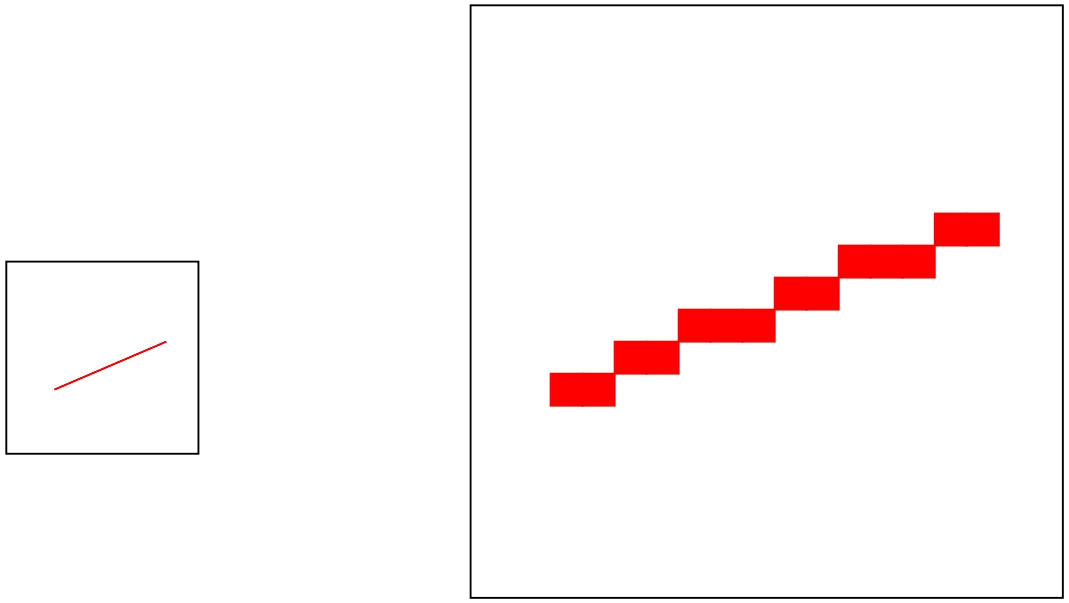
Introduction

The *digital* ***drflerential analyzer (DDA) is a scan-conversion line algorithm based on*** calculating either y axis or ***x axis.We sample the line at unit in- Straight linesegment with*** tervals in one coordinate and determine corresponding integer values nearest the five sampling positions along line path for the other coordinate.

Line Drawing

- Line drawing is fundamental to computer graphics.
- We must have fast and efficient line drawing functions.

Rasterization Problem: Given only the two end points, how to compute the intermediate pixels, so that the set of pixels closely approximate the ideal line.

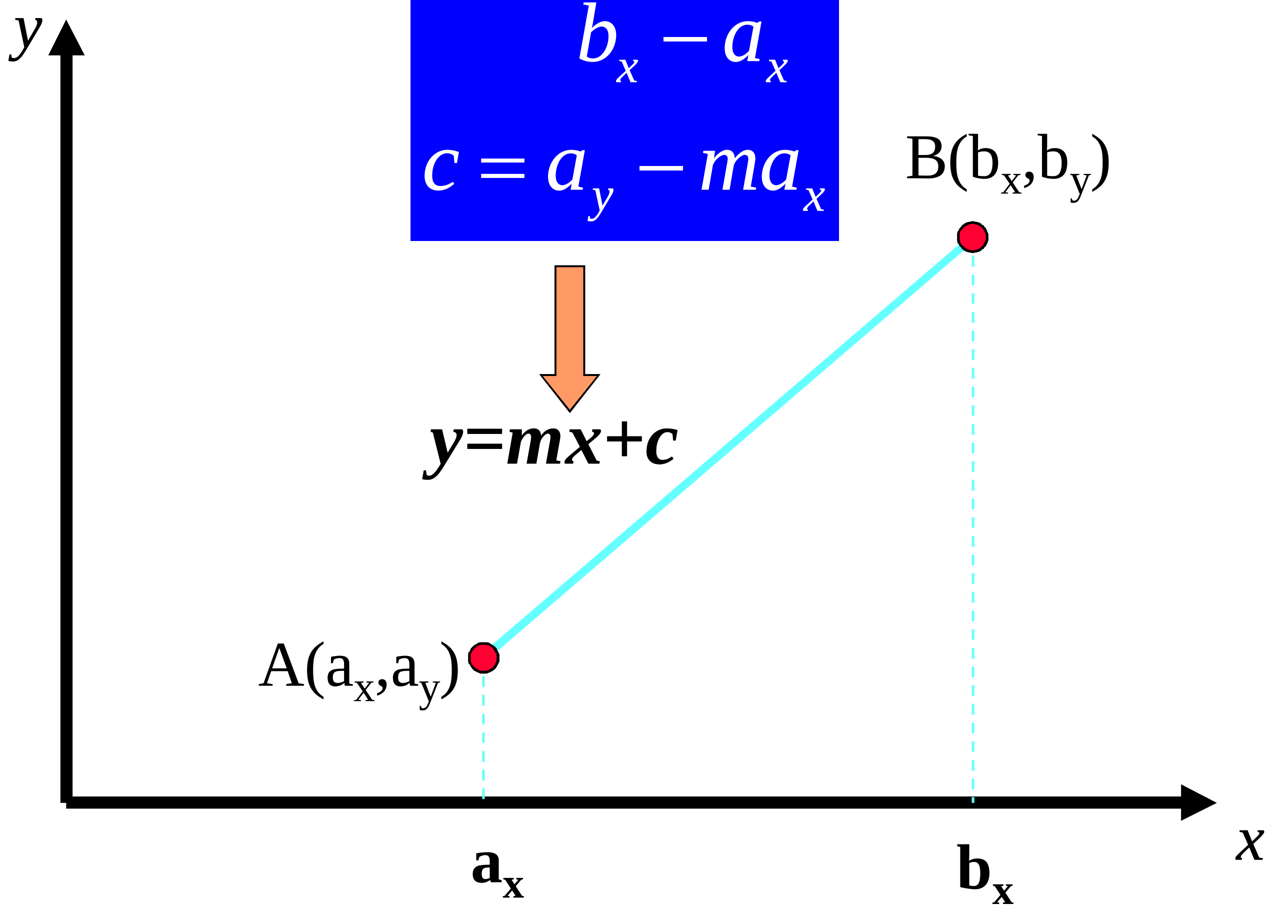


Line Drawing - Analytical Method

$$m = \frac{b_y - a_y}{b_x - a_x}$$

$$c = a_y - ma_x$$

$$y = mx + c$$



Line Drawing - Analytical Method

```
double  m = (double)(by-ay)/(bx-ax);
double  c =  ay - m*ax;
double y;
int iy;
for (int x=ax ;   x <=bx ;  x++) {
     y = m*x + c;
     iy = round(y);
     setPixel (x, iy);
}
```

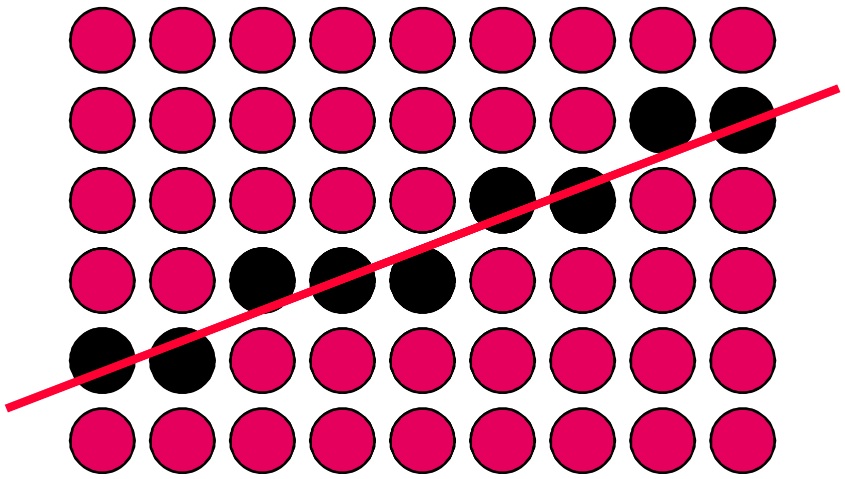
- Directly based on the analytical equation of a line.
- Involves floating point multiplication and addition
- Requires round-off function.

Incremental Algorithms

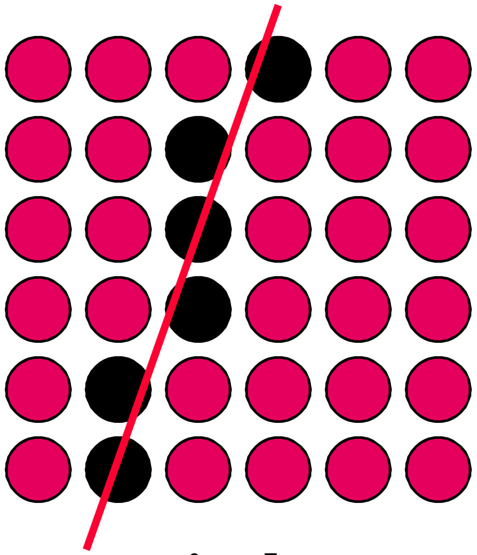
I have got a pixel on the line (Current Pixel).
How do I get the next pixel on the line?

Compute one point based on the previous point:

(x_0, y_0)……………………..$(x_k, y_k) \Rightarrow (x_{k+1}, y_{k+1})$ …….

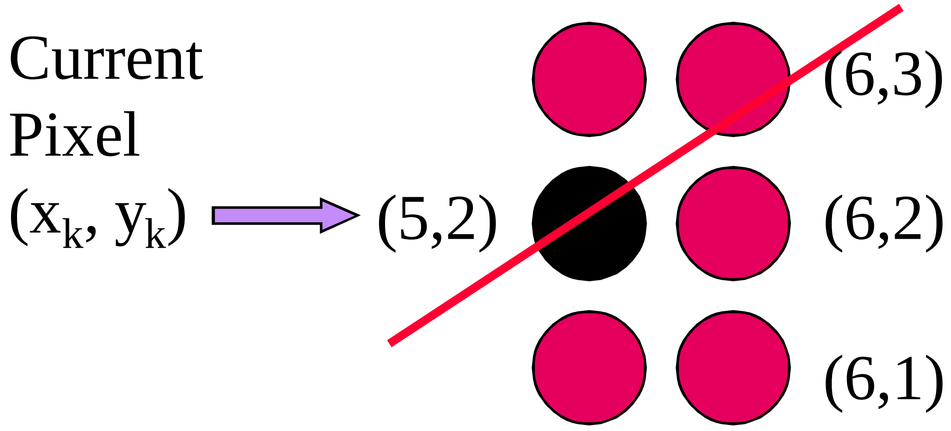


Next pixel on next column
(when slope is small)



Next pixel on next row
(when slope is large)

Incrementing along x



To find $(x_{k+1}, y_{k+!})$:

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

y_{k+1} = **?**

- **Assumes that the next pixel to be set is on the next column of pixels (Incrementing the value of x !)**
- **Not valid if slope of the line is large.**

Line Drawing - DDA

Digital Differential Analyzer Algorithm is an incremental algorithm.

Assumption: Slope is less than 1 (Increment along x).

Current Pixel = (x_k, y_k).

(x_k, y_k) lies on the given line. $\longrightarrow$ $y_k = m.x_k + c$

Next pixel is on next column. $\longrightarrow$ $x_{k+1} = x_k + 1$

Next point (x_{k+1}, y_{k+1}) on the line $\longrightarrow$

$$\begin{aligned} y_{k+1} &= m.x_{k+1} + c \\ &= m(x_k + 1) + c \\ &= y_k + m \end{aligned}$$

Given a point (x_k, y_k) on a line, the next point is given by

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

$$\mathbf{y_{k+1} = y_k + m}$$

```
#include "device.h"

#define ROUND(a) ((int)(a+0.5))

void lineDDA (int xa, int ya, int xb, int yb)
{
  int dx = xb - xa, dy = yb - ya, steps, k;
  float xIncrement, yIncrement, x = xa, y = ya;

  if (abs (dx) > abs (dy)) steps = abs (dx);
  else steps = abs  dy);
  xIncrement = dx / (float) steps;
  yIncrement = dy / (float) steps;

  setPixel (ROUND(x), ROUND(y));
  for (k=0; k<steps; k++) {
    x += xIncrement;
    y += yIncrement;
    setPixel (ROUND(x), ROUND(y));
  }
}
```

Application

In computer graphics, a hardware or software implementation of a **digital differential analyzer (DDA)** is used for linear interpolation of variables over an interval between start and end point. DDAs are used for rasterization of lines, triangles and polygons.

In its simplest implementation the DDA algorithm interpolates values in interval $[(x_{start}, y_{start}), (x_{end}, y_{end})]$ by computing for each x_i the equations $x_i = x_{i-1}+1/m$, $y_i = y_{i-1} + m$, where $\Delta x = x_{end} - x_{start}$ and $\Delta y = y_{end} - y_{start}$ and $m = \Delta y/\Delta x$

Scope of Research

The DDA method can be implemented using floating point or integer arithmetic. The native floating-point implementation requires one addition and one rounding operation per interpolated value (e.g. coordinate x, y, depth, color component etc.) and output result.