BY: Mukunda Paudel
paudelmuku@gmail.com

COMPUTER GRAPHICS

Disclaimer

*This document is part of teaching materials for COMPUTER GRAPHICS under the Tribhuvan University syllabus for Bachelors of Science in Computer Science and Information Technology (**BSc. CSIT**).This document does not cover all aspect of learning COMPUTER GRAPHICS, nor are these be taken as primary source of information. As the core textbooks and reference books for learning the subject has already been specified and provided to the students, students are encouraged to learn from the original sources because this document cannot be used as a substitute for prescribed textbooks..*

*Various text books as well as freely available material from internet were consulted for preparing this document. Contents in This document are **copyrighted** to the instructor and authors of original texts where applicable.*

Chapter-2 Scan Conversion Algorithm

Output Primitives:

Initial phase of computer science→ computing and textual form printing

Later on → graphics evolved→ drawing of different objects like line, circle, ellipse, rectangle, spline curve etc.

Hence, these graphical components are called output primitives.

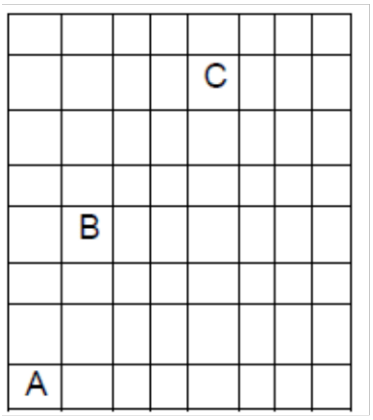
Output primitives are the geometric structure such as straight-line segments and polygon areas used to describe the shapes and color of the objects.

In case of the two-dimensional algorithm, we mostly deal with the line, circle, ellipse etc. as they are the basic output primitives.

Scan Converting a Point and a Straight Line

Point:

- Pixel is a unit square area identified by the coordinate of its lower left corner.
- Each pixel on the display surface has a finite size depending on the screen resolution & hence a pixel can't represent a single mathematical number.
- Origin of the reference coordinate system being located of the lower left corner of the display surface.
- The each pixel is accused by non-negative integer coordinate pair(x, y).
- The x values start at the origin &increase from left to right along a scan line & y values start at the bottom & increase upwards.



- In the above diagram the coordinate of pixel A:0,0 ,B:1,4 , C:4,7.C:4,7.
- A coding position (4. 2, 7. 2) is represented by C whereas (1.5, 4.2) is represented by B.
- In order to half a pixel on the screen we need to round off the coordinate to a nearest integer.

Line:

- Line drawing is accomplished by calculating intermediate point coordinates along the line path between two given end points.
- Screen pixel are referred with integer values, plotted positions may only approximate the calculate coordinates, what is pixel which are intensified are those which lie very close to the line path.
- In a high resolution system the adjacent pixels are so closely spread that the approximated line pixels lie very close to actual line path and hence the plotted lines appear to be much smooth-almost like straight line drown on paper.
- In low resolution system the same approximation technique causes to display with stair step appearance that is not smooth.

Line Drawing Algorithm

The equation of a straight line is Y=mX + b

Where **m** representing slope of the line and **b** as the y intercept

.

Consider the two end points of a line segment are (x_1, y_1) & (x_2, y_2) then the straight line can be written as

$$y_1 = \frac{y_2 - y_1}{x_2 - x_1} x_1 + b \quad \ldots\ldots\ldots\ldots\ldots\ldots (1)$$

We can determine the volume for the slope 'm' & y intercept 'b' with the following calculation.

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

$$b = y_1 - mx_1$$

$$\Rightarrow y_1 = mx_1 + b$$

For any given x interval Δx along a line, we can compute the corresponding y interval Δy from the above equation

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

Similarly, we can obtained the x interval Δx corresponding to a specified Δy as

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

For a line with slope magnitude |m|<1,

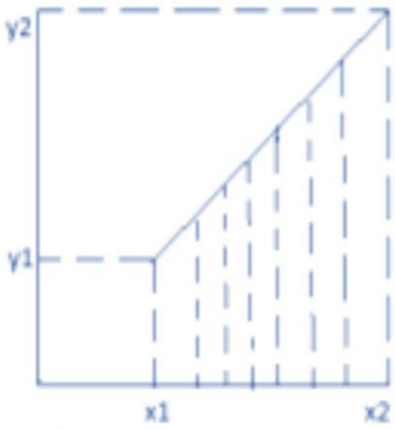
Δx can be set proportional to a small horizontal deflection voltage & the corresponding vertical deflection is then set proportional to Δy as calculate from the equation

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

For a line whose slopes have magnitudes |m|>1,

Δy can be set proportional to a small vertical deflection voltage with the corresponding horizontal deflection voltage set proportional to Δx calculate from the equation

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



For a line with m=1,

Then $\Delta x = \Delta y$ and vertical & horizontal deflection voltages are equal.

In each case a smooth line with slope 'm' is generated between specified end point.

DDA Algorithm (Digital Differential Analyzer)

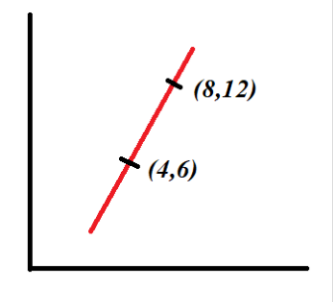
- DDA is a scan-conversion line algorithm base on either Δx or Δy.

The process of representing continuous graphics objects as a collection of discrete pixels is called ***scan conversion***. The graphics objects are continuous. The pixels used are discrete.

- We sample the line at unit interval in one direction (x if △x is greater than △y, otherwise in y direction) and determine the corresponding integer values nearest the line path for the other coordinate

Algorithm:

Consider the line having positive slope



The equation of the line is given,

$y = mx+b$ …….(i)

$m = (y2-y1)/(x2-x1)$ …….(ii)

For any interval $\triangle x$, corresponding interval is given by $\triangle y = m.\triangle x$.

Case I: If $|m| \leq 1$, we sample at unit x interval

i.e $\triangle x = 1$.

$$x_{k+1} = x_k+1$$

Then we compute each successive y-values, by setting $\triangle y = m$

$$y_{k+1} = y_k + m$$

The calculated y value must be rounded to the nearest integer

Case II: If $|m| > 1$, we sample at unit y-interval

i.e $\triangle y = 1$ and compute each successive x-values.

$$y_{k+1} = y_k + 1$$

Therefore, $1 = m.\triangle x$, and $\triangle x = 1/m$ (since, $m=\triangle y/\triangle x$ and $\triangle y=1$).

$$x_{k+1} = x_k + 1/m$$

The above equations are under the assumption that lines are to be processed from left endpoints (x_k, y_k) to right endpoints (x_{k+1}, y_{k+1}).

Advantages

- DDA algorithm is a faster method for calculating pixel position then the equation of a pixel position. Y = mx + b
- Simplex and fast method

Disadvantages

- Accumulation of round of error is successive addition of the floating point increments is used to find the pixel position but it take lot of time to compute the pixel position.
- Poor end point accuracy

Problem: 01 : Digitized the line with end points (0, 0) and (4, 5) using DDA.

Solution:
Given,
Starting Point: $P(x_1, y_1) = (0, 0)$
Ending Point: $Q(x_2, y_2) = (4, 5)$

Now Slope $m = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$ = (5-0) / (5-0) = (5/4) = 1.25 i.e. Slope |m|>1,

From DDA algorithm we have

$$Y_{k+1} = y_k + 1$$

$$x_{k+1} = x_k + (1/m)$$

Hence,

X	Y	X-Plot	Y-Plot	(X,Y)
0	0	0	0	(0,0)
0.8	1	1	1	(1,1)
1.6	2	2	2	(2,2)
2.4	3	2	3	(2,3)
3.2	4	3	4	(3,4)
4	5	4	5	(4,5)

Problem: 02: Consider a line from M (2, 1) to P (8, 3). Using DDA algorithm, rasterize this line.

Solution:

Given,

Starting Point: M $(x_1, y_1) = (2, 1)$
Ending Point: P $(x_2, y_2) = (8, 3)$

Now,

Slope m= $= \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$ = (3-1) / (8-2) = (1/3) = 0.333 i.e. slope |m|<1

From DDA algorithm we have

$$X_{k+1} = X_k + 1$$
$$Y_{k+1} = Y_k + m$$

Hence,

X	Y	X-Plot	Y-Plot	(X,Y)
2	1	2	1	(2,1)
3 (2+1)	1.33 (1+0.33)	3	1	(3,1)
4 (3+1)	1.66 (1.33+0.33)	4	2	(4,2)
5	1.993	5	2	(5,2)
6	2,326	6	2	(6,2)
7	2.659	7	3	(7,3)
8	2.999	8	3	(8,3)

Practice:

Q1. Consider a line from A (0, 0) to B (5, 5). Using DDA algorithm, rasterize this line.

Q2. Digitized the line with end points M (3, 7) and N (8, 3) using DDA.

Bresenham's Line Drawing Algorithm

Bresenham's Line Drawing Algorithm

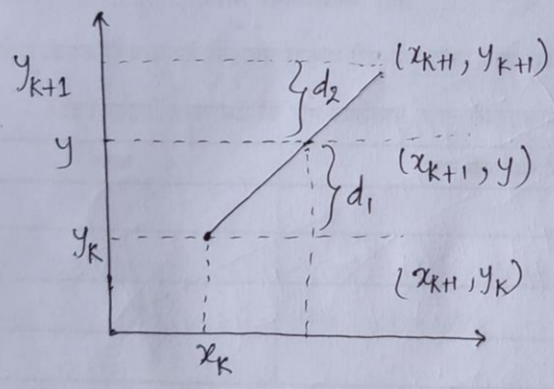
For +ve slope $|m| < 1$

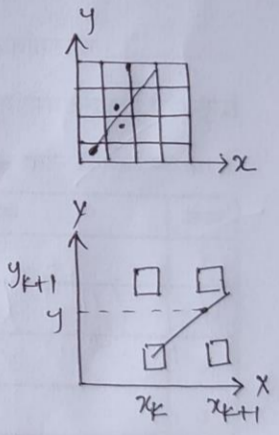
objective

To decide which of two possible pixel position is closer to the line path at each sample step.

If we consider the +ve slope < 1 then pixel position along line path are determined by Sampling x-interval.

Let us start from left end point (x_0, y_0) of a given line we step to each successive column (x position) & plot the pixel whose scan-line y-value is closest to the line path.





Let, (x_k, y_k) be the pixel position of starting point of line then next pixel to be plotted is either (x_{k+1}, y_k) or (x_{k+1}, y_{k+1}) [i.e. (x_{k+1}, y_k) or (x_{k+1}, y_k+1)]

At Sampling position x_{k+1} we label ~~vertical~~ vertical pixel separation from mathematical line path as d_1 and d_2.

The y-coordinate on the mathematical line at pixel column position x_{k+1} is calculated as:

$y = mx+b \Rightarrow m(x_k+1)+b$ — ①

From figure,

$d_1 = y - y_k$ and $d_2 = (y_k+1) - y$

difference between these two seperation is d_1-d_2, then

$d_1-d_2 = (y-y_k)-[(y_k+1)-y] \Rightarrow y-y_k-y_k-1+y$

$\Rightarrow 2y-2y_k-1$

from ①

$d_1-d_2 = 2[m(x_k+1)+b]-2y_k-1$

$\Rightarrow 2m(x_k+1)+2b-2y_k-1$ — ⑪

if $d_1-d_2 < 0$ i.e distance d_1 is small ($d_1 < d_2$) so pixel (x_{k+1}, y_k) is plotted $\Rightarrow (x_k+1, y_k)$

if $d_1-d_2 > 0$ i.e distance d_2 is small ($d_1 > d_2$) so pixel (x_{k+1}, y_{k+1}) is plotted.

$\Rightarrow (x_k+1, y_k+1)$

In ⑪ expression, slope (m) is present and 'm' gives float value so we need to remove it.

Put $m = \frac{\Delta y}{\Delta x}$ and multiply both side by Δx

$\Delta x(d_1-d_2) = 2.\Delta x.\frac{\Delta y}{\Delta x}(x_k+1)+2b\Delta x-2y_k\Delta x-\Delta x$

$\Rightarrow 2\Delta y x_k+2\Delta y+2b\Delta x-2y_k\Delta x-\Delta x$

from this expression we can decide the nearest distance the pixel. Hence consider the decision parameter as P_k for the k^{th} step/position.

Therefore, $P_k = \Delta x(d_1-d_2) = 2\Delta y x_k+2\Delta y+2b\Delta x-2y_k\Delta x-\Delta x$

Hence,

$$\boxed{P_K = 2\Delta y x_K - 2\Delta x y_K + C}$$

Where, $C = 2\Delta y + 2b\Delta x - \Delta x$ is constant in above expression which doesn't affect the decision.

for the K^{th} position P_K is decision parameter and it changes towards every position.

Now, To decide the next nearest pixel to be plotted, we have to find next P_K i.e. P_{K+1}

$$\boxed{P_{K+1} = 2\Delta y x_{K+1} - 2\Delta x y_{K+1}} \text{ or, } \boxed{P_{next} = 2\Delta y x_{next} - 2\Delta x y_{next}}$$

To find how decision parameter changes,

$$P_{K+1} - P_K = 2\Delta y x_{K+1} - 2\Delta x y_{K+1} - [2\Delta y x_K - 2\Delta x y_K]$$

$$= 2\Delta y x_{K+1} - 2\Delta x y_{K+1} - 2\Delta y x_K + 2\Delta x y_K$$

Case-I if $P_{K+1} - P_K < 0$, $\Rightarrow (x_{K+1}, y_K)$ i.e. $P_K < 0$

case-II if $P_{K+1} - P_K > 0$, $\Rightarrow (x_{K+1}, y_{K+1})$ i.e $P_K > 0$

Therefore,

$$P_{K+1} - P_K = 2\Delta y(x_K + 1) - 2\Delta x y_K - 2\Delta y x_K + 2\Delta x y_K$$

$$\Rightarrow 2\Delta y x_K + 2\Delta y - 2\Delta y x_K$$

$$\boxed{P_{K+1} = P_K + 2\Delta y}$$

Again, case-II

if $P_{k+1} - P_k \geq 0$ i.e. $P_k > 0 \Rightarrow$

$$P_{k+1} - P_k = [2\Delta y x_{k+1} - 2\Delta x y_{k+1}] - [2\Delta y x_k - 2\Delta x y_k]$$

$$\Rightarrow 2\Delta y (x_k + 1) - 2\Delta x (y_k + 1) - 2\Delta y x_k + 2\Delta x y_k$$

$$\Rightarrow 2\Delta y x_k + 2\Delta y - 2\Delta x y_k - 2\Delta x - 2\Delta y x_k + 2\Delta x y_k$$

$$P_{k+1} - P_k \Rightarrow 2\Delta y - 2\Delta x$$

$$\boxed{P_{k+1} = P_k + 2\Delta y - 2\Delta x}$$

Now, calculate the decision parameter for initial condition

we know, for k^{th} position

$$P_k = 2\Delta y x_k - 2\Delta x y_k + 2\Delta y + 2\Delta x b - \Delta x$$

let, P_0 is the initial decision parameter,

$$P_0 = 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y - 2\Delta x b - \Delta x$$

$$\Rightarrow 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y + 2\Delta x \left[y_0 - \frac{\Delta y}{\Delta x} \cdot x_0 \right] - \Delta x$$

$$\left[\because y = mx + b \right.$$
$$y_0 = m x_0 + b$$
$$b = y_0 - m x_0$$
$$\left. b = y_0 - \frac{\Delta y}{\Delta x} * x_0 \right]$$

$$\Rightarrow 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y + 2\Delta x y_0 - 2\Delta y x_0 - \Delta x$$

$$\boxed{P_0 = 2\Delta y - \Delta x}$$

Which is required initial decision parameter

#.

Summary of Bresenham's Line Algorithm.

For $m \leq 1$

1) $P_0 = 2\Delta y - \Delta x$

$P_K = 2\Delta y x_K - 2\Delta x y_K + c$

$c = 2m + 2b - 1$

2) if $P_K < 0$

$y_{K+1} = y_K \Rightarrow$ Plotting pixel $(x_K + 1, y_K)$

$x_{K+1} = x_K + 1$

and $P_{K+1} = P_K + 2\Delta y$

3) if $P_K \geq 0$

$x_{K+1} = x_K + 1$

$y_{K+1} = y_K + 1 \Rightarrow$ plotting pixel $(x_K + 1, y_K + 1)$

and $P_{K+1} = P_K + 2\Delta y - 2\Delta x$

4) $P_0 = 2\Delta y - \Delta x$