Computer Graphics

Lab 2

Bresenham's Algorithm

Advantages of DDA

- It calculates the pixel positions faster than the calculations performed by using the equation y=mx +b.
- Multiplication is eliminated as the x and y increments are used to determine the position of the next pixel on a line

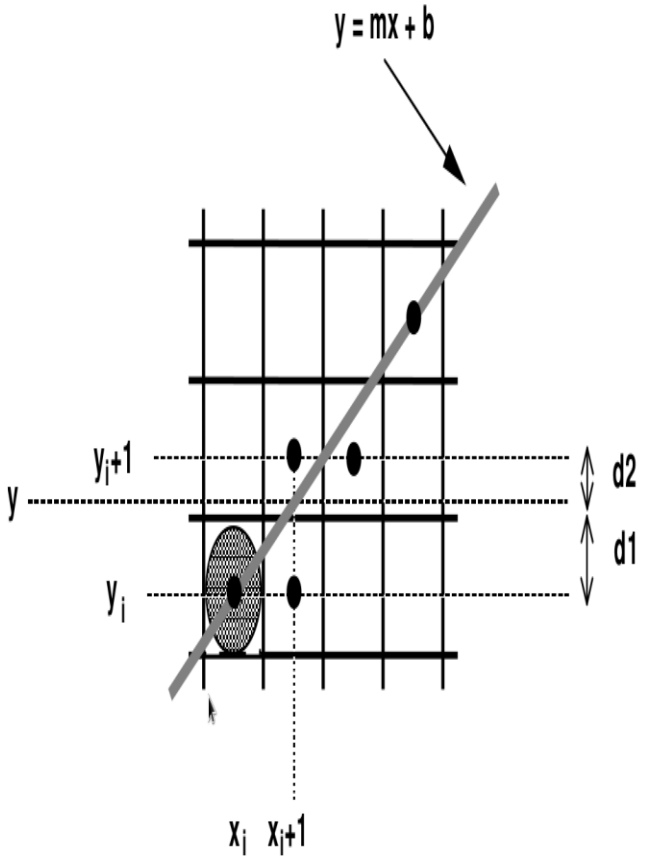
Disadvantages of DDA

- The rounding and floating point operations are time consuming.
- The round-off error which results in each successive addition leads to the drift in pixel position, already calculated

Bresenham's Line Algorithm

- The Bresenham algorithm is another incremental scan conversion algorithm. The big advantage of this algorithm is that, it uses only integer calculations. Moving across the x axis in unit intervals and at each step choose between two different y coordinates.
- For example, as shown in the following illustration, from position (2, 3) you need to choose between (3, 3) and (3, 4). You would like the point that is closer to the original line.
- If the pixel to be displayed occurs at a position (X_i, Y_i) then the next pixel is either at (X_i+1,Y_i) or (X_i+1,Y_i+1)
- The 'Y' coordinate at the pixel position Xk+1 can be obtained from $Y=m(X_i+1)+b$

- Separation between (X_{i+1},Y_i) and (X_{i+1},Y) is d1
- separation between (X_{i+1},Y) and (X_{i+1}, Y_{i+1}) is d2 then
- $d1 = y - y_i$ and
- $d2 = (Y_i+1) - Y$



- $Y=m(X_i+1)+b$ . . . . eq 1
- $d1=y - y_i$
- $d1=m(x_i+1)+b - Y_i$ (from eqn (1) .....(2)
- And $d2= (Y_i+1) - Y$
- $=(Y_i+1) - [m(X_i+1)+b]$
- $=(Y_i+1) - m(X_i+1) - b$ .....(3)
- The difference is given as
- d1-d2 =
- $= m(X_i+1)+b-Y_i-[(Y_i+1)-m(X_i+1)-b]$
- $=m(X_i+1)+b-Y_i-(Y_i+1)+m(X_i+1)+b$
- $d1-d2 = 2m(X_i+1)-2Y_i+2b - 1$ .....(4)

- **A decision parameter P_i can be obtained by substituting m= dy/dx in equation 4**

- **$d1 - d2 = 2m(X_i+1) - 2Y_i + 2b - 1$**

  $= 2\, dy/dx\, (X_i+1) - 2Y_i + 2b - 1$

  $= \dfrac{2\, dy(X_i+1) - 2\, dx.Y_i + 2b.dx - dx}{dx}$

- **$dx(d1-d2) = 2\, dy(X_i+1) - 2\, dx.Y_i + 2b.dx - dx$**

  $= 2\, dyX_i + 2\, dy - 2\, dx.Y_i + 2b.dx - dx$

  $= 2\, dyX_i - 2\, dx.Y_i + c$

- **Where, $dx(d1-d2) = P_i$ and**

  $c = 2\, dy + dx(2b-1)$

  $P_i = 2\, dyX_i - 2\, dx.Y_i + c$ . . . . … …(5)

- if P_i > = 0 , then d1-d2 >0 ,d1>d2 , Y= Y_i +1 .
- if P_i < 0 , then d1-d2 <0 ,d1<d2 , Y= Y_i .

- At i+1 step, the value of P_i is given as
- $P_{i+1} = 2\,dyX_{i+1} - 2\,dx.Y_{i+1} + C$.....(6) (from 5)

- Eq 6 – eq 5
- $P_{i+1} - P_i = (2\,dyX_{i+1} - 2\,dx.Y_{i+1} + c) - (2\,dyX_i - 2\,dx.Y_i + c)$

  $= 2dy(X_{i+1} - X_i) - 2\,dx(Y_{i+1} - Y_i)$ .....(7)

- Since $x_{i+1} = X_i + 1$ The eqn 7 becomes

- $P_{i+1} - P_i = 2dy(X_i + 1 - X_i) - 2\,dx(Y_{i+1} - Y_i)$

  $= 2dy - 2\,dx(Y_{i+1} - Y_i)$

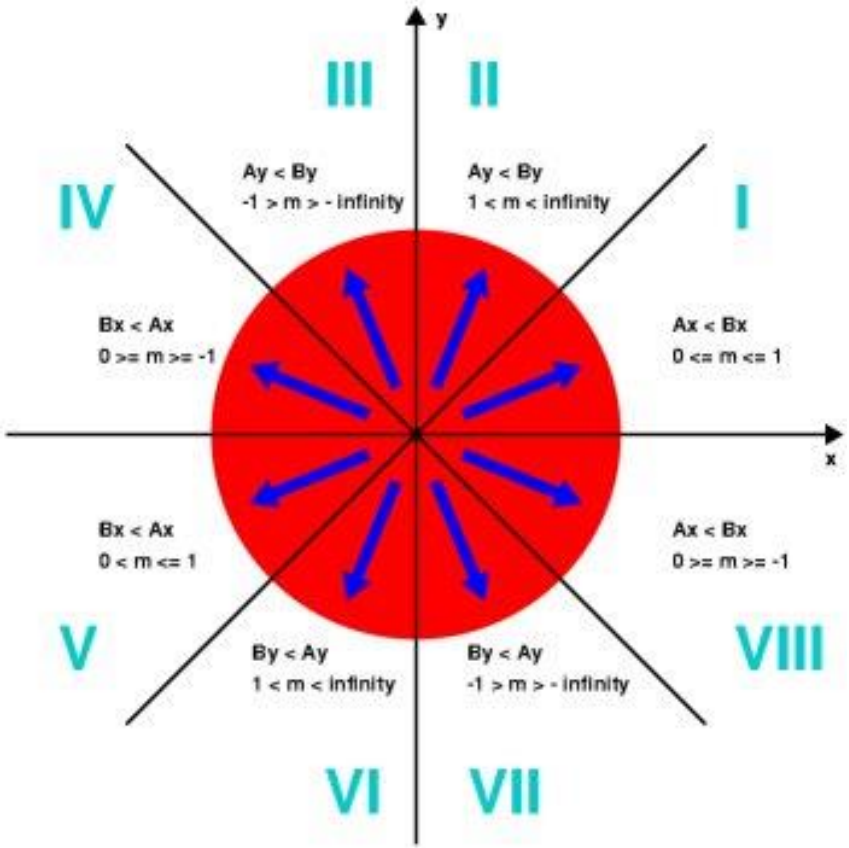
- $P_{i+1} = P_i + 2dy - 2\,dx(Y_{i+1} - Y_i)$ .....(8)
- Where $(Y_{i+1} - Y_i)$ is either 0 or 1 based on the sign of Pk.
- **The starting parameter P0 at the pixel position (X0,Y0) is given as**
- **P0 = 2dy – dx** .....(9)

- y=mx+b is the eq of line
- $P_i = 2\,dyX_i - 2\,dx.Y_i + c$
- $c = 2\,dy + dx(2b-1)$
- $P_0 = 2\,dyX_0 - 2\,dx.Y_0 + c$

  $= 2\,dyX_0 - 2\,dx.Y_0 + 2\,dy + dx(2b-1)$
- $Y_0 = mX_0 + b \rightarrow b = Y_0 - mX_0 \rightarrow b = Y_0 - dyX_0/dx$
- $P_0 = 2\,dyX_0 - 2\,dx.Y_0 + 2\,dy + dx(2(Y_0 - dyX_0/dx)-1)$

  $= 2\,dyX_0 - 2\,dx.Y_0 + 2\,dy + 2dxY_0 - 2dyX_0 - dx$
- $P_0 = 2\,dy - dx$

The OCTANTS demo demonstrates octants and shows the general direction for drawing lines for each octant.



Algorithm :

- **Step 1** – Input the two end-points of line, storing the left end-point in (x0,y0)
- **Step 2** – Plot the point (x0,y0).
- **Setp3** - determine the initial value of the decision parameter by calculating the constants |dx|,| dy|, 2|dy| and 2|dy|-2|dx| as

  $P0 = 2dy - dx$
- **Step 4** – At each X_i along the line, starting at i = 0, perform the following test

  if $\mathbf{P_i} < 0$, the next point to plot is $(X_i + 1, Y_i)$ and $\mathbf{P_{i+1}} = \mathbf{P_i} + 2dy$

  Otherwise, $(X_i, +1\ Y_i + 1)$ and $\mathbf{P_{i+1}} = \mathbf{P_i} + 2dy - 2dx$
- **Step 5** – Repeat step 4 (dx – 1) times.

Advantages of Bresenham's Alg.

- It uses only integer calculations
- So, it is faster than DDA

Example 1 :

- Let the given end points for the line be (30,20) and (40, 28)

solution:

- M = dy/dx = (y2 – y1)/(x2-x1) = (28 – 20)/(40-30) = 0.8
- dy = 8 and dx = 10 Major=x , Minor=y
- Bx > Ax increment one to x
- By > Ay increment one to y
- The initial decision parameter P0 is

  P0 = 2dy – dx = 2(8) – 10 = 16 – 10 = 6

  P0 = 6
- The constants 2dy and 2dy-2dx are

  2dy = 2(8) = 16 2dy-2dx = 2(8)- 2(10) =16 – = 20

  2dy = 16 2dy – 2dx = - 4
- Shimaa.gamal@mu.edu.eg

The starting point (x0, y0)=(30,20) and the successive pixel positions are given in the following table

i	P_i	(X_{i+1} , Y_{i+1})
0	6	(31,21)
1	2	(32,22)
2	-2	(33,22)
3	14	(34,23)
4	10	(35,24)
5	6	(36,25)
6	2	(37,26)
7	-2	(38,26)
8	14	(39,27)
9	10	(40,28)

- End points (30,20) (40,28)
- dx=x2-x1=40-30=10 dy= y2-y1=28-20=8
- P0=2dy-dx=2X8-10 = 6 >0 pt(31,21)
- Pi+1= pi+2dy-2dx= 6+16-20 =2 >0 (32,22)
- Pi+1=2+16-20 =-2 <0 (33,22)
- Pi+1= pi+2dy =-2+16 = 14 >0 (34,23)
- Pi+1= pi+2dy-2dx=14+16-20=10>0 (35,24)
- Pi+1=10+16-20=6 >0 (36,25)
- Pi+1= 6+16-20=2>0 (37,26)
- Pi+1=2+16-20=-2 <0 (38,26)
- Pi+1= pi+2dy=-2+16=14>0 (39,27)
- Pi+1=pi+2dy-2dx= 14+16-20=10>0 (40,28)