Computer Graphics

To plot some pixels on a computer to draw an ~~impo~~ image

To plot some points on a graph to draw an image

Q. What is pixel?

Elementary part of system & also known as picture element.

→ Comp. graphics is an rendering tool ~~of~~ for generation of images & manipulation of images.

It deals with hardware & software supports for generating the images.

The ~~for~~ four major operations that can be performed in comp. graphics are —

(1) Imaging : Refers to representation of 2D images.

(2) Modeling : representation of 3D images

(3) Rendering : generation of 2D images from 3D images

(4) Animation : Simulation of seq. of images over the time

→ Resolution: Max. no. of pixels that can be displayed on comp. screen without overlapping.

→ Aspect ratio: No. of vertical pixels to the no. of horizontal pixels necessary to produce equal length lines in both horizontal & vertical directions on the screen.

→ Frame buffer: It is also known as a refresh buffer that holds the set of intensity value of all the screen points (pixels). The size of frame buffer is (resolution x (bits/pixel)).

→ Scanned conversion: It is the area of comp. graphics that is responsible for converting continuous figures into its discrete approximation.

Scanned conversion
- Random scan
- Roster scan

$y = mx + c$

① DDA (Digital Differential Analyzer)

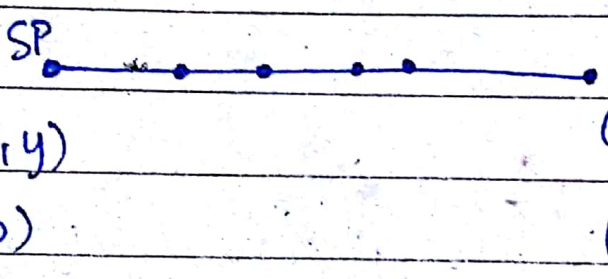
It is a mechanical device that solves differential equation by using numerical method.

It is a scanned conversion line drawing algo. based on two parameters Δx & Δy.

~~This~~

→ Screen consists of pixels

→ Every pixel is ~~sepe~~ is represented by ~~$y = mx + c$~~ (x, y).



It is used to find out all the intermediate pts.

$y = mx + c$

$$\text{slope} = \frac{y_2 - y_1}{x_2 - x_1} = \frac{y_{k+1} - y_k}{x_{k+1} - x_k}$$

~~In this slope = 0 grey~~

$(x_k, y_k) \rightarrow$ SP

$(x_{k+1}, y_{k+1}) \rightarrow$ EP

Case 1

$m < 1$

$x \rightarrow$ unit interval

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

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

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

y move with slope diff.

case 2

$m > 1$

$y \rightarrow$ unit interval

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

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

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

Case 3

$m = 1$

x & y both have unit interval

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

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

(1) Calculate slope, m

(2) If $m < 1$

x changes in unit interval & y moves with deviations

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

(3) If $m > 1$

x moves with deviations

y changes in unit interval

(4) If $m = 1$

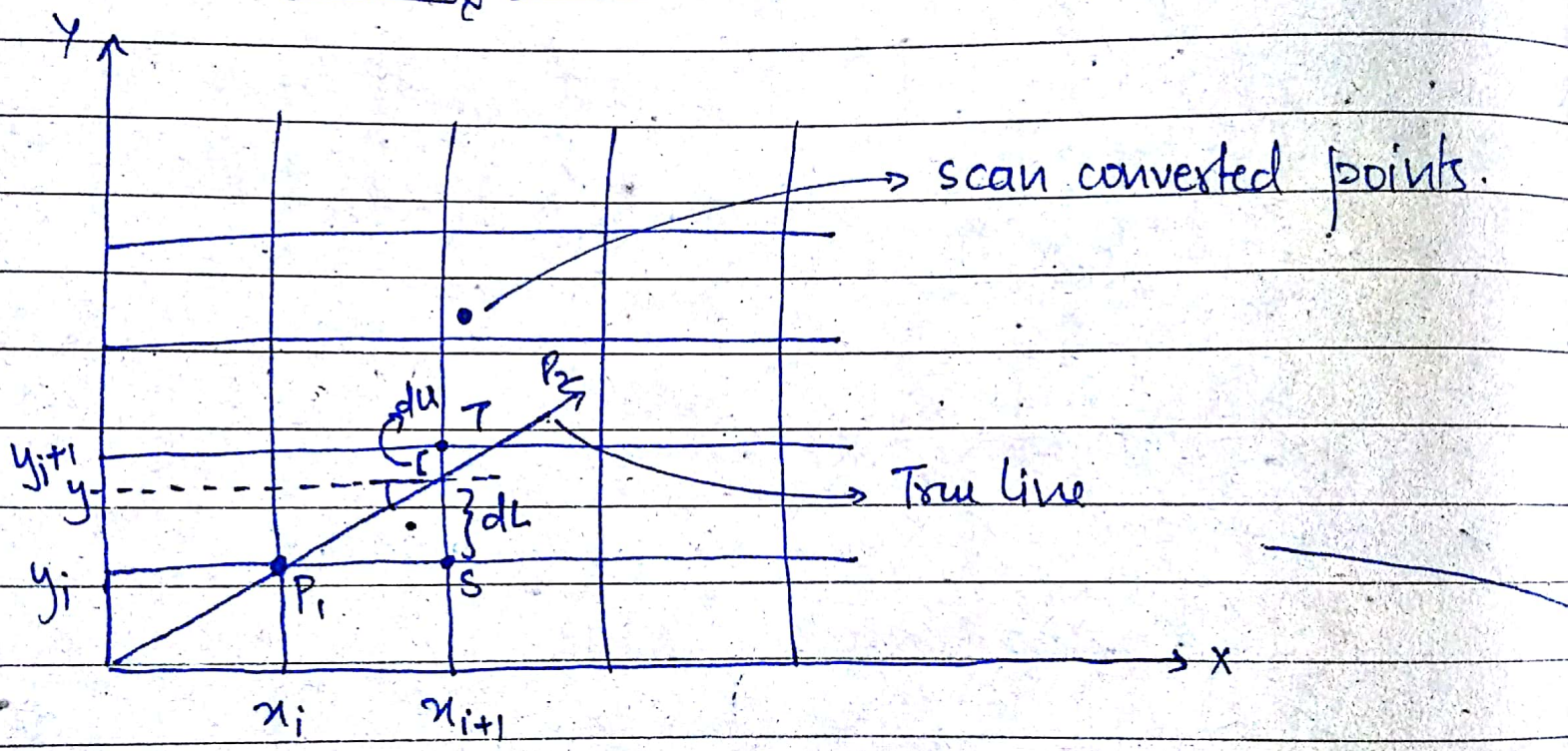
$(x_{k+1}, y_{k+1}) = \left(x_k + \frac{1}{m}, y_k + 1\right)$

(4) If $m = 1$

x & y both moves in unit intervals

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

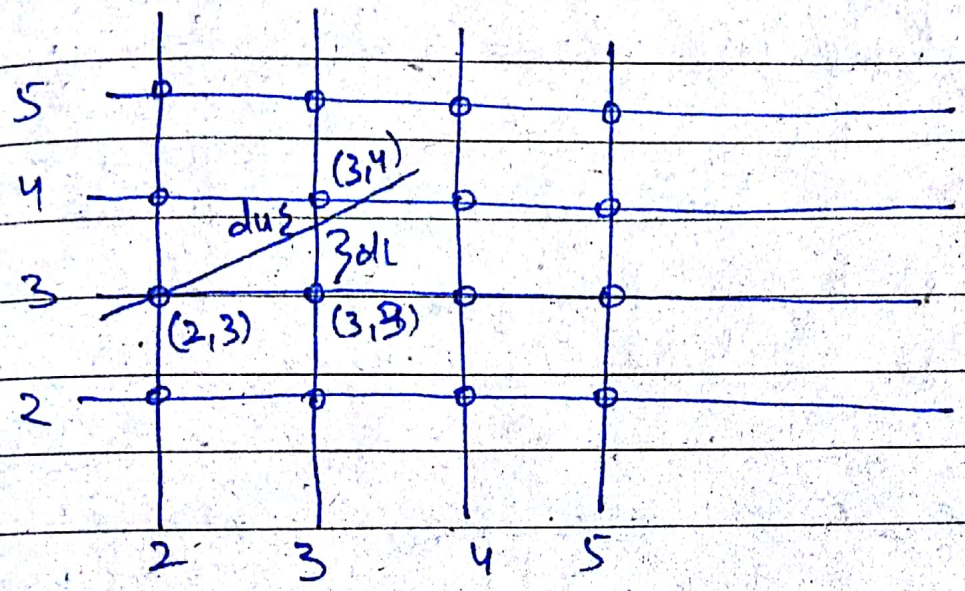
Bresenham's line drawing



This algorith is used to overcome the problem of ground func^n given in DDA algorithm, because it resists the computations.

It is an efficient & accurate rastur line algorithm developed by Bresenham's that scan converts the line using incremental integer calculations that can be adapted to display circles & other curves.

It is a highly efficient incremental method for scan converting lines.
It produces mathamatically accurate results using int. addition, sub. & multiplication by 2 which can be accomplished by simple shift operation.

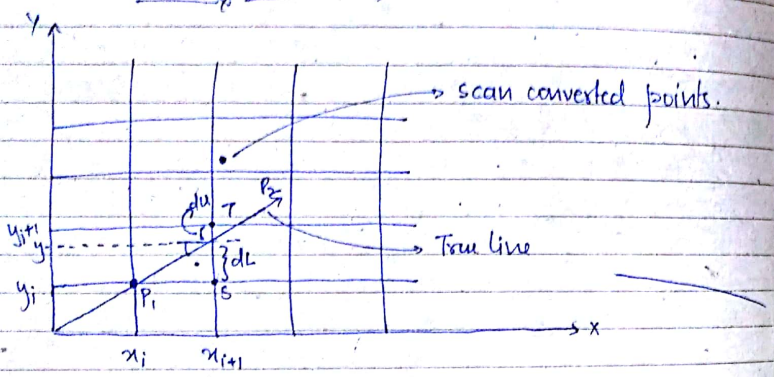


Starting pt. (2,3)
Second pt. is not a pixel so we take nearest pt. & it is (3,4)

Assume that we want to scan convert the line shown in figure where $0 < m < 1$
We start with pixel $P_1'(x_1', y_1')$ & then select the subsequent pixels as we work to the right one pixel posn at a time in horizontal dirn towards $P_2'(x_2', y_2')$

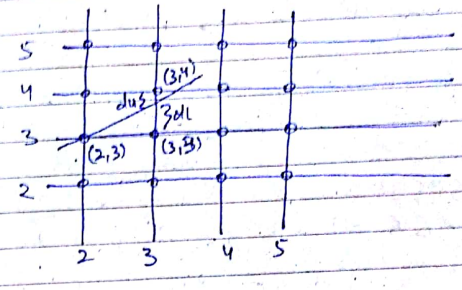
(4) If $m=1$
x & y both moves in unit intervals
$(x_{k+1}, y_{k+1}) = (x_k+1, y_k+1)$

Bresenham's line drawing



This algorith is used to overcome the problem of ground func^n given in DDA algorithm. Because it resists the computations.
It is an efficient & accurate rastur line algorithm developed by Bresenham's that scan converts the line using incremental integer calculations that can be adapted to display circles & other curves.

It is a highly efficient incremental method for scan converting lines.
It produces mathamatically accurate results using int. addition, sub. & multiplication by 2 which can be accomplished by simple shift operation.



Starting pt. (2,3)
Second pt. is not a pixel so we take nearest pt. & it is (3,4)

Assume that we want to scan convert the line shown in figure where $0<m<1$
We start with pixel $P_1'(x_1', y_1')$ & then select the subsequent pixels as we work to the right dis^n, one pixel pos^n at a time in horizontal dis^n towards $P_2'(x_2', y_2')$

Once a pixel is chosen at any step then next pixel is either one to its right (which constitutes of lower bound or upper bound) due to limit on slope (m). The line is best approximated by those pixels that falls on the least distance from its true path a b/w P_1' & P_2'. Using the notations of figure the coordinates of the last chosen pixel upon entering the value of 'i' are (x_i, y_i).

Now our task is to choose the next pixel b/w the bottom pixel (S) & the upper pixel (T).

If 'S' is chosen, we have $x_{i+1} = x_i + 1$

$$y_{i+1} = y_i$$

If 'T' is chosen, we have $x_{i+1} = x_i + 1$

$$y_{i+1} = y_i + 1$$

The actual y coordinate of the line at $x = x_{i+1}$

$$y = m x_{i+1} + b$$

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

~~Actual~~ The distance from S to actual line in y dir^n is

$$d_l = y - y_i$$

The distance from T to actual line in y dir^n $d_u = (y_{i+1} - y)$

Difference b/w these 2 distances value is —

$$d_L - d_U = d$$

when $d_L - d_U < 0$, we have $d_L < d_U$ $ closest pixel is (S). Similarly when $d_L - d_U > 0$, we have $d_L > d_U$ $ closest pixel is T.

If $d_L - d_U = 0$ then ~~close~~ T is chosen.

$$d_L - d_U = (y - y_i) - [(y_i + 1) - y]$$
$$= y - y_i - y_i - 1 + y$$
$$d_L - d_U = 2y - 2y_i - 1$$

But, $y = m(x_i + 1) + b$

$$d_L - d_U = 2(m(x_i + 1)) + 2b - 2y_i - 1$$

Substituting (m) by dy/dx $\Delta y/\Delta x$ & introducing a parameter decision

$$d_i = \Delta x (d_L - d_U)$$

$$d_L - d_U = \frac{2\Delta y}{\Delta x}(x_i + 1) + 2b - 2y_i - 1$$

$$\Delta x (d_L - d_U) = d_i = 2\Delta y (x_i + 1) + 2b\Delta x + 2y_i \Delta x - \Delta x$$

$$c = 2\Delta y + 2b\Delta x - \Delta x$$

$$\therefore d_i = 2\Delta y x_i - 2y_i \Delta x + c$$

Similarly, $d_{i+1} = 2\Delta y\, x_{i+1} - 2\Delta x\, y_{i+1} + c$

Difference b/w d_i & d_{i+1} — $x_{i+1} = x_i + 1$

$$d_{i+1} - d_i = 2\Delta y - 2\Delta x (y_{i+1} - y_i)$$

→ If the chosen pixel is top pixel i.e. T
i.e. $d_i \geq 0$
then $y_{i+1} = y_i + 1$

So $d_{i+1} = d_i + 2\Delta y - 2\Delta x$

→ If the chosen pixel is bottom pixel i.e. S
i.e. $d_i < 0$
then $y_{i+1} = y_i$
So, $d_{i+1} = d_i + 2\Delta y$

∴ Next coordinate
$(x_i + 1, y_i)$

⇒ d_1, the base value for the recursive formula from the original definition of d_i

$$d_i = \Delta x [2m(x_i + 1) + 2b - 2y_i - 1]$$

$$d_1 = \Delta x [2m(x_1+1) + 2b - 2y_1 - 1]$$
$$= \Delta x [2(mx_1 + m + b - y_1) - 1]$$

As $y_i = mx_i + b$

So $y_1 = mx_1 + b$

$$mx_1 + b - y_1 = 0$$

$$\therefore d_1 = \Delta x [2m - 1]$$
$$= 2m\Delta x - \Delta x$$
$$= 2\frac{\Delta y}{\Delta x} \cdot \Delta x - \Delta x$$

$$\boxed{d_1 = 2\Delta y - \Delta x}$$

Bresenham algorithm for scan converting of lines between $P_1'(x_1', y_1')$ to $P_2'(x_2', y_2')$ with

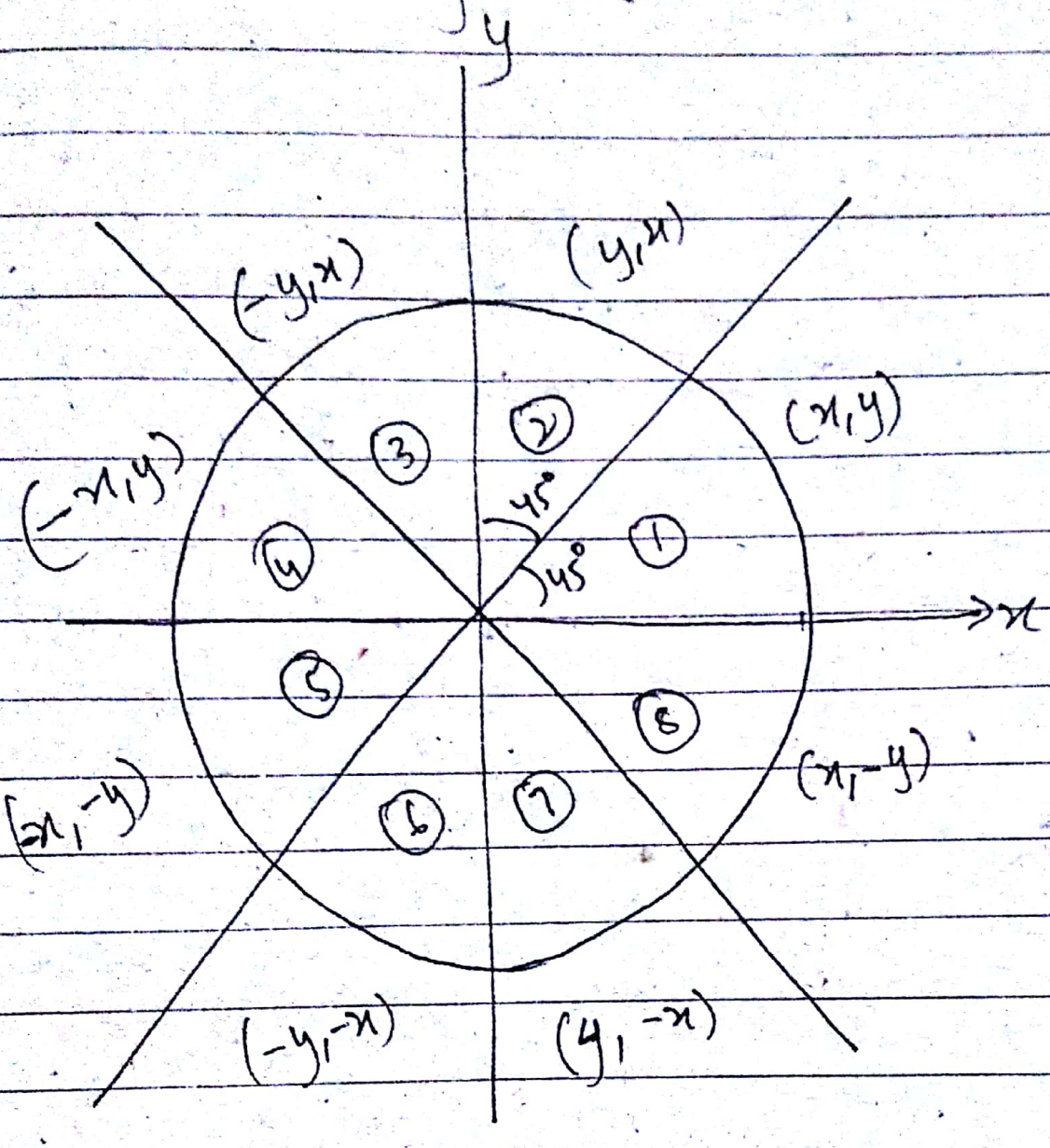
$x_1' < x_2'$ & $0 < m < 1$

```
int x = x1' , y = y1' ;
int dx = x2' - x1' ;
int dy = y2' - y1' ;

int dT = 2(dy - dx);
int dS = 2dy ;
int d = 2dy - dx ;
```

```
Set Pixel (x,y)
while( x < x2' )
  {
     x++;
     if (d<0)
       d = d+ds;
   else
     { y++;
       d = d + dT;
     }
    set Pixel(x,y);
  }
```

Scan Converting Circle



A circle is a symmetrical figure. Any circle drawing algo. can take advantage of circle symmetry to plot the 8 points for each value that algo. calculates.
8-way symmetry is used by reflecting each calculated point around 45° axis.

For eg. if pt. ① in figure is calculated by the circle algo then the 7 more pts. would be found by reflection. Reflection

$P_1 = (x, y)$ $P_2 = (y, x)$
$P_3 = (-y, x)$ $P_4 = (-x, y)$
$P_5 = (-x, -y)$ $P_6 = (-y, -x)$
$P_7 = (y, -x)$ $P_7 = (x, -y)$

Defining the circle

There are two standard methods of methematically defining a circle centered at origin.

→ The first method defines a circle with II order polynomial eqn.

Representation of circles

↳ Polynomial method
↳ Trignometric method

Polynomial Method

Polynomial eq^n is used to represent the circle.

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

↳ radius.

Polynomial eq^n's can be used to find out y coordinate for the known x-coordinate.

Therefore scan converting a circle using polynomial eq^n is achieved by stepping or moving 'x' from 0 to $\frac{r}{\sqrt{2}}$

$ each y coordinate is found by evaluating $\sqrt{r^2 - x^2}$ for each step of 'x'.

, This generates 1/8 portion of the circle. and remaining portion can be obtained just by reflection.

Trignometric method.

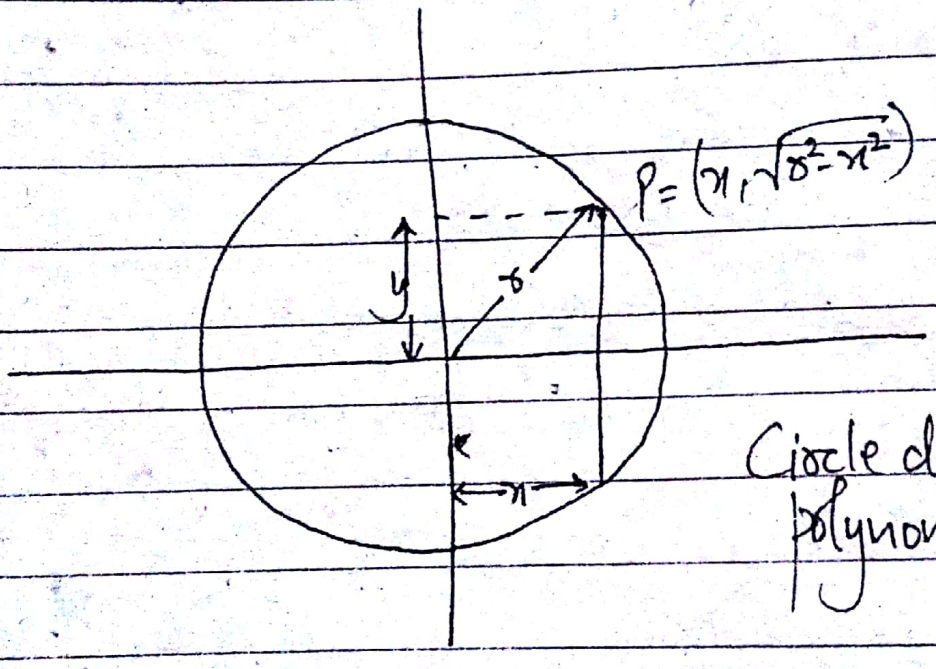
Circle is represented by the use of trignometric functions.

$$x = r\cos\theta$$

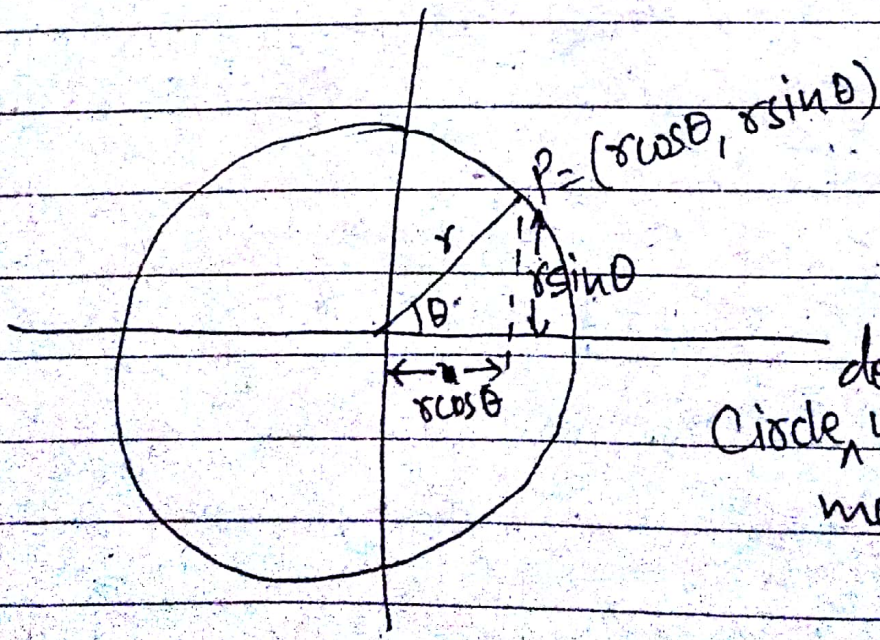
$$y = r\sin\theta$$

where θ = current angle.

Scan converting a circle using trigonometric method is achieved by stepping θ from 0 to $\pi/4$ radians. and each value of x & y is calculated.
This method is more inefficient than the polynomial method because the computation of values of $\sin\theta$ & $\cos\theta$ is even more time consuming than solving a polynomial.

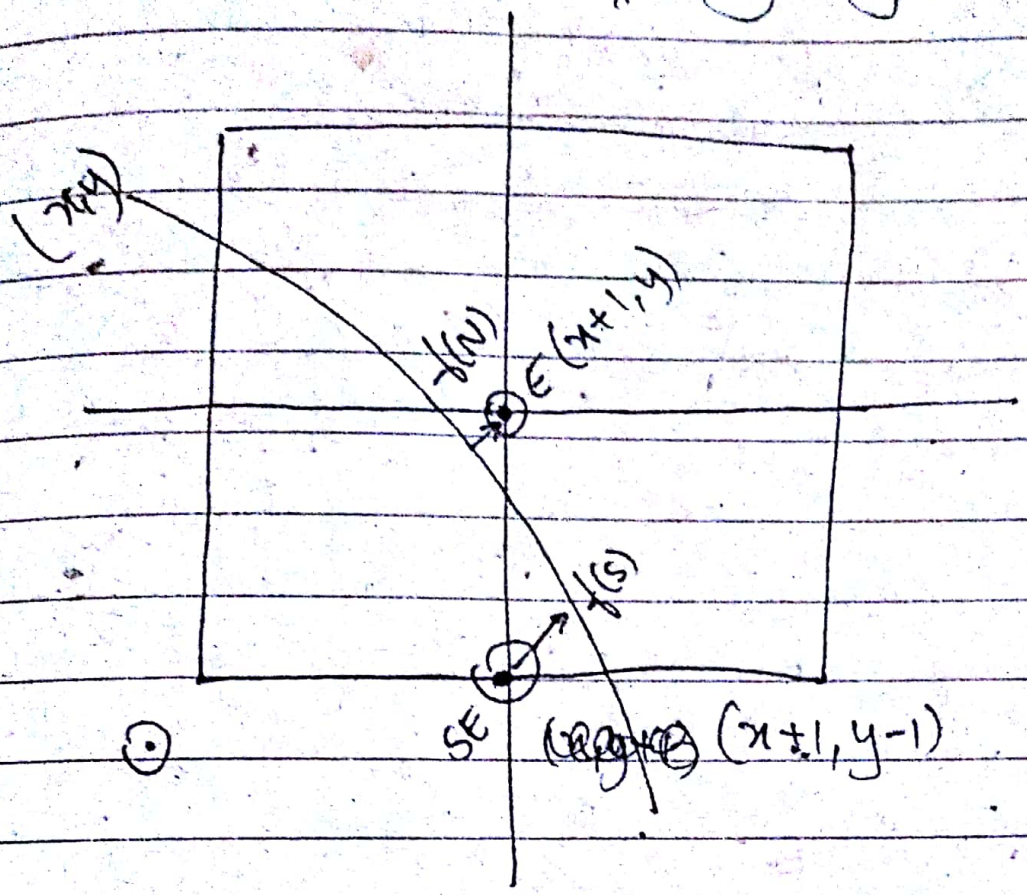


Circle defined using polynomial method



Circle defined using trigonometric method.

Bresenham's circle drawing algorithm



First calculate the error in f(N) & f(S) less error will be selected.

$$x^2+y^2=R^2$$

→ eqⁿ of circle centered at origin with radius R

$$f(x,y) = x^2+y^2-R^2$$

→ initial point is (x_i, y_i)

$$f(N) = (x_i+1, y_i)$$

$$f(S) = (x_i+1, y_i-1)$$

$$f(N) = (x_i + 1)^2 + (y_i)^2 - R^2$$
$$f(S) = (x_i + 1)^2 + (y_i - 1)^2 - R^2$$

Decision Parameter

$$\boxed{d_i = f(N) + f(S)}$$

We have a convention that if a pt. is inside the circle then it's distance from the circle is always negative.

If a point is outside the circle then it's distance from the circle is always +ve.

$d_i = f(N) + f(S)$

(East) (SE)

+ve −ve

If $d \leq 0$ → East point is selected.

$x_{i+1} = x_i + 1$ } f(S) is dominating

$y_{i+1} = y_i$ }

If $d > 0$ → South East point is selected

$x_{i+1} = x_i + 1$ } f(N) is dominating

$y_{i+1} = y_i - 1$ }

We know,

$$f(N) = (x_i+1)^2 + (y_i)^2 - R^2$$

$$f(S) = (x_i+1)^2 + (y_i-1)^2 - R^2$$

$$d_i = f(N) + f(S)$$

$$\boxed{d_i = 2(x_i+1)^2 + (y_i)^2 + (y_i-1)^2 - 2R^2} \quad \text{---------- (1)}$$

$i \rightarrow i+1$

$$d_{i+1} = 2(x_{i+1}+1)^2 + (y_{i+1})^2 + (y_{i+1}-1)^2 - 2R^2$$

$$d_{i+1} - d_i = 2\left((x_i+2)^2 - (x_i+1)^2\right) + (y_{i+1})^2 - (y_i)^2 + \left((y_{i+1}-1)^2 - (y_i-1)^2\right)$$

$$d_{i+1} = d_i + 2(2x_i+3) + \left((y_{i+1}+y_i)(y_{i+1}-y_i)\right) + \left((y_{i+1}-1+y_i-1)(y_{i+1}-1-y_i+1)\right)$$

$$d_i = f(N) + f(S)$$

$$d_i = 2(x_i+1)^2 + y_i^2 + (y_i-1)^2 + 2R^2$$

$$\boxed{d_{i+1} = 2(x_{i+1}+1)^2 + (y_{i+1})^2 + (y_{i+1}-1)^2 - 2R^2}$$

$$d_{i+1} - d_i = 2\left[(x_{i+1}-1)^2 - (x_i+1)^2\right] + (y_{i+1}^2 - y_i^2) + \left[(y_{i+1}-1)^2 - (y_i-1)^2\right]$$

Case 1 : $d \le 0$

$$x_{i+1} = x_i + 1$$
$$y_{i+1} = y_i$$

$$d_{i+1} - d_i = 2\left[(x_i+2)^2 - (x_i+1)^2\right] + 0 + 0$$
$$\Delta d_E = 2[2x_i + 3]$$

Case 2 : $d > 0$

$$x_{i+1} = x_i + 1$$
$$y_{i+1} = y_i - 1$$
$$\Delta d_{SE} = 4(x_i - y_i) + 10$$

d_{start} / d_{first}

$$\boxed{\begin{matrix} x = 0 \\ y = R \end{matrix}}$$

$$d_i = d_0 = 2(x_i+1)^2 + y_i^2 + (y_i-1)^2 - 2R^2$$
$$= 2 + R^2 + R^2 + 1 - 2R - 2R^2$$
$$\boxed{d_0 = 3 - 2R}$$

decision parameter