UNIT-1:

chapter 2 Output Primitives

* Output Primitives

To describe structure of basic object is reffered to as output primitive. Each output primitive is specified with input coordinate, data and other information about the way that the object is to be displayed.
A picture is completely specified by the set of intensities for the pixel position in the display.
Digital devices display a straight line segment by plotting the discrete points between the two end points.

Discrete coordinate position along line the path are calculated from the equations of the line for a raster video display the line color (intensity) is then loaded into frames buffer at the corresponding pixel coordinates. Reading from the frame buffer the video controller that plots the screen pixel positions are referenced according to scanline (rows) number and columns numbers.

Scanlines are numbered consecutively from 0 starting at the top of screen and pixel columns are numbered from 0 left to right across each Scanline

Scan line no (y) ↓ height of Screen	(0,0)					5
0	0,0	1,0	2,0	3,0	4,0	5,0
1	0,1	1,1	2,1	3,1	4,1	5,1
2	0,2	1,2	2,2	3,2	4,2	5,2
3	0,3	1,3	2,3	3,3	4,3	5,3
4	0,4	1,4	2,4	3,4	4,4	5,4
5	0,5	1,5	2,5	3,5	4,5	5,5

(x) → width of screen

resolution = width * height = 6×6

classmate

Date
Page

To load an intensity value into buffer at a position corresponding to column 'x' along scanline 'y' setpixel(x, y)

To retrieve the current frame buffer intensity setting for a specified location we use a low level function getpixel(x, y).

Point Ploting:

It is accomplished by converting a single coordinate position given by a application program into appropriate operation for output device with a CRV monitor for example the electron beam is turned on to illuminate the screen phosper at the selected location.

Line Drawing:

It is accomplished by calculating intermediate positions along the line path between two specified end point positions. An output device is then directed to fill up the positions both end points.

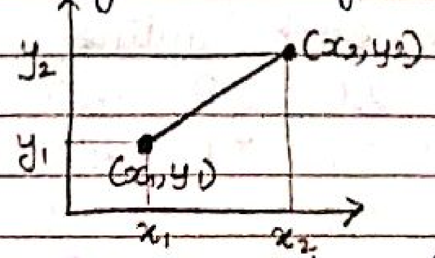
Line Drawing Algorithms:

The certain slope intercepts for straight line is

$y = mx + b$ where m = slope of line

b = y intercept

Given that two end points of line segment are specified at positions x_1, y_1 and x_2, y_2 as in figure below.



we can determine the values for slope 'm' and y intercept 'b' with the following calculations.

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

$$b = y - m \cdot x$$

For any given x interval Δx along the line, we can compute the corresponding y interval

$$\Delta y = m \cdot \Delta x \quad \text{— ①}$$

we can obtain the x interval Δx corresponding to specified Δy

$$\Delta x = \frac{\Delta y}{m} \quad \text{— ②}$$

for a lines with slope magnitude $|m| < 1$.
Δx can be set proportional to a small horizontal deflection voltage and a corresponding vertical deflection is then set proportional to Δy as calculated from eqn ①

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

For lines whose slope magnitude > 1 ($|m| > 1$). Δy can be set proportional to a small vertical deflection voltage with the corresponding horizontal deflection voltage set proportional to Δx as calculated from eqn ②

i.e: $$\Delta x = \frac{\Delta y}{m}$$

For lines with $m = 1$, then $\Delta x = \Delta y$ and the horizontal and vertical deflection voltages are equal.

1. Consider three different classical systems with resolution of
1) 640 X 480
2) 1280 X 1024
3) 2560 X 2048
a) What size is a frame buffer (in bytes) for each of this systems to store 12 bits per pixel?

b) How much storage in byte is required for each system if 24 bits per pixel is to be stored?

Solutions:

1. For 640×480 resolution

a) Total number of pixels required for 640×480 pixel resolution $= 640 \times 480$ pixels.

Since 1 pixel can store 12 bits therefore the size of frame buffer in bits $= 640 \times 480 \times 12$ bits

and in bytes $= \frac{640 \times 480 \times 12}{8}$ bytes

$= 4,60,800$ bytes.

b) Total number of pixels required for 640×480 pixels $\Big\} = 640 \times 480$

Since 1 pixel can store 24 bits, therefore the size of frame buffer $= 640 \times 480 \times 24$ bits

and in bytes $= \frac{640 \times 480 \times 24}{8} = 921600$ bytes

2. For 1280×1024 resolution

a) Total number of pixels required for 1280×1024 resolution $= 1280 \times 1024$ pixels

Since 1 pixels store 12 bits

$\therefore$ The size of frame buffer in bits $= 1280 \times 1024 \times 12$ bit

And in bytes $= \frac{1280 \times 1024 \times 12}{8}$

$= 1966080$ bytes

b) Total number of pixels required for 1280×1024 $= 1280 \times 1024$

Since 1 pixel can store 24 bits, therefore the size of frame buffer $= 1280 \times 1024 \times 24$ bits and in bytes = $\frac{1280 \times 1024 \times 24}{8}$ bytes

$= 5242880$ bytes

3 For 2560×2048

a) Total number of pixels required for 2560×2048 pixel resolution $= 2560 \times 2048$ pixels.
Since 1 pixel can store 12 bits therefore the size of frame buffer in bits $= 2560 \times 2048 \times 12$ bits.
and in bytes $= \frac{2560 \times 2048 \times 12}{8}$ bytes

$= 7864320$ bytes

b) Total number of pixels required for 2560×2048 pixel resolution $= 2560 \times 2048$ pixels
Since 1 pixel can store 24 bits therefore the size of frame buffer in bits $= 2560 \times 2048 \times 24$ bits
and in bytes $= \frac{2560 \times 2048 \times 24}{8}$ bytes

$= 15728640$ bytes.

2 Consider three different classical systems with resolution of
1) 660×480
2) 1280×1024
3) 2550×2048
Determine the size of frame buffer that is needed for this system to store 12 bits per pixel.

Solution:

1. For 660×480
Total number of pixels required for 660×480 pixel resolution $= 660 \times 480$ pixels.
Since 1 pixel can store 12 bits therefore the size of frame buffer in bits $= 660 \times 480 \times 12$ bits

and in bytes $= \frac{660 \times 480 \times 12}{8}$ bytes

$= 475200$ bytes

For 2550×2048

Total number of pixels required for 2550×2048 pixel resolution $= 2550 \times 2048$ pixels.

Since 1 pixel can store 12 bits therefore the size of frame buffer in bits $= 2550 \times 2048 \times 12$ bits.

And in bytes $= \frac{2550 \times 2048 \times 12}{8}$ bytes

$= 7833600$

3. A larger printer is capable of printing 2 pages of size 10×11 inch per second at resolution of 500 pixels per inch. How many bits per second does such device require

Soln: The storage required per page in pixels

$= 10 \times 11$ inch/second

$= 10 \times 11 \times 500 \times 500$ pixels/second

$= 27500000$ pixels/second

Since the laser printer is capable of printing 2 pages per second so the bits per second (bps) required by the printer is

$= 2 \times 10 \times 11 \times 500 \times 500$ pixels/second

$= 55000000$ pixels/second

$= 5.5 \times 10^7$ pixels/second

$= 5.5 \times 10^7 t$ bits/second [Assume 1 pixel $= t$ bits]

Line Drawing Algorithms

1) Digital Differential Analyzer Algorithm (DDA)
The DDA Algorithm is a Scan Conversion line algorithm based on the calculation of either Δx or Δy. The line at unit intervals of 1 cordinate and determines the corresponding integer value nearest the line path for other cordinate. A line with a positive slope, if the slope is less than or equal to 1 ($m \le 1$) at unit x interval ($\Delta x = 1$) and computes each successive y value as

$$y_{k+1} = y_k + m \quad \text{—— (1)}$$

subscript k takes integer value starting from 1, until last point and increases by 1 until final point is reached. m can be any real value between 0 and 1 and calculated y value should be rounded to the nearest integer value.
For lines with positive slope greater than 1 we reverse slope of x any y ($\Delta y = 1$) and calculate each successive x value as:

$$x_{k+1} = x_k + \frac{1}{m} \quad \text{—— (2)}$$

Equations (1) and (2) are based on assumption that the lines are processed from left end point to right end point. If this is reversed so that starting point is on right, then either we have $\Delta x = -1$ and

$$y_k = y_k - m \quad \text{—— (3)}$$

OR when the slope is greater than 1 we have $\Delta y = -1$ and $x_{k+1} = x_k - \frac{1}{m}$ —— (4)

If the absolute value of slope is less than 1 and the start end point is at left we said that the $\Delta x = 1$ and calculate the y value with equation ①.

When the start end point is at right with same slope we set $\Delta x = -1$ and obtain y position with equation ③.

Similarly when absolute value of negative slope is greater than 1 we use $\Delta y = -1$ and the eqn ④ or we use $\Delta y = 1$ and eqn ②.

Advantages of DDA. Impt

1) It is the simplest algorithm and it does not require special skills for algorithm.
2) It is the faster method for calculating pixel position since we are not using $y = mx + c$ directly.
3) It eliminates multiplication in equation with the help of raster characteristics. Thus increments are applied in x or y direction to find pixel position along line path.

Disadvantages Impt

1) It is orientation dependent therefore endpoint accuracy is poor.
2) It uses floating point arithmetic equation therefore it is time consuming.

Algorithm: Impt

step 1: Read the line end points (x_1, y_1) and (x_2, y_2) such that they are not equal. (If equal then plot that pixel and exit) step2: Compute dx and dy where

$$dx = \Delta x = |x_2 - x_1|$$

$$dy = \Delta y = |y_2 - y_1|$$

step3: Approxiamate the length of the line that is

if (dx ≥ dy) then
  length = dx
else
  length = dy

step4: Select the raster unit ie:

$$dx = \frac{(x_2 - x_1)}{length}$$

$$dy = \frac{(y_2 - y_1)}{length}$$

This makes either dx or dy equal to 1 because the length is either the absolute value of $(x_2 - x_1)$ or $(y_2 - y_1)$. Therefore incremental value for either x or y is 1.

step5: Round the values by using

$$x = x_1 + 0.5 * sign(dx)$$

$$y = y_1 + 0.5 * sign(dy)$$

Here the sign function makes the algorithm work in all quadrants. It works in -1, 0, 1 depending on whether argument is less than 0, greater than 0 or equal to 0. The factor 0.5 makes it possible to round the value in the integer function rather than truncating them.

step 6: Plot the points (integer(x), integer(y))

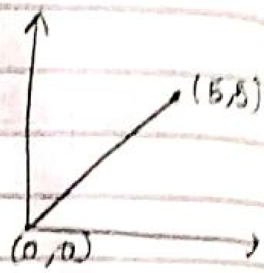
Assign i = 1
while(i <= length)
begin
  x = x + dx;
  y = y + dy

plot(int x, int y)
i = i+1;

end

Problems:

1) Draw a straight line segment joining the point (0,0) to (5,5) using DDA algorithm.



soln: step1: $(x_1, y_1) = (0,0)$
$(x_2, y_2) = (5,5)$
$\therefore x_1 = 0, y_1 = 0, x_2 = 5, y_2 = 5$

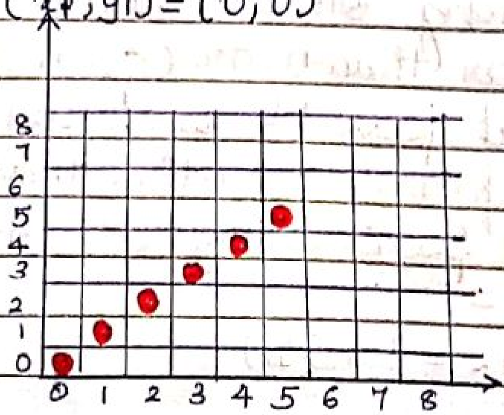
step2: Compute dx and dy
$dx = \Delta x = |x_2 - x_1| = |5-0| = 5$
$dy = \Delta y = |y_2 - y_1| = |5-0| = 5$

step3: Finding length.
Since dx = dy therefore
length = dx = 5

step4: selecting the raster unit
$dx = \frac{(x_2 - x_1)}{length} = \frac{5}{5} = 1$
$dy = \frac{(y_2 - y_1)}{length} = \frac{5}{5} = 1$

step5: Plot $(x_1, y_1) = (0,0)$

step6: Plot $(x_1, y_1) = (0,0)$



i = 1.

1) Is (i<=length)? $(1 \leq 5)$? Yes

$x = x + dx = 0 + 1 = 1$

$y = y + dy = 0 + 1 = 1$

plot $(x, y) = (1, 1)$

$i = i + 1 = 1 + 1 = 2$

Is $(2 \leq 5)$? YES

$x = x + dx = 1 + 1 = 2$

$y = y + dy = 1 + 1 = 2$

plot $(x, y) = (2, 2)$

$i = i + 1 = 2 + 1 = 3$

Is $(3 \leq 5)$? YES

$x = x + dx = 2 + 1 = 3$

$y = y + dy = 2 + 1 = 3$

plot $(3, 3)$

$i = i + 1 = 4$

Is $(4 \leq 5)$? YES

$x = x + dx = 3 + 1 = 4$

$y = y + dy = 3 + 1 = 4$

plot $(4, 4)$

$i = i + 1 = 5$

Is $(5 \leq 5)$? YES

$x = x + dx = 4 + 1 = 5$

$y = y + dy = 4 + 1 = 5$

plot $(5, 5)$

$i = i + 1 = 6$

Is $(6 \leq 5)$? $\boxed{\text{NO}}$

stop.

i	plot(x,y)	x	y
		0	0
	0,0		
		1	1
1			
		2	2
2			
	2,2		
		3	3
3			
	3,3		
		4	4
4			
	4,4		
		5	5
5			
	5,5		

Pseudocode of DDA Algorithm

```
1) Input x1, y1, x2, y2 //endpoints of the line
2). int dx = x2-x1, dy = y2-y1, step, i ;
    float xinc, yinc, x = x1, y = y1;
3) if ( abs(dx) >= abs (dy))
       step = abs (dx) ;
   else
       step = abs (dy) ;
4) xinc = dx/ float(step);
   yinc = dy/ float(step);
5) setpixel (x+0.5, y+0.5)
6) for(i=1; i<=step ; i++)
   {
       x = x + xinc ;
       y = y + yinc ;
       setpixel (x+0.5, y+0.5) ;
   }
```

Consider the line from (0,0) to (-6,-6). Use DDA Algorithm to rastarize the line.

Soln: <u>step1</u>: $(x_1, y_1) = (0,0)$

$(x_2, y_2) = (-6,-6)$

$x_1 = 0$. $y_1 = 0$ $x_2 = -6$ $y_2 = -6$.

<u>step2</u>:

$dx = \Delta x = |x_2 - x_1| = 6$

$dy = \Delta y = |y_2 - y_1| = 6$

<u>step3</u>:

since $dx = dy$

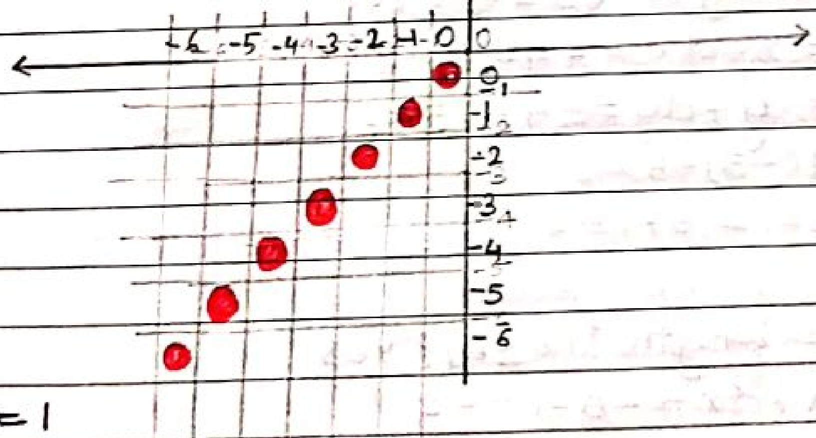
Length $= dx = 6$

<u>step4</u>:

$dx = \frac{(x_2 - x_1)}{\text{Length}} = \frac{-6}{6} = -1$

$dy = \frac{(y_2 - y_1)}{\text{Length}} = \frac{-6}{6} = -1$

step5: Plot $(x_1, y_1) = (0,0)$.



$i = 1$

i) Is ($i <=$ length)? ($1 \leq 6$)? Yes

$x = x + dx = 0 - 1 = -1$

$y = y + dy = 0 - 1 = -1$

plot $(-1, -1)$.

$i = i + 1 = 2$.

ii) Is ($i <=$ length)? ($2 \leq 6$)? Yes

Scanned by CamScanner

$x = x + dx = -1 - 1 = -2$

$y = y + dx = -1 - 1 = -2$

plot(x, y) = plot$(-2, -2)$

$i = i + 1 = 2 + 1 = 3$

iii) Is$(i <= length)$? $(3 \leq 6)$? YES

$x = x + dx = -2 - 1 = -3$

$y = y + dy = -2 - 1 = -3$

plot$(-3, -3)$

$i = i + 1 = 3 + 1 = 4$

iv) Is $(i <= length)$? $(4 \leq 6)$? YES

$x = x + dx = -3 - 1 = -4$

$y = y + dy = -3 - 1 = -4$

plot$(-4, -4)$

$i = i + 1 = 4 + 1 = 5$.

v) Is $(i <= length)$? $(5 \leq 6)$? YES

$x = x + dx = -4 - 1 = -5$

$y = y + dy = -4 - 1 = -5$

plot$(-5, -5)$

$i = i + 1 = 5 + 1 = 6$

vi) Is $(i <= length)$? $(6 \leq 6)$? YES

$x = x + dx = -5 - 1 = -6$

$x = y + dy = -5 - 1 = -6$

plot$(-6, -6)$

$i = i + 1 = 6 + 1 = 7$

vii) Is $(i \leq length)$? $(7 \leq 6)$? NO

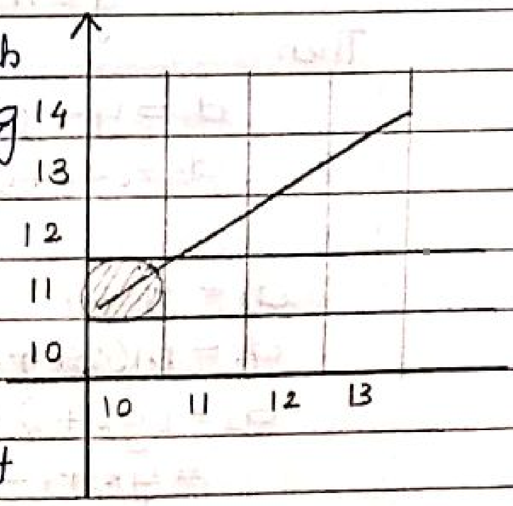
Stop.

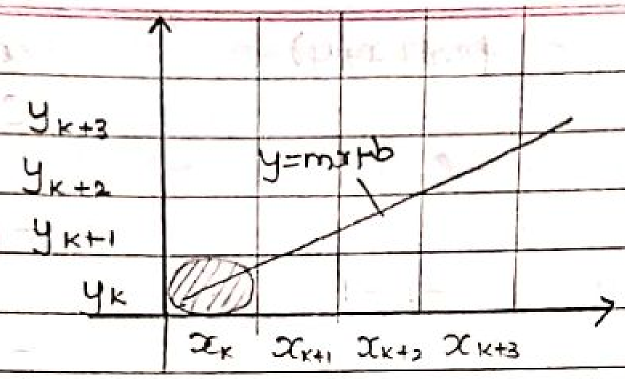
i	plot(x, y)	x	y
		0	0
	0, 0		
1		−1	−1
	−1, −1		
2		−2	−2
	−2, −2		
3		−3	−3
	−3, −3		
4		−4	−4
	−4, −4		
5		−5	−5
	−5, −5		
6		−6	−6
	−6, −6		

Bresenham's Line Algorithm

- An accurate and efficient line drawing algorithm that uses only incremental integer calculation.
- To understand the algorithm we consider lines with positive slope < 1

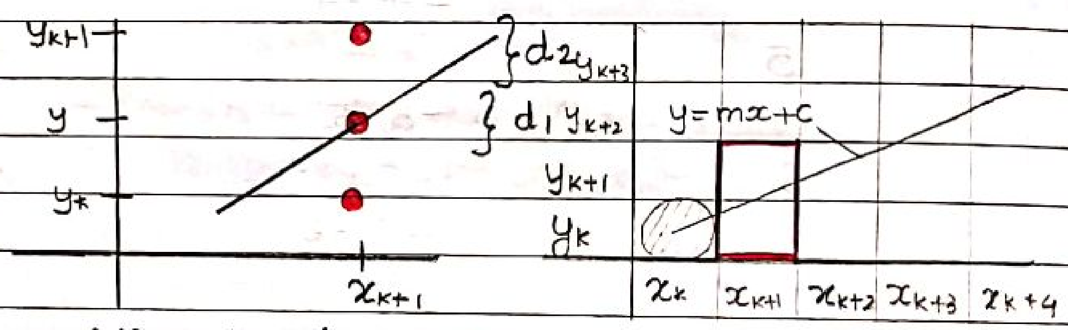
Pixel positions along a line path are then determined by sampling at unit x intervals.
Starting from the left endpoint (x_0, y_0) of a given line, we step to each successive column (x position) and plot the pixel whose scanline y value is closest to the line path.





To determine the pixel (x_k, y_k) to be displayed, we need to decide which pixel to plot in the column x_{k+1}

(x_{k+1}, y_k) or (x_k+1, y_k+1)



At sampling position (x_k+1), we label vertical pixel seperations from 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,

$$y = mx+b$$

$$y = m(x_k+1)+b \quad \text{——} \quad (1)$$

Then

$$d_1 = y - y_k$$

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

$$d_1 = y - y_k$$

$$d_1 = m(x_k+1)+b-y_k$$

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

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

$$d_2 = y_k+1-m(x_k+1)-b$$

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

$$= m(x_k+1)+b-y_k-y_k-1+m(x_k+1)+b$$

$$d_1-d_2 = 2m(x_k+1)-2y_k+2b-1 \quad \text{——} \quad (2)$$

A decision parameter P_k for the kth step in the line algorithm can be obtained by rearranging equation(2)

$$d_1-d_2 = 2m(x_k+1)-2y_k+2b-1 \quad \text{——} \quad (2)$$

After substituting, $m = \frac{\Delta y}{\Delta x}$

where Δx and Δy are the vertical and horizontal seperations of the endpoint positions and defining the decisions parameter, as,

$$P_k = \Delta x(d_1-d_2)$$

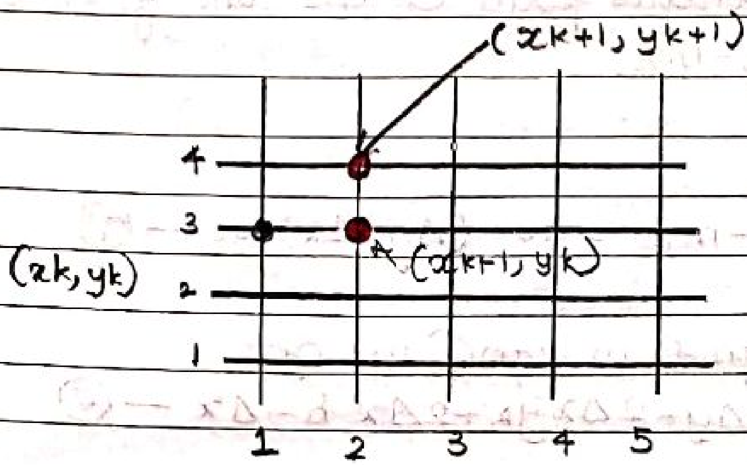
$$P_k = \Delta x(2m(x_k+1)-2y_k+2b-1)$$

$$= \Delta x\left(2 \cdot \frac{\Delta y}{\Delta x}(x_k+1)-2y_k+2b-1\right)$$

$$= 2\Delta y(x_k+1) + 2\Delta x \ldots$$

$$= \ldots \Delta y(x_k+1) - \Delta x(2y_k+2b \ldots$$

$$= \Delta y(x_k+1) \ldots b) + \Delta x \ldots$$



At steps $k+1$, P_k is evaluated as from equation(3)

$P_k = 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + C$ — (3)

$P_{k+1} = 2\Delta y \cdot x_{k+1} - 2\Delta x \cdot y_{k+1} + C$

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

$= 2\Delta y \cdot x_{k+1} - 2\Delta x \cdot y_{k+1} + \cancel{C} - 2\Delta y \cdot x_k + 2\Delta x \cdot y_k - \cancel{C}$

$= 2\Delta y (x_{k+1} - x_k) + 2\Delta x (y_{k+1} - y_k)$

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

$P_{k+1} - P_k = 2\Delta y (x_k + 1 - x_k) - 2\Delta x (y_{k+1} - y_k)$

$= 2\Delta y (1) - 2\Delta x (y_{k+1} - y_k)$

$P_{k+1} = P_k + 2\Delta y - 2\Delta x (y_{k+1} - y_k)$ — (4).

where $(y_{k+1} - y_k) = 0$ or 1 depending on P_k sign.

The first parameter P_0 is evaluated from eqn(5) at the starting pixel position (x_0, y_0) and with

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

$P_k = 2\Delta y (x_k + 1) - 2\Delta x y_k + 2\Delta x b - \Delta x$ — (5)

$P_k = 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + 2\Delta x b - \Delta x$ — (6)

we already know that

$y = mx + b$

Therefore for initial point on the line, $y_0 = mx_0 + b$

Therefore, $b = y_0 - mx_0$

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

therefore, $b = y_0 - mx_0 = y_0 - (\Delta y / \Delta x) x_0$ — (7).

Substituting eqn (7) in eqn (6) we get,

$P_k = 2\Delta y \cdot x_k + 2\Delta y - 2\Delta x y_k + 2\Delta x \cdot b - \Delta x$ — (6).

$$P_k = 2\Delta y \cdot x_k + 2\Delta y - 2\Delta x\, y_k + 2\Delta x\, [y_0 - (\Delta y/\Delta x) x_0] - \Delta x$$

Therefore initial decision parameter P_0 is,

$$P_0 = 2\Delta y \cdot x_0 + 2\Delta y - 2\Delta x \cdot y_0 + 2\Delta x\, [y_0 - (\Delta y/\Delta x) x_0] - \Delta x$$

$$= 2\Delta y \cdot x_0 + 2\Delta y - 2\Delta x\, y_0 + 2\Delta x\, y_0 - 2\Delta x (\Delta y/\Delta x) x_0 - \Delta x$$

$$= 2\Delta y\, x_0 + 2\Delta y - 2\Delta y\, x_0 - \Delta x$$

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

Bresenham's Line algorithm

step1: Input the two line endpoints and store the left endpoint in (x_0, y_0)

step2: Load (x_0, y_0) into the frame buffer, i.e plot the first point

step3: Calculate the constants $\Delta x, \Delta y, 2\Delta y$ and $2\Delta y - 2\Delta x$ and obtain the starting value for the decision parameter as $p_0 = 2\Delta y - \Delta x$

step4: At each x_k, along the line, starting at $k=0$, perform the following test.

If $p_k < 0$, the next point to plot is (x_{k+1}, y_k) and

$$p_{k+1} = p_k + 2\Delta y$$

Otherwise, the next point to plot is $(x_k \pm 1, y_k + 1)$ and

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x$$

step5: Repeat step4, Δx times.

Advantages of

1. Algorithm is fast
2. Uses only integer calculations

Disadvantages

1. It is meant only for basic line drawing.

1 Scan convert a line with endpoint (10,5) and (15,9)

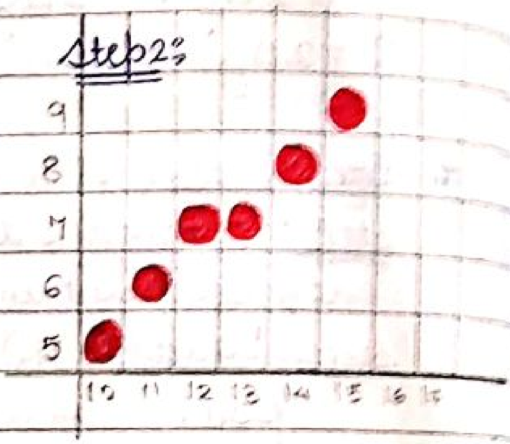
Soln: **step 1:**

$(x_0, y_0) = (10,5)$

$$m = \left|\frac{y_2 - y_1}{x_2 - x_1}\right| = \left|\frac{9-5}{15-10}\right| = \left|\frac{4}{5}\right| = 0.8$$

$m < 1$

step 2:



step 3:

$\Delta x = x_2 - x_1 = 5$

$\Delta y = y_2 - y_1 = 4$

$2\Delta y = 8$

$2\Delta y - 2\Delta x = 8 - 2\times 5$

$= -2$

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

$= 8 - 5$

$= 3$

P_0 is the initial decision parameter.

We plot the initial point $(x_0, y_0) = (10,5)$ and determine ~~step4:~~ the successive pixel position along the line path from the decision parameter as P_{k+1}. Hence

i) Hence for $k=0$, $P_k = P_0 = 3$, plot (x_{k+1}, y_{k+1})

$= (x_0+1, y_0+1)$

$= (11, 6)$

Is $(P_k < 0)$? $(P_0 < 0)$? $(3 < 0)$? No

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

$= 3 + 8 - 10$

$= 1$

ii) for $k=1$, $P_k = P_1 = 1$, plot (x_{k+1}, y_{k+1})

$= (12, 7)$

Is $(P_k < 0)$? $(P_1 < 0)$? $(1 < 0)$? No.

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

$= 1 + 8 - 10$

$= -1$

iii) $k=2, P_k = P_2 = -1$, plot (x_k+1, y_k+1)
$= (12+1, 7 ~~+0~~)$
$= (13, 7)$

Is $(P_k<0)$? $(-1<0)$? YES

$P_{k+1} = P_k + 2\Delta y$
$= -1+8$
$P_3 = 7$

iv) $k=3, P_k = P_3 = 7$

Is $(P_k<0)$? $(7<0)$? NO

plot $(x_k+1, y_k+1) = (13+1, 7+1)$
$= (14, 8)$

$P_{k+1} = P_k + 2\Delta y - 2\Delta x$
$= 7+8-10$
$P_4 = 5$

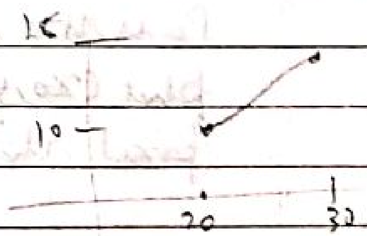
v) $k=4, P_k = P_4 = 5$

Is $(P_k<0)$? $(5<0)$? NO

plot $(x_k+1, y_k+1) = (14+1, 8+1)$
$= (15, 9)$

Stop.

k	P_k	plot (x,y) (10,5)
0	3	(11,6)
1	1	(12,7)
2	-1	(13,7)
3	7	(14,8)
4	5	(15,9)



2. Digitize the line with endpoints (20,10) and (30,18) using Bresenham's line drawing algorithm.

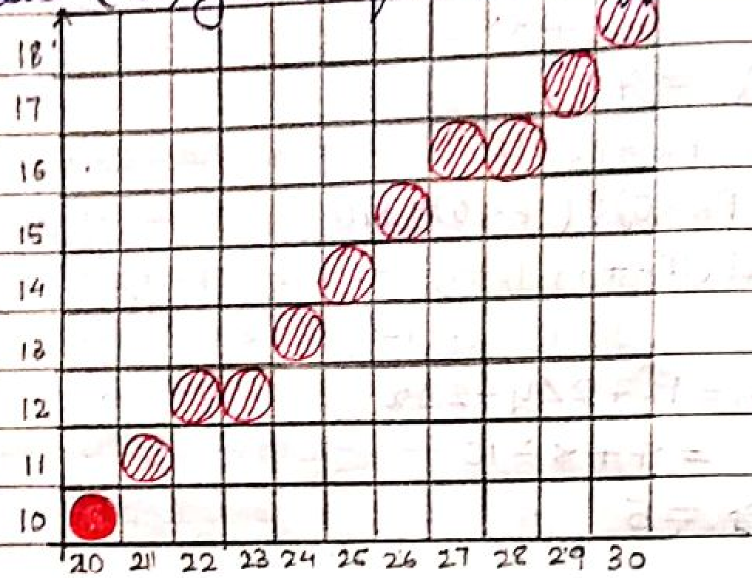
Solns step 1:

$(x_0, y_0) = (20, 10)$.

$$m = \left|\frac{y_2 - y_1}{x_2 - x_1}\right| = \left|\frac{18-10}{30-20}\right| = \left|\frac{8}{10}\right| = 0.8$$

$m < 1$.

step2: Load (x_0, y_0) in frame buffer. plot $(20,10)$



step3:

$\Delta x = 10$ (0 to 9 for k)

$\Delta y = 8$

$2\Delta y = 16$

$2\Delta y - 2\Delta x = 16 - 2\times 10$
$= 16 - 20$
$= -4$.

$P_0 = 2\Delta y - \Delta x$
$= 16 - 10$
$= 6$

P_0 is the initial decision parameter. Initially we plot (x_0, y_0) and further determine the successive pixel along the line.

step4:

i) $k=0$, $P_k = P_0 = 6$, plot (x_k+1, y_k+1)
$= (21, 11)$

Is $(P_k < 0)$? NO

$P_{k+1} = P_k + 2\Delta y - 2\Delta x$
$= 6 + 16 - 20$
$= 2$

2) $k=1, P_k = P_1 = 2$

Is $(P_k < 0)$? $(2<0)$? NO

$plot(x_k+1, y_k+1) = plot(22,12)$

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

$= 2+16-20$

$P_2 = -2$

3) $k=2, P_k = -2$

Is $(P_k<0)$? $(-2<0)$? YES

$plot(x_k+1, y_k) = (23,12)$

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

$= -2+16$

$P_3 = 14$

4) $k=3, P_k = P_3 = 14$

Is $(P_k<0)$? $(14<0)$? NO

$plot(x_k+1, y_k+1) = plot(24,13)$

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

$= 14+16-20$

$P_4 = 10$

5) $k=4, P_k = P_4 = 10$

Is $(P_k<0)$? $(10<0)$? NO.

$plot(x_k+1, y_k+1) = plot(25,14)$

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

$= 10+16-20$

$P_5 = 6$

6) $k=5, P_k = P_5 = 6$

Is $(P_k<0)$? $(6<0)$? NO.

$plot(x_k+1, y_k+1) = plot(26,15)$

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

$= 6-4$

$P_6 = 2$

7) $k=6, P_k = P_6 = 2$

Is $(P_k<0)$? $(2<0)$? NO.

$plot(x_k+1, y_k+1) = plot(27,16)$

$P_{k+1} = P_k + 2\Delta y - 2\Delta x$
$= 2 - 4$
$P_7 = -2$.

8) $k=7, P_k = P_7 = -2$
Is $(P_k < 0)$? $(-2<0)$? YES
plot (x_k+1, y_k) = plot$(28,16)$
$P_{k+1} = P_k + 2\Delta y$
$= -2+16$
$P_8 = 14$

9) $k=8, P_k = P_8 = 14$
Is $(P_k<0)$? $(14<0)$? NO
plot (x_k+1, y_k+1) = plot$(29,17)$
$P_{k+1} = P_k + 2\Delta y - 2\Delta x$
$= 14-4$
$= 10$

10) $k=9, P_k = P_9 = 10$
Is $(P_k<0)$? $(10<0)$? NO
plot (x_k+1, y_k+1) = plot $(30,18)$
Stop

K	P_k	plot(x, y)
-	-	(20,10)
0	6	(21,11)
1	2	(22,12)
2	-2	(23,12)
3	14	(24,13)
4	10	(25,14)
5	6	(26,15)
6	2	(27,16)
7	-2	(28,16)
8	14	(29,17)
9	10	(30,18)

Bresenham's hold normal for $m \le 1$.

classmate

Date

Page

3 Consider a line from (0,0) to (6,7). Use Bresenham's algorithm to rasterize the line.

soln: step 1:

$(x_0, y_0) = (0,0)$

$$m = \left|\frac{y_2 - y_1}{x_2 - x_1}\right| = \left|\frac{7-0}{6-0}\right| = \frac{7}{6} = 1.16$$

$m > 1$

step 2:

$\Delta x = x_2 - x_1 = 6$

$\Delta y = y_2 - y_1 = 7$

since $\Delta x < \Delta y$,

therefore exchange Δx and Δy

$\therefore \Delta x = 7$

$\Delta y = 6$

$2\Delta y = 12$

$2\Delta y - 2\Delta x = 12 - 2 \times 7$

$= 12 - 14$

$= -2.$

P_0 is the initial decision parameter. Initially we plot (x_0, y_0) and further determine successive pixel along the line.

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

$= 12 - 7$

$= 5$

step 4: Finding the points to be plotted.

1) $k=0$, $P_k = P_0 = 5$

Is $(P_k < 0)$? $(5 < 0)$? NO

plot$(x_k + 1, y_k + 1)$ = plot$(1,1)$

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

$= 5 - 2$

$P_1 = 3$

2) $k=1$, $P_k = P_1 = 3$

Is ($P_K<0$)? ($\cancel{3}<0$)? NO

plot (x_K+1, y_K+1) = plot (2, 2)

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

$= 3-2$

$P_2 = 1$

3) $k=2$, $P_K = P_2 = 1$

Is ($P_K<0$)? ($1<0$)? NO

plot (x_K+1, y_K+1) = plot (3, 3)

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

$= 1-2$

$P_3 = -1$

4) $k=3$, $P_K = P_3 = -1$

Is ($-1<0$)? YES

plot (x_K, y_K+1) = plot (3, 4)

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

$= -1+12$

$P_4 = 11$

5) $k=4$, $P_K = P_4 = 11$

Is ($11<0$)? NO

plot (x_K+1, y_K+1) = plot (4, 5)

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

$= 11-2$

$P_5 = 9$

6) $k=5$, $P_K = P_5 = 9$

Is ($9<0$)? NO

plot (x_K+1, y_K+1) = plot (5, 6)

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

$= 9-2$

$= 7$

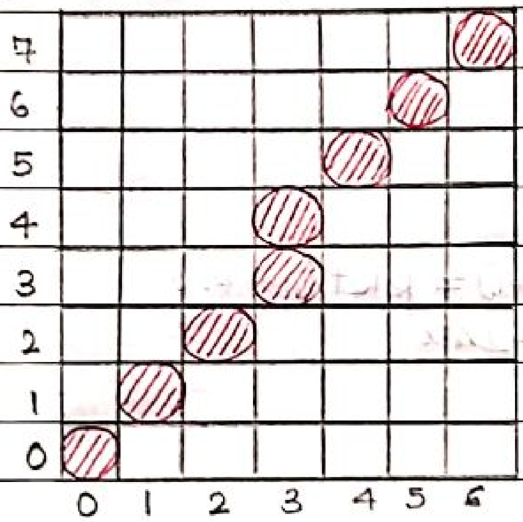
7) $k=6$, $P_K = P_6 = 7$

Is ($7<0$)? NO

plot (x_K+1, y_K+1) = plot (6, 7)

stop

k	P_k	plot (x,y)
–	–	(0,0)
0	5	(1,1)
1	3	(2,2)
2	1	(3,3)
3	-1	(3,4)
4	11	(4,5)
5	9	(5,6)
6	7	(6,7)



4. Digitize the line with end points (15,20) and (10,10) using Bresenham's line drawing algorithm.

Soln: step 1:

$(x_0, y_0) = (15, 20)$

$$m = \left|\frac{y_2 - y_1}{x_2 - x_1}\right| = \left|\frac{10-20}{10-15}\right| = \left|\frac{-10}{-5}\right| = 2$$

$m > 1$

step 2:

$\Delta y = |10 - 20| = 10$

$\Delta x = |10 - 15| = 5$

$\Delta x < \Delta y$

Since $\Delta x < \Delta y$ exchange Δx and Δy

$\Delta x = 10$

$\Delta y = 5$.

Scanned by CamScanner

$2\Delta y = 10$

$2\Delta y - 2\Delta x = 10 - 2\times 10$

$= 10 - 20$

$= -10.$

P_0 is the initial decision parameter. Initially we plot (x_0, y_0) and further determine successive points i.e: plot(10,10)

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

$= 10 - 10$

$= 0$

step4:

1) $K=0, P_K = P_0 = 0$

Is $(P_K < 0)$? NO.

plot (x_K+1, y_K+1) = plot (11,11)

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

$= 0 - 10$

$= -10.$

2) $K=1, P_K = P_1 = -10$

Is $(P_K < 0)$? $(-10 < 0)$? YES

plot (x_K, y_K+1) = plot (11, 12).

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

$= -10 + 10$

$P_2 = 0$

3) $K=2, P_K = P_2 = 0.$

Is $(P_K < 0)$? $(0 < 0)$? NO

plot (x_K+1, y_K+1) = plot (12,13)

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

$= 0 - 10$

$P_3 = -10.$

4) $K=3, P_K = P_3 = -10$

Is $(P_K < 0)$? $(-10 < 0)$? YES

plot (x_K, y_K+1) = plot (12,14)

$P_{k+1} = P_k + 2\Delta y$
$= -10 + 10$
$P_4 = 0.$

5) $k = 4$, $P_k = P_4 = 0$
Is $(0 < 0)$? NO
plot$(x_k + 1, y_k + 1)$ = plot $(13, 15)$
$P_{k+1} = P_k + 2\Delta y - 2\Delta x$
$= 0 - 10$
$P_5 = -10$

6) $k = 5$, $P_k = P_5 = -10$
Is $(-10 < 0)$? YES
plot$(x_k, y_k + 1)$ = plot$(13, 16)$
$P_{k+1} = P_k + 2\Delta y$
$= \cancel{0} -10 + 10$
$P_6 = \cancel{0}$

7) $k = 6$, $P_k = P_6 = 0$
Is $(0 < 0)$? NO
plot$(x_k + 1, y_k + 1)$ = plot$(14, 17)$
$P_{k+1} = P_k + 2\Delta y - 2\Delta x$
$= 0 - 10$
$P_7 = -10$

8) $k = 7$, $P_k = P_7 = -10$
Is $(-10 < 0)$? YES
plot$(x_k \cancel{+1}, y_k + 1)$ = plot$(14, 18)$
$P_{k+1} = P_k + 2\Delta y$
$= -10 + 10$
$P_8 = 0$

9) $k = 8$, $P_k = P_8 = 0$
Is $(0 < 0)$? NO
plot $(x_k + 1, y_k + 1)$ = plot$(15, 19)$
$P_{k+1} = P_k + 2\Delta y - 2\Delta x$
$= 0 - 10$
$P_9 = -10$

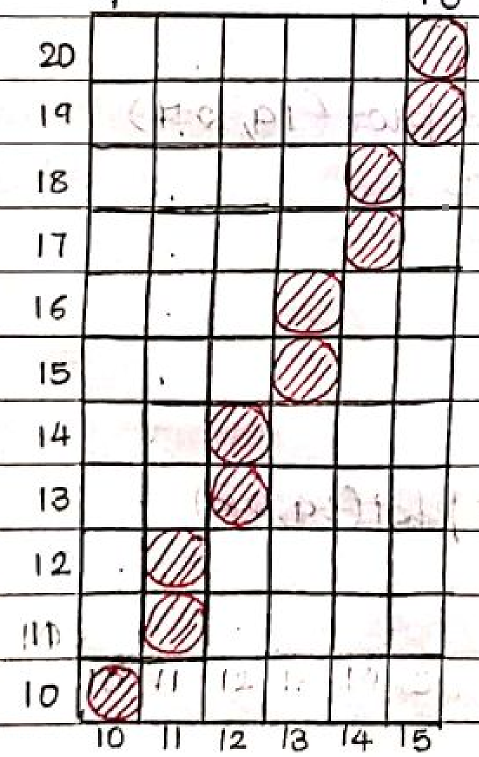
10) $K=9$, $P_K = P_9 = -10$

Is $(-10<0)$? YES

plot(x_K, y_K+1) = plot$(15,20)$

stop.

K	P_K	plot(x, y)
—	—	(10,10)
0	0	(11,11)
1	-10	(11,12)
2	0	(12,13)
3	-10	(12,14)
4	0	(13,15)
5	-10	(13,16)
6	0	(14,17)
7	-10	(14,18)
8	0	(15,19)
9	-10	(15,20)



Midpoint circle Algorithm

Algorithm:

step1: Input radius r and circle center (x_c, y_c) and obtain the first point of the circumference of a circle centered on the origin as $(x_0, y_0) = (0, r)$

step2: Calculate the initial value of the decision parameter as $P_0 = \frac{5}{4} - r$ OR $P_0 = 1 - r$

step3: At each x_k position, starting at $k=0$, perform the following test:

If $P_k < 0$, the next point along the circle centered on $(0,0)$ is (x_c+1, y_c) and $P_{k+1} =$

otherwise, the next point along the circle is (x_k+1, y_k-1) and $P_{k+1} = P_k + 2x_{k+1} + 1 - 2y_{k+1}$

Where, $2x_{k+1} = 2x_k + 2$ and $2y_{k+1} = 2y_k - 2$

step4: Determine symmetry points in the other seven octants.

step5: Move each calculated pixel position (x, y) onto the circular path centered on (x_c, y_c) and plot the coordinate values:

$$x = x + x_c$$
$$y = y + y_c$$

step6:

```
void circleMidpoint(int xCentre, int yCentre, int radius)
{
  int x=0, y=radius;
  int p=1-radius);
  circlePlotPoints(xCentre, yCentre, x, y);
  while(x<y)
  {
    x++;
    if(p<0)
        p+=2*x+1;
```

```
else
{
        y--;
        p+=2*(x-y)+1;
}
circlePlotPoints(xCenter, yCenter, x, y);
}
}
/* End of circle midpoint function */

void circlePlotPoints(int xCentre, int yCentre, int x, int y)
{
    setPixel(xCenter + x, yCenter + y);
    setPixel(xCenter - x, yCenter + y);
    setPixel(xCenter + x, yCenter - y);
    setPixel(xCenter - x, yCenter - y);
    setPixel(xCenter + y, yCenter + x);
    setPixel(xCenter - y, yCenter + x);
    setPixel(xCenter + y, yCenter - x);
    setPixel(xCenter - y, yCenter - x);
}
```

1 Calculate the points to draw a circle with radius 5 and centre at (0, 0)

Soln: step1: $(x_0, y_0) = (0, r) = (0, 5)$

step2: Calculate initial value

$P_0 = 1 - r = 1 - 5 = -4$

$2x_0 = 2 \times 0 = 0 \qquad 2y_0 = 2 \times 5 = 10$

step3:

$k = 0$; $P_k = 0$? Yes, $P_0 = -4 \therefore P_0 < 0$

$\therefore$ Next pixel point is $(x_k + 1, y_k) = (0+1, 5) = (1, 5)$

$\because P_k < 0$, $P_0 < 0$. $\qquad 2x_{k+1} = 2 \times 1 = 2 \quad 2y_{k+1} = 2 \times 5 = 10$

$\therefore P_0 < 0$

$P_{k+1} = P_k + 2x_{k+1} + 1 = -4 + 2 + 1 = -1$

$P_{0+1} = P_1 = -1$.

The initial value of decision parameter is P_0 which is calculated as $1 - r$. For the circle centered on the coordinate origin the initial point is

$(x_0 + x_c, y_0 + y_c) = (0+0, 5+0) = (0, 5)$

And the initial parameter-terms for calculating parameters are

Continue till $x < y$.

$2x_0 = 2 \times 0 = 0$

$2y_0 = 2 \times 5 = 10$

$k = 1$, $P_k = -1$

Is $(P_k < 0)$? $P_1 < 0$? Yes

$\therefore$ Next pixel point to plot is $(x_k + 1, y_k) = (1+1, 5) = (2, 5)$

$2x_{k+1} = 2 \times 2 = 4$

$2y_{k+1} = 2 \times 5 = 10$

$\because P_1 < 0$

$\therefore P_{k+1} = P_k + 2x_{k+1} + 1$

$P_2 = P_1 + 2x_{k+1} + 1$

$= -1 + 4 + 1$

$= 4$

$K=2,\ P_K = 4$

Is $(P_K < 0)$? $(4<0)$? No.

$\therefore$ Next pixel to plot is $(x_K+1, y_K-1) = (2+1, 5-1)$
$= (3,4)$

$2x_{K+1} = 2\times 3 = 6$

$2y_{K+1} = 2\times 4 = 8 \qquad \therefore P_K > 0.$

$P_{K+1} = P_K + 2x_{K+1} + 1 - 2y_{K+1}$
$= 4 + 6 + 1 - 8$
$= 3$

$k=3$

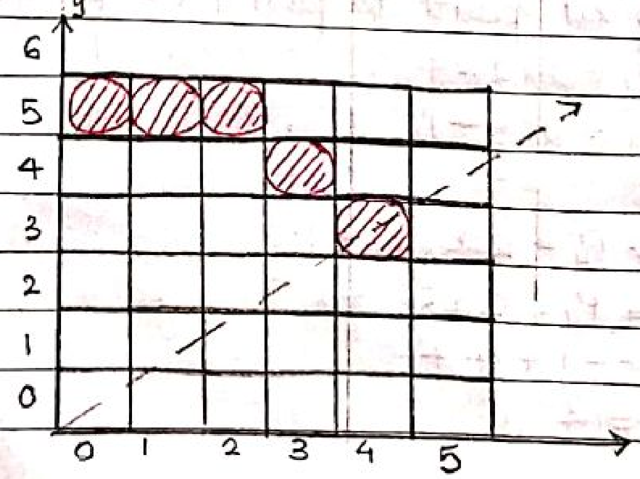
$P_K<0$? $P_3<0$? No

$\therefore$ Next pixel to plot is (x_K+1, y_K-1)
$= (3+1, 4-1)$
$= (4,3)$

stop, since x is found to be greater than y that is $4>3$ therefore we stop.

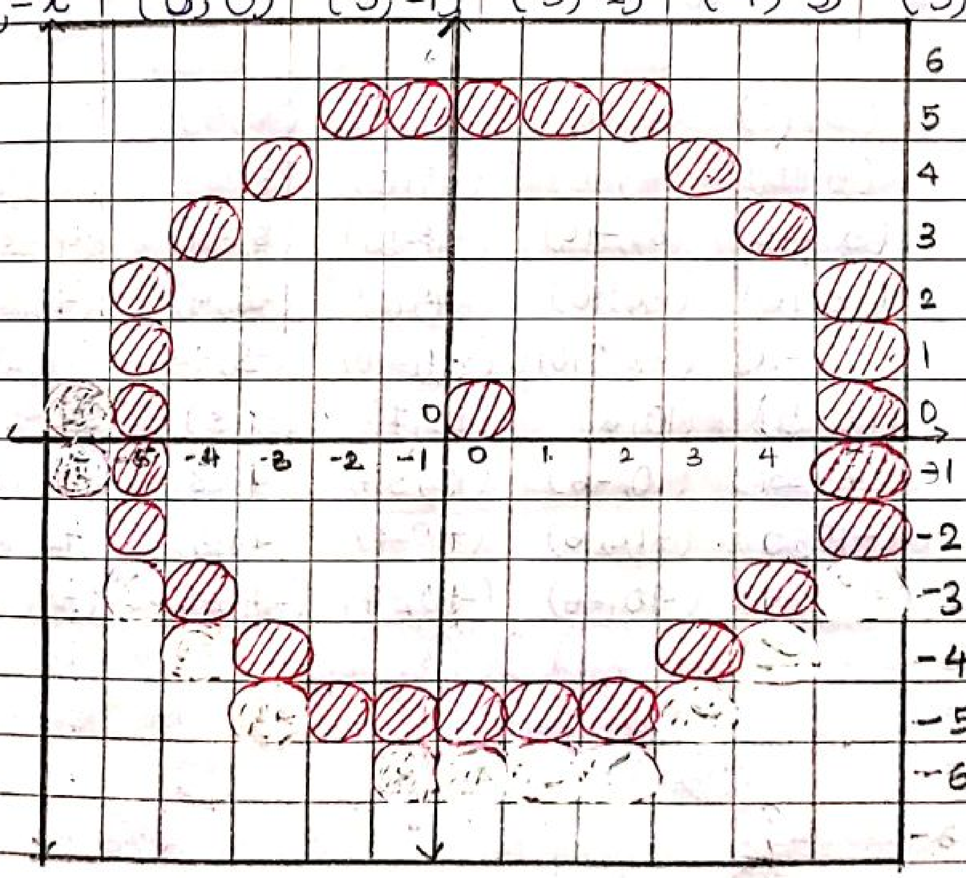
Table:

K	P_K	Plot (x_K, y_K)	$2x_{K+1}$	$2y_{K+1}$
.		(0,5)		
0	-4	(1,5)	2	10
1	-1	(2,5)	4	10
2	4	(3,4)	6	8
3	3	(4,3)	8	6



Now using the 8 point symmetry we get all point on circle as below.

	(0,5)	(1,5)	(2,5)	(3,4)	(4,3)
x, y	(0,5)	(1,5)	(2,5)	(3,4)	(4,3)
x, -y	(0,-5)	(1,-5)	(2,-5)	(3,-4)	(4,-3)
-x, -y	(0,-5)	(-1,-5)	(-2,-5)	(-3,-4)	(-4,-3)
-x, y	(0,5)	(-1,5)	(-2,5)	(-3,4)	(-4,3)
y, x	(5,0)	(5,1)	(5,2)	(4,3)	(3,4)
-y, -x	(-5,0)	(-5,-1)	(-5,-2)	(-4,-3)	(-3,-4)
-y, x	(-5,0)	(-5,1)	(-5,2)	(-4,3)	(-3,4)
y, -x	(5,0)	(5,-1)	(5,-2)	(4,-3)	(3,-4)



2. Plot a circle centered at (5,5) having radius of 5 unit using the midpoint circle algorithm.

Soln: The first point to plot is $(x_0+x_c, y_0+y_c) = (0+5, 5+5)$

$= (5,10)$

The initial value of the decision parameter is

$P_0 = 1-r = 1-5 = -4$

And the initial increment terms for calculating decision parameter is

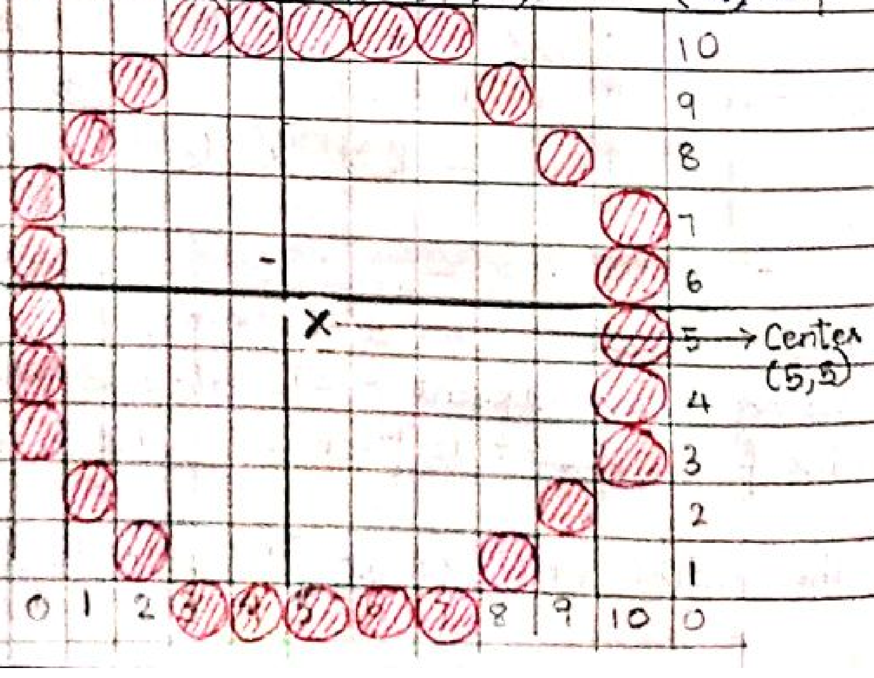
$2x_0 = 2 \times 0 = 0$

$2y_0 = 2 \times 5 = 10$.

Solve as before

K	P_k	Plot(x_k, y_k)	$2x_{k+1}$	$2y_{k+1}$	actual points$(x+x_c, y+y_c)$
		(0,5)			(5,10)
0	-4	(1,5)	2	10	(5,10)
1	-1	(2,5)	4	10	(7,10)
2	4	(3,4)	6	8	(8,9)
3	3	(4,3)	12	6	(9,8)

	(5,10)	(6,10)	(7,10)	(8,9)	(9,8)
x,y	(5,10)	(6,10)	(7,10)	(8,9)	(9,8)
x,-y	(5,0)	(6,0)	(7,0)	(8,1)	(9,2)
-x,y	(5,10)	(4,10)	(3,10)	(2,9)	(1,8)
-x,-y	(5,0)	(4,0)	(3,0)	(2,1)	(1,2)
y,x	(10,5)	(10,6)	(10,7)	(9,8)	(8,9)
y,-x	(10,5)	(10,4)	(0,3)	(9,2)	(8,1)
-y,x	(0,5)	(0,6)	(0,7)	(1,8)	(2,9)
-y,-x	(0,5)	(0,4)	(0,3)	(1,2)	(2,1)



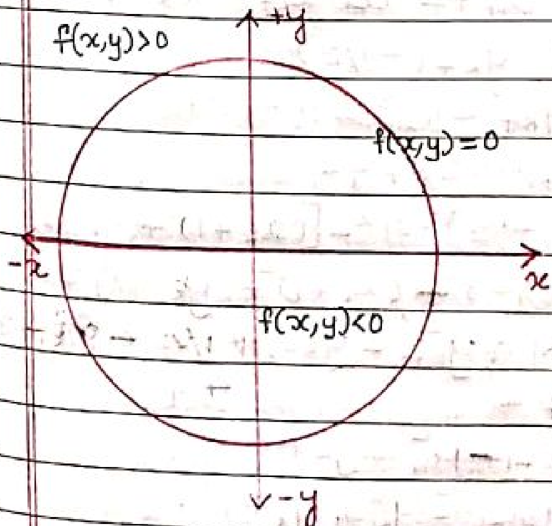
Midpoint circle algorithm

As in the raster line algorithm, we sample at unit intervals and determine the closest pixel position to the specified circle path at each step.
For a given radius r and screen center position (x_c, y_c), calculate pixel positions around a circle path centered at the coordinate origin $(0,0)$. Then each calculated position (x,y) is moved to its proper screen position (x,y) is moved to its proper screen position by adding x_c to x and y_c to y. To apply the midpoint method we define a circle function as

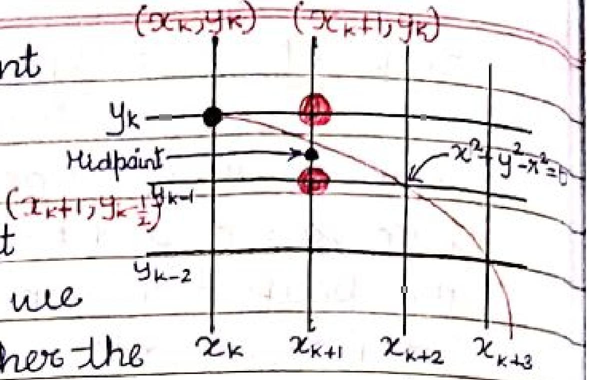
$$f_{circle}(x,y) = x^2 + y^2 - r^2$$

Any point (x,y) on the boundary of the circle with radius r satisfies the equation $f_{circle}(x,y)=0$. If the point is in the interior of the circle, the circle function is negative. And if the point is outside the circle, the circle function is positive

$$f_{circle}(x,y) = \begin{cases} <0; & \text{if } (x,y) \text{ is inside circle boundary} \\ =0; & \text{if } (x,y) \text{ is on the circle boundary} \\ >0; & \text{if } (x,y) \text{ is outside circle boundary.} \end{cases}$$



The tests in the above equations are performed for the mid positions between pixels near the circle path at each sampling step. The circle function is the decision parameter in the midpoint algorithm.



The figure shows the midpoint between the two candidate pixels at sampling position x_k+1. Assuming we have just plotted the pixel at (x_k, y_k) we next need to determine whether the next pixel at position (x_k+1, y_k) or the one that at position (x_k+1, y_k-1) is closer to the circle.

Decision parameter is the circle function evaluated at the midpoint between these 2 pixels.

$$P_k = f_{circle}(x_k+1, y_k-1/2)$$

$$\therefore P_k = (x_k+1)^2+(y_k-1/2)^2-r^2 \quad (\because x^2+y^2-r^2 = f_{circle})$$

If $P_k<0$, this midpoint is inside the circle and the pixel on scan line y_k is closer to circle boundary. Otherwise, the midpoint is outside or on the circle boundary, and we select the pixel on scanline y_{k-1}.

Successive decision parameters are obtained using incremental calculations. We obtain a recursive expression for the next decision parameter by evaluating the circle function at sampling position. $x_{k+1}+1 = x_k+1+1 = x_k+2$ as $x_{k+1} = x_k+1$.

$$P_{k+1} = f_{circle}(x_{k+1}+1, y_{k+1}-1/2)$$

$$= [(x_k+1)+1]^2+(y_{k+1}-1/2)^2-r^2$$

$$\therefore P_{k+1} = [(x_k+1)+1]^2+(y_{k+1}-1/2)^2-r^2$$

$$P_{k+1}-P_k = [(x_k+1)+1]^2+(y_{k+1}-1/2)^2-r^2-[(x_k+1)^2+(y_k-1/2)^2-r^2]$$

$$P_{k+1}-P_k = [(x_k+1)+1]^2+(y_{k+1}-1/2)^2-\cancel{r^2}-(x_k+1)^2-(y_k-1/2)^2+\cancel{r^2}$$

$$P_{k+1}-P_k = \cancel{(x_k+1)^2}+2(x_k+1)+1+y_{k+1}^2-y_{k+1}+\cancel{1/4}-\cancel{(x_k+1)^2}$$

$$1-y_k^2+y_k-\cancel{1/4}$$

$$= 2(x_k+1)+y_{k+1}^2-y_{k+1}-y_k^2+y_k+1$$

$$= 2(x_k+1)+y_{k+1}^2-y_{k+1}-y_k^2+y_k+1$$

$$P_{k+1}-P_k = 2(x_k+1)+(y_{k+1}^2-y_k^2)-(y_{k+1}-y_k)+1$$

Scanned by CamScanner

$\therefore P_{k+1} = P_k + 2(x_k+1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$

where y_{k+1} is either y_k or y_{k-1} depending on the sign of P_k

Increments for obtaining P_{k+1} are either $2x_{k+1}+1$ (if P_k is negative) or $2x_{k+1}+1-2y_{k+1}$ (if P_k is positive)
Evaluation of the terms $2x_{k+1}$ and $2y_{k+1}$ can also be done incrementally as

$$2x_{k+1} = 2x_k + 1 + 1 = 2x_k + 2$$

$$2y_{k+1} = 2y_k - 2$$

At the start position $(0, r)$, these two terms have the values 0 and $2r$ respectively.
Each successive value is obtained by adding 2 to the previous value of $2x$ and subtracting 2 from the previous value of $2y$. The initial decision parameter P_0 is obtained by evaluating the circle function at the start position $(x_0, y_0) = (0, r)$

$$P_0 = f_{circle}(x_k+1, y_k - 1/2)$$

$$\therefore P_0 = f_{circle}(0+1, r-1/2)$$

$$\therefore P_0 = f_{circle}(1, r-1/2)$$

$$\therefore P_0 = 1^2 + \left(r - \frac{1}{2}\right)^2 - r^2$$

$$\therefore P_0 = 1 + \cancel{r^2} - r + \frac{1}{4} - \cancel{r^2}$$

$$\therefore P_0 = 1 - r + \frac{1}{4}$$

$$= \left(1 + \frac{1}{4}\right) - r$$

$$\boxed{\therefore P_0 = \frac{5}{4} - r}$$

If the radius r is specified as an integer,

$$P_0 = 1 - r \quad (\text{for } r \text{ an integer})$$

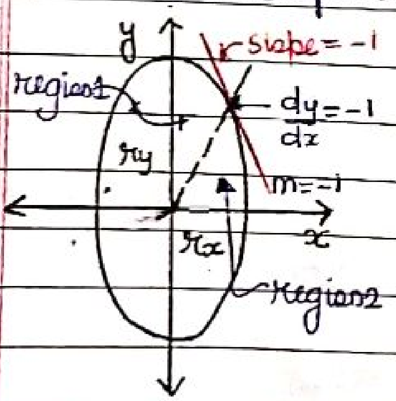
since all increments are integers

Ellipse Generating Algorithm

An ellipse is an elongated circle. Therefore, elliptical curves can be generated by modifying circle-drawing procedures to take into account the different dimensions of an ellipse along the major and minor axis.

* Midpoint Ellipse algorithm

The approach here is similar to that used in displaying a raster circle. Given parameters r_x, r_y and (x_c, y_c), we determine points (x, y) for an ellipse in standard position centered on the origin and then we shift the points so the ellipse is centered at (x_c, y_c).
The midpoint ellipse method is applied throughout the first quadrant in two parts. The figure below shows the decision of the first quadrant according to the slope of an ellipse with $r_x < r_y$. we process this quadrant by taking unit steps in the direction where the slope of the curve has a magnitude less than 1, and taking unit steps in the y direction where the slope has a magnitude greater than 1.



Ellipse processing regions. Over region 1, the magnitude of the ellipse slope is less than 1, over region 2, the magnitude of the ellipse slope is greater than 1.

Region 1 and region 2 can be processed in various ways. we can start at position $(0, r_y)$ and step clockwise along the elliptical path in the first quadrant, shifting from unit steps in y when slope becomes less.. than -1. Alternatively, we could start at $(r_x, 0)$ and select points in the counter clockwise order, shifting from unit steps in y to unit steps in x when the slope becomes greater than -1. With parallel processors, we could calculate pixel position in the two regions simultaneously. As an example of a sequential imple-mentation of the midpoint algorithm, we take start position at $(0, r_y)$ and step along the ellipse path in clockwise order throughout the first quadrant. We define an ellipse function from equation (1) with $(x_c, y_c) = (0,0)$ as

$$f_{ellipse}(x,y) = r_y^2 x^2 + r_x^2 y^2 - r_x^2 r_y^2 \quad \text{— (1)}$$

This has the following properties:

$$f_{ellipse}(x,y) \begin{cases} <0; & p(x,y) \text{ is inside the ellipse boundary} \\ =0; & p(x,y) \text{ is on the ellipse boundary} \\ >0; & p(x,y) \text{ is outside the ellipse boundary} \end{cases}$$

Thus, the ellipse function $f_{ellipse}(x,y)$ serves as the decision parameter in the midpoint algorithm. At each sampling position, we select the next pixel along the ellipse with path according to the origin of ellipse function evaluated at the midpoint between the 2 candidate pixels.

Starting at $(0, r_y)$ we take unit steps in the x direction until we reach the boundary between region 1 and region 2. Then we switch to unit steps in the y direction over the remainder of the curve in the first quadrant. At each step, we need to test the value of the slope of the curve. The ellipse slope is calculated from equation (1)

$$f_{ellipse}(x,y) = r_y^2 x^2 + r_x^2 y^2 - r_x^2 r_y^2$$

$$\frac{df}{dx} = 2r_y^2 x + 2y\, r_x^2 \frac{dy}{dx}$$

Since, $\frac{df}{dx} = 0$.

$$\therefore 0 = 2r_y^2 x + 2y \frac{dy}{dx} r_x^2$$

$$-2r_y^2 x = 2y \frac{dy}{dx} r_x^2$$

$$\frac{dy}{dx} = \frac{-2r_y^2 x}{2y\, r_x^2}$$

At the boundary between region 1 and region 2, $\frac{dy}{dx} = -1$ and

$$\therefore -1 = \frac{-2r_y^2 x}{2y\, r_x^2}$$

$$2r_y^2 x = 2r_x^2 y$$

Therefore, we move out of region 1 whenever,

$$2r_y^2 x \geq 2r_x^2 y$$

Figure (2) shows the midpoint between the 2 candidate pixels at sampling positions x_k+1 in the first region. Assuming position (x_k, y_k) has been selected at the previous step, we determine the next position along the ellipse path by evaluating the decision parameter (i.e the ellipse function equation (1)) at this midpoint.

$$p1_k = f_{ellipse}(x_k+1, y_k - 1/2)$$

$$\therefore p1_k = r_y^2 (x_k+1)^2 + r_x^2 (y_k - 1/2)^2 - r_x^2 r_y^2$$

If $p1_k < 0$, the midpoint is inside the ellipse and the pixel on scanline y_k is closest to the ellipse boundary. Otherwise, the midpoint is outside or on the ellipse boundary, and we select the pixel on scanline $y_k - 1$.

At the next sampling position $(x_{k+1}+1 = x_k+1+1 = x_k+2)$, the decision parameter for region 1 is evaluated as,

$$p1_{k+1} = f_{ellipse}(x_{k+1}+1, y_{k+1} - 1/2)$$

$$p1_{k+1} = r_y^2 [(x_k+1)+1]^2 + r_x^2 (y_{k+1} - 1/2)^2 - r_x^2 r_y^2$$

$$p1_{k+1} = r_y^2 [(x_k+1)^2 + 2(x_k+1)+1] + r_x^2 [y_{k+1} - 1/2]^2 - r_x^2 r_y^2$$

$$p1_{k+1} = r_y^2 (x_k+1)^2 + 2r_y^2(x_k+1) + r_y^2 + r_x^2 [y_{k+1} - 1/2]^2 - r_x^2 r_y^2$$

OR $p1_{k+1} - p1_k = r_y^2 (x_k+1)^2 + 2r_y^2 (x_k+1) + r_y^2 + r_x^2 [y_{k+1} - 1/2]^2 - r_x^2 r_y^2$

$$- r_x^2 r_y^2 - r_y^2 (x_k+1)^2 - r_x^2 (y_k - 1/2)^2 + r_x^2 r_y^2$$

$$p1_{k+1} - p1_k = 2r_y^2 (x_k+1) + r_y^2 + r_x^2 [(y_{k+1} - 1/2)^2 - (y_k - 1/2)^2]$$

$$\therefore p1_{k+1} = p1_k + 2r_y^2 (x_k+1) + r_y^2 + r_x^2 [(y_{k+1} - 1/2)^2 - (y_k - 1/2)^2]$$

where y_{k+1} is either y_k or $y_k - 1$, depending on the sign of $p1_k$.

Decision parameters are incremented by the following amounts:

$$\text{increment} = \begin{cases} 2r_y^2 x_{k+1} + r_y^2 & , \text{if } p1_k < 0 \\ 2r_y^2 x_{k+1} + r_y^2 - 2r_x^2 y_{k+1} & , \text{if } p1_k \geq 0 \end{cases}$$

As in the circle algorithm, increments for the decision parameters can be calculated using only addition and subtraction, since values for the terms $2r_y^2 x$ and $2r_x^2 y$ can also be obtained incrementally. At the initial position $(0, r_y)$ the two terms evaluate to,

$$2r_y^2 x = 0 \quad \text{——} (2)$$

$$2r_x^2 y = 2r_x^2 r_y \quad \text{——} (3)$$

As x and y are incremental updated values are obtained by adding $2r_y^2$ to equation (2) and subtracting $2r_x^2$ from equation (3). The updated values are compared at each step and we move from region1 to region2 when condition $2r_y^2x \geq 2r_x^2y$ is satisfied.

In region1, the individual value of the decision parameter is obtained by evaluating the ellipse function at the start position $(x_0, y_0) = (0, r_y)$

$$p1_0 = f_{ellipse}(1, r_y - 1/2)$$

$$\therefore p1_0 = r_y^2(1)^2 + r_x^2(r_y - 1/2)^2 - r_x^2 r_y^2$$

$$\therefore p1_0 = r_y^2 + r_x^2(r_y^2 - r_y + 1/4) - r_x^2 r_y^2$$

$$\therefore p1_0 = r_y^2 + r_x^2 r_y^2 - r_x^2 r_y + 1/4\, r_x^2 - r_x^2 r_y^2$$

$$\therefore p1_0 = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2$$

Over region2, we sample at unit steps in the negative of direction, and the midpoint is now taken between horizontal pixels at each step (fig 3). For this region, the decision parameter is evaluated as,

$$p2_k = f_{ellipse}(x_k + 1/2, y_k - 1)$$

$$\therefore p2_k = r_y^2(x_k + 1/2)^2 + r_x^2(y_k - 1)^2 - r_x^2 r_y^2$$

If $p2_k > 0$, the mid position is inside or on the ellipse boundary and we select pixel position x_{k+1}.

To determine the relationship between successive decision parameters in region 2, we evaluate the ellipse function at the next sampling step $y_{k+1} - 1 = y_k - 2$

$$p2_{k+1} = f_{ellipse}(x_{k+1} + 1/2, y_{k+1} - 1)$$

$$p2_{k+1} = r_y^2(x_{k+1} + 1/2)^2 + r_x^2[(y_k - 1) - 1]^2 - r_x^2 r_y^2$$

OR

$$p2_{k+1} - p2_k = r_y^2(x_{k+1} + 1/2)^2 + r_x^2[(y_k - 1) - 1]^2 - r_x^2 r_y^2 - r_y^2(x_k + 1/2)^2 - r_x^2(y_k - 1)^2 + r_x^2 r_y^2.$$

$$\therefore p2_{k+1} - p2_k = r_y^2(x_{k+1} + 1/2)^2 + r_x^2(y_k - 1)^2 - 2r_x^2(y_k - 1) + r_x^2 - r_y^2(x_k + 1/2)^2 - r_x^2(y_k - 1)^2.$$

Scanned by CamScanner

$\therefore \; p2_{k+1} - p2_k = r_y^2 (x_{k+1} + 1/2)^2 - 2r_x^2 (y_k - 1) + r_x^2 - r_y^2 (x_k + 1/2)^2$

$$p2_{k+1} = p2_k - 2r_x^2 (y_k - 1) + r_x^2 + r_y^2 \left[(x_{k+1} + 1/2)^2 - (x_k + 1/2)^2 \right]$$

with x_{k+1} set either to x_k or x_k+1, depending on the sign of $p2_k$.

When we enter region2, the initial position (x_0, y_0) is taken as the last position selected in region1. and the initial decision parameter in region2 is then,

$$p2_0 = f_{ellipse}(x_0 + 1/2, y_0 - 1)$$

$$p2_0 = r_y^2 (x_0 + 1/2)^2 + r_x^2 (y_0 - 1)^2 - r_x^2 r_y^2.$$

To simplify the calculations of $p2_0$, we could select pixel positions in counter clockwise order starting at $(r_x, 0)$. Unit steps would then be taken in the positive y direction upto the last position selected in region1.

Assuming r_x, r_y and the ellipse center are given in integer screen coordinates, we only need incremental integer calculations to determine values for the decision parameters in the midpoint ellipse algorithm. The increments r_x^2, r_y^2 and $2r_y^2$ are evaluated once at the beginning of the procedure

Midpoint Ellipse Algorithm

step1: Input r_x, r_y and ellipse center (x_c, y_c) and obtain first point on the ellipse as

step2: Calculate the initial value of the decision parameter in region1 as

$$P1_0 = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2$$

step3: At each x_k position in region1, starting at $k=0$ perform the following test
if $(P1_k < 0)$; the next point along the ellipse centered on $(0,0)$ is (x_k+1, y_k) and

$$P1_{k+1} = P1_k + 2r_y^2 x_{k+1} + r_y^2$$

otherwise the next point along the ellipse is (x_k+1, y_k-1) and

$$P1_{k+1} = P1_k + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_y^2$$

with, $2r_y^2 x_{k+1} = 2r_y^2 x_k + 2r_y^2$

$$2r_x^2 y_{k+1} = 2r_x^2 y_k - 2r_x^2$$

And continue until $2r_y^2 x \geq 2r_x^2 y$

step4: Calculate the initial value of the decision parameter in region2 using the last point (x_0, y_0) calculated in region1 as

$$P2_0 = r_y^2 (x_0 + 1/2)^2 + r_x^2 (y_0 - 1)^2 - r_x^2 r_y^2$$

step5: At each y_k position in region 2 starting at $k=0$ perform the following test
If $(P2_k > 0)$ the next point along the ellipse centered at $(0,0)$ is (x_k, y_{k-1}) and

$$P2_{k+1} = P2_k - 2r_x^2 y_{k+1} + r_x^2$$

otherwise the next point along the ellipse is (x_k+1, y_k-1) and

$$P2_{k+1} = P2_k + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_x^2$$

using the same incremental calculations on x and y in region 1.

step 6: Determine the symmetry points in the other three quadrants.

step 7: Move each calculated pixel position (x, y) on to the elliptical path sentered on (x_c, y_c) and plot the coordinate values

$$x = x + x_c$$
$$y = y + y_c$$

step 8: Repeat the steps for region 2 until $2r_y^2 x \geq 2r_x^2 y$ (Last point has to be $(r_x, 0)$).

1. Trace the midpoint ellipse algorithm for the given example. Initial center coordinate $(x_0, y_0) = (0, 5)$, $r_x = 6$, $r_y = 4$.

soln: Given

$(x_c, y_c) = (0, 5)$
$r_x = 6$
$r_y = 4$.

step 1:
Region 1: $(x_0, y_0) = (0, r_y) = (0, 4)$ $\therefore x_0 = 0$ & $y_0 = 4$

Initially;

$$2r_y^2 x = 2 \times (4)^2 \times 0 = 0$$
$$2r_y^2 = 2 \times (4)^2 = 32$$
$$2r_x^2 = 2 \times (6)^2 = 72$$

step 2: calculating initial decision parameter for region 1.

$$P1_0 = r_y^2 - r_x^2 r_y + 1/4\, r_x^2$$
$$= (4)^2 - (6)^2 4 + \frac{1}{4} (6)^2$$
$$= 16 - 36 \times 4 + 9$$
$$= -119$$

classmate
Date
Page

step3: At each x_k position in region 1 do as follows:-

k=0) Is $P1_0 < 0$? $-119 < 0$? YES

$\therefore$ Next point to plot is $(x_k+1, y_k) = (1,4)$

$2r_y^2 x_{k+1} = 2\times(4)^2\times(1) = 32$

$2r_x^2 y_{k+1} = 2\times(6)^2\times(4) = 288$

$2r_y^2 x_{k+1} \ngeq 2r_x^2 y$ so continue next parameter

$P1_{k+1} = P1_k + 2r_y^2 x_{k+1} + r_y^2$ OR $P1_{k+1} = P1_k + 2r_y^2 x_k + 2r_y^2 + r_y^2$

$= -119 + (32\times1) + (4)^2$ $\quad = -119 + (32\times0) + 32 + (4)^2$

$= -71$ $\quad = -71$

k=1) Is $P1_1 < 0$? $-71 < 0$? YES

$\therefore$ Next point to plot is $(x_k+1, y_k) = (2,4)$

Is $(2r_y^2 x_{k+1} \geq 2r_x^2 y_{k+1})$

$64 \ngeq 288$. NO.

$P1_{k+1} = P1_k + 2r_y^2 x_{k+1} + r_y^2$

$= -71 + 2(4)^2(2) + (4)^2$

$= -71 + 64 + 16$

$= 9.$

k=2) Is $P_2 < 0$? $9 < 0$? No

$\therefore$ Next point to plot is $(x_k+1 ; y_k-1) = (3,3)$

Is $(2r_y^2 x_{k+1} \geq 2r_x^2 y_{k+1})$

$(2(4)^2(3) \geq 2(6)^2(3))$?

$(96 \ngeq 216)$? NO.

$P1_{k+1} = P1_k + 2r_y^2 x_{k+1} - 2r_x^2 \cdot y_{k+1} + r_y^2$

$= 9 + 2(4)^2(3) - 2(6)^2(3) + (4)^2$

$= 9 + 96 - 216 + 16$

$= -95$

k=3) Is $P_3 < 0$? $-95 < 0$? YES

$\therefore$ Next point to plot is $(x_k+1, y_k) = (4,3)$.

Is $(2r_y^2 x_{k+1} \geq 2r_x^2 y_{k+1})$?

$(128 \ngeq 216)$ $\therefore$ continue to next point.

$P1_{k+1} = P1_k + 2r_y^2 x_{k+1} + r_y^2$

$= -95 + 128 + 16$

$= 49.$

Scanned by CamScanner

classmate
Date
Page

$k = 4$, Is $P_4 < 0$? $49 < 0$? NO

Next point to plot is $(x_k+1, y_k-1) = (5,2)$

Is $(2r_y^2 x_{k+1} \geq 2r_x^2 y_{k+1})$

$(160 \geq 144)$? YES.

Therefore stop.

For region 2

step 1: $(x_0, y_0) = (5, 2)$ $x_0 = 5$, $y_0 = 2$.

step 2: Calculate the initial decision parameter for region 2

$$P2_0 = r_y^2 (x_0 + 1/2)^2 + r_x^2 (y_0 - 1)^2 - r_x^2 r_y^2$$

$$= -56$$

step 3:

$k=0$, Is $P2_0 > 0$? $-56 > 0$? No

$\therefore$ Next point to plot is $(x_k+1, y_k-1) = (6,1)$

Is $(2r_y^2 x_{k+1} \geq 2r_x^2 y_{k+1})$?

$192 \geq 72$. YES so continue to next point.

$$P2_{k+1} = P2_k + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_x^2$$

$$= 100$$

$k=1$, Is $P2_1 > 0$? $100 > 0$? YES

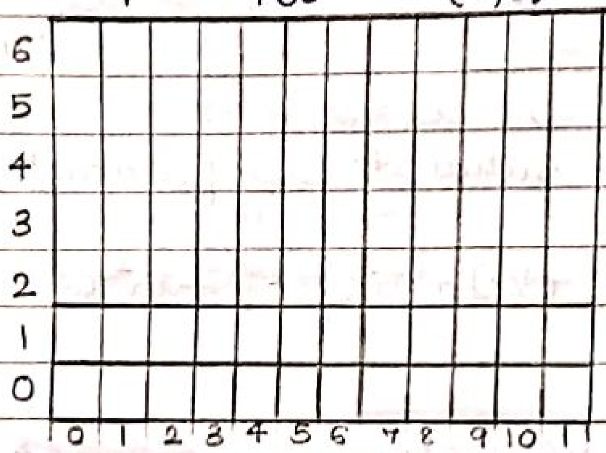
$\therefore$ Next point to plot is $(x_k, y_k-1) = (6,0)$

Stop since we get back $(6,0)$

successive decision parameter value and position along ellipse path is calculated using midpoint ellipse method at $(0,0)$ as follows

The remaining positions along the ellipse path in the first quadrant are calculated as follow In region2 at center (0,0) as shown below

k	$P2_k$	(x_{k+1}, y_{k+1})	$2r_y^2 x_{k+1}$	$2r_x^2 y_{k+1}$
0	-56	(6,1)	192	72
1	-100	(6,0)	–	–



Successive decision parameters and values are calculated as follows:-

$(x_c, y_c) = (0,5)$. $x_c = 0$, $y = 5$

Hence the points to plot in region1 are $x = x + x_c$ and $y = y + y_c$

k	(x_{k+1}, y_{k+1})	Actual points to plot $(x = x + x_c, y = y + y_c)$
	(0,4)	(0,9)
	(1,4)	(1,9)
	(2,4)	(2,9)
	(3,3)	(3,8)
	(4,3)	(4,8)
	(5,2)	(5,7)

k	(x_{k+1}, y_{k+1})	Actual points to plot $(x + x_c)$ $y = y + y_c$
0	(6,1)	(6,6)
1	(6,0)	(6,5)

Scanned by CamScanner

using the (four symmetry) we get all the points to be plotted on ellipse as below.

(-x, y)

(-x, -y)

(x, -y)

Scanned by CamScanner

Polygon Filling

Difference between Flood-fill and Boundary-fill

Flood Fill	Boundary Fill.
1. These algorithm are used to colour an entire area of connected pixels with the same color.	1. These algorithm are used to colour an area with pixels of a certain colour as boundary.
2. It is also known as forest-fire fill algorithm because it spreads from a seed in all the direction.	2. It is not called as a forest-fire fill algorithm as here painting is continued till the boundary of figure is reached.

Procedure for flood-fill algorithm for filling 4-connected region is as follows:-

```
void floodfill (int x, int y, int fillColor, int OldColor)
{
 if (getPixel (x,y) == OldColor)
 {
    setColor(fillColor);
    setPixel (x,y);
    floodFill ( x+1, y, fillColor, OldColor);
    floodFill ( x-1, y, fillColor, OldColor);
    floodFill ( x, y+1, fillColor, Oldcolor);
    floodFill ( x, y-1, fillColor, OldColor);
 }
}.
```

```
void floodfill8Connect(int x, int y, int fillColor, int OldColor)
{
  if( getPixel(x, y) == OldColor )
  {
     setColor(fillColor);
     setPixel (x,y) ;
     floodfill8Connect(x, y-1, fillColor, Oldcolor);
     floodfill8Connect(x, y+1, fillColor, OldColor);
     floodfill8connect( x-1, y, fillcolor, OldColor);
     floodfill8Connect( x+1, y, fillColor, OldColor);
     floodfill8Connect(x-1, y-1, fillColor, OldColor);
     floodfill8Connect(x+1, y-1 , fillColor, OldColor);
     floodfill8Connect(x-1, y+1, fillColor, OldColor);
     floodfill8Connect(x+1, y+1, fillColor, OldColor);
  }
}
```

* Recursive approach.

Pseudocode of boundary fill access pixel spans for a 4-connected area.

```
Void boundaryFill(int x, int y, int fill, int boundary)
{
  int current;
  current = getPixel (x,y);
  if ((current != boundary) && (current != fill))
  {
    setColor(fill);
    setPixel (x,y);
    boundaryFill (x+1, y, fill, boundary);
    boundaryFill (x-1, y , fill, boundary);
    boundaryFill (x, y+1, fill, boundary);
```

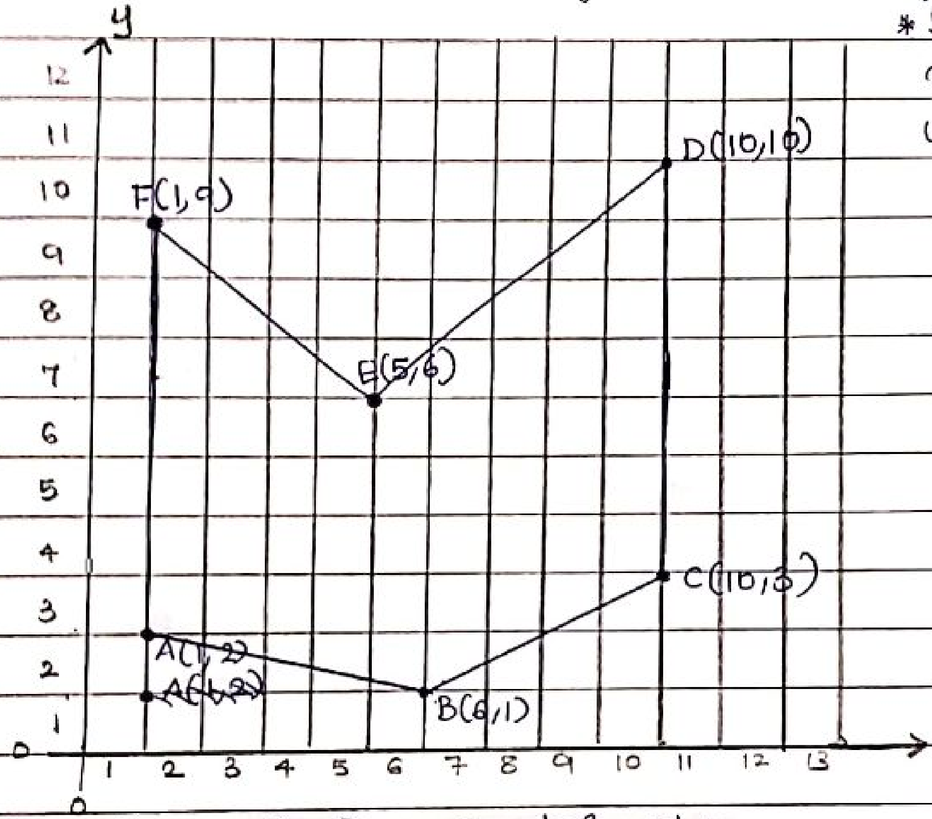
```
        boundaryFill (x, y-1, fill, boundary);
    }
}


Void boundaryFill(int x, int y, int fill, int boundary)
{
    int current;
    current = getPixel (x, y);
    if((current != boundary) && (current != fill))
    {
        setColor(fill);
        setPixel (x, y);
        boundaryFill(x+1, y, fill, boundary);
        boundaryFill(x-1, y, fill, boundary);
        boundaryFill(x, y-1, fill, boundary);
        boundaryFill(x, y+1, fill, boundary);
        boundaryFill(x-1, y+1, fill, boundary);
        boundaryFill(x-1, y-1, fill, boundary);
        boundaryFill (x+1, y+1, fill, boundary);
        boundaryFill (x+1, y-1, fill, boundary);
    }
}
```

Scanline

1. Generate the sorted edge table and active edge list to fill the polygon having coordinates A(1,2), B(6,1), C(10,3), D(10,10), E(5,6), F(1,9) using scanline polygon fill.

Soln:



* Note xmin is the x value associated with y min.

* Note xmin is the x value associated with ymin.

A, B ⇒ (1,2) (6,1) $m = \frac{1-2}{6-1} = \frac{-1}{5}$

B, C ⇒ (6,1) (10,3) $m = \frac{3-1}{10-6} = \frac{2}{4} = \frac{1}{2}$

C, D ⇒ (10,3) (10,10) $m = \frac{10-3}{10-10} = \frac{7}{0}$ undefined

D, E ⇒ (10,10) (5,6) $m = \frac{6-10}{5-10} = \frac{-4}{-5} = \frac{4}{5}$

E, F ⇒ (5,6) (1,9) $m = \frac{9-6}{1-5} = \frac{3}{-4} = \frac{-3}{4}$

F, A ⇒ (1,9) (1,2) $m = \frac{2-9}{1-1} = \frac{-7}{0}$ = undefined

TUTORIALS :

The positions to be calculated as follows for DDA algorithm.

Quadrant	$m <= 1$	$m > 1$
First	$x = x + 1$ $y = y + m$	$x = x + \frac{1}{m}$ $y = y + 1$
Second	$x = x - 1$ $y = y + m$	$x = x - \frac{1}{m}$ $y = y + 1$
Third	$x = x - 1$ $y = y - m$	$x = x - \frac{1}{m}$ $y = y - 1$
Fourth	$x = x + 1$ $y = y - m$	$x = x + \frac{1}{m}$ $y = y - 1$

4/2/20

Tutorial week-2

1. Consider a line from (20,10) to (28,5). Use the simple DDA algorithm to rasterize the line.

Soln: step1: $(x_1, y_1) = (20, 10)$

$(x_2, y_2) = (28, 5)$

$\therefore x_1 = 20, y_1 = 10, x_2 = 28, y_2 = 5$

step2: $dx = |x_2 - x_1| = |28 - 20| = 8$

$dy = |y_2 - y_1| = |5 - 10| = 5.$

step3: length $= dx = 8$

step4: $dx = \frac{(x_2 - x_1)}{\text{length}} = \frac{8}{8} = 1$

$dy = \frac{(y_2 - y_1)}{\text{Length}} = \frac{-5}{8} = -0.625$

step5: plot(20,10)

step6: $x =$

i) $i = 1$, Is ($i <=$ length)? ($1 <= 8$)? YES

$x = x + dx = 20 + 1 = 21$

$y = y + dy = 10 + (-0.625) = 9.375$

plot(21, 9)

$i = i + 1 = 1 + 1 = 2$

ii) $i = 2$, Is ($i <=$ length)? ($2 <=$ length)? YES

$x = x + dx = 28 + 1 = 22$

$y = y + dy = 9.5 - 0.625 = 8.875$

plot(30, 9).

iii) $i=3$, Is ($3 <=$ length)? YES

$x = x + dx = 22 + 1 = 23$

$y = y + dy = 8.75 - 0.625 = 8.125$

plot (23, 8.125)

iv) $i=4$, Is ($4 <= 8$)? YES

$x = x + dx = 23 + 1 = 24$

$y = y + dy = 8.125 - 0.625 = 7.5$

plot (24, 8)

v) $i=5$, Is ($5 <= 8$)? YES

$x = x + dx = 24 + 1 = 25$

$y = y + dy = 7.5 - 0.625 = 6.875$

plot (25, 7)

vi) $i=6$, Is ($6 <= 8$)? YES

$x = x + dx = 25 + 1 = 26$

$y = y + dy = 6.875 - 0.625 = 6.25$

plot (26, 6)

vii) $i=7$, Is ($7 <= 8$)? YES

$x = 26 + 1 = 27$

$y = 6.25 - 0.625 = 5.625$

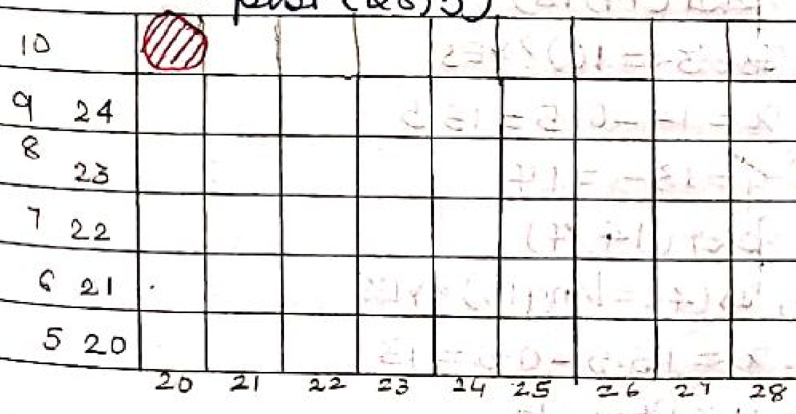
plot (27, 6)

viii) $i=8$, Is ($8 <= 8$)? YES

$x = 27 + 1 = 28$

$y = 5.625 - 0.625 = 5$

plot (28, 5)



2. Consider a line from (15, 20) to (10,10). Use the DDA algorithm to rasterize this line.

Soln: $slope = \frac{y_2 - y_1}{x_2 - x_1} = \frac{10-20}{10-15} = \frac{-10}{-5} = 2$ $\frac{10-20}{10-15}$

slope, m>1

step1: $(x_1, y_1) = (15, 20)$

$(x_2, y_2) = (10, 10)$

$x_1 = 15, y_1 = 20, x_2 = 10, y_2 = 10$

step2: $dx = |x_2 - x_1| = |10-15| = 5$

$dy = |y_2 - y_1| = |10-20| = 10$

step3: length = 10.

step4: $dx = \frac{(x_2 - x_1)}{length} = \frac{-5}{10} = -0.5$

$dy = \frac{(y_2 - y_1)}{length} = \frac{-10}{10} = -1.$

step5: plot (15, 20)

step6: 1) i=1, Is (1<=10)? YES

$x = x + dx = 15 - 0.5 = 14.5$

$y = 20 - 1 = 19$

plot (15, 19)

2) i=2, Is (2<=10)? YES

$x = 14.5 - 0.5 = 14$

$y = 19 - 1 = 18$

plot (14, 18)

3) i=3, Is (3<=10)? YES

$x = 14 - 0.5 = 13.5$

$y = 18 - 1 = 17$

plot (14, 17)

4) i=4, Is (4<=length)? YES

$x = 13.5 - 0.5 = 13$

$y = 17 - 1 = 16$

plot (13, 16)

5) i=5, Is (5<=10) ? YES
$x = 13-0.5=12.5$
$y = 16-1=15$
plot(13,15)

6) i=6, Is (6<=10) ? YES
$x=12.5-0.5=12$
$y = 15-1=14$
plot(12,14)

7) i=7, Is (7<=10) ? YES
$x=12-0.5= 11.5$
$y=14-1=13$
plot(12,13)

8) i=8, Is (8<=10) ? YES
$x=11.5-0.5=11$
$y = 13-1=12$
plot(11,12)

9) i=9, Is (9<=10) ? YES
$x=11-0.5=10.5$
$y=12-1=11$
plot(11,11)

10) i=10, Is (10<=10) ? YES
$x=10.5-0.5=10$
$y=11-1=10$
plot(10,10).

~~9) i.~~

i	plot(x,y)	x	y
	(15,20)	15	20
1	(15,19)	15	19
2	(14,18)	14	18
3	(14,17)	14	17
4	(13,16)	13	16
5	(13,15)	13	15
6	(12,14)	12	14
7	(12,13)	12	13
8	(11,12)	11	12
9	(11,11)	11	11
10	(10,10)	10	10

1. Indicate the raster location that can be chosen by scan converting a line (1,1) to (8,5).

Soln: **step1:**

$(x_0, y_0) = (1,1)$

$m = \left| \frac{y_2 - y_1}{x_2 - x_1} \right| = \left| \frac{5-1}{8-1} \right| = \frac{4}{7} = 0.571$

$m < 1$

step2:

$\Delta y = |y_2 - y_1| = 4$

$\Delta x = |x_2 - x_1| = 7$

$2\Delta y = 8$

$2\Delta y - 2\Delta x = 8 - 7 \times 2$

$= 8 - 14$

$= -6$

P_0 is the initial permutation. Initially we plot (x_0, y_0). i.e (1,1) and further determine the points-

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

$= 8 - 7$

$= 1$

step4:

1) $K = 0, P_K = P_0 = 1$.

Is $(P_K < 0)$? $(1 < 0)$? NO

plot$(x_K + 1, y_K + 1)$ = plot(2,2).

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

$= 1 - 6$

$P_1 = -5$

2) $K = 1, P_K = P_1 = -5$

Is $(-5 < 0)$? YES

plot$(x_K + 1, y_K)$ = plot(3,2)

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

$= -5 + 8$

$P_2 = 3$.

3) $k=2, P_k = P_2 = 3$

Is $(P_k < 0)$? $(3<0)$? NO

plot $(x_k+1, y_k+1) =$ plot $(4, 3)$

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

$= 3 - 6$

$P_3 = -3$

4) $k=3, P_k = -3$

Is $(-3<0)$? YES

plot $(x_k+1, y_k) =$ plot $(5,3)$

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

$= -3+8$

$P_4 = 5$

5) $k=4, P_k = P_4 = 5$

Is $(5<0)$? NO

plot $(x_k+1, y_{k+1}) =$ plot $(6,4)$

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

$= 5 - 6$

$P_5 = -1$

6) $k=5, P_k = P_5 = -1$

Is $(-1<0)$? YES

plot $(x_k+1, y_k) =$ plot $(7,4)$

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

$= -1+8$

$P_6 = 7$.

7) $k=6, P_k = P_6 = 7$

Is $(7<0)$? NO

plot $(x_k+1, y_k+1) =$ plot $(8,5)$

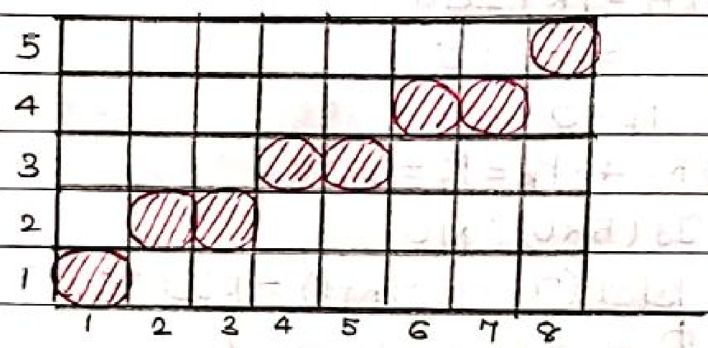
$P_{k+1} = P_k + 2\Delta y - 2\Delta x$

$= 7 - 6$

$= 1$

stop.

k	P_k	plot (x, y)
–	–	(1,1)
0	1	(2,2)
1	−5	(3,2)
2	3	(4,3)
3	−3	(5,3)
4	5	(6,4)
5	−1	(7,4)
6	7	(8,5)



3) Digitize line with endpoints (20,10) and (30,18) using DDA algorithm.

Soln: $\text{slope} = \frac{y_2 - y_1}{x_2 - x_1} = \left|\frac{18-10}{30-20}\right| = \left|\frac{8}{10}\right| = 0.8 < 1.$

step 1: $(x_1, y_1) = (20,10)$, $(x_2, y_2) = (30,18)$ $\therefore x_1 = 20$, $y_1 = 10$, $x_2 = 30$, $y_2 = 18$.

step 2: $dx = |y_2 - x_1| = |30 - 20| = 10$

$dy = |y_2 - y_1| = |18 - 10| = 8.$

step 3: length $= 10$

step 4: $dx = \frac{(x_2 - x_1)}{\text{length}} = \frac{10}{10} = 1$

$dy = \frac{(y_2 - y_1)}{\text{length}} = \frac{8}{10} = 0.8$

step 5: plot (20,10).

step 6: Plot the remaining points

1) i=1, Is (1<=10)? YES

$x = x + 1 = 20 + 1 = 21$

$y = y + 0.8 = 10 + 0.8 = 10.8$

plot (21, 11)

2) i=2, Is (2<=10)? YES

$x = 21 + 1 = 22$

$y = 10.8 + 0.8 = 11.6$

plot (22, 12)

3) i=3, Is (3<=10)? YES

$x = 22 + 1 = 23$

$y = 11.6 + 0.8 = 12.4$

plot (23, 12)

4) i=4, Is (4<=10)? YES

$x = 23 + 1 = 24$

$y = 12.4 + 0.8 = 13.2$

plot (24, 13)

5) i=5, Is (5<=10)? YES

$x = 24 + 1 = 25$

$y = 13.2 + 0.8 = 14.0$

plot (25, 14)

6) i=6, Is (6<=10)? YES

$x = x + 1 = 25 + 1 = 26$

$y = y + 1 = 14 + 0.8 = 14.8$

plot (26, 15)

7) i=7, Is (7<=10)? YES

$x = x + 1 = 26 + 1 = 27$

$y = 14.8 + 0.8 = 15.6$

plot (27, 16)

8) i=8, Is (8<=10)? YES

$x = x + 1 = 27 + 1 = 28$

$y = 15.6 + 0.8 = 16.4$

plot (28, 16)

9) i=9, Is (9<=10)? YES

$x = 28 + 1 = 29$

$y = 16.4 + 0.8 = 17.2$

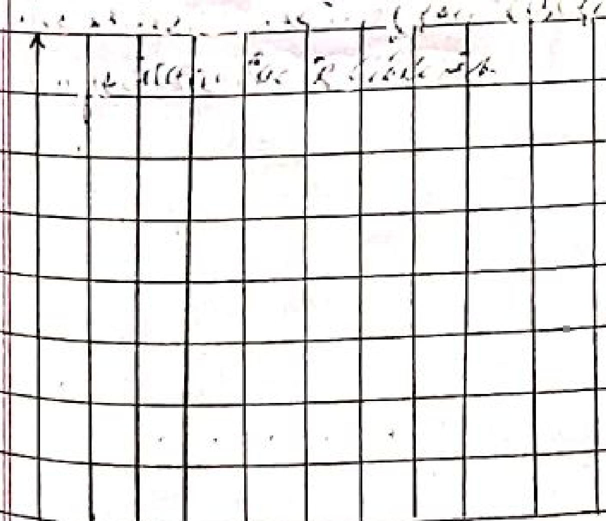
plot (29, 17)

10) i=10, Is (10<=10)? YES

$x = 29 + 1 = 30$

$y = 17.2 + 0.8 = 18$

plot (30, 18).



4) Consider a line from (0,0) to (6,7). Use the simple DDA algorithm to rasterize the line.

Soln: $m = \left|\frac{y_2-y_1}{x_2-x_1}\right| = \left|\frac{7-0}{6-0}\right| = \frac{7}{6}$.

step1: $(x_1, y_1) = (0,0)$ $(x_2, y_2) = (6,7)$

step2: $dx = |x_2 - x_1| = 6-0=6$ $dy = |y_2 - y_1| = 7-0=7$.

step3: length = 7.

step4: $dx = \frac{x_2 - x_1}{\text{Length}} = \frac{6}{7} = 0.857$

$dy = \frac{y_2 - y_1}{\text{Length}} = \frac{7}{7} = 1$

step5: plot (0,0)

step6: calculate the remaining points

1) $i=1$, Is $(1 <= 7)$? YES

$x = x + 0.857 = 0.857$

$y = 0+1 = 1$

plot(1,1)

2) $i=2$, Is $(2 <= 7)$? YES

$x = 0.857 + 0.857 = 1.714$

$y = 1+1 = 2$

plot (2,2)

3) $i=3$, Is $(3 <= 7)$? YES

$x = 1.714 + 0.857 = 2.571$

$y = 2+1 = 3$

plot (3,3)

4) $i=4$, Is $(4 <= 7)$? YES

$x = 2.571 + 0.857 = 3.428$

$y = 3+1 = 4$

plot (3,4)

5) $i=5$, Is $(5 <= 7)$? YES

$x = 3.428 + 0.857 = 4.285$

$y = 4+1 = 5$

plot(4,5)

6) $i=6$, Is $(6 <= 7)$? YES

$x = 4.285 + 0.857 = 5.142$

$y = 5+1 = 6$

plot (5,6)

7) $i=7$, Is $(7 <= 7)$? YES

$x = 5.142 + 0.857 = 5.999$

$y = 6+1 = 7$

plot (6,7)

i	plot(x,y)	x	y
	(0,0)	0	0
1	(1,1)	1	1
2	(2,2)	2	2
3	(3,3)	3	3
4	(3,4)	3	4
5	(4,5)	4	5
6	(5,6)	5	6
7	(6,7)	6	7

5) Digitize the line with endpoints (15,7) and (25,15) using DDA algorithm.

Soln: $m = \left|\frac{y_2 - y_1}{x_2 - x_1}\right| = \left|\frac{15-7}{25-15}\right| = \frac{8}{10} = 0.8 < 1$

step 1: $(x_1, y_1) = (15,7)$ $(x_2, y_2) = (25,15)$

step 2: $dx = |x_2 - x_1| = |25-15| = 10$ $dy = |y_2 - y_1| = |15-7| = 8$

step 3: length = 10

step 4: $dx = \frac{x_2 - x_1}{\text{length}} = \frac{10}{10} = 1$ $dy = \frac{y_2 - y_1}{\text{length}} = \frac{8}{10} = 0.8$

step 5: plot (15,7)

step 6: Plot the remaining points

1) i=1, Is(i<=length)(1<=10)? yes
$x = 15+1 = 16$, $y = 7+0.8 = 7.8$
plot(16,8)

2) i=2, Is(2<=10)? YES
$x = 17$ $y = 7.8+0.8 = 8.6$
plot(17,9)

3) i=3, Is(3<=10)? YES
$x = 18$ $y = 8.6+0.8 = 9.4$
plot(18,9)

4) i=4, Is(4<=10)? YES
$x = 19$ $y = 9.4+0.8 = 10.2$
plot(19,10)

5) i=5, Is(5<=10)? YES
$x = 20$ $y = 10.2+0.8 = 11$
plot(20,11)

6) i=6, Is(6<=10)? YES
$x = 21$ $y = 11+0.8 = 11.8$
plot(21,12)

7) i=7, Is(7<=10)? YES
$x = 22$ $y = 11.8+0.8 = 12.6$
plot(22,13)

8) i=8, Is(8<=10)? YES
$x = 23$ $y = 12.6+0.8 = 13.4$
plot(23,13)

9) $i=9$, Is $(9<=10)$? YES

$x=24 \quad y=13.4+0.8=14.2$

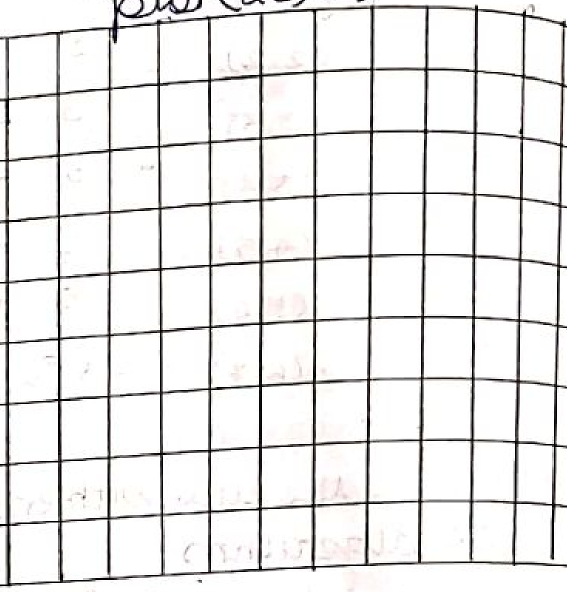
plot(24, 14)

10) $i=10$, Is $(10<=10)$? YES

$x=25 \quad y=14.2+0.8=15$

plot(25,15)

i	plot(x, y)	x	y
	(15,7)	15	7
1	(16,8)	16	8
2	(17,9)	17	9
3	(18,9)	18	9
4	(19,10)	19	10
5	(20,11)	20	11
6	(21,12)	21	12
7	(22,13)	22	13
8	(23,13)	23	13
9	(24,14)	24	14
10	(25,15)	25	15



6) Consider a line from (250,100) to (240,105). Use the simple DDA algorithm to rasterize this line.

Soln: $m=\left|\dfrac{y_2-y_1}{x_2-x_1}\right|=\left|\dfrac{105-100}{240-250}\right| = \dfrac{5}{10} = 0.5 < 1$

step 1: $(x_1, y_1)=(250,100) \quad (x_2, y_2)=(240,105)$

step 2: $dx=|x_2-x_1|=10 \quad dy=|y_2-y_1|=5$

step 3: length $=10$

step 4: $dx=\dfrac{(x_2-x_1)}{\text{Length}}=\dfrac{-10}{10}=-1$

$dy=\dfrac{(y_2-y_1)}{\text{Length}}=\dfrac{5}{10}=0.5$

step 5: plot (250,100)

step 6: Plot the remaining points.

1) $i=1$, Is $(1<=10)$? YES

$x=250-1=249$

$y=100+0.5=100.5$

plot(249,101)

2) $i=2$, Is $(2<=10)$? YES

$x=249-1=248$

$y=100.5+0.5=101$

Plot (248,101)

3) $i=3$, Is $(3<=10)$? YES
$x=247$ $y=101.5$
plot(247,102)

4) $i=4$, Is $(4<=10)$? YES
$x=246$ $y=102$
plot (246,102)

5) $i=5$, Is $(5<=10)$? YES
$x=245$ $y=102.5$
plot(245,103)

6) $i=6$, Is $(6<=10)$? YES
$x=244$ $y=103$
plot(244,103)

7) $i=7$, Is $(7<=10)$? YES
$x=243$, $y=103.5$
plot (243,104)

8) $i=8$, Is $(8<=10)$? YES
$x=242$ $y=104$
plot (242,104)

9) $i=9$, Is $(9<=10)$? YES
$x=241$ $y=104.5$
plot (241,105)

10) $i=10$, Is $(10<=10)$? YES
$x=240$ $y=105$
plot(240,105).

i	plot(x,y)	x	y
	(250,100)	250	100
1	(249,101)	249	101
2	(248,101)	248	101
3	(247,102)	247	102
4	(246,102)	246	102
5	(245,103)	245	103
6	(244,103)	244	103
7	(243,104)	243	104
8	(242,104)	242	104
9	(241,105)	241	105
10	(241,105)	240	105

7) Consider a line from (10,10) to (15,20). Use the simple DDA Algorithm to rasterize this line.

Soln: $m = \left|\frac{y_2-y_1}{x_2-x_1}\right| = \left|\frac{20-10}{15-10}\right| = \frac{10}{5} = 2 > 1$

step1: $(x_1,y_1)=(10,10)$ $(x_2,y_2)=(15,20)$

step2: $dx=|x_2-x_1|=5$ $dy=|y_2-y_1|=10$

step3: length $=10$

step4: $dx=\frac{(x_2-x_1)}{\text{length}} = \frac{5}{10} = 0.5$

$dy = \frac{(y_2-y_1)}{\text{length}} = \frac{10}{10} = 1$

step5: plot(10,10)

step6: Plot the remaining points

1) i=1, Is(1<=10)? YES

$x=10+0.5=10.5$ $y=10+1=11$

plot(11,11)

2) i=2, Is(2<=10)? YES

$x=11$ $y=12$

plot(11,12)

3) i=3, Is(3<=10)? YES

$x=11.5$ $y=13$

plot(12,13)

4) i=4, Is(4<=10)? YES

$x=12$ $y=14$

plot(12,14)

5) i=5, Is(5<=10)? YES

$x=12.5$ $y=15$

plot(13,15)

6) i=6, Is(6<=10)? YES

$x=13$ $y=16$

plot(13,16)

7) i=7, Is(7<=10)? YES

$x=13.5$ $y=17$

plot(14,17)

8) i=8, Is(8<=10)? YES

$x=14$ $y=18$

plot(14,18)

9) i=9, Is(9<=10)? YES

$x=14.5$ $y=19$

plot(15,19)

10) i=10, Is(10<=10)? YES

$x=15$ $y=20$

plot(15,20)

i	plot(x,y)	x	y
	(10,10)	10	10
1	(11,11)	11	11
2	(11,12)	11	12
3	(12,13)	12	13
4	(12,14)	12	14
5	(13,15)	13	15
6	(13,16)	13	16
7	(14,17)	14	17
8	(14,18)	14	18
9	(15,19)	15	19
10	(15,20)	15	20