**Chapter II Scan Conversion of a line**

A line connects two points. It is a basic element in graphics. To draw a line, you need two points between which you can draw a line. In the following three algorithms, we refer the one point of line as X0,Y0X0,Y0 and the second point of line as X1,Y1X1,Y1.

DDA Algorithm

Digital Differential Analyzer DDADDA algorithm is the simple line generation algorithm which is explained step by step here.

**Step 1** − Get the input of two end points (X0,Y0)(X0,Y0) and (X1,Y1)(X1,Y1).

**Step 2** − Calculate the difference between two end points.

dx = X1 - X0

dy = Y1 - Y0

**Step 3** − Based on the calculated difference in step-2, you need to identify the number of steps to put pixel. If dx > dy, then you need more steps in x coordinate; otherwise in y coordinate.

if (absolute(dx) > absolute(dy))

Steps = absolute(dx);

else

Steps = absolute(dy);

**Step 4** − Calculate the increment in x coordinate and y coordinate.

Xincrement = dx / (float) steps;

Yincrement = dy / (float) steps;

**Step 5** − Put the pixel by successfully incrementing x and y coordinates accordingly and complete the drawing of the line.

for(int v=0; v < Steps; v++)

{

x = x + Xincrement;

y = y + Yincrement;

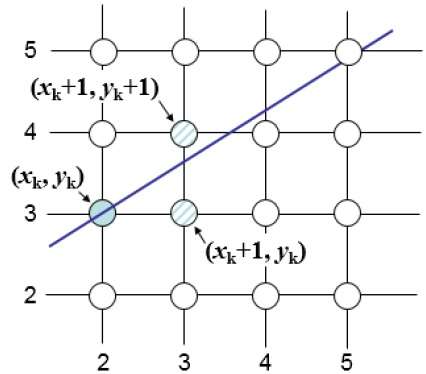
putpixel(Round(x), Round(y));

}

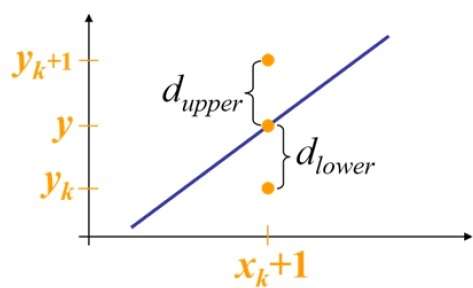
Bresenham’s Line Generation

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 2,32,3 you need to choose between 3,33,3 and 3,43,4. You would like the point that is closer to the original line.



At sample position Xk+1,Xk+1, the vertical separations from the mathematical line are labelled as dupperdupper and dlowerdlower.



From the above illustration, the y coordinate on the mathematical line at xk+1xk+1 is −

Y = m$Xk$+1$Xk$+1 + b

So, dupperdupper and dlowerdlower are given as follows −

dlower=y−ykdlower=y−yk

=m(Xk+1)+b−Yk=m(Xk+1)+b−Yk

and

dupper=(yk+1)−ydupper=(yk+1)−y

=Yk+1−m(Xk+1)−b=Yk+1−m(Xk+1)−b

You can use these to make a simple decision about which pixel is closer to the mathematical line. This simple decision is based on the difference between the two pixel positions.

dlower−dupper=2m(xk+1)−2yk+2b−1dlower−dupper=2m(xk+1)−2yk+2b−1

Let us substitute *m* with *dy/dx* where *dx* and *dy* are the differences between the end-points.

dx(dlower−dupper)=dx(2dydx(xk+1)−2yk+2b−1)dx(dlower−dupper)=dx(2dydx(xk+1)−2yk+2b−1)

=2dy.xk−2dx.yk+2dy+2dx(2b−1)=2dy.xk−2dx.yk+2dy+2dx(2b−1)

=2dy.xk−2dx.yk+C=2dy.xk−2dx.yk+C

So, a decision parameter PkPk for the *k*th step along a line is given by −

pk=dx(dlower−dupper)pk=dx(dlower−dupper)

=2dy.xk−2dx.yk+C=2dy.xk−2dx.yk+C

The sign of the decision parameter PkPk is the same as that of dlower−dupperdlower−dupper.

If pkpk is negative, then choose the lower pixel, otherwise choose the upper pixel.

Remember, the coordinate changes occur along the x axis in unit steps, so you can do everything with integer calculations. At step k+1, the decision parameter is given as −

pk+1=2dy.xk+1−2dx.yk+1+Cpk+1=2dy.xk+1−2dx.yk+1+C

Subtracting pkpk from this we get −

pk+1−pk=2dy(xk+1−xk)−2dx(yk+1−yk)pk+1−pk=2dy(xk+1−xk)−2dx(yk+1−yk)

But, xk+1xk+1 is the same as (xk)+1(xk)+1. So −

pk+1=pk+2dy−2dx(yk+1−yk)pk+1=pk+2dy−2dx(yk+1−yk)

Where, Yk+1–YkYk+1–Yk is either 0 or 1 depending on the sign of PkPk.

The first decision parameter p0p0 is evaluated at (x0,y0)(x0,y0) is given as −

p0=2dy−dxp0=2dy−dx

Now, keeping in mind all the above points and calculations, here is the Bresenham algorithm for slope m < 1 −

**Step 1** − Input the two end-points of line, storing the left end-point in (x0,y0)(x0,y0).

**Step 2** − Plot the point (x0,y0)(x0,y0).

**Step 3** − Calculate the constants dx, dy, 2dy, and 2dy–2dx2dy–2dx and get the first value for the decision parameter as −

p0=2dy−dxp0=2dy−dx

**Step 4** − At each XkXk along the line, starting at k = 0, perform the following test −

If pkpk < 0, the next point to plot is (xk+1,yk)(xk+1,yk) and

pk+1=pk+2dypk+1=pk+2dy

Otherwise,

(xk,yk+1)(xk,yk+1)

pk+1=pk+2dy−2dxpk+1=pk+2dy−2dx

**Step 5** − Repeat step 4 dx–1dx–1 times.

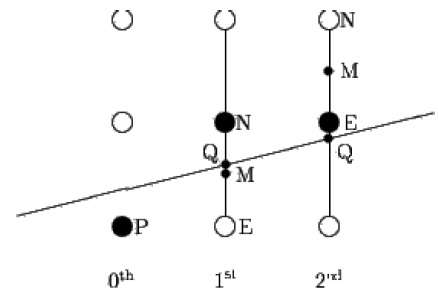
For m > 1, find out whether you need to increment x while incrementing y each time.

After solving, the equation for decision parameter PkPk will be very similar, just the x and y in the equation gets interchanged.

Mid-Point Algorithm

Mid-point algorithm is due to Bresenham which was modified by Pitteway and Van Aken. Assume that you have already put the point P at x,yx,y coordinate and the slope of the line is 0 ≤ k ≤ 1 as shown in the following illustration.

Now you need to decide whether to put the next point at E or N. This can be chosen by identifying the intersection point Q closest to the point N or E. If the intersection point Q is closest to the point N then N is considered as the next point; otherwise E.



To determine that, first calculate the mid-point Mx+1,y+½x+1,y+½. If the intersection point Q of the line with the vertical line connecting E and N is below M, then take E as the next point; otherwise take N as the next point.

In order to check this, we need to consider the implicit equation −

Fx,yx,y = mx + b - y

For positive m at any given X,

* If y is on the line, then Fx,yx,y = 0
* If y is above the line, then Fx,yx,y < 0
* If y is below the line, then Fx,yx,y > 0

