13.01.19

Ref. Book :

Computer Graphics (C version) 4th edition

- Donald Hearn, M. Pauline Baker.

——x——

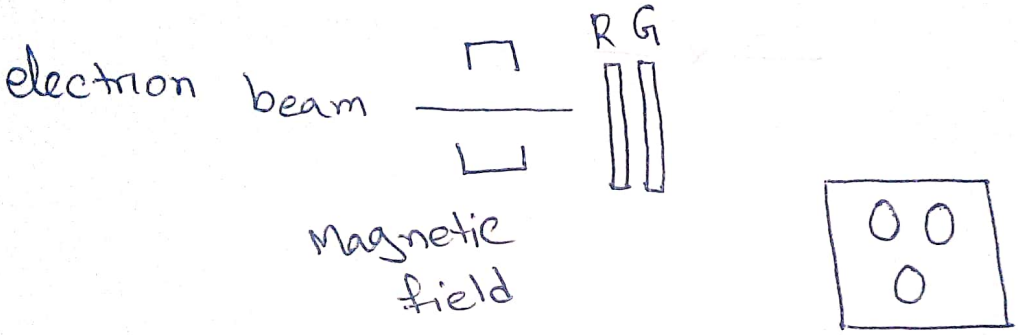
14.01.19

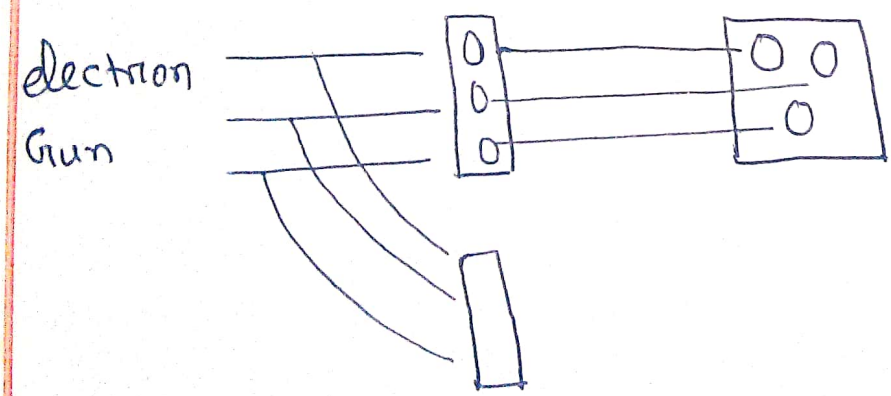
CRT

Interlaced (i)

Random Scan Displays

Color CRT:





LCD:

3D:

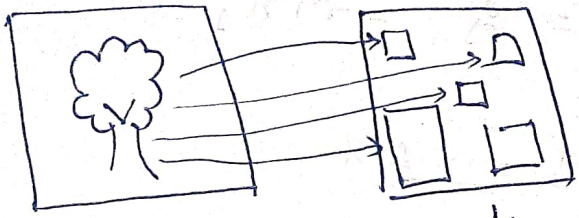
Creates Different image in different eyes.

—x—

16.01.19.

2.4

* Coordinate Representations:



→ Local / Modeling Coordinate.

Graphics Functions:

Premitive

Geometric Transformations

Viewing Transformations

——x——

"Chapter-3"

* Points & lines:

* line drawing algorithms:

$$y = mx + c$$

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

$$b = y_1 - m x_1$$

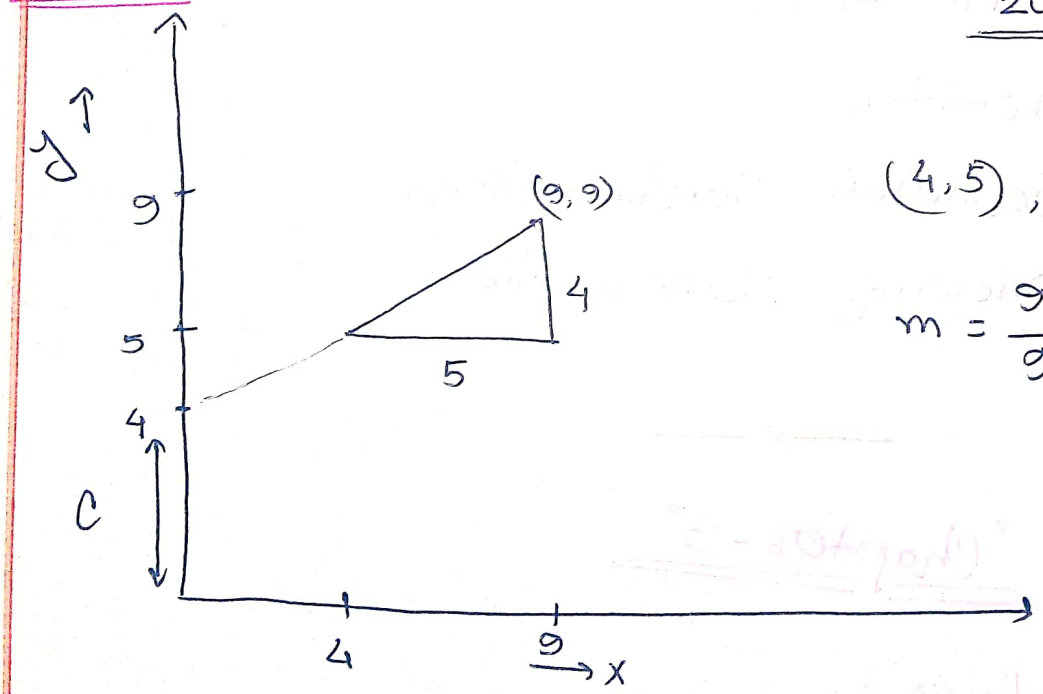
$$\Delta y = m \Delta x$$

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

——X——

DDA:

20.01.19



$(4,5)$, $(9,9)$

$$m = \frac{9-5}{9-4} = \frac{4}{5} = 0.8$$

x	y	Nearest integer
4	5	5
5	5.8	6
6	6.6	7
7	7.4	7
8	8.2	8
9	9	9

$(4,5)$, $(9,10)$

$$\Rightarrow m = \frac{10-5}{9-4} = \frac{5}{5} = 1$$

$(4,5)$, $(9,11)$

$$\Rightarrow m = \frac{6}{5} = 1.2 > 1$$

calculate x here, with respect to y as the change of y is greater.

Now, $\frac{1}{1.2} = 0.8 \left(\frac{1}{m} \text{ if } m>1\right)$

y	x	nearest integers
5	4	4
6	4.8	5
7	5.6	6
8	6.4	6
9	7.2	7
10	8	8
11.	8.8	9

eqⁿ. 3.6, 3.7

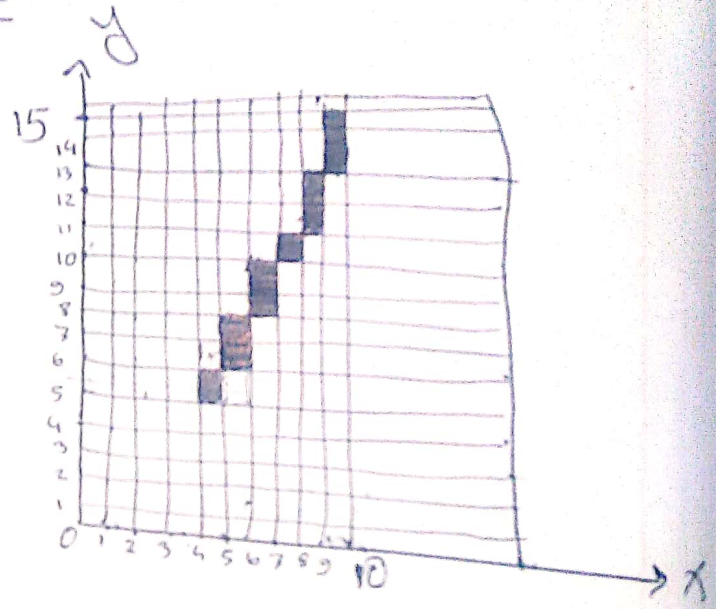
$(5, 6), (10, 15)$

$$m = \frac{15-6}{10-5} = 1.8$$

$\left(\frac{1}{m}\right)$

Now, $\frac{1}{1.8} = 0.6$

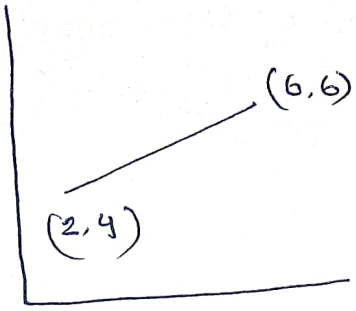
y	x	nearest integer
6	5	5
7	5.6	6
8	6.2	6
9	6.8	7
10	7.4	7
11	8	8
12	8.6	9
13	9.2	9
14	9.8	10
15	10.4	10



—x—

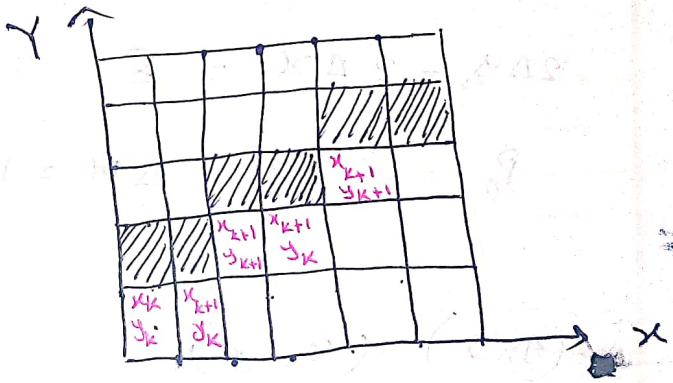
21.01.19.

Bresenham's Algorithm:



x_k , y_k

x_{k+1} , y_k / y_{k+1}



$\Delta x = 4$

$\Delta y = 2$

P = Decision parameter

$2\Delta y = 4$

$2\Delta y - 2\Delta x = 4 - 8 = -4$

$P_0 = 2\Delta y - \Delta x = 4 - 4 = 0$

→ (5, 6), (10, 15)

$\Delta x = 5$

$\Delta y = 9$

$2\Delta y = 18$

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

$P_0 = 2\Delta y - \Delta x = 18 - 5 = 13$

→ (4, 5), (9, 9).

$\Delta x = 5$

$\Delta y = 4$

$2\Delta y = 8$

$2\Delta y - 2\Delta x = 8 - 10 = -2$

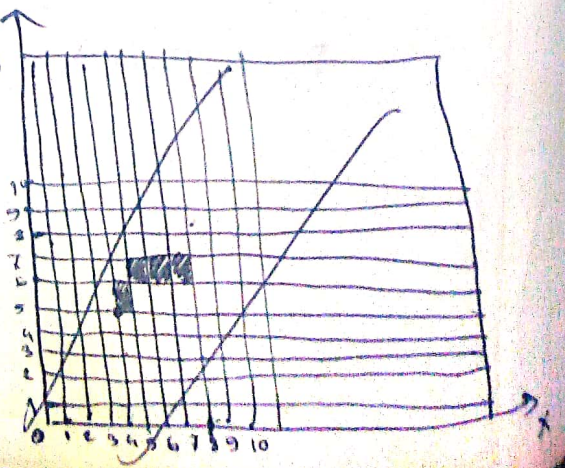
$P_0 = 2\Delta y - \Delta x = 8 - 5 = 3$

$P_{k+1} = 3 + 2\Delta y - 2\Delta x$

$= 3 + 8 - 10 = 1$

~~$P_{k+1} = 1 + 8 - 10 = -1$~~

~~$P_{k+1} = -1 + 8 - 10 = -3$~~



~~(20,10), (30, 18)~~

$P_{k+1} = 1 + 8 - 10 = -1$

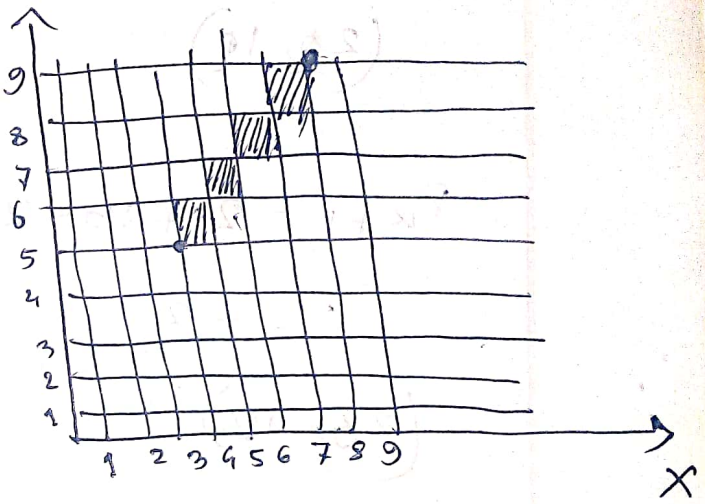
$P_{k+1} = P_k + 2\Delta y = -1 + 8$

$= 7$

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

$= 7 + 8 - 10$

$= 5$



⇒ (20, 10), (30, 18)

$\Delta x = 10$

$\Delta y = 8$

$2\Delta y = 16$

$2\Delta y - 2\Delta x = 16 - 20 = -4$

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

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

~~(20, 10)~~ (21, 11)

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

$$= 6 + 16 - 20$$

$$= 2$$

$(22, 12)$

$$P_{k+1} = 2 + 16 - 20$$

$$= -2$$

$(23, 12)$

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

$$= -2 + 16$$

$$= 14$$

~~$(23, 12)$~~ $(24, 13)$

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

$$= 14 + 16 - 20$$

$$= 10$$

~~$(24, 13)$~~ $(25, 14)$

$P_{k+1} = 10 + 16 - 20$

$= 6$

~~(25, 14)~~ (26, 15)

$P_{k+1} = 6 + 16 - 20$

$= 2$

~~(26, 15)~~ (27, 16)

$P_{k+1} = 2 + 16 - 20$

$= -2$

~~26, 1~~ ~~(27, 15)~~ (28, 16)

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

$= -2 + 16$

$= 14$

~~(28, 16)~~ (29, 17)

$P_{k+1} = 14 + 16 - 20 = 10$

~~(29, 17)~~ (30, 18)

$P_{k+1} = 10 + 16 - 20$

$= 6$

(30, 18)

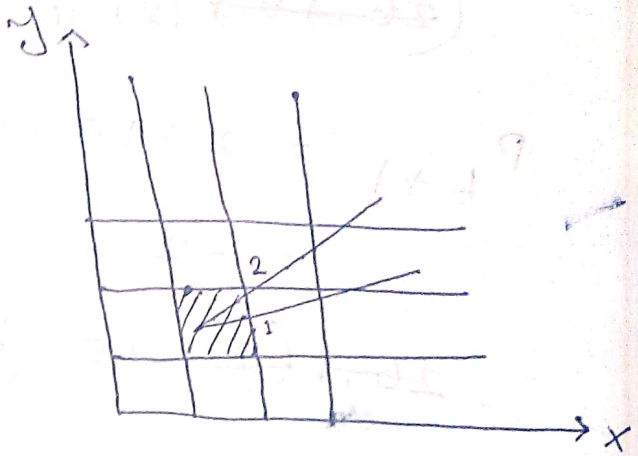
———— x ————

Calculation:

23.01.19

$(x_i + 1), (y_i)$

$(x_i + 1), (y_i + 1)$



$d_1 = y - y_1$

$d_2 = (y_{i+1}) - y$

~~$y = m_i + 1$~~

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

$\therefore d_1 = m(x_i + 1) + b - y_i$

$d_2 = (y_i + 1) - y = y_i + 1 - m(x_i + 1) - b$

$d_1 - d_2 = 2m(x_i + 1) - 2y_i + 2b - 1$

if $d_1 - d_2 < 0$ then $y_{i+1} \leftarrow y_i$

if $d_1 - d_2 > 0$ then $y_{i+1} \leftarrow y_i + 1$

(more floating point calculation)

CT-1 : (27.01.19) ⟶ 10:50 am.

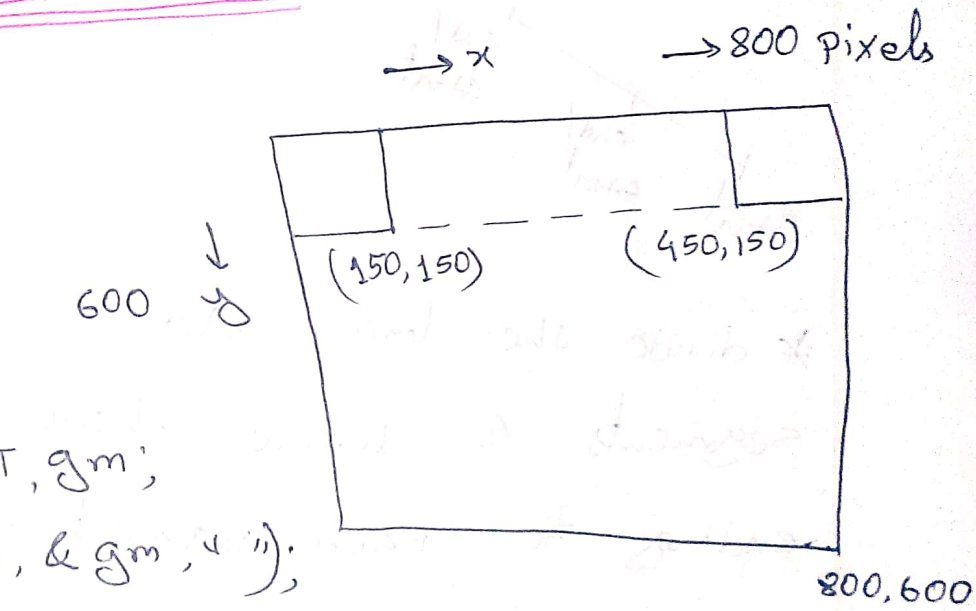
CRT, LCD ⟶ How it works?

DDA, Breserham's algorithm.

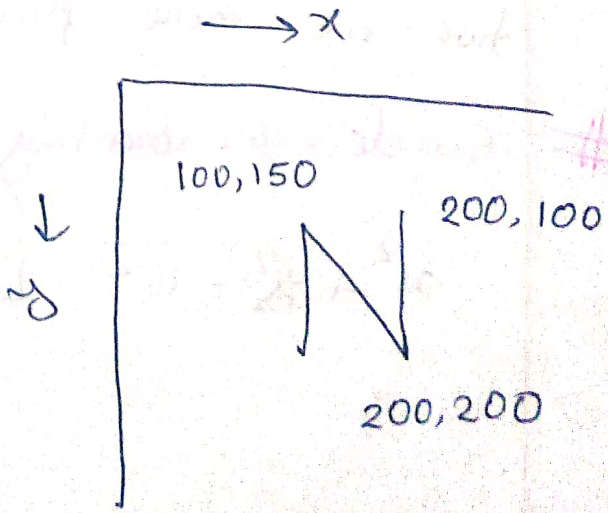
—x—

"LAB"

23.01.19



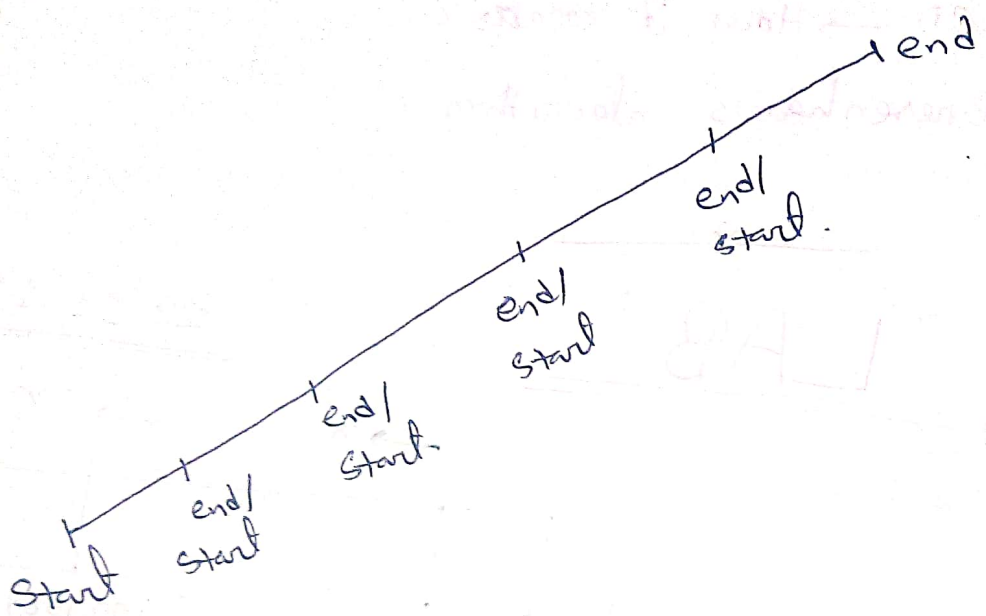

```
int main()
{ int gd=DETECT, gm;
  initgraph(&gd, &gm, " ");
  line(150,150,450,150);
  line(150,200,450,200);
  line(150,250,450,250);
  getch();
  closegraph();
}
```



—x—

27.01.19.

Parallel line Algorithm:



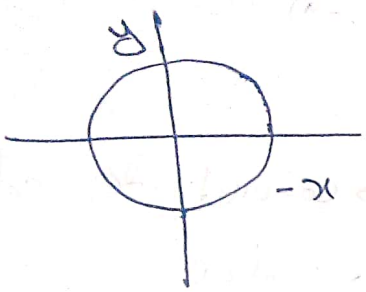
* divide the line into some predefined segments & define their starting & ending for each segment. Then use line drawing algorithm for every segment with two or more processors parallely.

Circle - Generating algorithm:

$x^2 + y^2 = r^2$ [if the center is at (0,0)].

$$\Rightarrow y^2 = r^2 - x^2$$

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



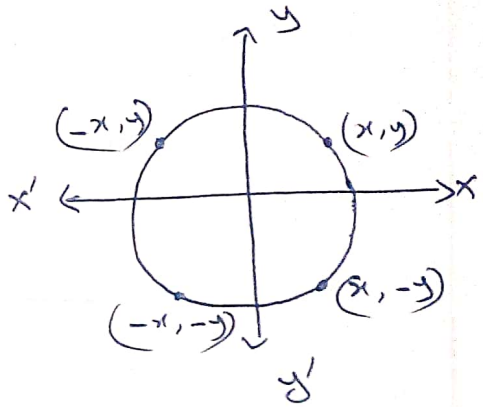
* If first square then, square root. So an expensive algorithm.

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

(when the center of a circle is at (0,0))

If not, then,

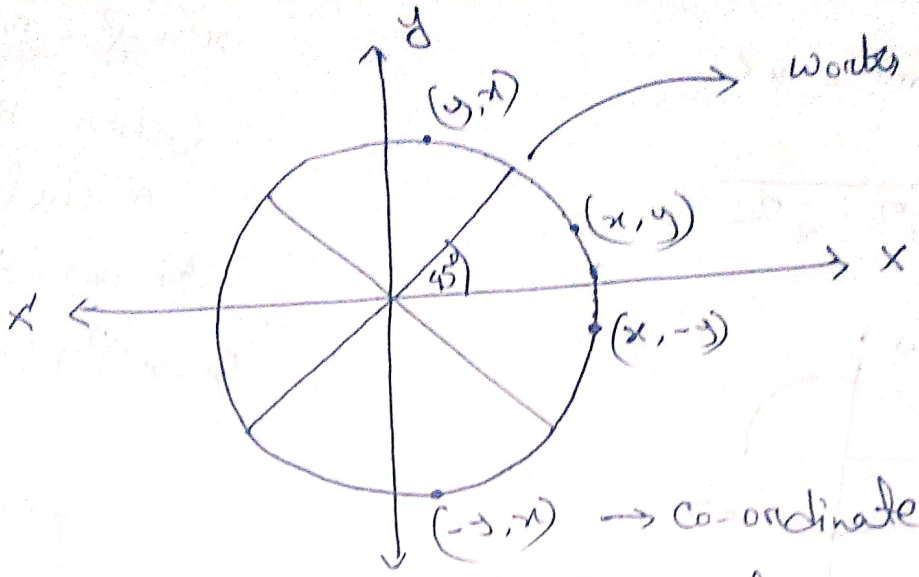
$$(x - x')^2 + (y - y')^2 = r^2$$



* If the center is not at (0, 0), then

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

$$\left.\begin{aligned} x &= x_c + r\cos\theta \\ y &= y_c + r\sin\theta \end{aligned}\right\}$$ still the algorithm is slow

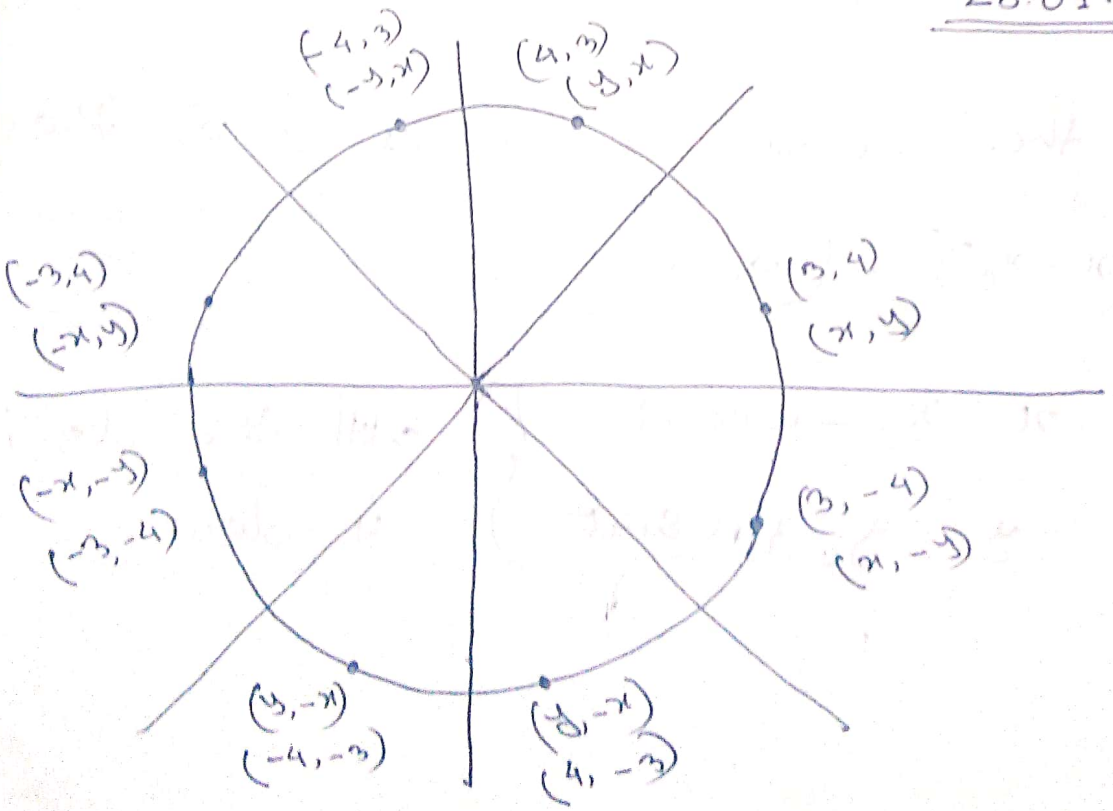


Words as mirror.

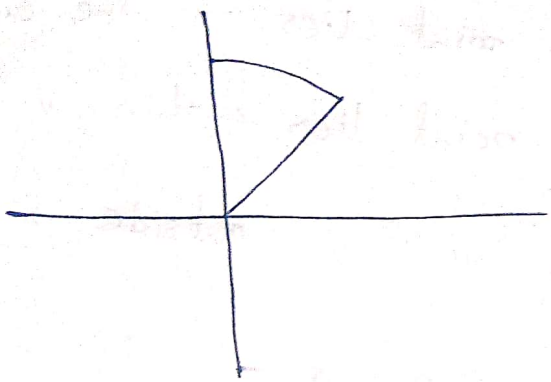
→ Co-ordinate calculation

→ value

28.01.19



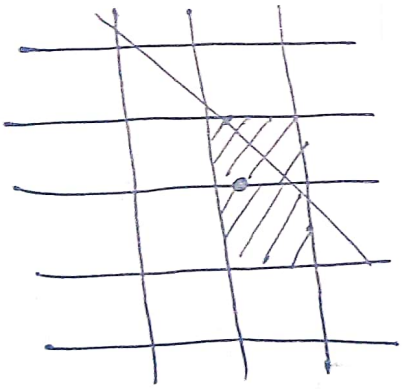
(i)



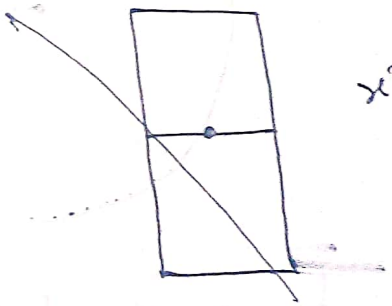
always $x = x + 1$

$y = y$ or $y - 1$

if $x^2 + y^2 > r^2$ then the point is out of the circle, if $x^2 + y^2 < r^2$ then the point is in the circle.



$x^2 + y^2 < r^2$



$x^2 + y^2 > r^2$

$(x, y) = (0, 10)$ $\quad r = 10$

$P_0 = 1 - r = 1 - 10 = -9$

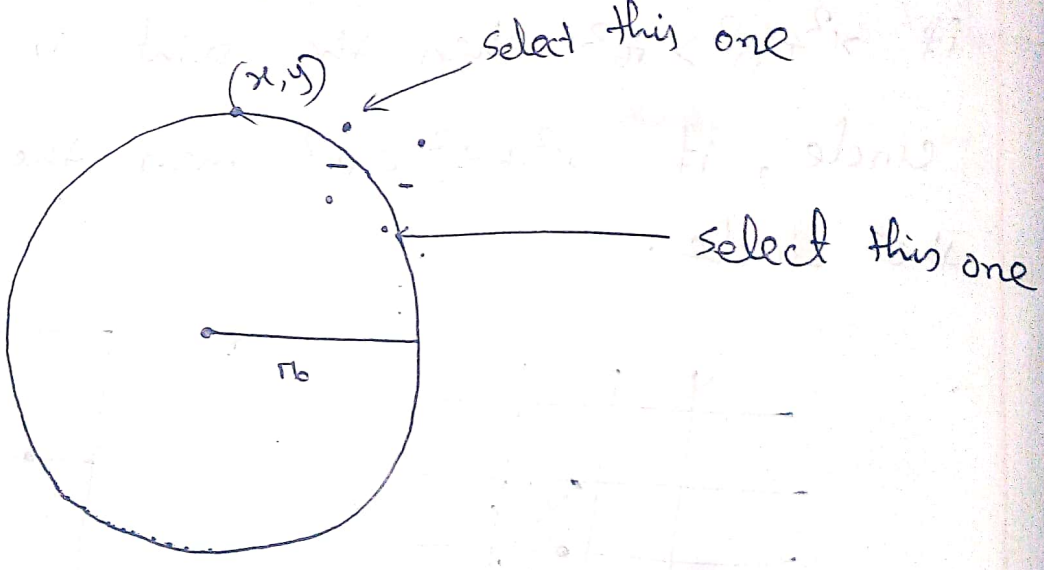
$2x_0 = 0$ $\quad 2y_0 = 0$

— x —

30.01.19

1. $f(x,y) = 0$ → The point lies on the circle
2. $f(x,y) < 0$ → The point lies within " "
3. $f(x,y) > 0$ → The " " outside " ".

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



Midpoint circle Algorithm:

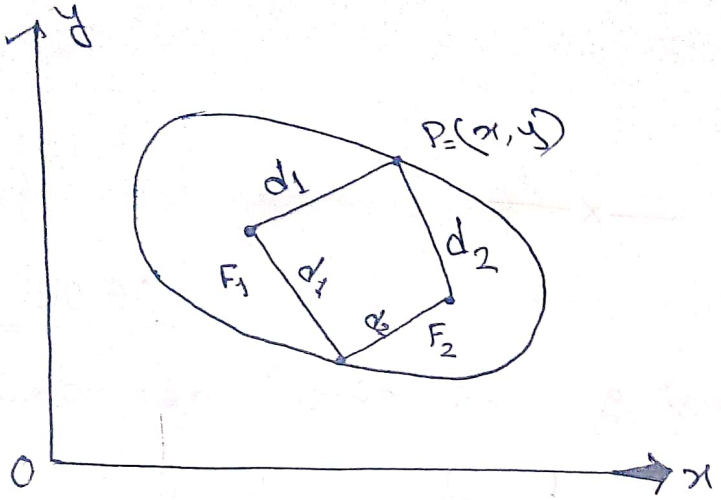
$$P_k = f(x_k + 1, y_k - \frac{1}{2})$$

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

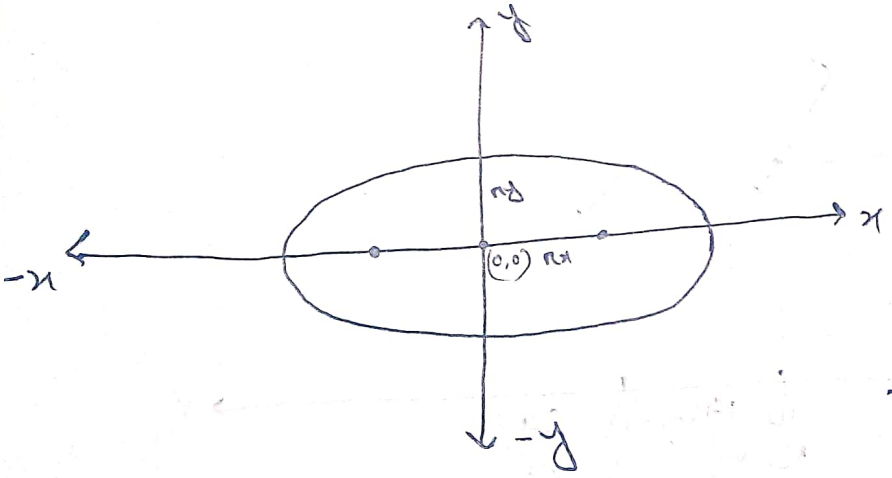
——X——

03.02.19.

Ellipse - generating Algorithm:



$d_1 + d_2 = \text{constant}$ (for every point, P)



$x = x_c + r_x \cos\theta$

$y = y_c + r_y \sin\theta$

Ellipse ⟶ H.C.

(Algorithm + derivation)

For computer screen $(x, y_{max} - y)$.

———x———

04.02.19

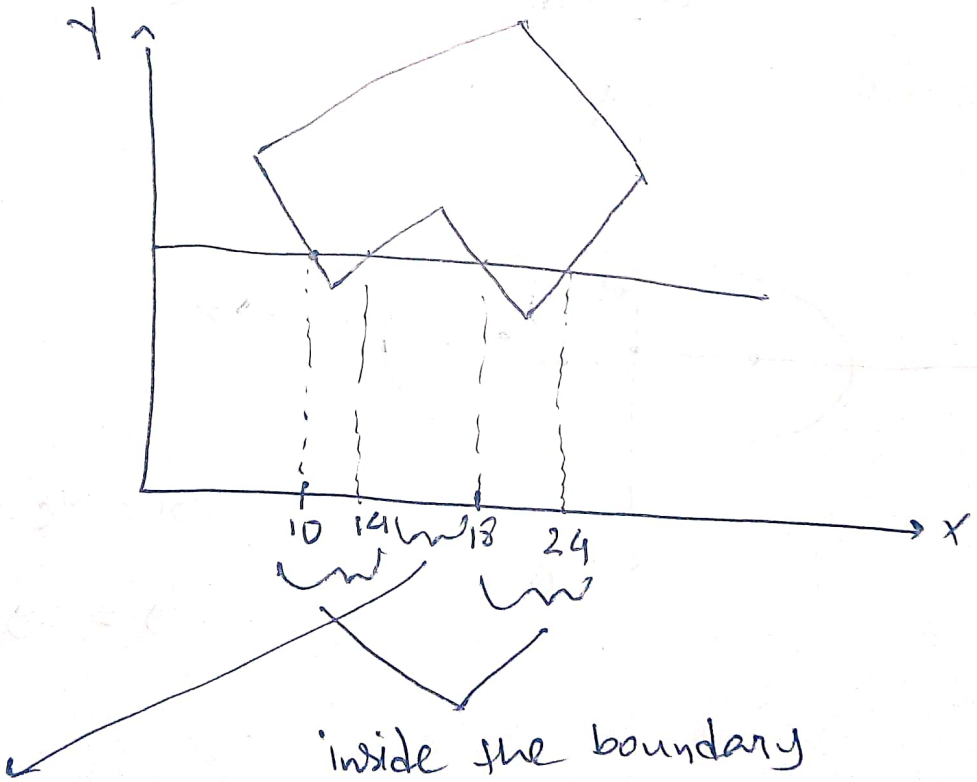
CT-1.

—x—

06.02.19

Pixel Addressing and object geometry:

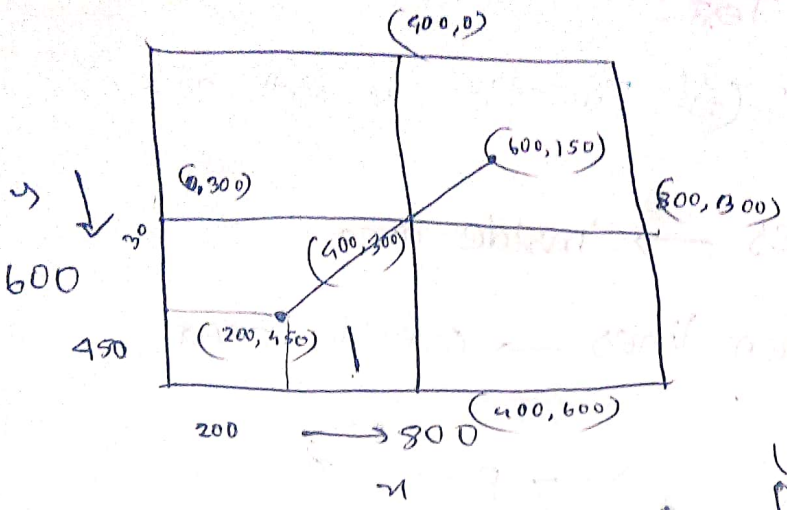
* use Inside part of a closed area



—x—

"LAB"

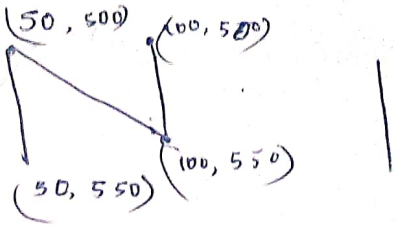
06.02.19.



50
50
100
140

x = 1-400

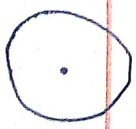
y = 450-600

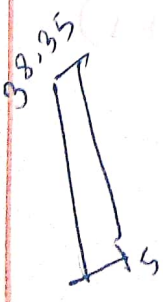


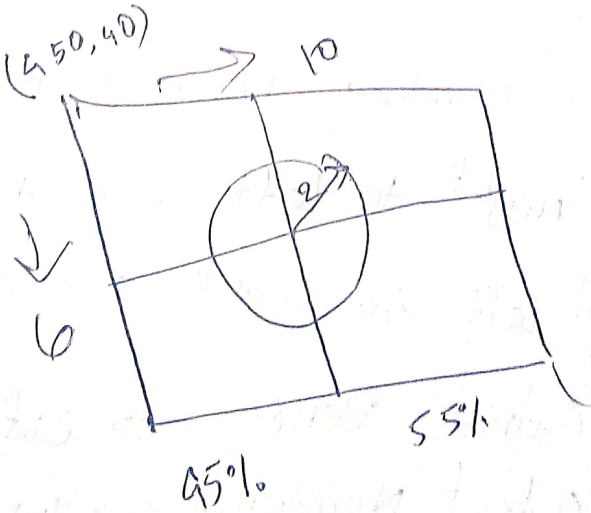
(45)

DDA Algorithm:

putpixel (int x, int y, ~~color~~ int color)







450 - 500 → x

100 - 150 - y

—X—

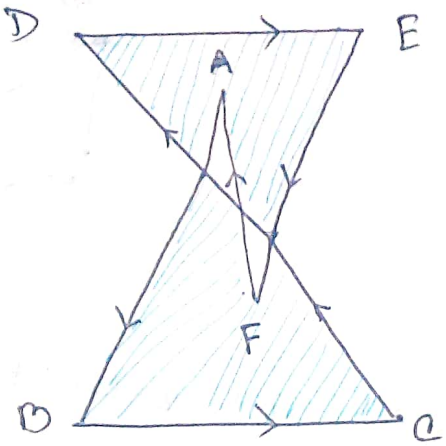
11.02.19.

Inside - Outside Test :

* Odd - Even Rule: (If Direction is not given)

If intersects odd lines → inside area

If intersects even lines → outside area.



inside area

* Nonzero - Winding - Number Rule: (If direction given)

$v = 0$

If right to left → +1 ($v = 0 + 1$)

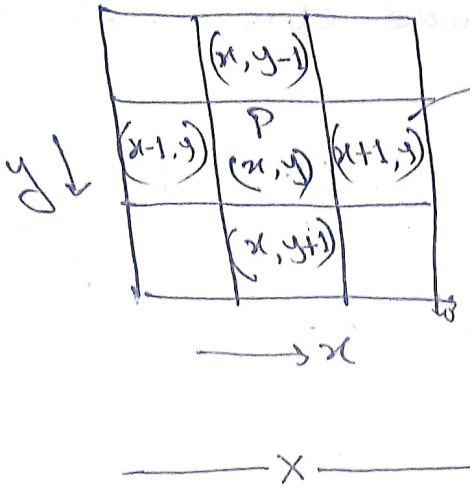
elseif left to right → −1

After loop, Output zero → outside area

Output Nonzero → Inside area

Curve lines :

Boundary fill Algorithm :



→ Neighborhood

* If inside the boundary, color it, else leave.

17.02.19

18.02.19

Floodfill :

Font

Sherif, Sans Sherif

"Chapter-4"

* Line Attribute:

  butt caps
  round caps
  projecting square caps } Single line width increase

  miter join
  Round join
  bevel join } Multiple line width increase.

* Line Gap:

Pen and brush options:

CT-2

Till now

wednesday - 27.02.19.

2nd period.

—x—

20.02.19.

Curve Attributes:

* Color & Grayscale level:

$$\text{intensity} = 0.5\,(\min(r,g,b) + \max(r,g,b))$$

* Area fill Attributes:

Marker Attributes:

Anti Aliasing:

$$f_s = 2 f_{max}$$

$$\Delta x_0 = \frac{\Delta x_{cycle}}{2}$$

Used to reduce staircase line.

Super Sampling

$$\text{pixel}_{color} = (5 \cdot red + 4 \cdot blue)/9$$

1	2	1
2	4	2
1	2	1

"LAB" 20.02.19

24.02.19

"Chapter - 5"

Basic Transformations:

Translation.

$x' = x + t_x$,

$y' = y + t_y$.

(t_x, t_y) translation vector.

——x——

25.02.19

Rotation

Scaling

27.02.19.

3x3 Matrix:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$P' = T(t_x, t_y) \cdot P$$

Rotation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

* Translation:

$$P = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad P' = \begin{bmatrix} x_1' \\ x_2' \end{bmatrix} \quad T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

* Rotation:

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

* Scaling:

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

$$P' = S \cdot P$$

General Pivot-Point Rotation:

$$\begin{bmatrix} \cos\theta & -\sin\theta & x_r(1-\cos\theta) \\ \sin\theta & \cos\theta & \\ 0 & 0 & \end{bmatrix}$$

$$A \cdot B \cdot C = (A \cdot B) \cdot C = A \cdot (B \cdot C)$$

In case of Rotation & scaling → order should be maintained.

* Reflection:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Reflect with respect to x-axis.

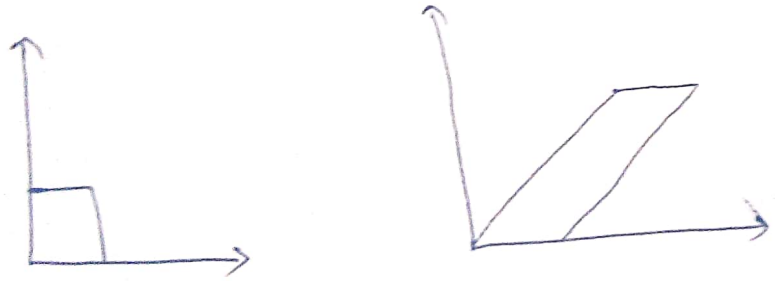
$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Reflect.
with respect to y-axis.

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Reflect with respect to both x and y-axis

* Shear:



$$\begin{bmatrix} 1 & Sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$x' = x + Sh_x \cdot y$

$y' = y$

Shear with respect to x-axis.

———x———