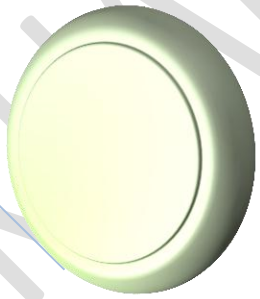


COMPUTER GRAPHICS

THIRD CLASS

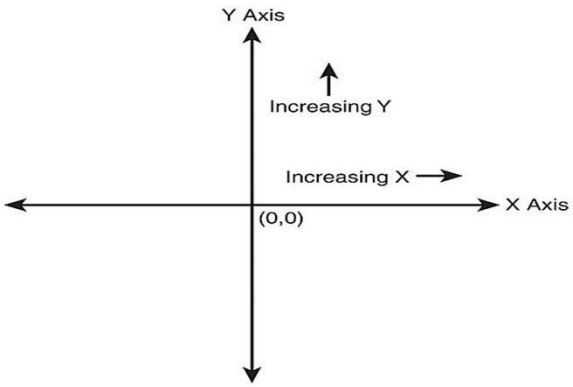
University of Diyala/ College of education for pure science / Computer science department

Dr. Adil I. KHALIL
18/10/2017

2. DRAWING ELEMANTARY FIGURES (PART 1)

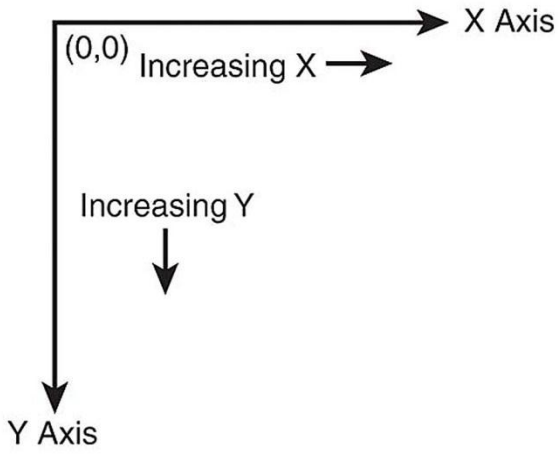
Plotting point

All graphical computing systems use some sort of *graphics coordinate system* to specify how points are arranged in a window or on the screen. Graphics coordinate systems typically spell out the *origin* (0, 0) of the system, as well as the axes and directions of increasing value for each of the axes.



The traditional **mathematical coordinate system** familiar to most of us is shown in this figure → .

The coordinate system of the screen is a Cartesian coordinate system. The origin (0,0) is at the top left of the screen. Positive x increases toward the right and positive y increases toward the bottom. **For you, what does this mean?**



All values in the windows coordinate system are positive integers. The **coordinate system of the screen is** shows in this figure →.

In order to draw a picture on a raster display, we must determine the corresponding points in the frame buffer that make up the picture. To perform this task we must write scan conversion point plotting algorithms.

To draw a point on the display screen, a point plotting procedure is required. We assume the availability of the command: **Putpixel (x , y , color).**

Horizontal lines

The screen coordinates of the points on a horizontal line are obtained by keeping the value of (y) constant and repeatedly incrementing the (x) value by one unit as in the following algorithm:

Input: Xstart, Xend, Yspecified.
Output: Horizontal line.

```
{
for x= Xstart to Xend
putpixel (x , yspecified , color);
}
```

If Xstart > Xend then replace Xend by Xstart and vice versa in for loop in the above algorithm.

H.W.: Write complete program in order to draw horizontal line?

Vertical lines

The screen coordinates of the points on a vertical line are obtained by keeping the value of (x) constant and repeatedly incrementing the (y) value by one unit as in the following algorithm

Input: Ystart,Yend, Xspecified.
Output: Vertical line.

```
{
for y= Ystart to Yend
putpixel (Xspecified , y , color);
}
```

Diagonal lines

To draw a diagonal line with a slope equal to (+1), we need only repeatedly increment by one unit both x and y values from the start to end pixels as shown in algorithm

Input: Xstart,Ystart, Xend, Yend.
i=0
Output: Diagonal line.

```
{
while (xstart + i) ≤ Xend )
{
putpixel (xstart + i, ystart + i, color);
i= i+1;
}
}
```

To draw a line with slop equals to (-1), replace (ystart+i) by (ystart-i) in algorithm above.

H.W.: Write complete program in order to draw diagonal line using algorithm above with slope (+1) and slope (-1)?

Line drawing algorithms

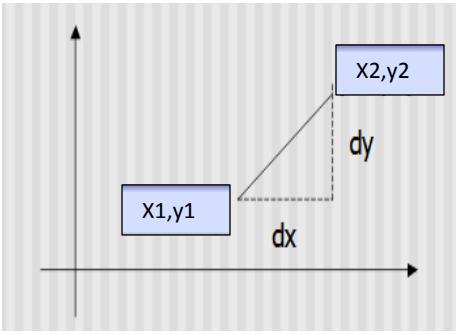
The choice of algorithm depends on:

1- The speed of line generation.
2- The appearance of the line.

Therefore, to understand these criteria better let's look at several different line generating algorithms.

1. **Direct method**

If the two endpoints used to specify a line are (X1,Y1) and (X2,Y2) ,then the equation of the line is used to describe the X , Y coordinates of all the pointes that lie between these two endpoints.



The equation of the straight line is :

$$y = mx + b$$

Where (*m*) is the slope of the line $m = \frac{\Delta y}{\Delta x} = \frac{y2-y1}{x2-x1}$

and (*b*) is a constant which represents y-intercept → $b = y - mx$

The Y values of the pointes of the line can be calculated using the above equation, by incrementing X values from X1 to X2 and substitute it in the line equation.

Note : The slope between any two points (X,Y) on the line and (X1,Y1) is the same as the slope between (X2,Y2) and (X1,X2).

$$m = \frac{y2 - y1}{x2 - x1} = \frac{y - y1}{x - x1}$$

H.W.: Write complete program in order to draw arbitrary line using direct method?

2. DDA (Digital Differential Analyzer) line algorithm

The DDA algorithm generates lines from their differential equations.

We calculate the length of the line in the X direction (number of pointes) by the equation :

ABS (X2-X1)

and calculate the length of the line in the Y direction (number of pointes) by the Equation :

ABS (Y2-Y1)

Where *ABS* is a function takes the positive of the arguments.

The Length estimates is equal to the larger of the magnitudes of the above two equations.
The increment steps ($X_{increment}$ and $\boldsymbol{Y_{increment}}$) are used to increment the X and Y coordinates for the next pointes to be plotted

$$Xincrement = \frac{x2 - x1}{larger\ length} \qquad Yincrement = \frac{y2 - y1}{larger\ length}$$

Algorithm DDA

Start

If ABS(X2-X1) > ABS (Y2-Y1) Then
Length=ABS (X2-X1)
Else
Length=ABS (Y2-Y1)

$X_{increment}$ *= (X2-X1) / Length*

$Y_{increment}$*= (Y2-Y1) / Length*

X=X1+ 0.5 × sign ($X_{increment}$*)*
Y=Y1+ 0.5 × sign ($\boldsymbol{y_{increment}}$*)*
For I=1 to Length
Begin
Plot (Int (X) , Int (Y))
X=X+ $X_{increment}$
Y=Y+ $\boldsymbol{Y_{increment}}$
End

Finish

Note: ***Sign*** *is a function returns (-1,0,+1) as it's argument is (<0,=0,>0).*
Using the Sign function makes the algorithm work in all quadrants.

Example: Consider the line from (0,0) to (5,5) Use DDA to rasterize the line.

Sol 1 :

X1=0 ; Y1=0 ; X2=5 ; Y2=5 ; length=5

dX=1 ; dY=1 ; X=0.5 ; Y=0.5

I	Plot	X	Y
		0.5	0.5
1	(0,0)	1.5	1.5
2	(1,1)	2.5	2.5
3	(2,2)	3.5	3.5
4	(3,3)	4.5	4.5
5	(4,4)	5.5	5.5

Note : the integer part of X and Y are used in plotting the line.

This would normally have the effect of truncating rather than rounding so we initialize the DDA with the value 0.5 in each of the fractional parts to achieve true rounding. One advantage of this arrangement is that it allows us to detect changes in X and Y and hence to avoid plotting the same point twice. The DDA algorithm is faster than the direct use of the line equation since it calculates points on the line without any floating point multiplication. However, a floating point addition is still needed in determining each successive point. Furthermore, cumulative error due to limited precision in the floating point representation may cause calculated points to drift away from their true position when the line relatively long.

Features of DDA

1- The algorithm is orientation dependent
2- The end point accuracy deteriorates
3- The algorithm suffer from the fact that it must be performed using floating point arithmetic

تعمل هذه الخوارزمية بشكل جيد عندما يكون $dx >= dy$ ولكنها بطيئة جداً على الحاسوب حيث أنها تتطلب عمليات على أرقام ذات فاصلة عائمة .أما إذا كان $dx < dy$ فإن المستقيم يصبح خشناً جداً .

H.W. : Consider the line from (0,0) to (-8,-4) . Evaluate the DDA algorithm.

H.W.: Could you find another way to calculate DDA algorithm?