Chapter 2 Scan-Conversion

Picture can be described in several ways, in raster display, picture is completely specified by set of intensity values of pixel positions.

Points and Lines:

Points are plotted by converting co-ordinate position to appropriate operations for the output device (e.g.: in CRT monitor, the electron beam is turned on to illuminate the screen phosphor at the selected location.).

Pixel and Pixel Position:

One dot or picture element of the raster is known as pixel. The pixel (a word invented from "picture element") is the basic unit of programmable color on a computer display or in a computer image.

Pixel Position is referenced by scan line number and column number.

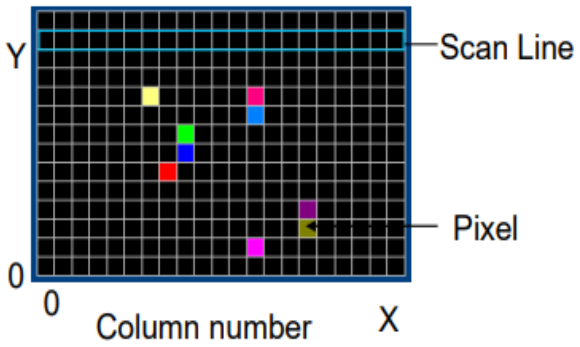


Figure 2.1 pixel representing raster system

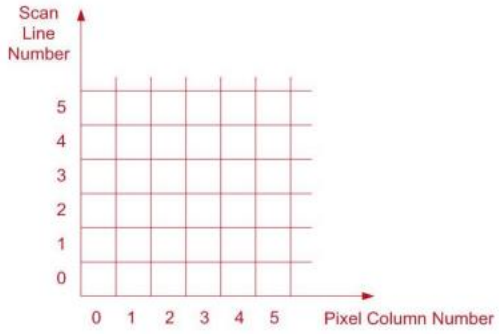


Figure 2.2 pixel position

Lines are plotted by calculating intermediate positions along the line path between two specified endpoint positions.

Screen locations are referenced with integer values, so plotted positions may only approximate actual line positions between two specified endpoints which are known as **jaggies**. E.g.: position (10.48, 20.51) is referenced as (10, 21).

Figure 2.3 screen locations

Chapter 2 Scan-Conversion

Line Drawing Algorithms:

The slope-intercept equation of a straight line is:

$y = mx + b$

Where m = slope of line and b = y-intercept.

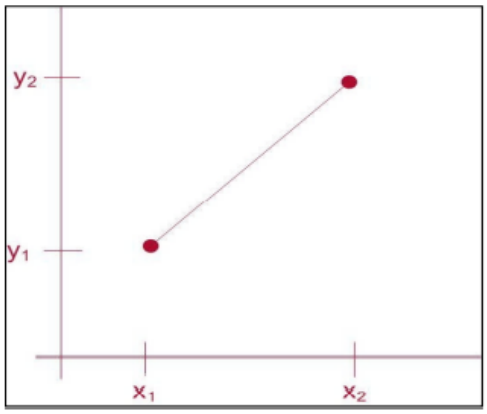


Figure 2.4 Straight line equation

For any two given points (x_1, y_1) and (x_2, y_2)

$$\text{slope (m)} = \frac{y_2 - y_1}{x_2 - x_1}$$

$$\therefore b = y - \frac{y_2 - y_1}{x_2 - x_1} x \quad \text{from above equation}$$

At any point (x_k, y_k)

$$y_k = mx_k + b \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 1$$

At (x_{k+1}, y_{k+1}),

$$y_{k+1} = mx_{k+1} + b \quad \ldots\ldots\ldots\ldots\ldots 2$$

subtracting 1 from 2 we get-

$$y_{k+1} - y_k = m(x_{k+1} - x_k)$$

Here $(y_{k+1} - y_k)$ is increment in y as corresponding increment in x.

$$\therefore \Delta y = m . \Delta x \quad \ldots\ldots\ldots\ldots\ldots 3$$

$$\text{or } m = \frac{\Delta y}{\Delta x} \quad \ldots\ldots\ldots\ldots\ldots\ldots 4$$

When the value of m is calculated using equation 4, test for three cases can be performed as

Case I: for |m| < 1,

Δx can be set proportional to unit horizontal deflection voltage & the corresponding vertical deflection is set proportional to Δy from equation 3.

Case II: for |m| > 1,

Δy can be set proportional to unit vertical deflection voltage & the corresponding horizontal deflection is set proportional to Δx from equation 3.

Case III: for |m| = 1,

Δx = Δy, the horizontal deflection voltage & the vertical deflection are equal.

Chapter 2 Scan-Conversion

Digital Differential Analyzer (DDA) Line Algorithm:

It is a scan conversion line algorithm based on calculating either Δx or Δy using equation m = Δy/Δx.

We sample the line at unit interval in one direction (x if Δx is greater than Δy otherwise in y direction) and determine corresponding integer values nearest the path for the other co-ordinate.

Cases:

For the line with the **positive slope**:

For the assumption that lines are to be processed form left endpoints(X_k, Y_k) to right endpoints (X_{k+1}, Y_{k+1}).

Case I: if **|m| ≤ 1,** we sample at unit x interval i.e. Δx = 1.

$$\mathbf{X_{k+1} = X_k + 1}$$

Then we compute each successive y-values, by setting

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

$$\mathbf{Y_{k+1} = Y_k + m}$$

The calculated y value must be rounded to the nearest integer

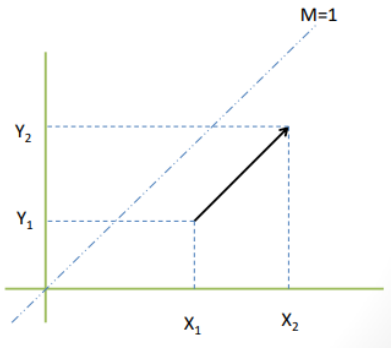


Figure 2.5 positive slope |m| ≤ 1 moving left to right

Case II: if **|m| > 1,** we sample at unit y interval i.e. Δy = 1.

$$\mathbf{Y_{K+1} = Y_K + 1}$$

Then we compute each successive y-values, by setting

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

$$\mathbf{X_{K+1} = X_K + 1/m}$$

The calculated x value must be rounded to the nearest integer.

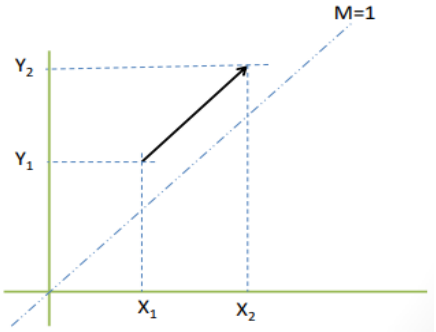


Figure 2.6 positive slope |m| > 1 moving left to right

For the assumption that lines are to be processed form right endpoints (X_k, Y_k) to left endpoints (X_{k+1}, Y_{k+1}).

Case I: if **|m| ≤ 1,** we sample at unit x interval i.e. Δx = -1.

$$\mathbf{X_{k+1} = X_k - 1}$$

Then we compute each successive y-values, by setting

$\Delta y = m * \Delta x$

$$Y_{k+1} = Y_k - m$$

The calculated y value must be rounded to the nearest integer

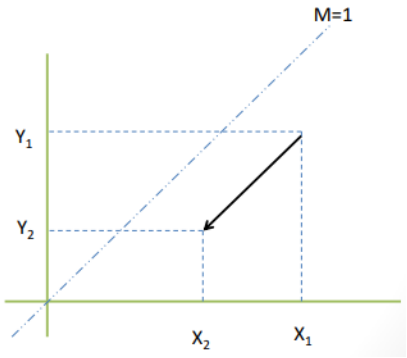


Figure 2.7 positive slope $|m| \leq 1$ moving right to left

Case II: if **$|m| > 1$,** we sample at unit y interval i.e. $\Delta y = -1$.

$$Y_{K+1} = Y_K - 1$$

Then we compute each successive y-values, by setting

$\Delta x = \Delta y / m$

$$X_{K+1} = X_K - 1/m$$

The calculated x value must be rounded to the nearest integer

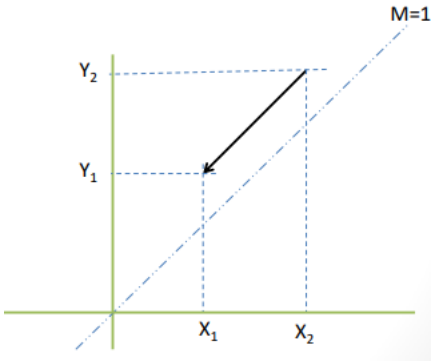


Figure 2.8 positive slope $|m| > 1$ moving left to right

Similarly for the line with the **negative slope:**

- **LEFT END TO RIGHT END POINT**

a. $|m| \leq 1$ → sampling in x-direction

$$\Delta x = 1$$
$$x_{k+1} = x_k + 1 \quad \because \Delta y = m\Delta x$$
$$y_{k+1} = y_k + m$$

b. $|m| > 1$ → Sampling in y- direction (Reverse the role of x and y)

$$\Delta y = -1$$
$$x_{k+1} = x_k - \frac{1}{m}$$
$$y_{k+1} = y_k - 1$$

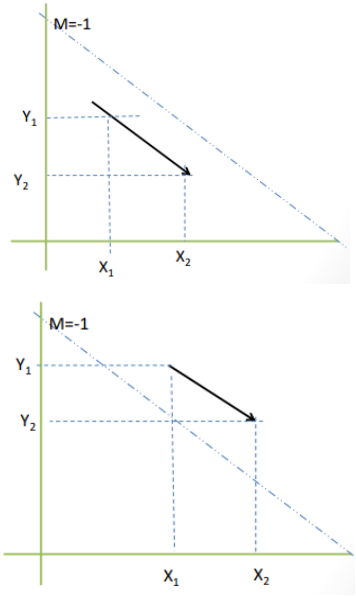


Figure 2.9 negative slope $|m| \leq 1$ & $|m| > 1$ moving left to right

- **RIGHT END TO LEFT END POINT**

a. |m |≤ 1 → sampling in x-direction

$$\Delta x = -1$$
$$x_{k+1} = x_k - 1$$
$$y_{k+1} = y_k - m$$

b. |m|>1→ Sampling in y- direction

$$\Delta y = 1 \quad \therefore \Delta y = m\Delta x$$
$$x_{k+1} = x_k + \frac{1}{m}$$
$$y_{k+1} = y_k + 1$$

Algorithm:

Consider one point of the line as (X1, Y1) and the second point of the line as (X2, Y2).

1. Start
2. Declare x_1, y_1, x_2, y_2, dx, dy, steps as integer variables and x, y, x_{inc}, y_{inc} as floating point.
3. Enter value of x_1, y_1, x_2, y_2.
4. Calculate dx = x2 –x1.
5. Calculate dy = y2 –y1
6. If absolute (dx) > absolute(dy)
   - Then steps = absolute (dx)
   - otherwise steps = absolute (dy)
7. Perform:
   - $x_{inc} = \frac{dx}{steps}$
   - $y_{inc} = \frac{dy}{steps}$
   - assign x = x1
   - assign y = y1
8. plot (x, y)
9. Do for k = 1 to steps times
   - x = x + x_{inc}
   - y = y + y_{inc}
   - plot (Round (x), Round (y))
10. stop

Chapter 2 Scan-Conversion

Numerical:

Q1. Digitize a Line with end point A(2,3) and B(6,8) , using DDA.

Answer.

Using DDA algorithm,

(x1, y1) = (2, 3)

(x2, y2) = (6, 8)

dx = x2 – x1 = 6 – 2 = 4

dy = y2 – y1 = 8 – 3 = 5

Since, dy > dx, therefore steps = dy =5

Here, Slope (M) = $\frac{(8-3)}{(6-2)}$ = 1.25 here Slope is positive and greater than 1 and moving left to right.

Xinc = dx/steps = 4/5 = 0.8

Yinc = dy/steps = 5/5 = 1

So, we use $x_{k+1} = x_k + \frac{1}{m}$

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

K	$X_{k+1} = X_k + 0.8$	$Y_{k+1} = Y_k + 1$	(x, y)
1	=2 + 0.8 = 2.8 ~ 3	4	(3, 4)
2	=2.8 + 0.8= 3.6 ~ 4	5	(4, 5)
3	=3.6 + 0.8 = 4.4 ~ 4	6	(4, 6)
4	= 4.4 + 0.8 = 5.2 ~ 5	7	(5, 7)
5	= 5.2 + 0.8 = 6	8	(6, 8)

Advantages:

- It is simple to implement.
- It is faster method than direct use of line equation y = mx + c. (Removes multiplication operation and only involve incremental operations of x or y direction).
- It requires no specific skills for implementation.

Disadvantages:

- A floating point addition is still needed in determining each successive point which is time consuming.
- The accumulation of round off error in successive addition of the floating point increments may cause the calculated pixel position to drift away from the true line path for long line segment this effect is known as **aliasing effect.**

Questions:

1. Digitize the line from A(1,1) to B(10,8) using scan conversion line algorithm.
2. Digitize the line with endpoints A(1,7) and B(6,3) using digital differential analyzer line drawing algorithm. Show all necessary steps.
3. Digitize the line with end points (1,2) and (5,6) using digital differential analyzer method.(TU).
4. Draw a line using DDA with two end points A(0,0) and B(-10,8).
5. Digitize line with start at (5,3) and end at (1,5) using DDA.

Bresenham's Line Algorithm:

The BLA is a more efficient method used to plot pixel position along a straight-line path.

Advantage of BLA over DDA

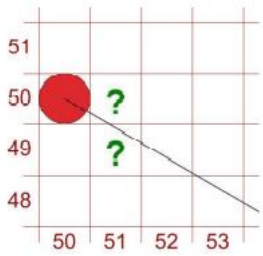
- In DDA algorithm each successive point is computed in floating point, so it requires more time and more memory space. While in BLA each successive point is calculated in integer value or whole number. So it requires less time and less memory space.
- In DDA, since the calculated point value is floating point number, it should be rounded at the end of calculation but in BLA it does not need to round, so there is no accumulation of rounding error.
- Due to rounding error, the line drawn by DDA algorithm is not accurate, while in BLA line is accurate.
- DDA algorithm cannot be used in other application except line drawing, but BLA can be implemented in other application such as circle, ellipse and other curves.

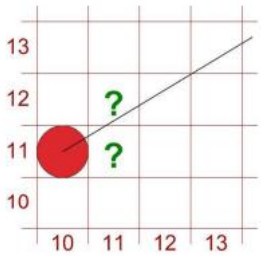
Its idea is to move across the x-axis in unit intervals and at each step choose between two y coordinates e.g. which pixel to draw?

(11, 11) or (11, 12)?

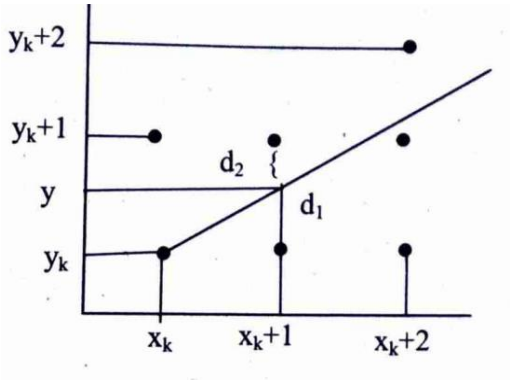
(51, 50) or (51, 49)?

The solution is to choose that which is closer to the original line.





BLA for slope positive and |m| ≤ 1



Pixel positions are determined by sampling at unit x intervals. Starting from left end point (x0, y0) step to each successive column (x samples) and plot the pixel whose scan line y value is closest to the line path.

Let us assume that pixel (x_k, y_k) is already plotted assuming that the sampling direction is along X-axis then the choice after (x_k, y_k) could be (x_k +1, y_k) or (x_k+1, y_k+1). Thus, the common equation of the line is

$$y = m(x_k + 1) + b$$

Then

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

And

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

Difference between separations

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

Let us define decision parameter as, $p_k = \Delta x(d_1 - d_2)$

Sign of p_k is same as that of d_1-d_2 for Δx>0 (left to right sampling)

$$p_k = \Delta x(d_1 - d_2) = 2\Delta y(x_k + 1) - 2\Delta x y_k + 2\Delta x b - \Delta x$$
$$= 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + \Delta x\,(2b - 1)$$
$$= 2\Delta y x_k - 2\Delta x y_k + 2\Delta y + \Delta x\,(2b - 1)$$
$$= 2\Delta y x_k - 2\Delta x y_k + c.$$

Where c=2Δy + Δx (2b-1), is a constant which is independent of pixel position

Case I:

If $p_k \geq 0$, then d2 < d1, which implies that y_k +1 is nearer than y_k. So, pixel at (y_k + 1) is better to choose which reduce error than pixel at y_k. This determines next co-ordinate to plot is (x_k +1, y_k +1).

Case II:

If $p_k < 0$, then d2 > d1, which implies that y_k is nearer than $y_k + 1$. So, pixel at y_k is better to choose which reduce error than pixel at $(y_k + 1)$. This determines next co-ordinate to plot is $(x_k + 1, y_k)$.

Now, similarly, pixel as $(x_k + 2)$ can be determined whether it is $(x_k + 2, y_k + 1)$ or $(x_k + 2, y_k + 2)$ by looking the sign of decision parameter p_{k+1} assuming pixel as (x_{k+1}) is known.

Now, K + 1th term

$$p_{k+1} = 2\Delta y.x_{k+1} - 2\Delta x.y_{k+1} + c$$

Here,

$$\begin{aligned} p_{k+1} - p_k &= 2\Delta y x_{k+1} - 2\Delta x y_{k+1} + c - (2\Delta y x_k - 2\Delta x y_k + c) \\ &= 2\Delta y(x_{k+1} - x_k) - 2\Delta x(y_{k+1} - y_k) \text{ where } x_{k+1} = x_k + 1 \\ &= 2\Delta y(x_k + 1 - x_k) - 2\Delta x\,(y_{k+1} - y_k) \\ &= 2\Delta y - 2\Delta x(y_{k+1} - y_k) \end{aligned}$$

This implies that decision parameter for the current column can be determined if the decision parameter of the last column is known.

Here,

If $p_k \geq 0$ (i.e. d2 < d1) which implies $y_{k+1} - y_k = 1$ that is, at $p_k \geq 0$, the pixel to be plotted is **$(x_k + 1, y_k + 1)$** and **$P_{k+1} = P_k + 2\,\Delta y - 2\Delta x$**

If $p_k < 0$ (i.e. d2 > d1) which implies $y_{k+1} - y_k = 0$ that is, at $p_k < 0$, the pixel to be plotted is **$(x_k + 1, y_k)$** and **$P_{k+1} = P_k + 2\,\Delta y$**

Initial Decision Parameter (p_0):

Let, $X_0 = 0$, $Y_0 = 0$

$$\begin{aligned} p_k &= 2\Delta y x_k - 2\Delta x y_k + 2\Delta y + \Delta x(2b - 1) \\ p_0 &= 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y + \Delta x(2b - 1) \\ \text{But } b &= y_0 - m x_0 = y_0 - \frac{\Delta y}{\Delta x} x_0 \\ &= 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y + 2\Delta x y_0 - 2\Delta y x_0 - \Delta x \\ &= 2\Delta y - \Delta x \end{aligned}$$

BLA Algorithm:

1. Input two points (x_1,y_1) and (x_2,y_2)
2. Compute $\Delta x = | x_2 - x_1 |$ **&** $\Delta y = |y_2 - y_1|$
3. If $(x_2 > x_1)$ lx=1 else lx= - 1
4. If $(y_2 > y_1)$ ly=1 else ly= - 1
5. Plot first point (x_1,y_1)
6. If $(\Delta x > \Delta y)$ **{** /* i.e. when **|m|<1** */

   → calculate $p_0 = 2\Delta y - \Delta x$

   → Starting at k =0 to Δx times , repeat

   If $p_k<0$ /* next point (x_k+1,y_k) */

   $x_{k+1} = x_k+lx$, $y_{k+1}= y_k$

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

   else /* next point $(x_k + 1, y_k+1)$ */

   $x_{k+1} = x_k+lx$, $y_{k+1}= y_k +ly$

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

}ENDIF

7. Else /* i.e. when **|m|>1** */

{

→ calculate $p_0 = 2\Delta x - \Delta y$

→ Starting at k =0 to Δy times , repeat

If $p_k<0$ /* next point (x_k,y_k+1) */

$x_{k+1} = x_k$

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

$P_{k+1} = p_k + 2\Delta x$

else /* next point $(x_k + 1, y_k+1)$ */

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

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

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

}

Chapter 2 Scan-Conversion

Numerical:

Q1. Digitize the line with end points (20, 10) and (30, 18) using BLA.

Solution:

Here, Starting point of line = (x1, y1) = (20, 10)

And Ending point of line = (x2, y2) = (30, 18)

Thus, slope of line, m = Δy / Δx = y2-y1 / x2-x1 = (18-10) / (30-20) = 8/10

As the given points, it is clear that the line is moving left to right with the positive slop

|m| = 0.8 < 1

Thus,

The initial decision parameter (P0) = 2Δy - Δx = 2*8 – 10 = 6

Since, for the Bresenham's Line drawing Algorithm of slope, |m| ≤ 1, we have

If $P_k < 0$ (i.e. d1-d2 is Negative)
then,
$X_{k+1} = X_k + 1$
$Y_{k+1} = Y_k$
$P_k = P_k + 2\,\Delta y$

If $P_k \geq 0$
then,
$X_{k+1} = X_k + 1$
$Y_{k+1} = Y_k + 1$
$P_k = P_k + 2\,\Delta y - 2\Delta x$

k	P_k	X_{k+1}	Y_{k+1}	(X_{k+1}, Y_{k+1})
0.	6	20+1 = 21	11	(21, 11)
1.	6 + 2*8 -2*10 = 2	21+1 = 22	12	(22, 12)
2.	2 + 2*8 -2*10 = -2	22+1 = 23	12	(23, 12)
3.	-2 + 2*8 = 14	23+1 = 24	13	(24, 13)
4.	14 + 2*8 -2*10 = 10	24+1 = 25	14	(25, 14)
5.	10 + 2*8 -2*10 = 6	25+1 = 26	15	(26, 15)
6.	6 + 2*8 -2*10 = 2	26+1 = 27	16	(27, 16)
7.	2 + 2*8 -2*10 = -2	27+1 = 28	16	(28, 16)
8.	-2 + 2*8 = 14	28+1 = 29	17	(29, 17)
9.	14 + 2*8 -2*10 = 10	29+1 = 30	18	(30, 18)

Q2. Digitize the given line endpoints (15, 15) and (10, 18) using Bresenham's line drawing algorithm.

Solution:

Here, $(X_1,Y_1)= (15,15)$ &
$(X_2,Y_2)= (10,18)$

$M= (18-15)/(10-15) = 3/-5$ –ve slope and $|M| < 1$

$\Delta x = |10-15| = 5$
$\Delta y = |18-15| = 3$

here,
Initial $(P_{0)} = 2\,\Delta y - \Delta x$
$= 2x3-5$
$= 1$

If $P_k <$
$X_{k+1} = X_k - 1$
$Y_{k+1} = Y_k$
$\mathbf{P_k = P_k + 2\,\Delta y}$

If $P_k \geq 0$
$X_{k+1} = X_k - 1$
$Y_{k+1} = Y_k + 1$
$\mathbf{P_k = P_k + 2\,\Delta y - 2\Delta x}$

(15, 15)

K	P_k	X_{k+1}	Y_{k+1}	(X_{k+1}, Y_{k+1})
0	1	14	16	(14,16)
1	= 1 + 2x3 - 2x5 =-3	13	16	(13,16)
2	= -3 + 2x3 = 3	12	17	(12,17)
3	= 3 + 2x3 – 2x5 = -1	11	17	(11,17)
4	= -1 + 2x3 = 5	10	18	**(10,18)**

Q3. Digitize line with endpoints (3, 10) and (6, 2) using Bresenham's Line Drawing Algorithm.

Solution:

Here, $(X_1,Y_1)= (6,2)$ &
$(X_2,Y_2)=(3,10)$

$M= (10-2)/(3-6) = 8/-3$ –ve slope and $|M| > 1$

$\Delta x = |3-6| = 3$
$\Delta y = |10-2| = 8$

here,
Initial $(P_{0)} = 2\,\Delta x - \Delta y$
$= 2x3-8$
$= -2$

If $P_k < 0$
$X_{k+1} = X_k$
$Y_{k+1} = Y_k + 1$
$\mathbf{P_k = P_k + 2\,\Delta x}$

If $P_k \geq 0$
$X_{k+1} = X_k - 1$
$Y_{k+1} = Y_k + 1$
$\mathbf{P_k = P_k + 2\Delta x - 2\,\Delta y}$

(6, 2)

K	P_K	X_{k+1}	Y_{k+1}	(X_{k+1}, Y_{k+1})
0	-2	6	3	(6,3)
1	=-2+2x3 = 4	5	4	(5,4)
2	=4+6-16=-6	5	5	(5,5)
3	=-6+6=0	4	7	(4,7)
4	=0+6-16=-10	4	8	(4,8)
5	=-10+6= -4	4	9	(4,9)
6	=-4 +6 =2	3	10	**(3,10)**

Q4. Digitize the given line endpoints (10, 10) and (20, 5) using Bresenham's line drawing algorithm.

Q5. Digitize the line with end points (15, 5) and (30, 10) using BLA.

Q6. Digitize line with endpoints (3, 10) and (6, 2) using Bresenham's Line Drawing Algorithm.

Circle:

A circle is defined as the set of points that are all at a given distance r from a center position (x_c, y_c). The equation of circle is:

$$(x-x_c)^2+(y-y_c)^2=r^2$$

The points along circumference could be calculated by stepping along x-axis:

$$y=y_c\pm\sqrt{r^2-(x_c-x)^2}$$

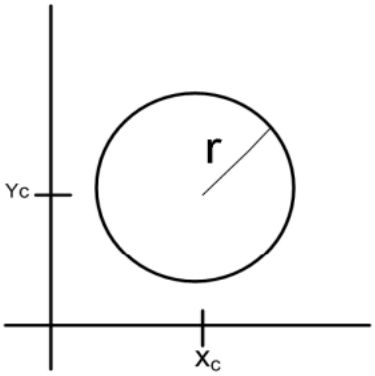


Figure 2.10 circle at center (x_c, y_c) and radius r

For the circle of center at origin and radius r:

$$y=\pm\sqrt{r^2-x^2}$$

Mid-Point Circle Algorithm:

In the mid-point circle algorithm we use eight-way symmetry so only ever calculate the points for the top right eighth of a circle, and then use symmetry to get the rest of the points.

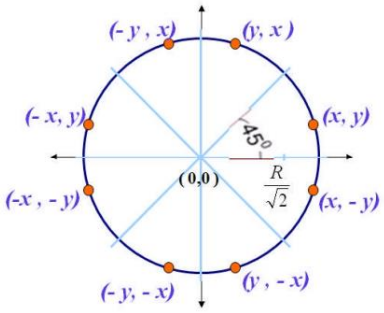


Figure 2.11 Eight way symmetry of circle

The equation of circle with center (0, 0) & radius (r) is,

$x^2 + y^2 = r^2$Any point (x, y) satisfies following conditions

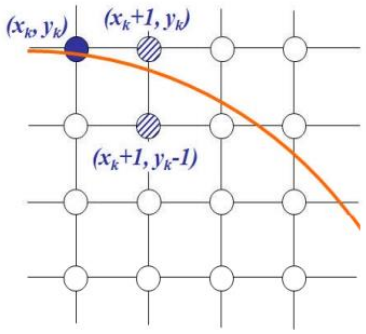
Circle function defined as:

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

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

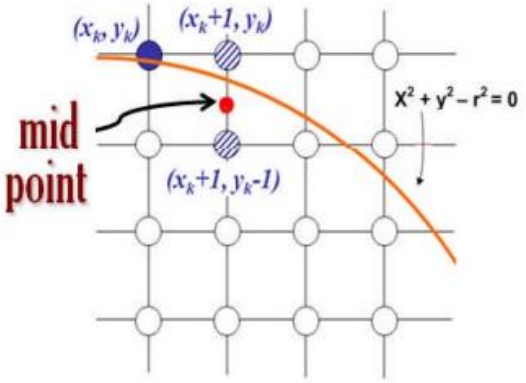
By evaluating this function at the midpoint between the candidate pixels we can make our decision

Assume that we have just plotted point (x_k, y_k)
The next point is a choice between (x_k+1, y_k) and (x_k+1, y_k-1)
We would like to choose the point that is nearest to the actual circle



Decision parameter is the function evaluated at the mid-point of these two points.

$$p_k = f_{circle}(x_k + 1, y_k - \frac{1}{2})$$
$$= (x_k + 1)^2 + (y_k - \frac{1}{2})^2 - r^2$$



Similarly,

$$p_{k+1} = f_{circle}(x_{k+1} + 1, y_{k+1} - \frac{1}{2})$$
$$= [(x_k + 1) + 1]^2 + (y_{k+1} - \frac{1}{2})^2 - r^2$$

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

$if\ p_k < 0$

$$y_{k+1} = y_k$$
$$P_{k+1} = P_k + 2x_{k+1} + 1$$

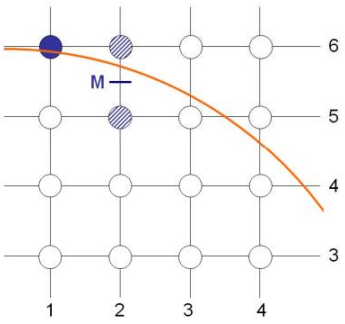


Figure 2.12 Mid-point lies inside circle

Else

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

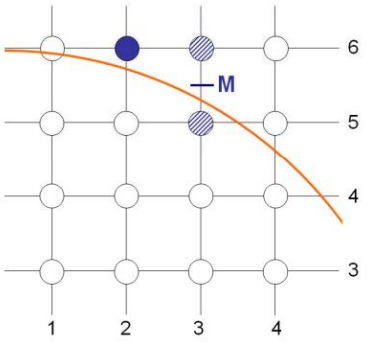


Figure 2.13 mid-point lies outside the circle

For the initial decision parameter, starting point is (0, r), then,

i.e.,
$(x_0,y_0) = (0,r)$

Here ,
$F_{circle}(1,r-1/2) = P_0$

Now,
$P_0 = 1^2+(r-1/2)^2 - r^2$
$= 1 + r^2 - r + 1/4 - r^2$
$= 5/4 - r$
$P_0 \approx 1-r$

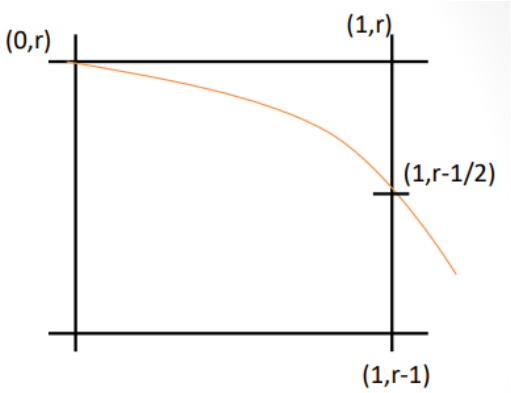
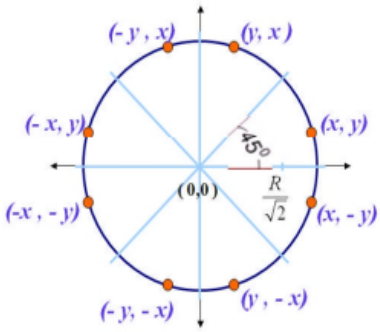


Figure 2.14 Circle of radius r and center at (0,0)

Algorithm:

1. Input radius r and circle center (x_c, y_c) and obtain the first point on the circumference of a circle centered on the origin as
   $(x_0,y_0) = (0,r)$
2. Calculate the initial value of the decision parameter as
   $P_0 = 5/4 - r$
3. At each x_k position, starting at k = 0, perform the following test:
   If $p_k < 0$ /* next point (x_k+1,y_K)*/
   $x_{k+1} = x_k + 1$
   $y_{k+1} = y_k$
   $P_{k+1} = p_k + 2x_{k+1} + 1$

   else /*next point $(x_k+1,y_K -1)$ */
   $x_{k+1} = x_k + 1$
   $y_{k+1} = y_k - 1$
   $P_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$
   Where $2x_{k+1} = 2x_k + 2$ and $2y_{k+1} = 2y_k - 2$



4. Determine the symmetry points in the other seven octants.
5. Move each calculated pixel position (x,y) onto the circular path centered on (x_c,y_c) and plot the co-ordinate values:
   $x = x + x_c$, $y = y+y_c$
6. Repeat steps 3 through 5 until $x \geq y$

Chapter 2 Scan-Conversion

Q1.

Draw a circle with radius $r = 8$, centering at (0,0). Initial point is

$$(x_0, y_0) = (0, 8)$$

Initial decision parameter is

$$p_0 = 1 - r = 7.$$

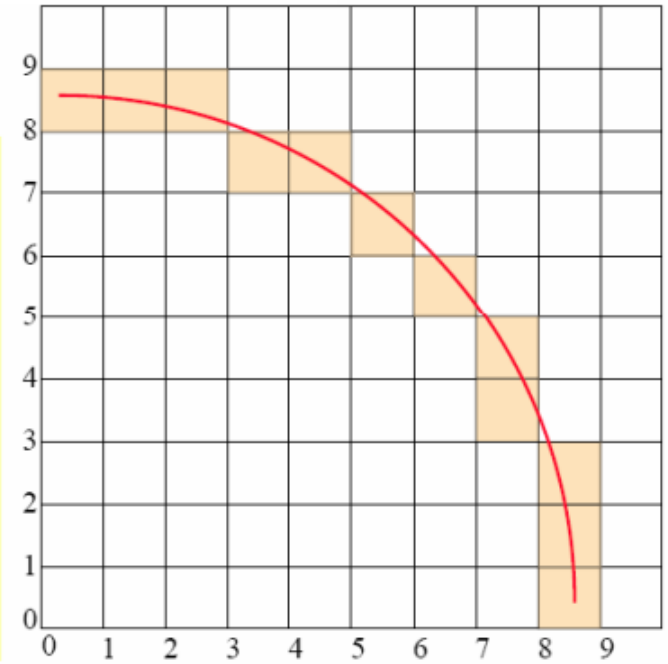
Successive pixels of the Midpoint circel are listed in the following table:

k	p_k	(x_{k+1}, y_{k+1})	$2x_{k+1}$	$2y_{k+1}$
0	-7	(1,8)	2	16
1	-4	(2,8)	4	16
2	1	(3,7)	6	14
3	-6	(4,7)	8	14
4	3	(5,6)	10	12
5	2	(6,5)	12	10

Successive pixels in a Midpoint circle.

PLOT

initial pixel → (0,8)
(1,8)
(2,8)
(3,7)
(4,7)
(5,6)
(6,5)
(7,4)
(7,3)
(8,2)
(8,1)
end pixel → (8,0)



Q2. Digitize a circle with radius 9 and center at (6, 7).

Here, the initial decision parameter (P0)

P0 =1 – r = 1-9 = - 8

Since, for the Midpoint Circle Algorithm of starting point (0, r) & centre at origin (0, 0) rotating at clockwise direction, we have

If P < 0

Plot $(x_k + 1, y_k)$

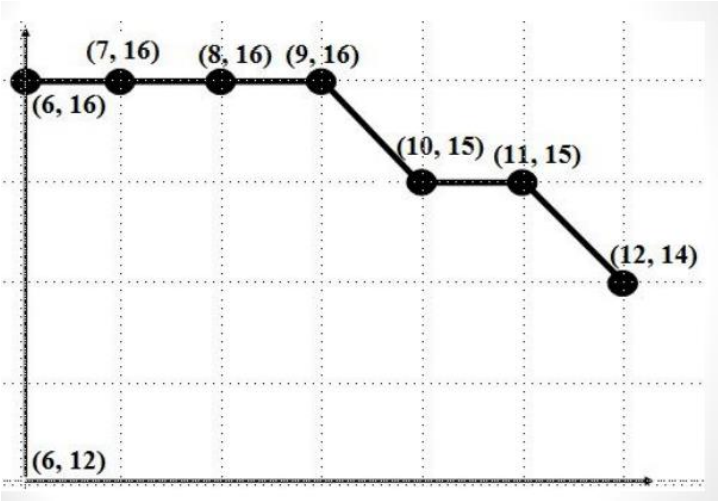
$P_{k+1} = P_k + 2x_{k+1} + 1$

Else (P ≥ 0)

Plot $(x_k + 1, y_k - 1)$

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

k	P_k	X_{k+1}	Y_{k+1}	(X_{k+1}, Y_{k+1}) At (0, 0)	(X_{k+1}, Y_{k+1}) At (6, 7)
0.	-8	0+1 = 1	9	(1, 9)	(1+6, 9+7) = (7, 16)
1.	= -8 + 2*1 +1= -5	1+1 = 2	9	(2, 9)	(2+6, 9+7) = (8, 16)
2.	= -5 + 2*2 +1= 0	2+1 = 3	8	(3, 8)	(3+6, 8+7) = (9, 15)
3.	= 0 + 2*3 - 2*8 +1=-9	3+1 = 4	8	(4, 8)	(4+6, 8+7) = (10,15)
4.	= -9 + 2*4 +1= 0	4+1 = 5	7	(5, 7)	(5+6, 8+7) = (11,15)
5.	= 0 + 2*5 - 2*7 +1=-3	5+1 = 6	7	(6, 7)	(6+6, 7+7) = (12,14)
6.	= -3 + 2*6 +1= 10	6+1 = 7	6	(7, 6)	(7+6, 6+7) = (13,13)



Q3. Digitize a circle with radius 12 and center at (6, 8).

Chapter 2 Scan-Conversion

Ellipse:

An ellipse is defined as the set of points such that the sum of the distances from two fixed point/ positions (foci) is same for all points.

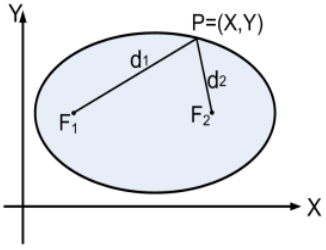
Elongated circle

Equation of ellipse:

$$d_1 + d_2 = constant$$

F1→(x_1, y_1), F2→(x_2, y_2)

$$\sqrt{(x-x_1)^2+(y-y_1)^2}+\sqrt{(x-x_2)^2+(y-y_2)^2} = constant$$

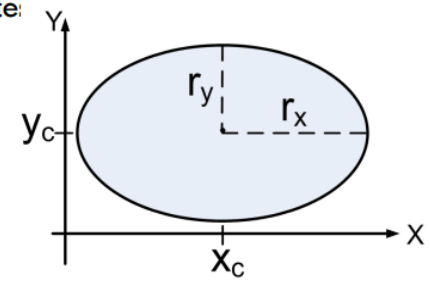


General Equation

$$Ax^2 + By^2 + Cxy + Dx + Ey + F = 0$$

In terms of ellipse center coordinate:

$$\left(\frac{x-x_c}{r_x}\right)^2 + \left(\frac{y-y_c}{r_y}\right)^2 = 1$$



In polar co-ordinate

$$x = x_c + r_x \cos\theta$$
$$y = y_c + r_y \sin\theta$$

Ellipse function is defined as:

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

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

From ellipse tangent slope:

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

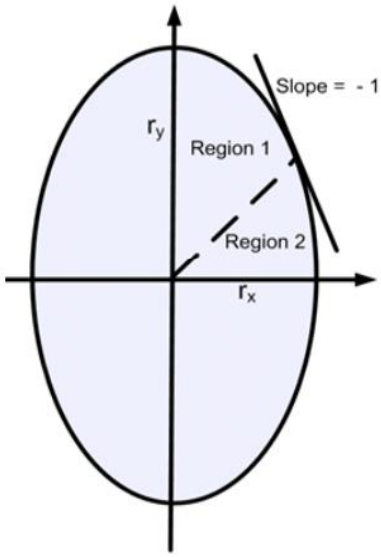
At boundary region (dy/dx= -1)

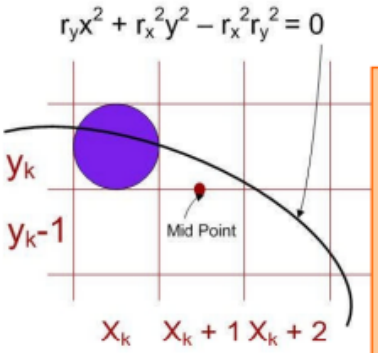
$$2r_y^2 x = 2r_x^2 y$$

Start from $(0, r_y)$, take x samples to boundary between 1 and 2

Switch to sample y from boundary between 1 and 2

(i.e whenever $2r_y^2 x \geq 2r_x^2 y$)





In the region 1

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

$$= r_y^2(x_k + 1)^2 + r_x^2(y_k - \frac{1}{2})^2 - r_x^2 r_y^2$$

For next sample

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

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

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

Thus increment

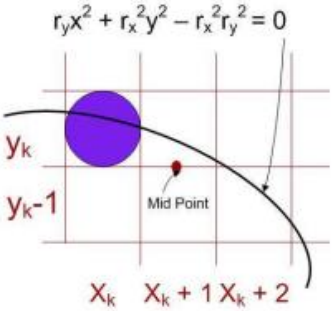
$$increment = \begin{cases} 2r_y^2 x_{k+1} + r_y^2, & if \quad p1_k < 0 \\ 2r_y^2 x_{k+1} + r_y^2 - 2r_x^2 y_{k+1}, & if \quad p1_k \geq 0 \end{cases}$$

- For increment calculation; Initially:

$$2r_y^2 x = 0$$

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

- Incrementally:
  Update x by adding $2r_y^2$ to first equation and update y by subtracting $2r_x^2$ to second equation

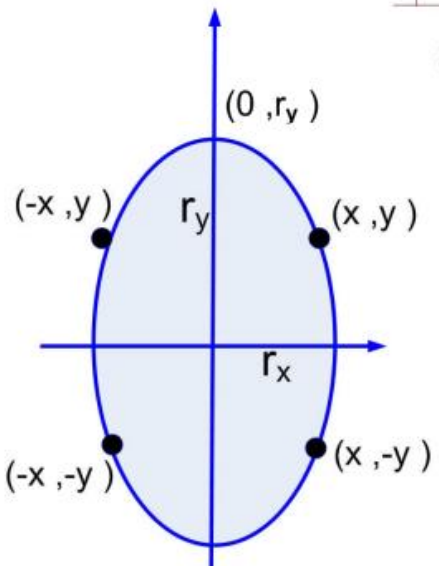


Initial value

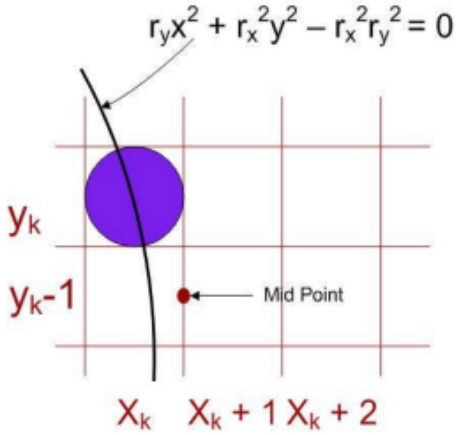
$$p1_0 = f_{ellipse}\left(1, r_y - \frac{1}{2}\right)$$

$$= r_y^2 + r_x^2\left(r_y - \frac{1}{2}\right)^2 - r_x^2 r_y^2$$

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



Chapter 2 Scan-Conversion



For simplification calculation of $p2_0$ can be done by selecting pixel positions in counter clockwise order starting at $(r_x,0)$ and unit samples to positive y direction until the boundary between two regions

In the region 2

$$p2_k = f_{ellipse}(x_k + \frac{1}{2}, y_k - 1)$$
$$= r_y^2 (x_k + \frac{1}{2})^2 + r_x^2 (y_k - 1)^2 - r_x^2 r_y^2$$

For next sample

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

Initially

$$p2_0 = f_{ellipse}\left(x_0 + \frac{1}{2}, y_0 - 1\right)$$
$$p2_0 = r_y^2 \left(x_0 + \frac{1}{2}\right)^2 + r_x^2 (y_0 - 1)^2 - r_x^2 r_y^2$$

Algorithm:

1. Input r_x,r_y, and the ellipse center (x_c, y_c) and obtain the first point on an ellipse centered on the origin as
   $(x_0,y_0) = (0,r_y)$
2. Calculate the initial value of the decision parameter in region 1 as

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

3. At each x_k position in region 1, starting at k = 0, perform the following test:
   If $p1_k < 0$ /* next point (x_{k+1},y_k)

$$x_{k+1} = x_k + 1,\ y_{k+1} = y_k$$
$$p1_{k+1} = p1_k + 2r_y^2 x_{k+1} + r_y^2$$

Else /* next point (x_k+1,y_k-1) */

$$x_{k+1} = x_k + 1,\ y_{k+1} = y_k - 1$$
$$p1_{k+1} = p1_k + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_y^2$$

With

$$2r_y^2 x_{k+1} = 2r_y^2 x_k + 2r_y^2, \qquad 2r_x^2 y_{k+1} = 2r_x^2 y_k - 2r_x^2$$

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

4. Calculate the initial value of decision parameter in region 2 using the last point (x_0,y_0) calculated in region 1 as

$$p2_0 = r_y^2\left(x_0+\frac{1}{2}\right)^2 + r_x^2(y_0-1)^2 - r_x^2 r_y^2$$

5. At each y_k position in region 2, starting at k = 0, perform the following test:

If $p2_k \le 0$ /* next point (x_k+1, y_k-1) */

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

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

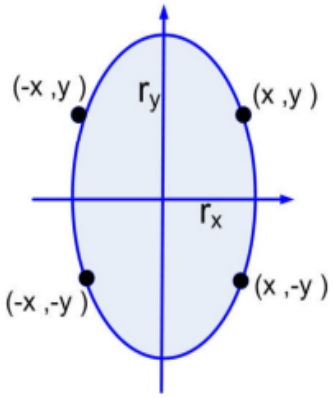
$$p2_{k+1} = p2_k + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_x^2$$

Else /*next point (x_k, y_k-1)*/

$x_{k+1} = x_k$

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

$$p2_{k+1} = p2_k - 2r_x^2 y_{k+1} + r_x^2$$



4. Using the same incremental calculations for x and y as in region 1. continue until y=0.
6. Determine the symmetry points in the other three quadrants.
7. Move each calculated pixel position (x,y) onto the elliptical path centered on (x_c,y_c) and plot the co-ordinate values:

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

Q1. Draw an ellipse with $r_x = 10$ and $r_y = 8$, centering at (0,0).

For region 1, the initial point for the ellipse centered on the origin is $(x_0, y_0) = (0, 8)$, and the initial decision parameter value is

$$p1_0 = r_y^2 - r_x^2 r_y + \frac{1}{4}r_x^2 = -711.$$

Successive midpoint decision-parameter values and the pixel positions along the ellipse are listed in the following table:

k	$p1_k$	(x_{k+1}, y_{k+1})	$2r_y^2 x_{k+1}$	$2r_x^2 y_{k+1}$
0	-711	(1,8)	128	1600
1	-519	(2,8)	256	1600
2	-199	(3,8)	384	1600
3	249	(4,7)	512	1400
4	-575	(5,7)	640	1400
5	129	(6,6)	768	1200
6	-239	(7,6)	896	1200
7	721	(8,5)	1024	1000

Table 3: Successive pixels in the region 1 of a Midpoint ellipse.

We now move out of region 1, since $2r_y^2 x > 2r_x^2 y$.

For region 2, the initial point is $(x_0, y_0) = (8, 5)$ and the initial decision parameter is

$$p2_0 = f_{\text{ellipse}}(8 + \frac{1}{2}, 4) = -704.$$

8

The remaining positions along the ellipse path in the first quadrant are then calculated as:

k	$p2_k$	(x_{k+1}, y_{k+1})	$2r_y^2 x_{k+1}$	$2r_x^2 y_{k+1}$
0	-704	(9,4)	1152	800
1	-252	(10,3)	1280	600
2	528	(10,2)	1280	400
3	228	(10,1)	1280	200
4	128	(10,0)	-	-

Successive pixels in the region 2 of a Midpoint ellipse.

Q2.

rx= 8 ry=6 and center (0,0)

For region 1: The initial point for the ellipse centered on the origin is $(x_0, y_0) = (0, 6)$, and the initial decision parameter value is

$$p1_0 = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2 = -332$$

Successive decision parameter values and positions along the ellipse path are calculated using the midpoint method as

k	$p1_k$	(x_{k+1}, y_{k+1})	$2r_y^2 x_{k+1}$	$2r_x^2 y_{k+1}$
0	-332	(1, 6)	72	768
1	-224	(2, 6)	144	768
2	-44	(3, 6)	216	768
3	208	(4, 5)	288	640
4	-108	(5, 5)	360	640
5	288	(6, 4)	432	512
6	244	(7, 3)	504	384

We now move out of region 1, since $2r_y^2 x > 2r_x^2 y$.

For region 2, the initial point is $(x_0, y_0) = (7, 3)$ and the initial decision parameter is

$$p2_0 = f\left(7 + \frac{1}{2}, 2\right) = \text{?}$$

Find Yourself

The remaining positions along the ellipse path in the first quadrant are then calculated as

k	$p2_k$	(x_{k+1}, y_{k+1})
0	?	(8, 2)
1	?	(8, 1)
2	?	(8, 0)

Filled Area Primitive:

If the boundary of some region is specified in a single color, we can fill the interior of this region, pixel by pixel, until the boundary color is encountered. This method, called the boundary-fill algorithm, is employed in interactive painting packages, where interior points are easily selected. This is a particularly useful technique for filling areas with irregular borders, such as a design created with a paint program.

Area Fill Algorithm

The algorithm makes the following assumptions one interior pixel is known, and pixels in boundary are known. Basically, a boundary-fill algorithm starts from an interior point (x, y) and sets the neighboring points to the desired color. This procedure continues until all pixels are processed up to the designated boundary for the area. There are two methods for processing neighboring pixels from a current point.

1. Four neighboring points:
   - These are the pixel positions that are right, left, above, and below the current pixel.
   - Areas filled by this method are called 4-connected.
2. Eight neighboring points:
   - This method is used to fill more complex figures.
   - Here the set of neighboring points to be set includes the four diagonal pixels, in addition to the four points in the first method.
   - Fill methods using this approach are called 8-connected.

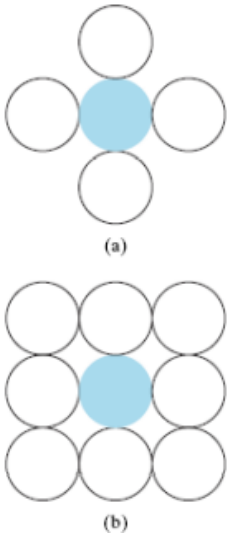


Figure 2. Fill methods applied to a 4-connected area (a) and to an connected area (b)

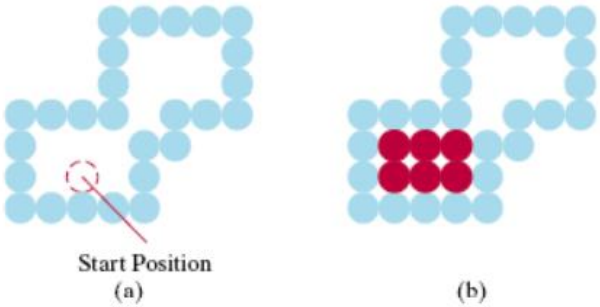


Figure 2. The area defined within the color boundary (a) is only partially filled in (b) using a 4-connected boundary-fill algorithm

The following procedure illustrates a recursive method for painting a 4-connected area with a solid color, specified in parameter **fillColor**, up to a boundary color specified with parameter **borderColor**.

We can extend this procedure to fill an 8- connected region by including four additional statements to test the diagonal positions ($x \pm 1, y \pm 1$).

```
void boundaryFill4 (int x, int y, int fillColor, int borderColor)
{
   int interiorColor;

   /* Set current color to fillColor, then perform following oprations. */
   getPixel (x, y, interiorColor);
   if ((interiorColor != borderColor) && (interiorColor != fillColor)) {
      setPixel (x, y);    // Set color of pixel to fillColor.
      boundaryFill4 (x + 1, y , fillColor, borderColor);
      boundaryFill4 (x - 1, y , fillColor, borderColor);
      boundaryFill4 (x , y + 1, fillColor, borderColor);
      boundaryFill4 (x , y - 1, fillColor, borderColor)
   }
}
```

Flood- fill Algorithm:

It is applicable when we want to fill an area that is not defined within a single color boundary. If fill area is bounded with different color, we can paint that area by replacing a specified interior color instead of searching of boundary color value. This approach is called flood fill algorithm.