Computer Graphics
EG678EX

2-D Algorithms

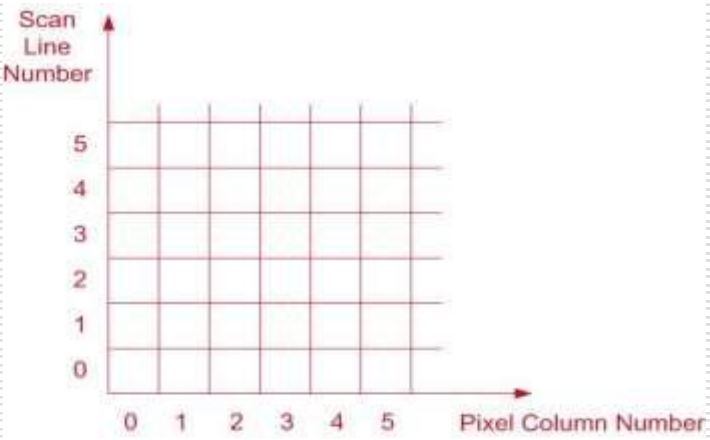
Prepared By: Dipesh Gautam

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
    - 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 → "***the jaggies***". E.g: position (10.48,20.51) → (10,21).

- Jaggies

- Pixel position: referenced by scan line number and column number



Line Drawing Algorithms

- Slope-Intercept Equation

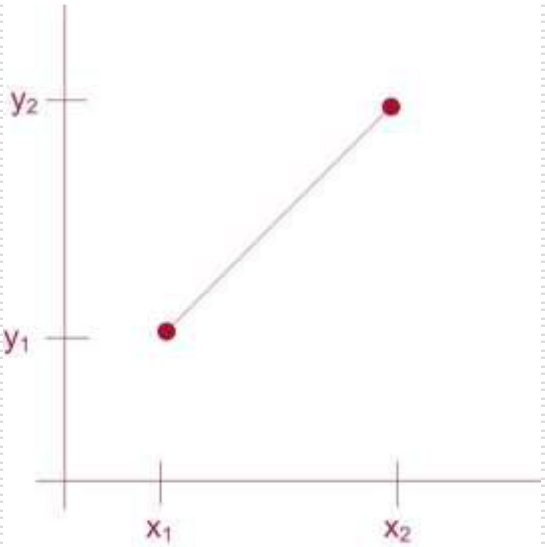
$$y = m.x + b$$

- Slope

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

- Intercept

$$b = y_1 - m.x_1$$



- Interval Calculation

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

- Analog System
  - |m| <1
    - Set Δx proportional to horizontal deflection voltage. Then

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

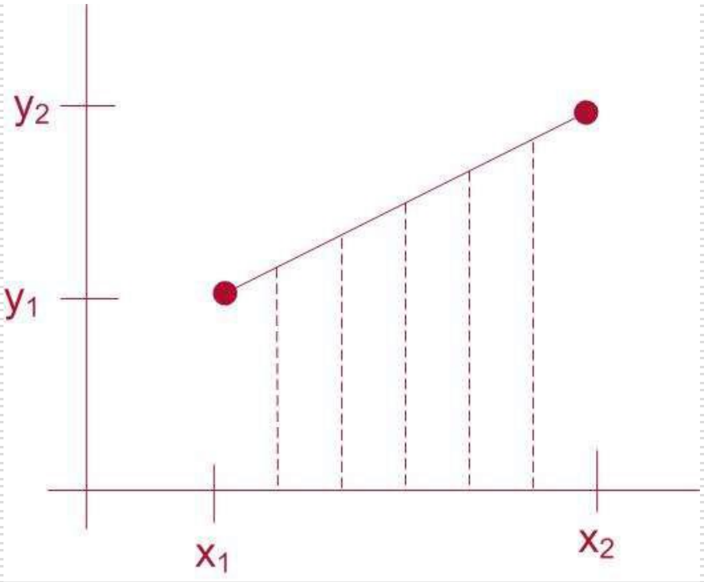
  - |m|>1
    - Set Δy set proportional to vertical deflection voltage. Then

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

  - |m|=1
    - Δx = Δy ➔ horizontal and vertical deflection voltages are equal

Digital System

- Sample a line at discrete positions and determine nearest pixel to the line at each sampled position



DDA Algorithm

- → Digital Differential Analyzer
  - Sample the line at unit intervals in one coordinate
  - Determine the corresponding integer values nearest the line path in another co-ordinate

DDA Algorithm (left to right)

- Slope $m = \dfrac{y_{k+1} - y_k}{x_{k+1} - x_k} = \dfrac{\Delta y}{\Delta x}$

- For |m|<1 (|Δy|< |Δx|)
  - Sample line at unit interval in x co-ordinate

$$y_{k+1} = y_k + m \qquad \Delta x = x_{k+1} - x_k = 1$$

- For |m|>1 (|Δy|> |Δx|)
  - Sample line at unit interval in y co-ordinate

$$x_{k+1} = x_k + \frac{1}{m} \qquad \Delta y = y_{k+1} - y_k = 1$$

DDA Algorithm (right to left)

- Slope $m = \dfrac{y_{k+1} - y_k}{x_{k+1} - x_k} = \dfrac{\Delta y}{\Delta x}$

- For |m|<1 (|Δy|< |Δx|)
  - Sample line at unit interval in x co-ordinate

$$y_{k+1} = y_k - m \qquad \Delta x = x_{k+1} - x_k = -1$$

- For |m|>1 (|Δy|> |Δx|)
  - Sample line at unit interval in y co-ordinate

$$x_{k+1} = x_k - \frac{1}{m} \qquad \Delta y = y_{k+1} - y_k = -1$$

DDA Algorithm

1. Input the two line endpoints and store the left endpoint in (x_0,y_0)
2. Plot first point (x_0,y_0)
3. Calculate constants Δx, Δy
4. If $|\Delta x| > |\Delta y|$ *steps* = $|\Delta x|$ else *steps* = $|\Delta y|$
5. Calculate XInc = $|\Delta x|$ / *steps* and *Y*Inc = $|\Delta y|$ / *steps*
6. At each x_k along the line, starting at k=0, Plot the next pixel at (xk + XInc, yk + YInc)
7. Repeat step 6 *steps* times

Pseudo Code

```
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));
      }
}
```

DDA Algorithm

- How about problem and performance ?
  - Assignment:
    - What's the performance problem in above pseudo code ?
    - How DDA performance can be improved ?