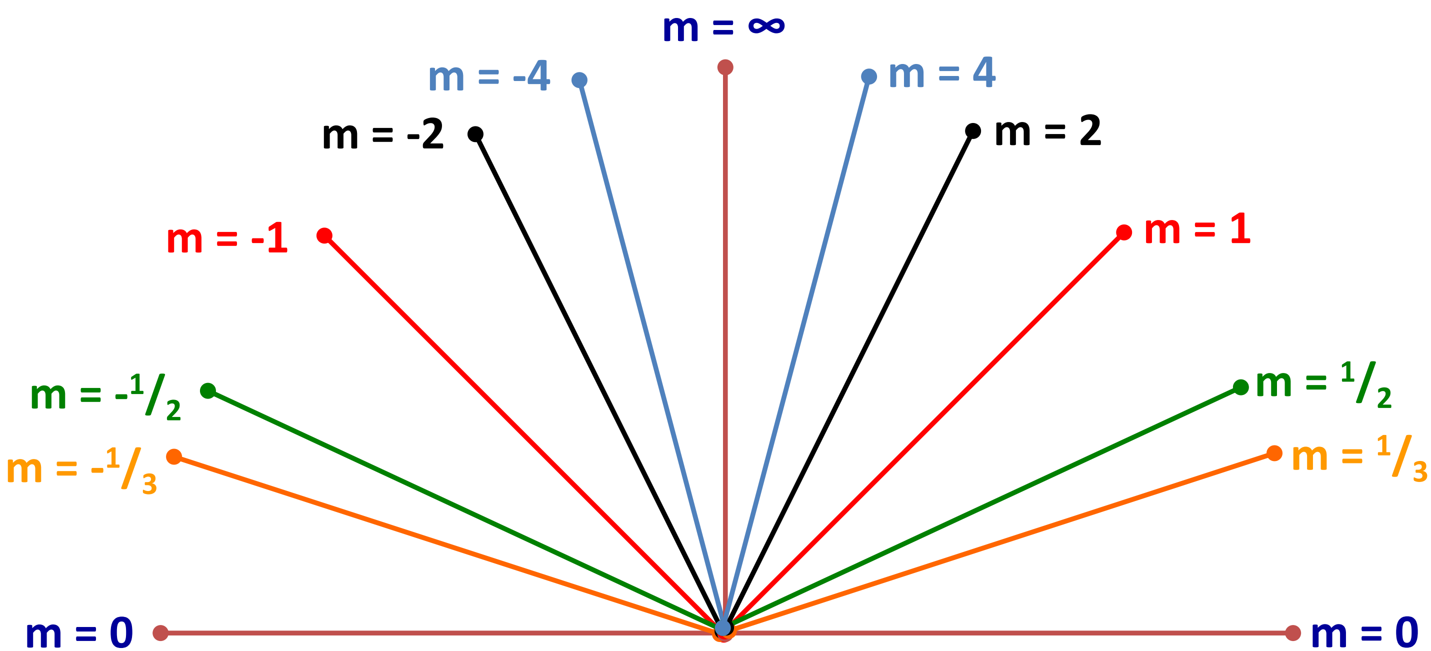
$$\Delta x = \frac{\Delta y}{m}$$

For $|m| = 1$

- Δx = Δy ➔ horizontal and vertical deflection voltages are equal

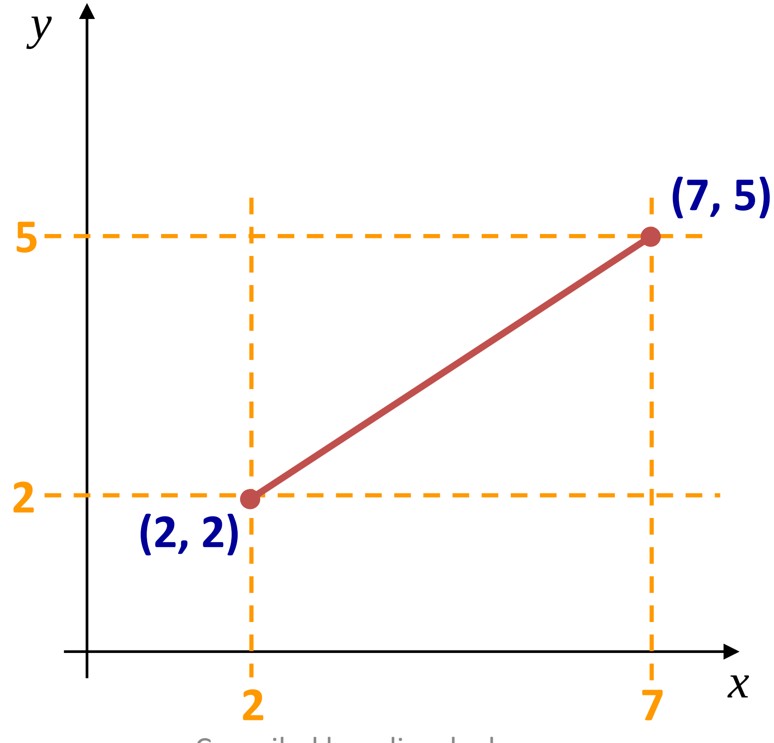
Lines and Slopes

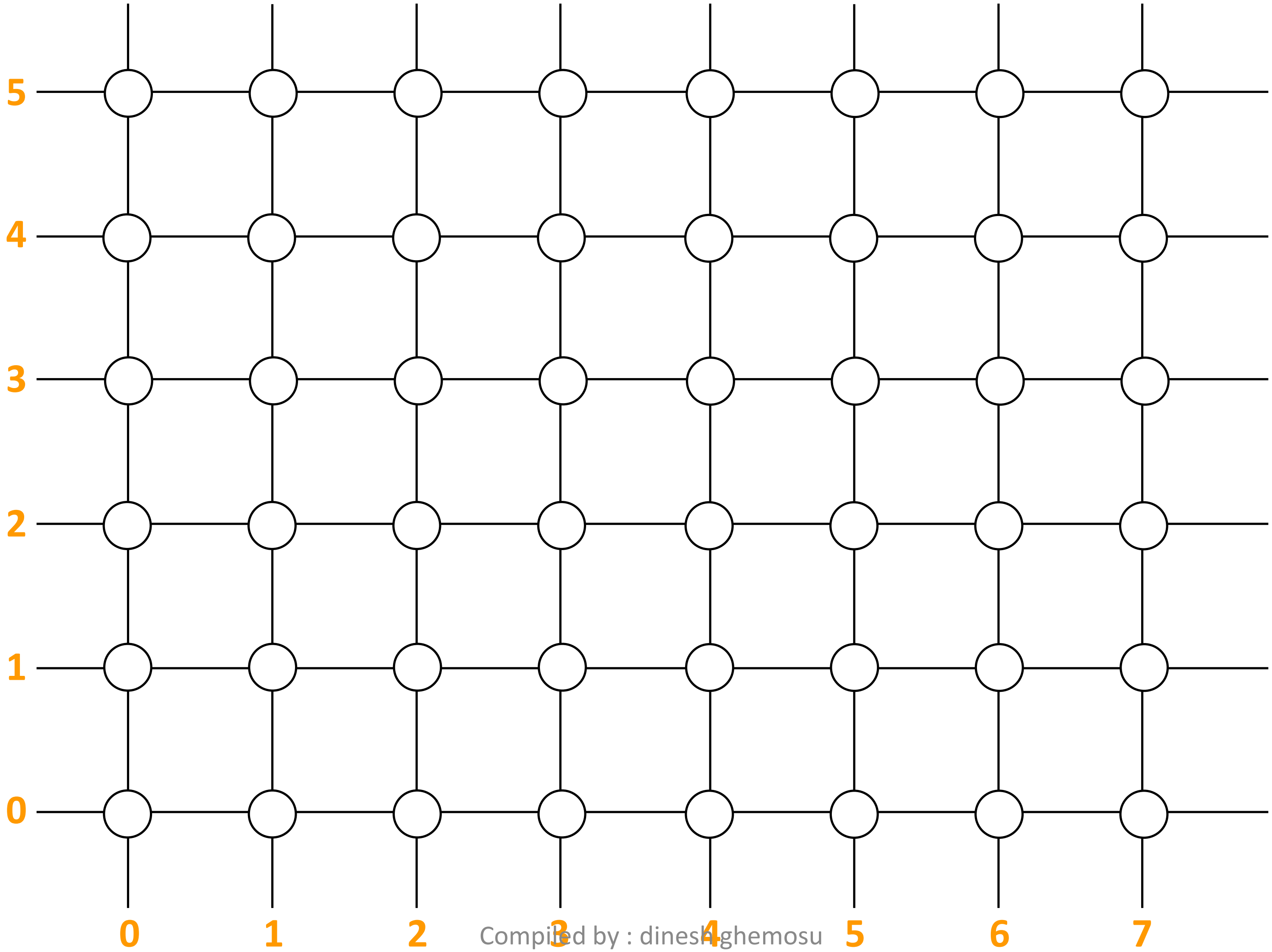
- The slope of a line (m) is defined by its start and end coordinates
- The diagram below shows some examples of lines and their slopes

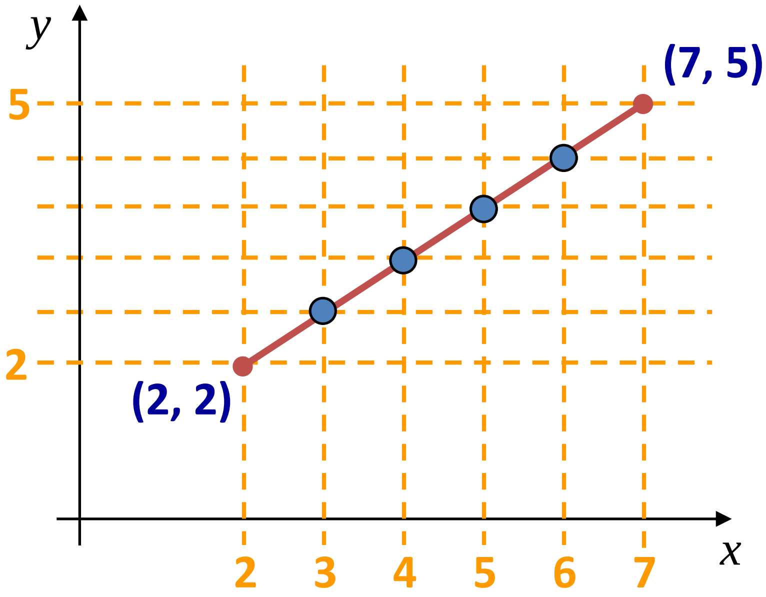


A very simple solution

- We could simply work out the corresponding y coordinate for each unit x coordinate
- Let's consider the following example:




5
4
3
2
1
0
0
1
2
3
4
5
6
7



- First work out m and b:

$$m = \frac{5-2}{7-2} = \frac{3}{5}$$

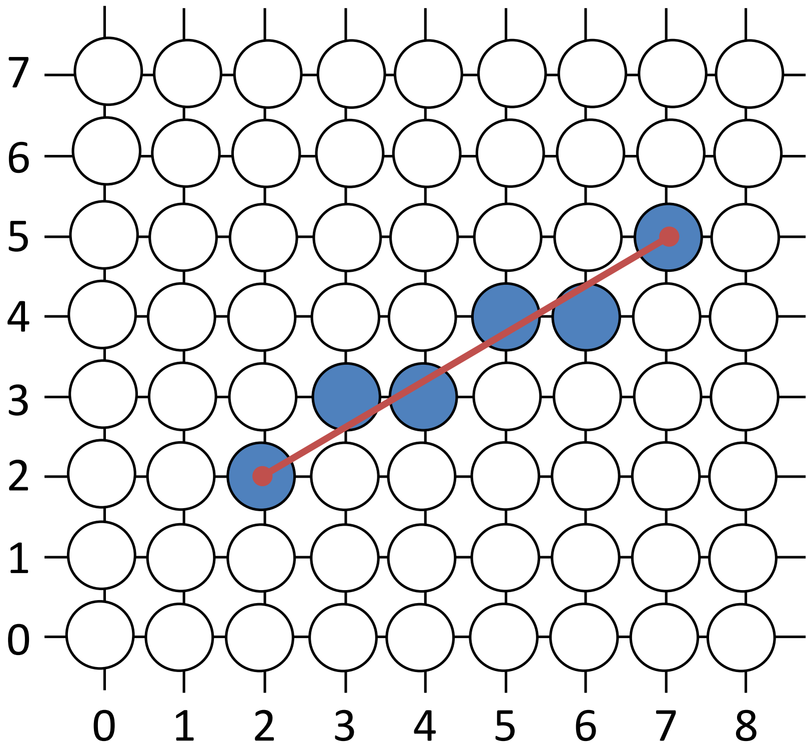
$$b = 2 - \frac{3}{5} * 2 = \frac{4}{5}$$

Now for each x value work out the y value:

$$y(3) = \frac{3}{5} \cdot 3 + \frac{4}{5} = 2\frac{3}{5} \qquad y(4) = \frac{3}{5} \cdot 4 + \frac{4}{5} = 3\frac{1}{5}$$

$$y(5) = \frac{3}{5} \cdot 5 + \frac{4}{5} = 3\frac{4}{5}$$

- Now just round off the results and turn on these pixels to draw our line.



$$y(3) = 2\frac{3}{5} \approx 3$$

$$y(4) = 3\frac{1}{5} \approx 3$$

$$y(5) = 3\frac{4}{5} \approx 4$$

$$y(6) = 4\frac{2}{5} \approx 4$$

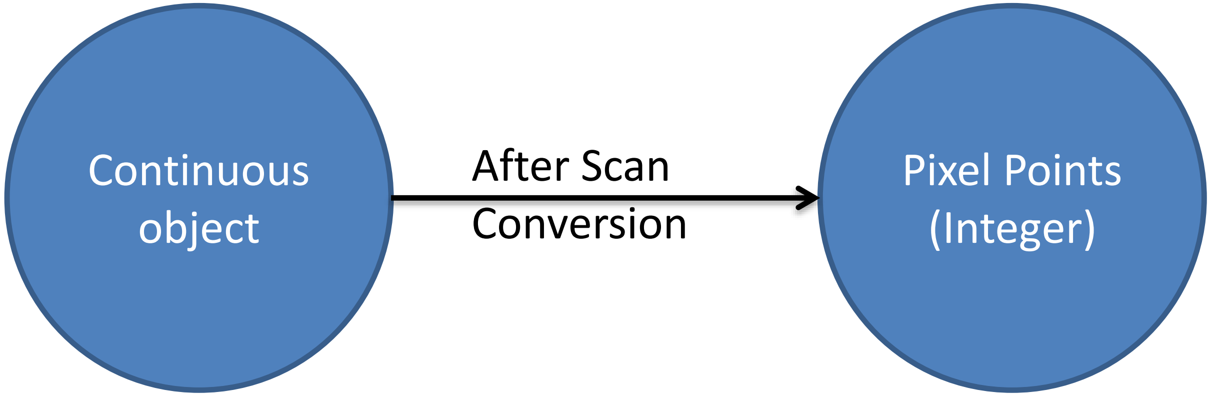
Some Terms

Vector Generation

- On our raster system, we can generate images by turning the pixels ON or OFF. The process of turning ON of the pixel for a line segment is called as **Vector Generation**.

Scan Conversion

- The process of conversion of the rasterized picture stored in a frame buffer to the rigid display pattern of video is called as **scan conversion**.



DDA Algorithm

- Digital Differential Analyzer (DDA) is a scan conversion line drawing algorithm based on calculating either Δx or Δy from the equation

$$\Delta y = m.\ \Delta x$$

- We sample the line at unit intervals in one coordinate and determine the corresponding integer values nearest the line path in another co-ordinate.

Consider a line with positive slope and *proceed from left to right*

Case I:

- If m <= 1, we sample x-coordinate. So Δx =1 and compute each successive y values as: $\Delta x = x_{k+1} - x_k = 1$
  - $y_{k+1} = y_k + m$ where $m = \frac{y_{k+1} - y_k}{x_{k+1} - x_k} = \frac{\Delta y}{\Delta x}$
  - Here k takes from starting point and increase by 1 until final end point.
  - m can be any real value between 0 and 1.

Case II:

- If m>1, we sample Δy =1 and calculate corresponding x value as: $x_{k+1} = x_k + \frac{1}{m}$ $\Delta y = y_{k+1} - y_k = 1$

Consider a line with positive slope that proceed from right to left.

Case I:

- If |m | <=1, we sample Δx = -1 and calculate

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

Case II:

- If |m|>1, we sample Δy = -1 and calculate

$$x_{k+1} = x_k - \frac{1}{m}$$

DDA Algorithm

Step 1: Input the line endpoints and store the left endpoint in (x_1, y_1) and right endpoint in (x_2, y_2).

Step 2: Calculate the values of dx and dy, $dx = x_2 - x_1$, $dy = y_2 - y_1$.

Step 3: if(abs(dx) > abs(dy))

- *steplength* = abs(dx)

else

- *steplength* = abs(dy)

Step 4: Calculate the values of x-increment and y-increment.

- xIncrement = dx/ *steplength*
- yIncrment = dy/ *steplength*

Step 5:Set $x = x_1$ and $y = y_1$

Step 6: Plot(x, y).

Step 7: for k=1 to *steplength* do

- x = x + xIncrement
- y = y + yIncrement
- Perform round off, and plot each calculated (x, y) i.e. Plot(round(x), round(y)).

Step 8: End

Advantages of DDA Algorithm

- Faster than direct use of the line equation since it calculates the line without any floating point multiplication i.e. only integer calculations.
- Simplest algorithm since does not require special skills for its implementation

Disadvantages of DDA Algorithm

- It is orientation dependent, due to this, the point accuracy is poor.
- A floating point addition is still needed in determining each successive point which is time consuming.
- Involves, continuous round offs which can cause the calculated pixel positions to drift away from the actual line path. It causes *jaggies.*

DDA Algorithm

Step 1: Input the line endpoints and store the left endpoint in (x_1, y_1) and right endpoint in (x_2, y_2).

Step 2: Calculate the values of dx and dy, $dx = x_2 – x_1$, $dy = y_2 – y_1$.

Step 3: if(abs(dx) > abs(dy))

steplength = abs(dx)

else

steplength = abs(dy)

Step 4: Calculate the values of x-increment and y-increment.

xIncrement = dx**/** *steplength*

yIncrment = dy**/** *steplength*

Step 5:Set $x = x_1$ and $y = y_1$

Step 6: Plot(x, y).

Step 7: for k=1 to *steplength* do

x = x + xIncrement

y = y + yIncrement

Perform round off, and plot each calculated (x, y) i.e. Plot(round(x), round(y)).

Step 8: End

Digitize the line with endpoints (1, 5) and (7, 2) using DDA algorithm.

Here, dx = 7-1=6, and dy = 2-5 = -3,

So, steplength = 6 (since abs(dx)>abs(dy)).

Therefore, xIncrement = dx/steplength = 6/6 = 1, and

yIncrement = dy/steplength = -3/6 = -1/2 = -0.5

Based on these values the intermediate pixel calculation is shown in the table below.

k	x_{k+1}	y_{k+1}	(x_{k+1}, y_{k+1})	Plot in screen (x_{k+1}, y_{k+1})
1	2	4.5	(2, 4.5)	(2, 5)
2	3	4	(3, 4)	(3, 4)
3	4	3.5	(4, 3.5)	(4,4)
4	5	3	(5,3)	(5,3)
5	6	2.5	(6,2.5)	(6,3)
6	7	2	(7,2)	(7,2)

Questions

- **Consider a line from (0, 0) to (6, 7). Using simple DDA algorithm, rasterize this line.**
- ***Digitize the line with endpoints (1, -6) and (4, 4) using DDA algorithm.***
- ***Digitize the line with endpoints (1, 6), (6, 10) using DDA algorithm.***
- ***Trace DDA algorithm for line with endpoints (1, 2), (5, 6).***
- ***Trace DDA algorithm for endpoints (1, 7), (6, 3).***