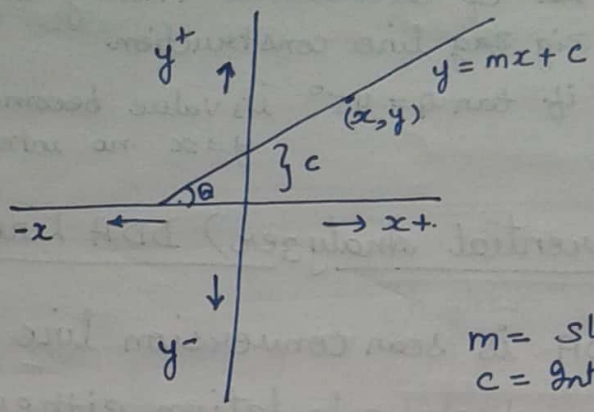
27.07.18



m = slope = tanθ.
c = Interception Yaxis

Line Equation is denoted by 'y = mx+c' where 'm' is the slope of line and 'c' is the intercept on the 'y' axis, such line passing through the point x,y. If line makes 45° angle on the x axis. It will pass through the origin. The equation of that line will be y = x.

Line Drawing Algorithm.

1. Direct Method:

① m = 1, c = 1

y = mx+c

x = 1 y = 1x1+1 = 2 (1,2)
x = 2 y = 1x2+1 = 3 (2,3)
x = 3 y = 1x3+1 = 4 (3,4)
⋮

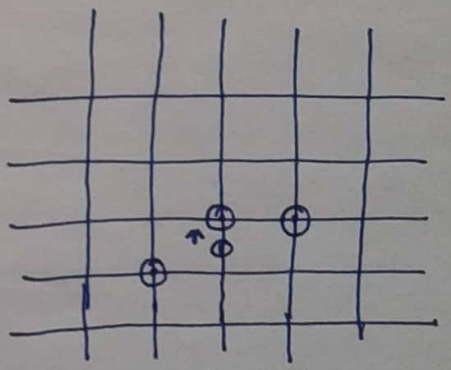


fig: ②

② if m = .5, c = ½

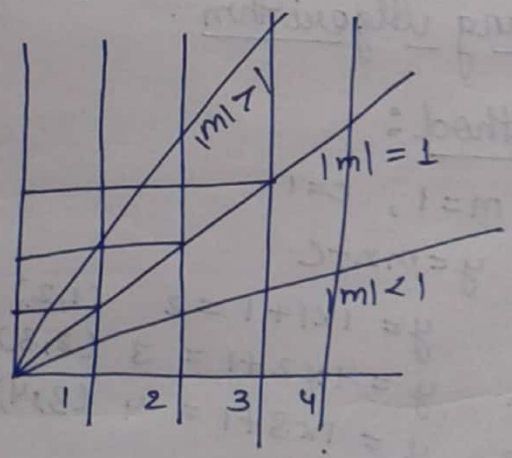
x = 1 y = .5x1 + ½ = 1 (1,1)
x = 2 y = .5x2 + ½ = (2,1.5) = (2,2).
x = 3 y = .5x3 + ½ = (3,2)
x = 4 y = .5x4 + ½ = (4,2.5) = (4,3)

Demerits - ① Floating numbers deviates location of pixel.
② Lot of calculations (Heavy Calculation)
③ zig zag line construction
if $\tan\theta = 45°$ its value becomes 1 then $y = x$ no interception.

28-07-2018

2. (Digital Differential Analyzer) DDA Line Drawing Algorithm

DDA is scan conversion line drawing algorithm, based on calculation either Δx or Δy. It generates a line using simultaneously incremental of Δx or Δy and calculate corresponding values of x and y. Such approach is characterized by performing calculation at each step using result from the preceding step. There may be following situations regarding the slope of line.



1. $|m| = 1$:

When the line pass through the origin and change in y will be equals to change in x.

2. $|m| < 1$:

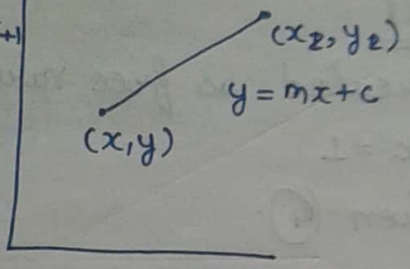
If this case, the change in x is higher than change in y, x would be treated as free variable and $\Delta x = 1$

Geometrical Aspects :

(Next Position) $K, K+1$

x_1 → Position of pixel

$x1$ → Increment of pixel position



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

$$y = \frac{\Delta y}{\Delta x} x + c$$

(ii) $|m| > 1$:

In this case the change in y is ~~happen~~ higher than the change in x, y would be treated as free variable and $\Delta y = 1$

If a line pass through a point (x, y), the equation of that line will be

$$y = mx + c \quad - (1)$$

if line pass through point (x_k, y_k)

$$y_k = mx_k + c \quad - (2)$$

Next position will (x_{k+1}, y_{k+1})

$$y_{k+1} = mx_{k+1} + c \quad - (3)$$

(3) & (2) Substracting (3) - (2).

$$y_{k+1} - y_k = (mx_{k+1} + c) - (mx_k + c)$$

$$y_{k+1} - y_k = mx_{k+1} + c - mx_k - c$$

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

$$\boxed{\Delta y = m \Delta x} \quad - (4).$$

Case 1: $|m| < 1$

x will be treated as free variable

$$\Delta x = 1$$

from equation (4).

$$\Delta y = m$$

$$\boxed{y_{k+1} - y_k = m} \rightarrow \quad y_{k+1} = y_k + m$$

Case 2: $|m| > 1$

y will be treated as free variable

$$\Delta y = 1$$

from equation (4)

$$1 = m \Delta x$$

$$\Delta x = \frac{1}{m}$$

$$x_{k+1} - x_k = \frac{1}{m}.$$

$$\boxed{x_{k+1} = x_k + \frac{1}{m}.}$$

01.08.2018

Tuesday

DDA Line Drawing Algorithm:

Algorithm:-

Step 1: Read line and points
(x_1, y_1) and (x_2, y_2)

Step 2: Calculate $dx = x_2 - x_1$, $dy = y_2 - y_1$

Step 3: if $abs(dy) > abs(dx)$ |∴ abs → absolute
then step = abs(dy)

otherwise
step = abs(dx)

Step 4: $x_{inc} = dx/step$, $y_{inc} = dy/step$ |∴ inc → Increment

Step 5: set pixel (round(x), round(y), color)

Step 6: $x = x + x_{inc}$, $y = y + y_{inc}$
setpixel (round(x), round(y), color)
Repeat step 6 until $x = x_2$, $y = y_2$

Numerical: Draw a line by using DDA Line Drawing Algorithm its ends point are (1,10) and (5,2)

$x_1 = 1$ $x_2 = 5$
$y_1 = 10$ $y_2 = 2$

$dx = 5-1 = 4$, $abs(dx) = 4$
$dy = 2-10 = -8$ $abs(dy) = 8$ (modulation)
Hence $abs(dy) < abs(dx)$ ∴ step 8

$x_{inc} = dx/step = 4/8 = .5$
$y_{inc} = dy/step = -8/8 = -1$

(x, y)		round
(1, 10)		(1, 10)
(1.5, 9)		(2, 9)
(2, 8)		(2, 8)
(2.5, 7)	→	(3, 7)
(3, 6)		(3, 6)
(3.5, 5)		(4, 5)
(4, 4)		(4, 4)
(4.5, 3)		(5, 3)
(5, 2)		(5, 2)

Advantage :

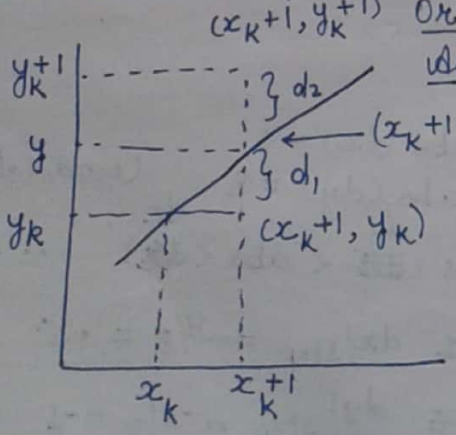
1. DDA Algorithm is a faster method for calculating pixel position than the direct method.
2. It eleminates the calculations upto some extent.

Disadvantages :

1. Rounding operation and floating point numbers still exist that actual line drawing is not possible.
2. The stair effect on the line can be seen in this algorithm

Bresenham's Line Drawing Algorithm.

Or Midpoint Line Drawing Algorithm



→ $d_1 > d_2$ → (x_k+1, y_k+1)

→ $d_1 < d_2$ → (x_k+1, y_k)

$p_k =$

Algorithm:

02-08-2018
Thursday

point (x_k+1, y) is on the line so it must be satisfied line equation $y = mx+b$.

$$y = m(x_k+1)+b \quad -①$$

$$d_1 = y - y_k$$

$$d_1 = [m(x_k+1)+b] - y_k \quad -②$$

$$d_2 = y_k+1 - y$$

$$d_2 = [y_k+1] - [m(x_k+1)+b] \quad -③$$

03-08-2018
Friday

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

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

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

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

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

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

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

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

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

p_k = decision parameter at k^{th} position

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

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

$$2\Delta y + \Delta x(2b-1) = K$$

K is the constant which is independent to the pixel position.

$$p_k = 2\Delta y x_k - 2\Delta x y_k + K \quad \text{——} \quad (4)$$

where $K = 2\Delta y + \Delta x(2b-1)$

at $k+1^{th}$ position equation will be

$$p_{k+1} = 2\Delta y x_{k+1} - 2\Delta x y_{k+1} + K \quad — \quad (5)$$

Subtracting (5) - (4) {changing equation sign}

$$p_{k+1} - p_k = 2\Delta y x_{k+1} - 2\Delta x y_{k+1} + K - 2\Delta y x_k + 2\Delta x y_k - K$$

$$= 2\Delta y (x_{k+1} - x_k) - 2\Delta x (y_{k+1} - y_k) \quad (6)$$

Case 1: $|m| < 1$

$\Delta x = 1$

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

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

~~where $p_k < 0$ and $d_1 < d_2$~~

if $p_k < 0$

position $(x_k + 1, y_k)$.

$y_{k+1} - y_k = 0$.

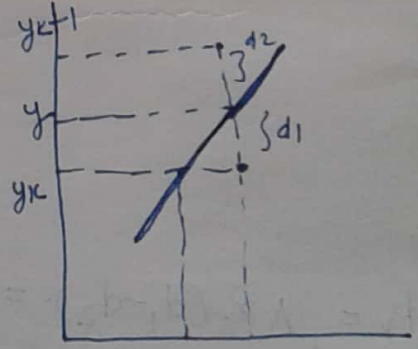
$$\boxed{p_{k+1} = p_k + 2\Delta y}$$

else

p_k ~~d1 d2~~

d_1 greater $p_k +$

d_2 greater $p_x -$



$p_k = d_1 - d_2$

$p_k < 0$ means $d_2 > d_1$

$p_k > 0$ means $d_1 > d_2$

position will be (x_{k+1} , y_{k+1})

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

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

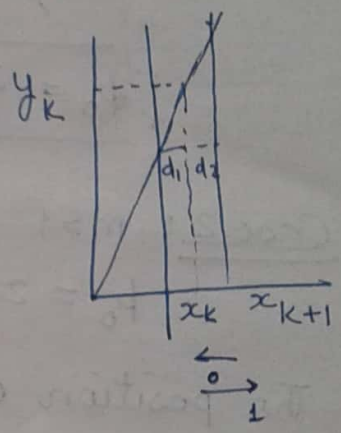
Case 2: $m > 1$

08-08-2018
Wednesday

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

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

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



If $p_k < 0$

the point will (x_k , y_{k+1})

$$x_{k+1} - x_k = 0$$

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

else

the point will be (x_{k+1} , y_{k+1})

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

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

First Decision Parameter at position (x_0 , y_0)

Case $m < 1$:-

From equation 4

$$p_k = 2\Delta y\, x_k - 2\Delta x\, y_k + K$$

$$p_0 = 2\Delta y\, x_0 - 2\Delta x\, y_0 + 2\Delta y + \Delta x(2b - 1)$$

$$y = mx + b$$

So. $b = (y_0 - mx)$

$$p_0 = 2\Delta y x_0 - 2x y_0 + 2\Delta y + \Delta x (2(y_0 - mx_0) - 1)$$

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

$$p_0 = 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y + \Delta x \left(2\left(y_0 - \frac{\Delta y}{\Delta x} \times x_0\right) - 1\right)$$

$$p_0 = 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y + \cancel{\Delta x} \left(\frac{2\Delta x y_0 - 2\Delta y x_0 - \Delta x}{\cancel{\Delta x}}\right)$$

$$p_0 = 2\cancel{\Delta y} x_0 - 2\cancel{\Delta x} y_0 + 2\Delta y + 2\cancel{\Delta x} y_0 - 2\cancel{\Delta y} x_0 - \Delta x$$

$$\boxed{p_0 = 2\Delta y - \Delta x}$$

Case 2: $m > 1$

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

The position of x and y will be changed and equation will be derived.

Algorithm:

Step 1: Input line and points (x_1, y_1) and (x_2, y_2).

Step 2: Calculate

$$\Delta y = y_2 - y_1$$

$$\Delta x = x_2 - x_1$$

Case 1: $m < 1$

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

Step 3: if $(p_k < 0)$

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

otherwise

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

Step 4: Repeat step 3 Δx times

Step 5: case 2 : $m > 1$

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

if $(p_k < 0)$

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

Otherwise

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

Step 6: Repeat step 5 Δy times.

09-08-2018
Thursday

Draw a line by using Breserham's Algorithm its two end point on (10,5) and (15,9).

$x_1 = 10$ $\quad x_2 = 15$

$y_1 = 5$ $\quad y_2 = 9$

$\Delta x = x_2 - x_1 = 15 - 10 = 5$

$\Delta y = y_2 - y_1 = 9 - 5 = 4$

$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{\Delta y}{\Delta x} = \frac{4}{5} = .8$

Here, $m < 1$

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

$= 2 \times 4 - 5$

$= 3$

Now, for getting next point, we calculate next decision parameter

$p_1 = p_0 + 2\Delta y - 2\Delta x$

$= 3 + 2 \times 4 - 2 \times 5$

$= 3 + 8 - 10$

$= 1$

Now, P_K is greater than 0, so applying the equation

$P<0$

(x_{K+1}, y_K)

else $P>0$

(x_{K+1}, y_{K+1}) ⇒ This equation will be used, because our else parts will become true, so coordinate become

(11, 6)

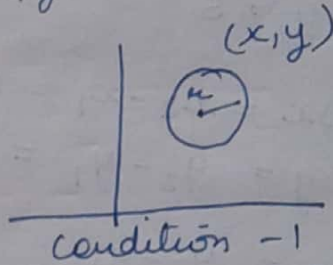
$$P_2 = P_1 + 2\Delta y - 2\Delta x$$
$$= 1 + 2\times 4 - 2\times 5$$
$$= 1 + 8 - 10$$
$$= -1$$

Circle Drawing Algorithm or Mid point Circle Drawing Algorithm

When a circle passing through a point (x, y) what will be the equation:

$$x^2 + y^2 = r^2$$

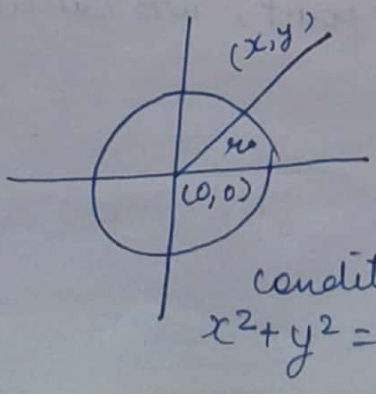
not satisfied



Explanation

When the circle passing through point (x, y) and center of the circle on the origin, radius of circle is r, the equation will be $x^2 + y^2 = r^2$.

satisfies.

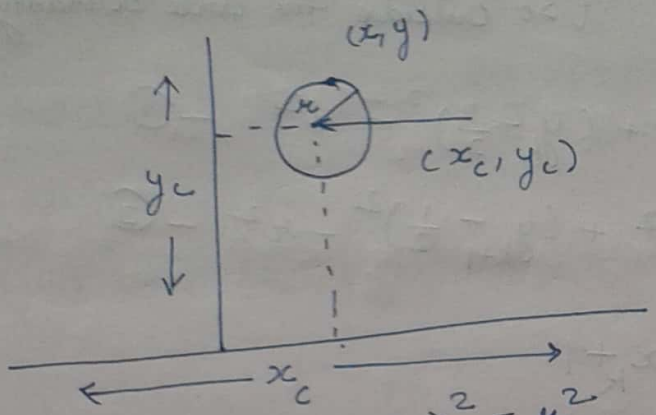


condition: 2

$x^2 + y^2 = r^2$

Explanation

When the circle moves from the origin and centre co-ordinates are x_c and y_c in that case, circle equation will be $(x-x_c)^2 + (y-y_c)^2 = r^2$



16.08.2018
Wednesday

Octant Symmetry of Circle

When we divide a circle in eight equal parts. This kind of symmetry is called Octant Symmetry of Circle. In Circle Drawing Algorithm, we generate the algorithm for single part and apply for remaining seven parts.

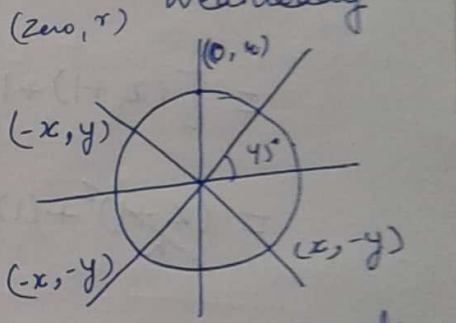
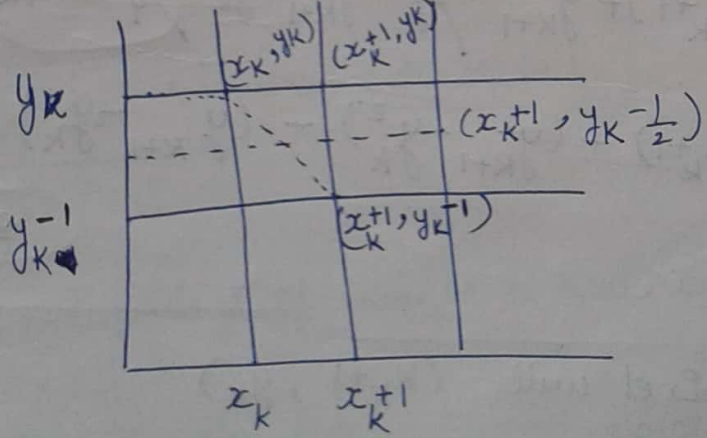


fig. octant Symmetry



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

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

$$p_k = (x_k+1)^2 + (y_k - \tfrac{1}{2})^2 - r^2 \quad -(1)$$

$$p_{k+1} = (x_{k+1})^2 + (y_{k+1} - \tfrac{1}{2})^2 - r^2 \quad -(2)$$

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

$$p_{k+1} = [(x_k+1)+1]^2 + (y_{k+1} - \tfrac{1}{2})^2 - r^2 \quad -(3)$$

$$= [(x_k+1)+1]^2 + (y_{k+1} - \tfrac{1}{2})^2 - [(x_k+1)^2 + (y_k - \tfrac{1}{2})^2 - p_k]$$

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

$$p_{k+1} = p_k + 2(x_k+1) + y_{k+1}^2 + \tfrac{1}{4} - 2 \times y_{k+1} \times \tfrac{1}{2} - [y_k^2 + \tfrac{1}{4} - 2 \times y_k \times \tfrac{1}{2}] + 1$$

$$p_{k+1} = p_k + 2(x_k+1) + y_{k+1}^2 + \tfrac{1}{4} - y_{k+1} - y_k^2 - \tfrac{1}{4} + y_k + 1$$

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

29-08-2018

if $p_k < 0$

the next pixel will (x_k+1, y_k)

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

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

$$\boxed{p_{k+1} = p_k + 2(x_k+1) + 1}$$

$$\boxed{\text{Or } p_{k+1} = p_k + 2x_{k+1} + 1}$$

else

next point will be (x_k+1, y_k-1)

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

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

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

$$= p_k + 2x_k + 2 + 1 - 2y_k + 2$$

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

$$\boxed{p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}}$$

First Decision Parameter at point $(0, r)$

$$(1, r - \tfrac{1}{2})$$

$$p_0 = (1)^2 + (r - \tfrac{1}{2})^2 - r^2$$

$$= 1 + r^2 + \tfrac{1}{4} - 2r \times \tfrac{1}{2} - r^2$$

$$\boxed{p_0 = \tfrac{5}{4} - r} \quad \text{or} \quad \boxed{p_0 = 1 - r}$$

Algorithm:

Step 1: Input radius of circle r and center point coordinates (x_c, y_c)

Step 2: Initial decision parameter at point $(0, r)$

$$p_0 = \tfrac{5}{4} - r$$

Step 3 : if ($p_k < 0$)

next position will be (x_{k+1}, y_k)

$$p_{k+1} = p_k + 2x_{k+1} + 1$$

else

$$p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$$

Step 4 : Determine Symmetry points in the seven other octants.

Step 5 : $x = x + x_c$

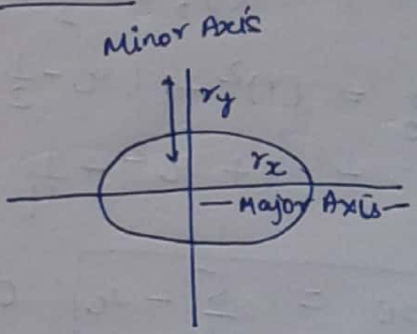
$y = y + y_c$

Step 6 : Repeat step 3 through 5 until $x \geq y$

30.08.2018
Thursday.

Ellipse Drawing Algorithm or Midpoint ellipse Drawing Algorithm.

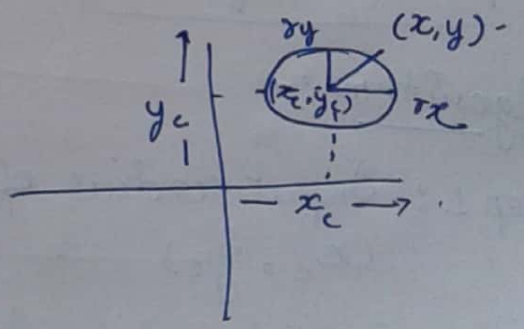
$$\frac{(x-x_c)^2}{r_x^2} + \frac{(y-y_c)^2}{r_y^2} = 1$$



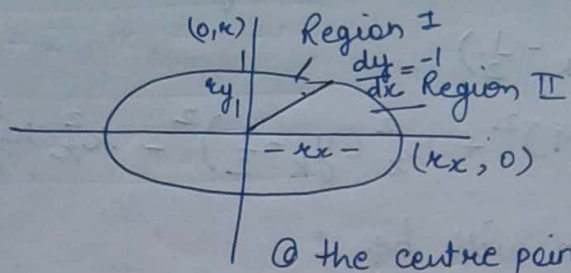
$$\frac{x^2}{r_x^2} + \frac{y^2}{r_y^2} = 1$$

or. $$x^2 r_y^2 + y^2 r_x^2 = r_x^2 r_y^2$$

after cross multiplication



Quadrant Symmetry of Ellipse



@ the centre point the value of

$$\frac{dy}{dx} = -1.$$

The condition on the center point of ellipse:

$$\frac{d}{dx}(r_y^2 x^2) + \frac{d}{dx}(r_x^2 y^2) = 0.$$

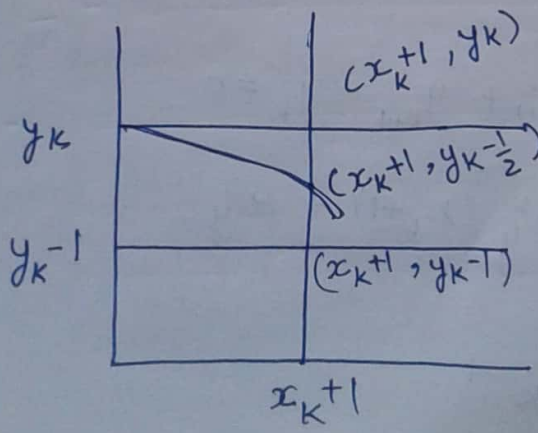
$$2x r_y^2 + 2 r_x^2 y \frac{dy}{dx} = 0.$$

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

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

$$+1 = + \frac{2x r_y^2}{2y r_x^2}$$

$$\boxed{x r_y^2 = y r_x^2}$$



ellipse equation will satisfy

$(x_k+1, y_k-\frac{1}{2})$

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

The next point will be :

$(x_{K+1}+1, y_{K+1}-\frac{1}{2})$

$$p_{1K+1} = r_y^2 ((x_K+1)+1)^2 + r_x^2 (y_{K+1}-\frac{1}{2})^2 - r_x^2 r_y^2$$

$$= r^2_y [(x_K+1)^2+1+2(x_K+1)] + r_x^2 (y_{k+1}-\frac{1}{2})^2 - r_x^2 r^2 y$$

$$p_{1K+1} = r^2_y [(x_K+1)^2+1+2(x_K+1)] + r_x^2 (y_{K+1}-\frac{1}{2})^2 - [r_y^2 (x_K+1)^2 + r_x^2 (y_{1K}-\frac{1}{2})^2 - p_{1K}]$$

$$p_{1K+1} = r_y^2 (x_K+1)^2 + r_y^2 + 2r_y^2 (x_k+1) + r_x^2 (y_{K+1}-\frac{1}{2})^2 - r_y^2 (x_K+1)^2 - r_x^2 (y_K-\frac{1}{2})^2 + p_{1K}$$

$$p_{1K+1} = p_K + 2r_y^2 (x_k+1) + r_y^2 + r_x^2 ((y_{K+1}-\frac{1}{2})^2 - (y_K-\frac{1}{2})^2)$$

27/09

if $(p_{1K} < 0)$ at this point $y_{K+1} - y_K = 0$

$$p_{1K+1} = p_{1K} + 2r_y^2 (x_k+1) + r_y^2$$

otherwise

$$y_{K+1} - y_K = 1$$

$$P_{1K+1} = P_{1K} + 2r^2y\,x_{K+1} - 2r^2x\,y_{K+1} + r^2y$$

First decision parameter in Region - I at point $(0, ry)$

$$\left(\frac{1}{x}, \frac{ry - \frac{1}{2}}{y}\right)$$

$$P_{10} = r^2y + r^2x\,(ry - \tfrac{1}{2})^2 - r^2x\,r^2y$$

$$P_{10} = r^2y - r^2x\,r^2y + \tfrac{1}{4}r^2x$$

In Region II

Point will be $\left(\frac{x_K + \frac{1}{2}}{x}, \frac{y_K - 1}{y}\right)$

$$P_{2K} = r^2y\,(x_K + \tfrac{1}{2})^2 + r^2x\,(y_K - 1)^2 - r^2x\,r^2y$$

Next point will be $\left(\frac{x_{K+1} + \frac{1}{2}}{x}, \frac{y_{K+1} - 1}{y}\right)$

$$P_{2\,K+1} = r^2y\,(x_{K+1} + \tfrac{1}{2})^2 + r^2x\,(y_{K+1} - 1)^2 + r^2x\,r^2y$$

$$P_{2\,k+1} = P_{2K} - 2r^2x\,(y_K - 1) + r^2x + r^2y\left[(x_K + \tfrac{1}{2})^2 - (x_K + \tfrac{1}{2})^2\right]$$

If $(P_{2K} > 0)$

$$\boxed{P_{2\,K+1} = P_{2K} - 2r^2x\,y_{K+1} + r^2x}$$

Otherwise

$$\boxed{P_{2K+1} = P_{2K} - 2r^2x\,y_{K+1} + r^2x + 2r^2y\,x_{K+1} + r^2x}$$

First decision parameter in Region - II

$$P_{20} = r^2y\,(x_0 + \tfrac{1}{2})^2 + r^2x\,(y_0 - 1)^2 - r^2x\,r^2y$$

Algorithm

Step 1: Input centre-point of ellipse that is (x_c, y_c) and radius of ellipse is r.

→ First point will be $(0, r_y)$

Step 2: Calculate first decision parameter in Region-I

$$p_{10} = r_y^2 - r_x^2 r_y^2 + \frac{1}{4} r_x^2$$

Step 3: Calculate

if $(p_{1k} < 0)$

$$p_{1k+1} = p_{1k} + 2r_y^2 (x_{1k}+1) + r_y^2$$

or

$$p_{1k+1} = p_{1k} + 2r_y^2 x_{k+1} + r_y^2$$

otherwise

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

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

Step 4: First decision parameter for Region II

$$p_{20} = r_y^2 (x_0 + \frac{1}{2})^2 + r_x^2 (y_0 - 1)^2 - r_x^2 r_y^2$$

Step 5: if $(p_{2k} > 0)$

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

otherwise

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

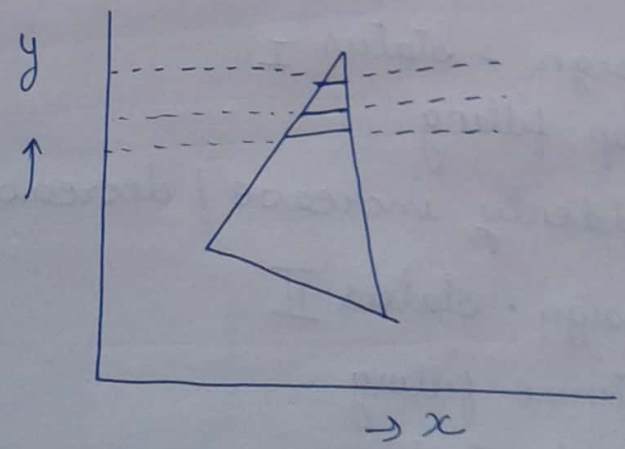
Step 6: Determine Symmetry for other three quadrant.
Step 7: $x = x + x_c$, $y = y + y_c$ until $2r^2 y x \geq 2r^2 x y$.

Filling Algorithm

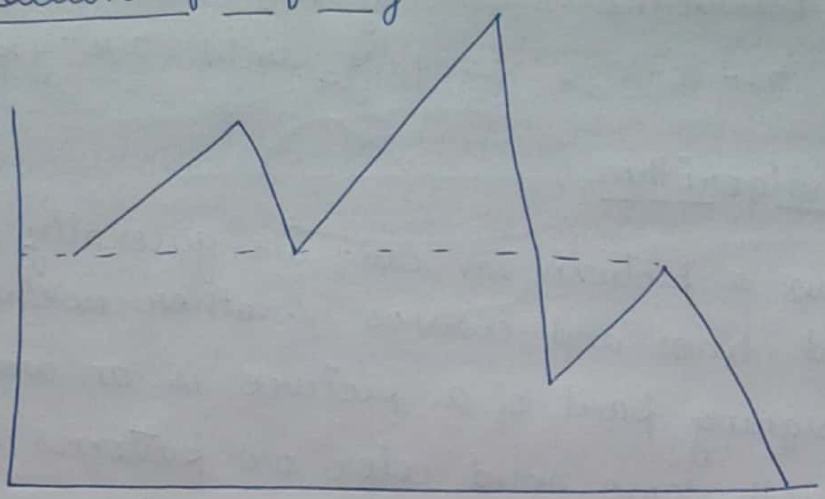
To draw a picture in CG, we generally use points straight lines and curves. Another useful construct for designing part of a picture is an area that is filled with same solid color or patterns the filling algorithm find interior part of algorithm the polygon and assigned a color or action. There are some methods which used to fill in the algorithm.

1. Scan line polygon filling Algorithm

This algorithm used for solid filling of a polygon when scan-line intersect the polygon H by ~~making~~ moving left to right it start the filling when again intersect with another H it stop the filling.



Another condition for filling



If the scan line passing through the vertex of intersect of two ellipse edges, we follow the following algorithm:

1. P when the scan line intersect with edge of polygon start the filling and is again intersect with edge of polygon it stop the filling.

2. If scan line passes through the intersect of two edges than we require following additional processing

i) Value of y monotonically increase / decrease

assign - status I

stop filling

ii) Value of suddenly increase / decrease

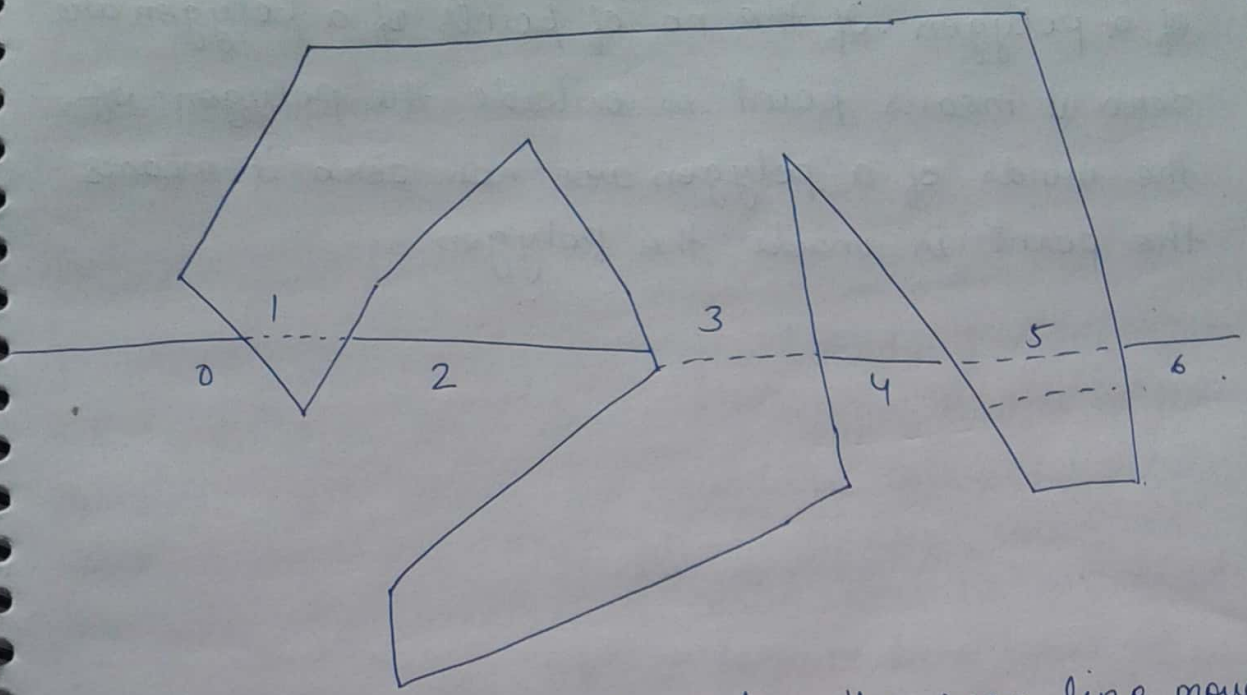
Assign. status II

Continue filling

(2) inside ≠ outside test

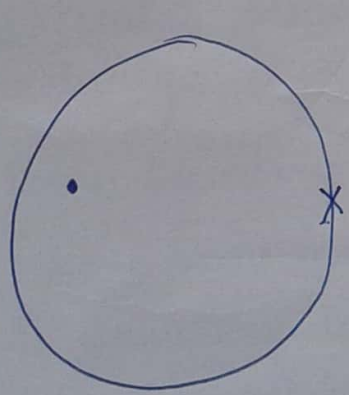
filling algorithm needs to identify intersect object region of an object.

i) odd - even Rule

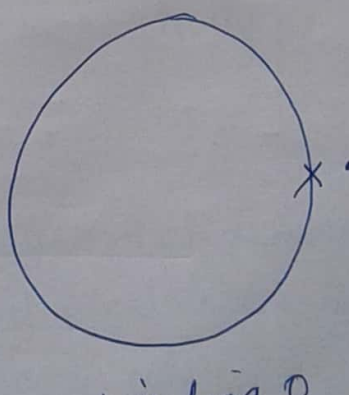


It is also called parity rule, the scan line move from left to right if the no. of intersection is odd it means scan ~~lf~~ line inside the polygon and if it even then scan line outside the polygon.

ii) Non-Zero Winding Rule



winding 1



winding 0

In this algorithm we count the number of winds of a polygon. If the no. of points of a polygon are zero it means point is outside the polygon. If the winds of a polygon are non-zero it means the point is inside the polygon.

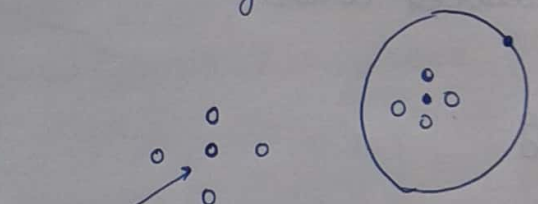
11·Oct·2018

Area Filling Algorithm

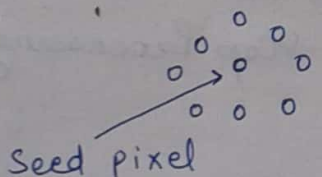
1. Boundary Fill Algorithm
2. Flood Fill Algorithm

Boundary Fill Algorithm

In this algorithm, we store the boundary color and take a point or pixel inside the polygon that is called the seed point, check the boundary color and pixel color is different, then fill the desired color in the pixel. This process will continue until we find the boundary color. We can follow two types of pattern for filling the polygon. One is the four point connectivity and second is the eight point connectivity



4 Pixel / Point connectivity



8 pixel / point Connectivity

* Short note

Step 1: Store boundary color

Step 2: Take a point inside polygon that is seed point

Step 3: If Boundary color and pixel color is different
Fill pixel by desired color
otherwise
Stop processing

2. Flood Fill Algorithm

In this algorithm we store the background of the polygon, Take a point inside the polygon i.e called seed point, if the background color and the pixel color is the same it means the pixel is unfilled and filled the desired color.
Use the four point connectivity or eight point connectivity to continue the process

Algorithm :

Step 1: Store background color

Step 2: Take a point inside the polygon that is seed point

Step 3: If background color and pixel color is same.
Fill pixel by desired color
otherwise
Stop Processing

Transformation

Matrix

Single Dimension

$\begin{bmatrix} 2 \\ 3 \\ 5 \end{bmatrix}_{3\times1}$ → column Matrix

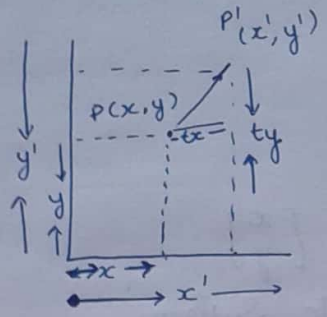
$\begin{bmatrix} 2 & 3 & 5 \end{bmatrix}_{1\times3}$ → row Matrix.

Two Dimension

$$\begin{bmatrix} 2 & 3 & 5 \\ 7 & 3 & 5 \\ 8 & 9 & 2 \end{bmatrix}_{3\times3}$$

1. Translation

Moving an object / pixel from one position / location to another that is called Translation.



P is the point having the coordinate x, y, it is translated to new position P' having the coordinate x', y'. tx is the translation factor on the x axis and ty is the translation factor on y axis.

$$x' = x + tx$$
$$y' = y + ty$$

$P' = P + T$ where $P' = \begin{bmatrix} x' \\ y' \end{bmatrix}$ $P = \begin{bmatrix} x \\ y \end{bmatrix}$ $T = \begin{bmatrix} tx \\ ty \end{bmatrix}$.

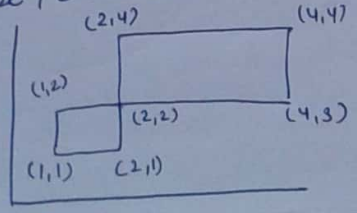
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} tx \\ ty \end{bmatrix}.$$

It can be expressed by Homogeneous coordinate System. * Question.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} x+tx \\ y+ty \\ 1 \end{bmatrix}$$ Product

2. Scaling

Increase / Decrease the size of an object is called scaling.



Initial points are (x, y) New points are (x', y')

S_x = Scaling factor on x Axis

S_y = Scaling factor on y Axis

$$x' = x \cdot S_x$$

$$y' = y \cdot S_y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Homogeneous Coordinate System

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$