Concept of Pixel and Resolution:

1.2.1 Pixel

A *pixel* may be defined as the smallest size object or color spot that can be displayed and addressed on a monitor. Any image that is displayed on the monitor is made up of thousands of such small pixels (also known as picture elements). The closely spaced pixels divide the image area into a compact and uniform two-dimensional grid of pixel lines and columns. Each pixel has a particular color and brightness value. Though the size of a pixel depends mostly on the size of the electron beam within the CRT, they are too fine and close to each other to be perceptible by the human eye. The finer the pixels the more the number of pixels displayable on a monitor-screen. However it should be remembered that the number of pixels in an image is fixed by the program that creates the image and not by the hardware that displays it.

1.2.2 Resolution

There are two distinctly different terms, which are often confused. One is *Image Resolution* and the other is *Screen Resolution*. Strictly speaking image resolution refers to the pixel spacing, i.e. the distance from one pixel to the next pixel. A typical PC monitor displays screen images with a resolution somewhere between 25 pixels per inch and 80 pixels per inch (ppi). In other words resolution of an image refers to the total number of pixels along the entire height and width of the image. For example a full-screen image with resolution 800×600 means that there are 800 columns of pixels, each column comprising 600 pixels, i.e. a total of $800 \times 600 = 480000$ pixels in the image area.

Scan convertion / Rasterization and introduction to line drawing:

Scan convertion or rasterization means the process to digitize a picture definition into a set of pixel intensity values for storage in frame buffer. This is a major task of the display processor.

In the previous chapters we have seen that in order to draw the primitive objects, one has to first *scan convert* the objects. This refers to the operation of finding out the location of pixels to be intensified and then setting the values of corresponding bits, in the graphics memory, to the desired intensity code. Each pixel on the display surface has a finite size depending on the screen resolution and hence a pixel cannot represent a single mathematical point. However, we consider each pixel as a unit square area identified by the coordinate of its lower left corner, the origin of the reference coordinate system being located at the lower left corner of the display surface. Thus each pixel is accessed by a non-negative integer coordinate pair (x, y). The x values start at the origin and increase from left to right along a scan line and the y values (i.e. the scan line numbers) start at bottom and increase upwards.

Line drawing is accomplished by calculating intermediate point coordinates along the line path between two given end points. Since screen pixels are referred with integer values, plotted positions may only approximate the calculated coordinates – i.e. pixels which are intensified are those which lie very close to the line path if not exactly on the. line path which is in the case of perfectly horizontal, vertical or 45° lines only. Standard algorithms are available to determine which pixels provide the best approximation to the desired line. Still, screen resolution is a big factor towards improving the approximation. In a high resolution system the adjacent pixels are so closely spaced that the approximated line-pixels lie very close to the actual line path and hence the plotted lines appear to be much smoother — almost like straight lines drawn on paper. In a low resolution system, the same approximation technique causes lines to be displayed with a "stairstep appearance" – not smooth (see Fig. 2.2 a).

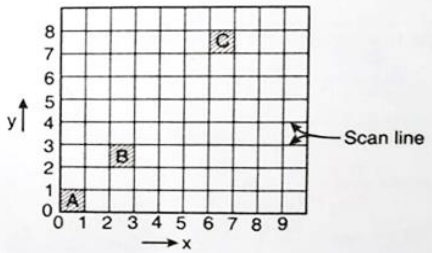


Fig. 2.1 *Array of square pixels on the display surface. Coordinate of pixel A: 0, 0; B: 2, 2; C: 6,7. A coordinate position (6.6,7.25) is represented by C, whereas (2.3,2.5) is represented by B. For plotting on the screen the calculated pixel coordinates are rounded off to nearest integers*

Abhishek Dey
Assistant Professor
Department of Computer Science
Bethune College, Kolkata

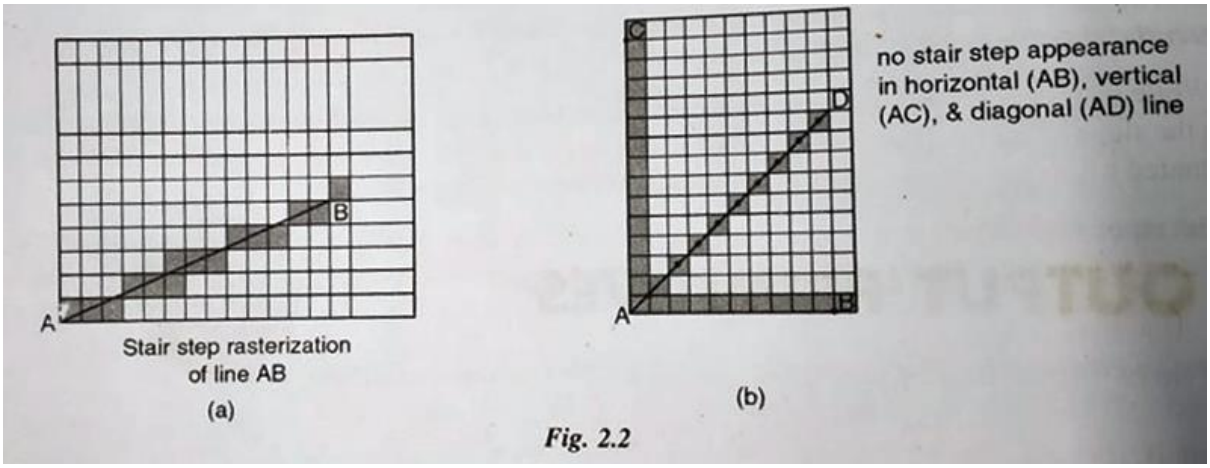


Fig. 2.2

Digital Differential Analyzer (DDA):

Assume that a straight line is to rasterized between two given end points (x_S, y_S) and (x_E, y_E).

Consider the slope intercept equation of straight line:

$$y = mx + c \quad \text{---------- (i)}$$

where, m = slope = $(y_E - y_S) / (x_E - x_S)$ and c is the y-intercept

For any two consecutive points (x_i, y_i) and (x_{i+1}, y_{i+1}) lying on this straight line, equation (i) will be satisfied.

Hence, $m = (y_{i+1} - y_i) / (x_{i+1} - x_i) = \Delta y / \Delta x$ ---------- (ii)

Thus, for any x interval Δx, we can compute corresponding y-interval Δy along the straight line.

We will initialize (x_i, y_i) with (x_S, y_S) and then we have to generate the subsequent points until we reach (x_E, y_E). Here, the main problem is to determine the increment factor of x and y [i.e. Δx and Δy] in each step until we reach (x_E, y_E). This can be done by checking absolute value of the line slope |m| as explained below:

Case 1:

If $|m| \leq 1$ then $\Delta y \leq \Delta x$, so we will sample the straight line at unit x-intervals.

Hence, $\Delta x = 1$ and $\Delta y = m$ [from equation (ii)]

Therefore, $\mathbf{x_{i+1} = x_i + 1}$ **and** $\mathbf{y_{i+1} = y_i + m}$

Case 2:

If $|m| > 1$ then $\Delta y > \Delta x$, so we will sample the straight line at unit x-intervals.

Hence, $\Delta y = 1$ and $\Delta x = 1/m$ [from equation (ii)]

Therefore, $\mathbf{y_{i+1} = y_i + 1}$ **and** $\mathbf{x_{i+1} = x_i + 1/m}$

As, m can be any real number (positive or negative) so the calculated values of y for case 1 and calculated values of x for case 2 must be rounded off to nearest integer values for plotting on the screen.

Abhishek Dey
Assistant Professor
Department of Computer Science
Bethune College, Kolkata

Pseudocode:

Procedure DDA (x_S, y_S, x_E, y_E) //Starting and ending points of straight line are passed as parameters

Begin

dx = (x_E – x_S), dy = (y_E – y_S)

If (Abs(dx) > Abs(dy))

Then

Steps = Abs(dx)

Else

Steps = Abs(dy)

EndIf

x_{INCR} = dx/Steps, y_{INCR} = dy/Steps // Calculation of x and y increment factors

x = x_S, y = y_S // Initialize 1st pixel

SetPixel(Round(x), Round(y)) // Plot 1st pixel

k=1

While (k ≤ Steps)

Begin

x = x + x_{INCR}

y = y + y_{INCR}

k = k+1

SetPixel(Round(x), Round(y))

EndWhile

End

Functions used:

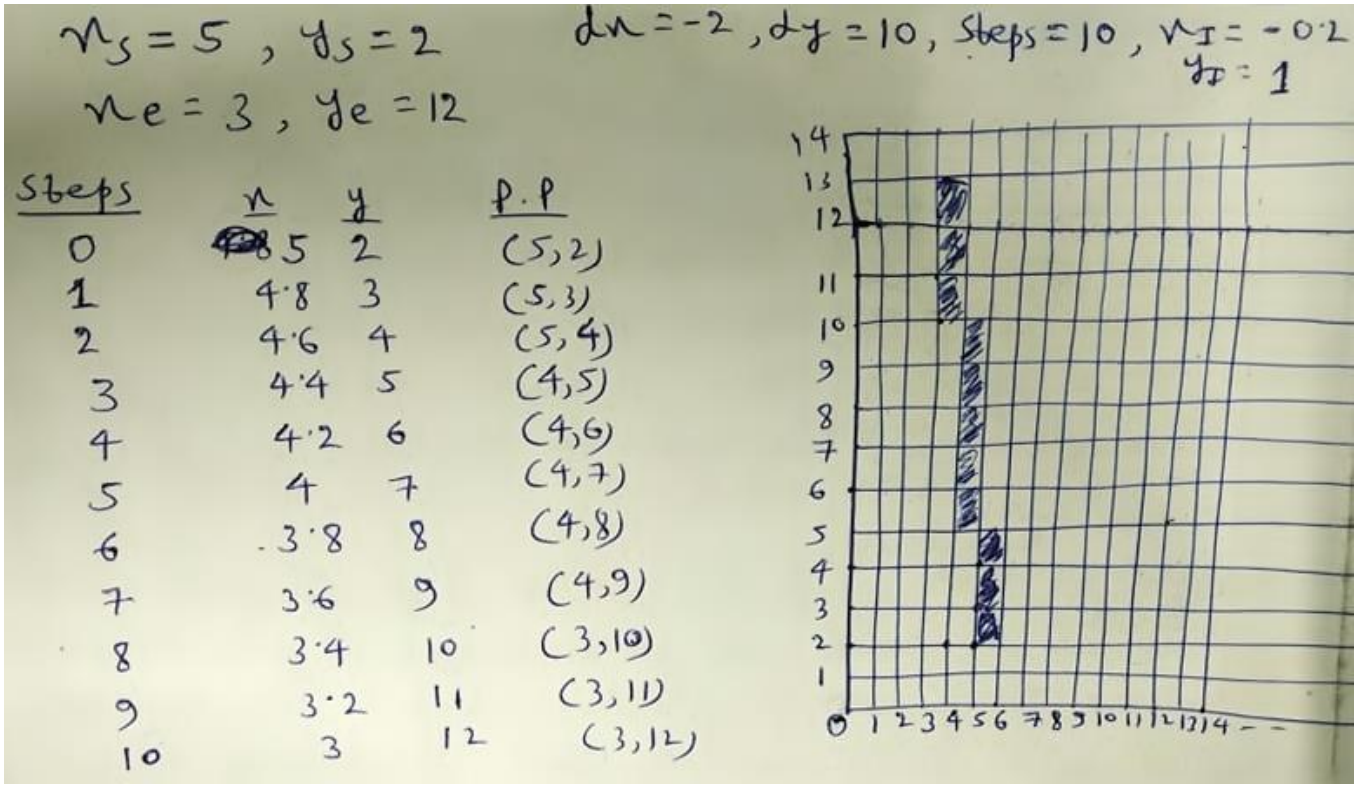
SetPixel(x, y) is used to plot the corresponding pixel defined by co-ordinate (x, y) on the screen.

Abs(arg) is used to determine the absolute value of the argument arg

Round(arg) is used to round off the argument arg to nearest integer value

Abhishek Dey
Assistant Professor
Department of Computer Science
Bethune College, Kolkata

An example:



Abhishek Dey
Assistant Professor
Department of Computer Science
Bethune College, Kolkata