Unit I

Graphics Primitives and Scan Conversion Algorithms

Introduction, graphics primitives - pixel, resolution, aspect ratio, frame buffer. Display devices, applications of computer graphics.
Introduction to OpenGL - OpenGL architecture, primitives and attributes, simple modelling and rendering of two- and three-dimensional geometric objects, GLUT, interaction, events and call-backs picking. (**Simple Interaction with the Mouse and Keyboard)**
Scan conversion: Line drawing algorithms: Digital Differential Analyzer (DDA), Bresenham. Circle drawing algorithms: DDA, Bresenham, and Midpoint.

Text Books:

1. S. Harrington, "Computer Graphics"‖, 2nd Edition, McGraw-Hill Publications, 1987, ISBN 0 – 07 – 100472 – 6.
2. Donald D. Hearn and Baker, "Computer Graphics with OpenGL", 4th Edition, ISBN-13: 9780136053583.
3. D. Rogers, "Procedural Elements for Computer Graphics", 2nd Edition, Tata McGraw-Hill Publication, 2001, ISBN 0 – 07 – 047371 – 4.

Reference Books:

1. J. Foley, V. Dam, S. Feiner, J. Hughes, "Computer Graphics Principles and Practice"‖, 2nd Edition, Pearson Education, 2003, ISBN 81 – 7808 – 038 – 9.
2. D. Rogers, J. Adams, "Mathematical Elements for Computer Graphics"‖, 2nd Edition, Tata McGraw Hill Publication, 2002, ISBN 0 – 07 – 048677 – 8.

Unit : 1
Graphics Primitives & scan Conversions

[Reference :- TB1]

Concepts –

→ Computer Graphics – are pictures created using Computers & the representation of image data by a computer specifically with the help from specialized graphic Hardware & software.

→ Computer Graphics studies the manipulation of visual & geometric information using Computational techniques.

→ Computer – electronic machine that accepts, processes, transforms & presents informations.

Computer Graphics – it involve technology to accept, process, transform & present information in a visual form that also concerns with producing images / Animations using computer.

Why do we need Computer Graphics :-

1) visualization – making visible presentation of numerical data, particularly graphical one.
2) Graphics is intresting
3) Requirement – e.g drawing of machines it is required to prepare drawing of a machine before the actual production.
4) entertain

Applications / Types of Computer Graphics :-

1] Interactive Computer Graphics :-

— it involves a two way communication between computer & user.
— The observer is given some control over the image by providing him with an input device.
e.g — video game controller of the ping pong game.
— Interactive computer Graphics affects our lives in a number of indirect ways.
e.g— it helps to train the pilots of our airplanes. we can create a flight simulator which may help the pilots to get trained not in a real aircraft but on grounds at the control of the flight simulator.

2) Non Interactive Computer Graphics:-
— also known as passive computer graphics.
— it is the computer graphics in which user does not have any kind of control over the image.
— image is merely the product of static stored program & will work according to the instructions given in the program linearly.
— The image is totally under the control of program instructions not under the user.
e.g screen savers

[Reference : TB1]

Applications of Computer Graphics :-

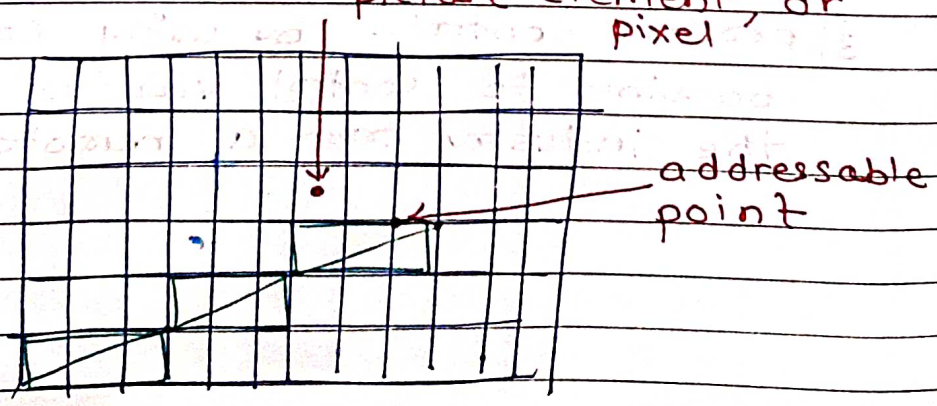
1] GUI - graphical user interfaces. a graphic, mouse oriented paradigm which allows the user to interact with a computer.
2] Data visualizations - charts & graphs
3] Computer Aided Design
4] Image processing
5] Cartography - CG is used to represent geographic maps, weather maps oceanographic charts, population density maps
6] plotting of Graphics & chart - In industry, business, government & educational organizations, computer graphics is most commonly used to create 2D & 3D graphs of mathematical physical & economic functions in form of histograms, bars & pie-charts. These graphs & charts are very useful for decision making.
7] Simulation & Animation - use of graphics in simulation makes mathematic models & mechanical system more realistic & easy to study. The interactive graphics supported by animation s/w proved their use in production of animated movies & cartoons films.
8] process control - by using computer it is possible to control various processes in the industry from a remote control room.

[Reference :- TB1]

Page No.
Date

Pixel :-

- A pixel (short for picture element), with the common abbreviation "pix" for "picture" is one of the many tiny dots that make up the representation of a picture in a computer's m/m.
- pixel is the smallest addressable part of the comp. screen.
- pixel is stored as a binary code representing a color.
- The code for a pixel can have between 1 & 32 bits of binary code.
- each information element is not really a dot, not a square, but an abstract sample.
- pixel in an image can be produced at any size without the appearance of visible dots or squares, but in many contexts they are reproduced as dots or squares & can be visibly distinct when not fine enough.
- The intensity of each pixel is variable in color system.
- each pixel has typically three or four dimensions of variability such as red, green & blue or cyan, magenta, yellow & black.



Pixel Depth :-

- it describes the no. of bits used to store each pixel.
- The greater the pixel depth, the more colors a pixel can have.
- color graphics vary in realism depending on resolution & pixel depth.
- greater the pixel depth, the bigger the file.

e.g.-

1) pixel depth monochrome
   monochrome graphics have one-bit pixel depth
   i.e pure blacks or pure white
2) Gray scale.
   - have more bit depth
   (no colors besides black, white, & grey)
3) 8 bit per pixel provides 256 color choices - typically of the web- that's why web graphics need some skillful preparation)
4) 24 or 32 bits per pixel provides thousands or millions of color choices (Typical of graphics and games s/w)

[Reference : TB1]

Frame Buffer :-

→ A block of memory, dedicated to graphics output that holds the contents of what will be displayed.

→ it is a large, contiguous piece of computer m/m. at a minimum there is one m/m bit for each pixel. ~~is~~ this amount of m/m

is called a bit plane. The picture is built up in the frame buffer one bit at a time

⇒ memory bit has only two states, single bit plane yeilds a black & white display.

⇒ frame buffer is a digital device & CRT is analog device. Conversion from a digital representation to an analog signal must take place when infom is read from the frame buffer & displayed on the raster CRT graphics device. have to use digital to analog convertor (DAC).

⇒ each pixel in frame buffer must be accessed & converted before it is visible on raster CRT.

Framebuffer in m/m :-

1) How much m/m do we need to allocate for framebuffer ?

— if we want framebuffer of 640 pixels by 480 pixels

allocate =

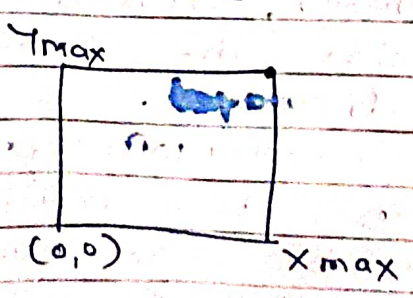
framebuffer = 640 * 480 bits.

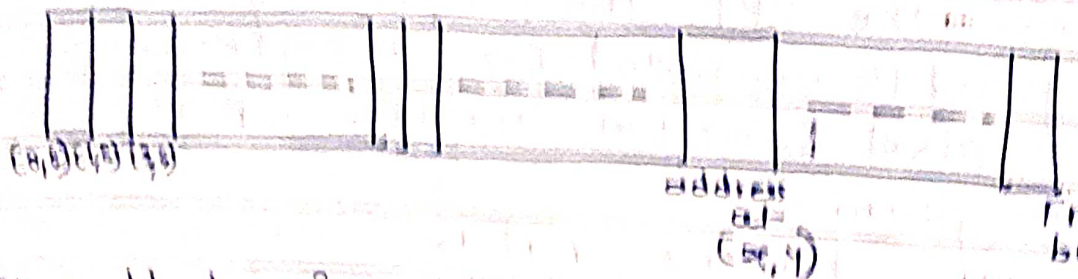
e.g put pixel (100, 200, 5)

x-coordinate = 100

Y-coordinate = 200

5- color no.





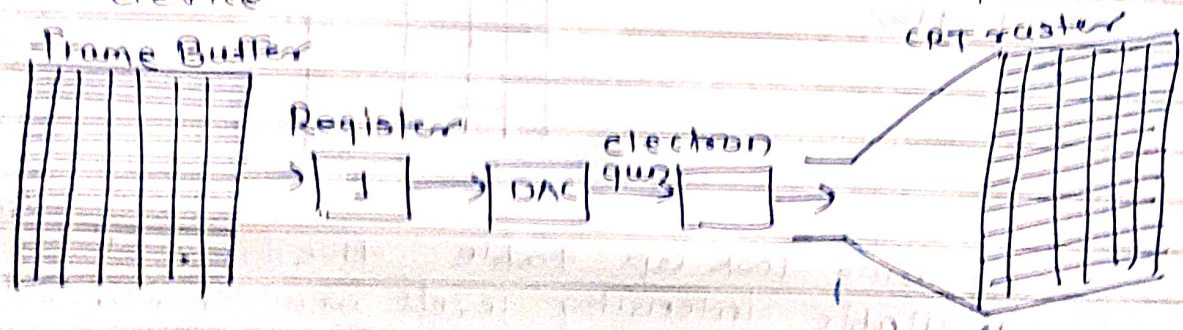
On a black & white system with one bit per pixel the frame buffer is commonly called as bitmap

* no of bits allocated per pixel in a buffer.

For system with multiple bits per pixel, frame buffer is often referred to as pixmap.

Black & white Frame Buffer:-

- If frame buffer stores one bit pixel info then the size of each element of frame buffer array is one bit. it is referred as bit plane.
- A single bit plane can be store only two values 0 & 1 thus it can yield blacks & white display.
- A single bit plane black & white frame buffer raster CRT graphic device

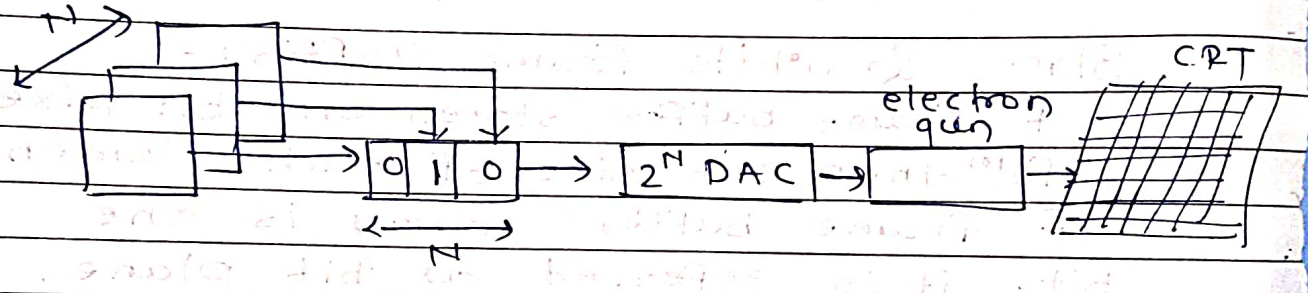


N-Bit plane Gray level Frame Buffer

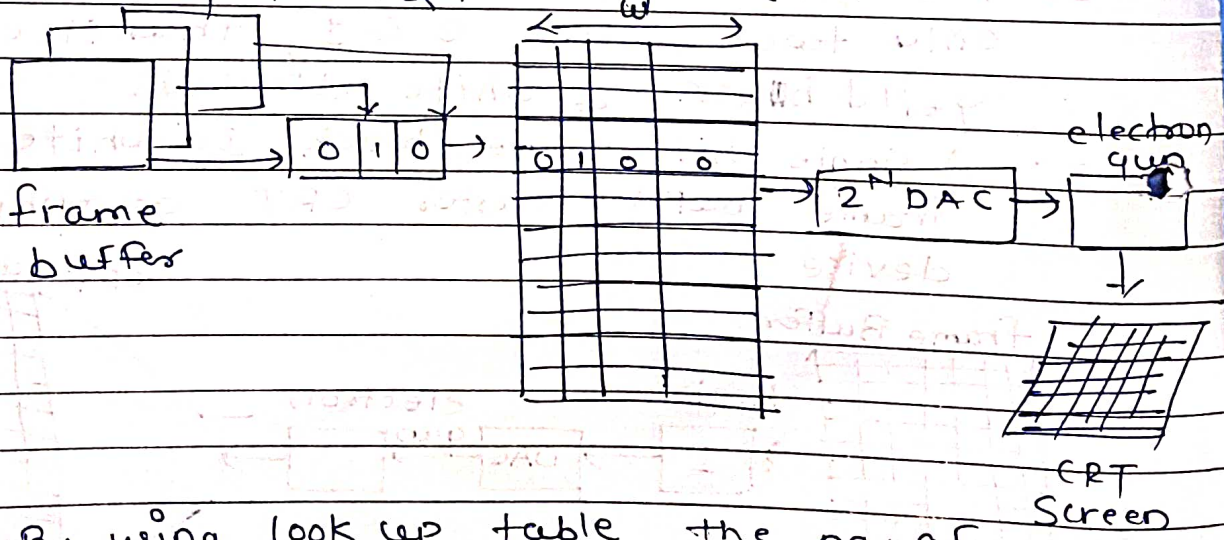
- The color of pixel can be incorporated in screen of frame buffer raster graphics device using additional bit planes.

— on the CRT the intensity of each pixel is controlled by a corresponding pixel location in each of N-bit planes.

— The binary value i.e 0 or 1 frame each of the N-bit plane is stored in register which is interpreted as an intensity level between 0 & 2^N where 0 represents dark & 2^N represents full intensity. This is then converted into analog voltage by using DAC i.e total 2^N intensity levels are possible.



Look Up Table & N-Bit plane :-



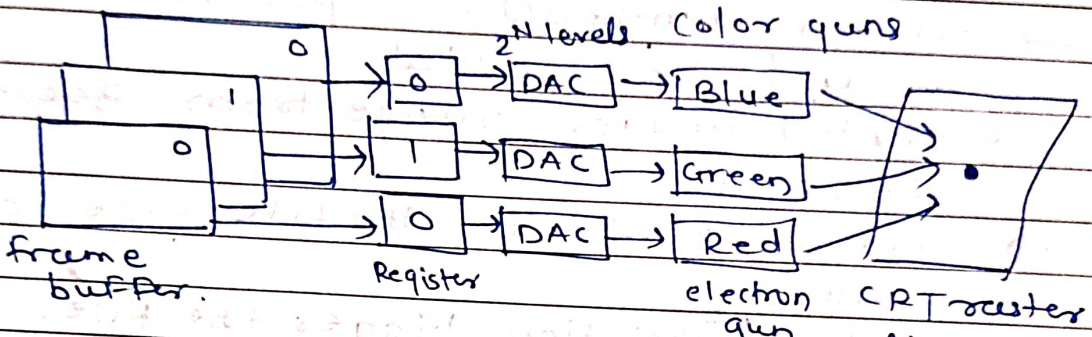
By using look up table the no. of available intensity levels are increased for achieving the modest increase in the m/m. after reading bit planes in the frame buffer resulting no. is used as an order into the look up table. The look up table has 2^N entries. each of which is w-bit wide.

Color & 3-Bit planes -

⇒ There are 3 primary Colors. simple color frame buffer is implemented with 3 bit plans. one for each primary order.

⇒ The bit plane drives an individual color gun for each of the three primary colors in color video

$2^3 = 8$ different colors.



	R	G	B
Black	0	0	0
Red	1	0	0
Green	0	1	0
Blue	0	0	1
cyan	0	1	1
Yellow	1	1	0
white	1	1	1
magneta	1	0	1

[Reference : TB1]

Resolution -

→ it is used as pixel count in digital imaging,

→ indicates the maximum no. of points that can be displayed without overlap on CRT.

→ It is defined as the no. of points per centimeter that can be plotted horizontally & vertically.

→ Resolution depends on the type of phosphor, the intensity to be displayed & the focusing & deflection system used in CRT.

→ Resolution refers to the density of dots on the screen or printed image & directly affects quality.

→ Resolution is measured in DPI (Dots per Inch)

→ Higher the resolution, the better the potential o/p.

→ Screens generally operate at around 70 - 100 dpl

→ printed images range from 300 to 2400 dpl.

→ Resolution affects the file size of an image

→ higher the resolution, bigger the file.

→ The visible resolution is limited to the maximum possible on the output device (screen or printer)

→ no matter how high the resolution of a photograph, it will show at the resolution of your screen or printer.

→ The convention is to describe the pixel resolution with the set of two positive integer numbers. where the 1st no. is number of pixel columns (width) & second is the no. of pixel rows (height)

e.g 640 × 480

VGA - 640 × 480

SVGA - 800 × 600

XGA - 1024 × 768

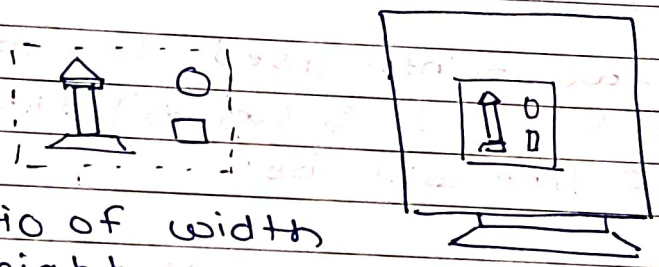
SXGA - 1400 × 1050

Resolution in Various media -

- DVDs have roughly 500 lines (or TV lines, or lines per picture height)
- High definition television has 1080 lines.
- 35 mm movie film is scanned for release on DVD at 1080 or 2000 lines as of 2005.
- [Reference : TBI]

ASPECT RATIO :-

- The ratio of the rectangle's width & height if different in orthogonal & window size — an desirable side effects caused by the independence of object, viewing parameters & workstation window specification concept of a viewport.



- The ratio of width to height.

e.g. if a direction in a slw manual tells you to "Hold down" the shift key while you resize a graphic in order to maintain the aspect ratio". it simply means that if you don't hold down the shift key you will stretch the image out of proportion.

- Aspect ratio of the screen determines the most efficient screen RESOLUTIONS & the most desirable shape for individual PIXELS, all of which may have to change upon the introduction of HIGH DEF" Television.

[Reference : TB1]

Page No.
Date

Plotting Primitives :-

Scan Conversions :-

— The process in which the object is represented as the collection of discrete pixels is called scan conversion.

lines —

→ point is represented by pair of numbers (x, y)
where x = horizontal distance
y = vertical distance

→ To specify a line we need two points.

— If two points used to specify a line are (x_1, y_1) & (x_2, y_2) then the equation of line will be

$$\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1} \quad \text{---------- (1)}$$

$$y = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) + y_1$$

This equation says that the slope bet^n any point on the line & point (x, y_1) is the same as the slope between (x_1, y_1) & (x_2, y_2).

multiplying by denominators we get

$$(y - y_1)(x_2 - x_1) = (y_2 - y_1)(x - x_1) \quad \text{---(2)}$$

by solving this for y

$$(y - y_1) = \frac{(y_2 - y_1)(x - x_1)}{(x_2 - x_1)}$$

$$= \frac{(y_2 - y_1)(x - x_1)}{(x_2 - x_1)} + y_1$$

i.e $$y = \frac{y_2 - y_1}{x_2 - x_1} * (x - x_1) + y_1$$

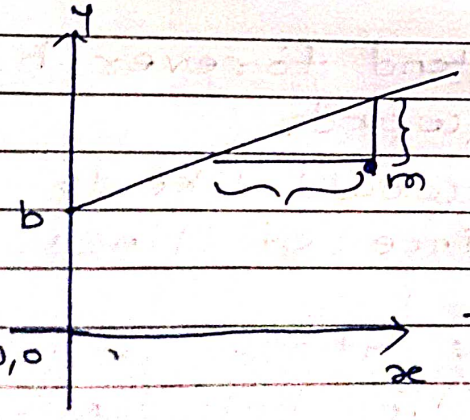
$$\therefore y = \underbrace{\frac{y_2 - y_1}{x_2 - x_1}}_{m} x \underbrace{- \frac{y_2 - y_1}{x_2 - x_1} x_1 + y_1}_{b}$$

$\therefore y = mx + b$ .... eqn called slope intercept form

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

$b = y_1 - mx_1$

Slope m is the change in height divided by the change in width for any two points on the line.



The intercept b is the height at which the line crosses the y-axis.

- we can get y intercept by having $(0, b)$

A different form of line equation is called general form.

$$(y_2-y_1)(x-x_1) = (y-y_1)(x_2-x_1)$$

$$(y_2-y_1)x - (x_2-x_1)y + x_2y_1 - x_1y_2 = 0$$

$$rx + sy + t = 0 \quad \text{——} (3)$$

values of r, s, t are

$r = y_2 - y_1$

$s = x_2 - x_1$

$t = x_2y_1 - x_1y_2$

if we compare equation (3) with slope intercept form i.e $y = mx + b$

$$rx + sy + t = 0$$

$$sy = -rx - t$$

$$y = \underbrace{-\frac{r}{s}}_{m}x \underbrace{- \frac{t}{s}}_{b}$$

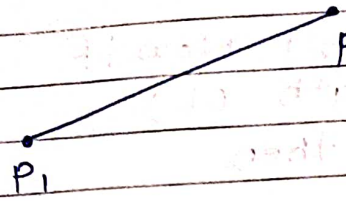
$$\therefore m = -\frac{r}{s} \quad \& \quad b = -\frac{t}{s}$$

[Reference : TB1]

line Segments :-

- There is a difference between line & line segment.
- The lines can extend forever both forward & backward.
- In graphics we would like to display only a piece of lines.
- let's consider only those points on a line which lie between two end points P_1 & P_2

called line Segment.



A line segment may be specified by its two end points. from these end points we can determine the equation of line.

If end points are $P_1 = (x_1, y_1)$ & P_2 $(x_2$ & $y_2)$ & these yeild eq^n

$y = mx + b$ then another point $P_3 = (x_3\ y_3)$ lies on the segment if

$= y_3 = mx_3 + b$

$= x_3$ is in between x_1 & x_2.

$= y_3$ is in between y_1 & y_2.

parametric form = x & y value on the line are given in terms of parameter u.

- Suppose we want the line segment between (x_1, y_1) & (x_2, y_2) then we need the x-coordinate to move uniformly from x_1 to x_2.

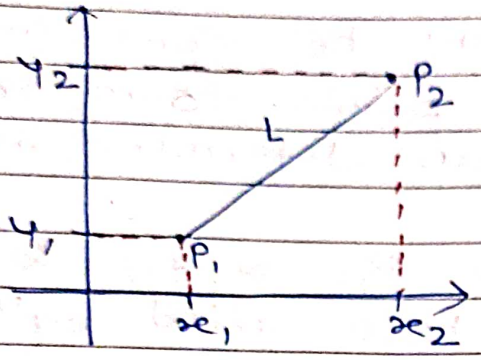
$x = x_1 + (x_2 - x_1)\,u$ where $u = 0$ to 1

when $u = 0$, $x = x_1$. As u increases to 1, x moves uniformly to x_2. Similarly we must have the y co-ordinate moving from y_1 to y_2 at the same time as x changes.

$y = y_1 + (y_2 - y_1)\,u$

To find the length of line segment use pythagorean theorem.

point $P(x_1, y_1)$ & $P_2(x_2, y_2)$ Now if we want to find length (L) of line segment P_1P_2 then



$$L^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2$$

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

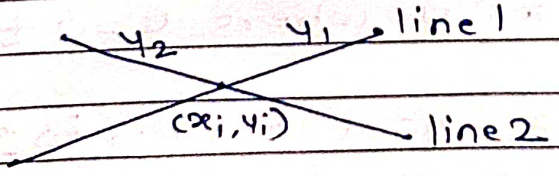
mid point of a line segment

$$= (x_m, y_m) = \left[\left(\frac{x_1 + x_2}{2}\right), \left(\frac{y_1 + y_2}{2}\right)\right]$$

Intersection of two lines :-

- To determine a point where two lines can cross

- By two lines crossing mean they share some common point. that point will satisfy the equation for the both lines.



- There are two lines line 1 & line 2 & both lines are intersecting at point (x_i, y_i) then we can represent line 1 by eq^n

line 1 = $y = m_1x + b_1$

line 2 $y = m_2x + b_2$

where m_1 & m_2 are slopes of line 1 & line 2 respectively.

if point (x_i, y_i) is shared by both the lines then the point (x_i, y_i) must satisfy both the line equations.

$\therefore y_i = m_1 x_i + b_1$ & $y_i = m_2 x_i + b_2$

$\therefore$ equating both over y_i

$$m_1 x_i + b_1 = m_2 x_i + b_2$$

$$m_1 x_i - m_2 x_i = b_2 - b_1$$

$$x_i = \frac{b_2 - b_1}{m_1 - m_2} \quad \text{---------- (1)}$$

equating the value of x_i in any one line equation we will get y_i as

$$y_i = m_1 x_i + b_1$$

$$= m_1 \left(\frac{b_2 - b_1}{m_1 - m_2}\right) + b_1$$

$$= \frac{m_1 b_2 - m_1 b_1}{m_1 - m_2} + b_1$$

$$= \frac{m_1 b_2 - m_1 b_1 + m_1 b_1 - m_2 b_1}{m_1 - m_2}$$

$$= \frac{m_1 b_2 - m_2 b_1}{m_1 - m_2}$$

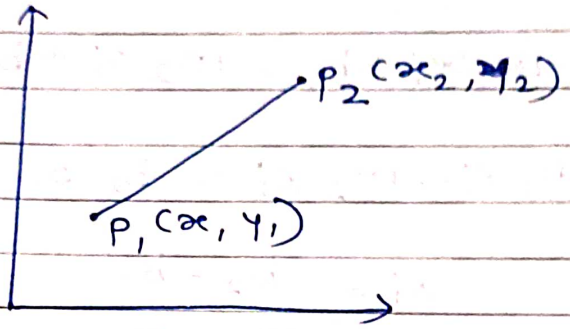
$\therefore$ point $\left(\frac{b_2 - b_1}{m_1 - m_2}, \frac{m_1 b_2 - m_2 b_1}{m_1 - m_2}\right)$ is intersection point.

[Reference : TB]

Vectors -

- Vector can be defined as a directed line segment which possess magnitude as well as direction.
- if two points P_1 & P_2 are given then vector V is defined as the difference between the two point positions.

for 2 Dimension :-



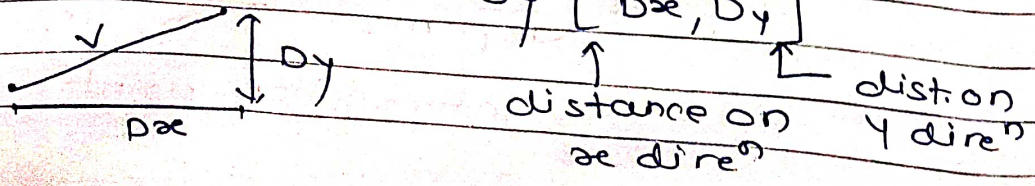
$$V = P_2 - P_1$$
$$= (x_2 - x_1, y_2 - y_1)$$
$$= (V_x, V_y)$$

$V_x, V_y \rightarrow$ projections of V on x & y axes

$$|V| = \sqrt{V_x^2 + V_y^2}$$

- line segment has a fixed position in space
- vector does not have a fixed position in space. The vector does not tell us the starting point. it tells how far to move and in which direction.

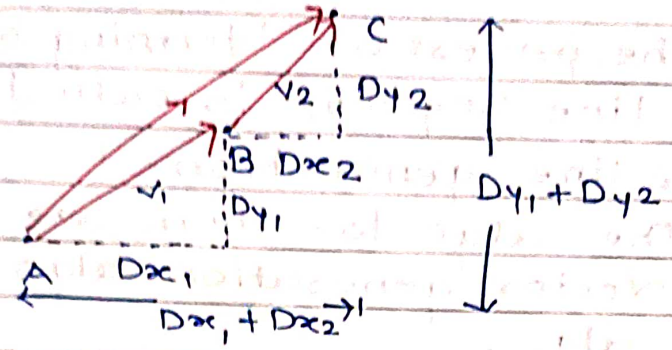
- A vector has a single direction & a length
- vector indicated by $[D_x, D_y]$



Page No.
Date

if vector V1 is $[Dx_1, Dy_1]$
vector V2 $[Dx_2, Dy_2]$

$$V_1 + V_2 = [Dx_1, Dy_1] + [Dx_2, Dy_2]$$
$$= [Dx_1 + Dx_2, Dy_1 + Dy_2]$$



we start at point A to B second vector B to C. then right hand side equation directly gives us A to C.

Multiplication of vectors-

- The vector can be multiplied by a number. The multiplication is performed by multiplying each component of a vector by a number.

let $V[Dx, Dy]$ be a vector which is to be multiplied by no. n then

$$nV = n[Dx, Dy]$$
$$= nDx, nDy \quad \text{—①}$$

The magnitude of the resultant vector is given by

$$|V| = (Dx^2 + Dy^2)^{1/2}$$

- The result of the multiplication changes the magnitude of a vector but preserves its direction.

Unit Vectors -

multiplication of a vector & the reciproca
of its length is equal to 1.
Such vectors are called Unit vectors.

[Reference : TBI]

Line Drawing Algorithms / vector generation

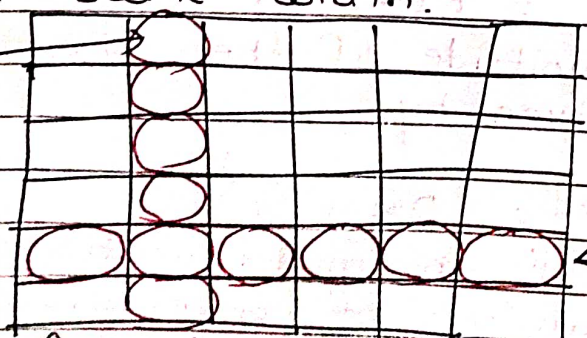
- The process of 'turning on' the pixels for a line segment is called vector generation or line generation.
- The algo for them are known as vector generation algo or line drawing algo.
-

Qualities of good line drawing Algo :-

1] The line should appear as a straight line & it should start & end accurately.
2] The line should be displayed with constant brightness along its length independent of its length & orientation
3] The line should be drawn rapidly.

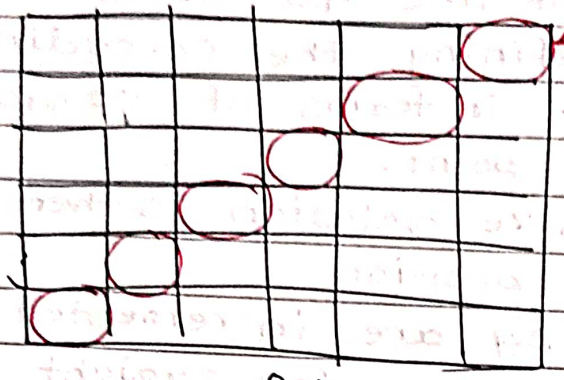
Vertical & Horizontal lines :-

- horizontal & vertical lines are straight & have same width.



- The 45° line is straight but its width is not constant.

— The line with any other orientation is neither straight nor has same width. Cases are due to finite resolution of display. &



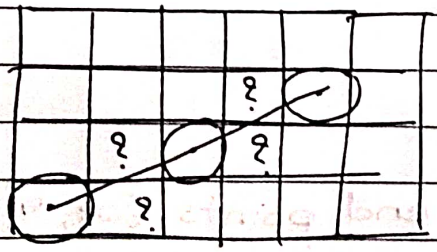
45° line.

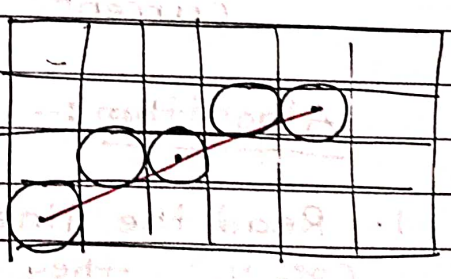
we have to accept approximate pixels in this situations

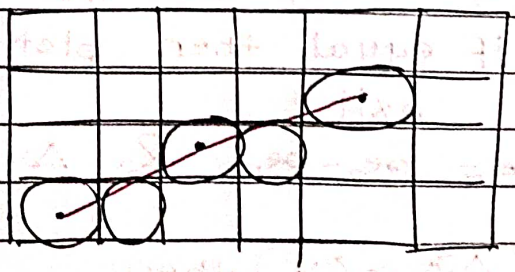
The brightness of the line is dependent on the orientation of the line.

— it is observe that the effective spacing between pixels for 45° line is greater than for the vertical & horizontal lines.

— This will make the vertical & horizontal lines appear brighter than 45° line.







[Reference : TB1]

Vector Generation / Digital Differential Analyzer (DDA) algo :-

- it is the simplest algo of line drawing.
- algo makes the use of differential equation of line to obtain recursion relation defining the co-ordinate of next point in terms of co-ordinate of current point.
- This recursive relation is then used to select pixel position.

if Δx & Δy are incremented in x & y direction then for straight line equation is,

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

x_1, x_2, y_1, y_2 are coordinates of end points

recursion relation obtained will be

$$y_{i+1} = y_1 + \Delta y$$

$$x_{i+1} = x_1 + \Delta x$$

where x_{i+1} & y_{i+1} are coordinates of next point. x_1 & y_1 are coordinates of current point.

Algorithm :-

1. Read the line & end points (x_1, y_1) & (x_2, y_2). they are not equal.
   (if equal then plot that point & exit)
2. $\Delta x = |x_2 - x_1|$ & $\Delta y = |y_2 - y_1|$
3. If $(\Delta x \geq \Delta y)$ then
   length = Δx
   else

length = Δy
end if

4. $\Delta x = (x_2 - x_1) / length$
$\Delta y = (y_2 - y_1) / length$
(This makes either Δx or Δy equal to 1 because length is either $|x_2 - x_1|$ or $|y_2 - y_1|$.
∴ the incremental value for either x or y is one)

5. $x = x_1 + 0.5 * sign(\Delta x)$
$y = y_1 + 0.5 * sign(\Delta y)$
(sign fun^n makes algo work in all quadrant. it returns $-1, 0, 1$ depending on whether its argument is $< 0, = 0, > 0$ respectively.
- factor 0.5 makes it possible to round the values in the integer function rather than truncating them
plot (Integer (x), Integer (y))

6. i = 1
while (i ≤ length)
{
x = x + Δx
y = y + Δy
plot (Integer (x), Integer (y))
i = i + 1
}

7. stop

example :-

line from (1,1) to (5,6).

→ end points of lines are (1,1) & (5,6)

$x_1 = 1, \quad y_1 = 1$

$x_2 = 5, \quad y_2 = 6$

length = abs $(y_2 - y_1)$ /

= abs $(6-1)$

= 5

∴ length = 5

$\Delta x = (x_2 - x_1) /$ length

$= \frac{5-1}{5}$

$= 0.8$

$\boxed{\Delta x = 0.8}$

$\Delta y = (y_2 - y_1) /$ length

$= \frac{6-1}{5}$

$\boxed{\Delta y = 1}$

initial value

$x = x_1 + 0.5 * \text{sign}(\Delta x)$

$y = y_1 + 0.5 * \text{sign}(\Delta y)$

$x = 1 + 0.5 * \text{sign}(0.8)$

$= 1.5$

$y = 1 + 0.5 * \text{sign}(1)$

$= 1.5.$

	plot	x	y			
1.	(1,1)	1.5	1.5	(4,5)	4.7	5.5
	(2,2)	2.3	2.5	(5,6)	5.5	6.5
	(3,3)	3.1	3.5			
	(3,4)	3.9	4.5	(6,7)	6.3	7.5

Page No. 7
Date

Advantages of DDA algorithm:-

1) It is the simplest algo & it does not require special skills for implementation.
2) logic is simple to understand.
3) Integer arithmetic is involved.
4) It is faster method for calculating pixel position.

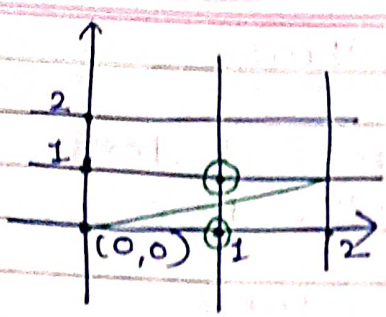
Disadvantages of DDA Algo:-

1) Floating point arithmetic in DDA algo is still time consuming.
2) Division logic is needed, which switches it towards hardware logic.
3) line is orientation dependent. i.e line which drawn in IInd & IVth quadrant tend to slightly diverted from the actual path of the line Sometimes extra pixels get activated which deteriorate (poor) the accuracy of end point.

[Reference :- TBI]

Bresenham's Line Generation Algorithm:-

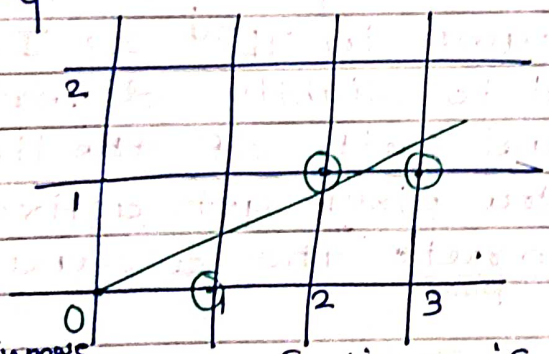
- This is another line generating algo.
- The distance between the actual line & the nearest grid location is called error & it is denoted by G.



first point (0,0) after displaying 1st point, we have to select next point.

- There are two candidate pixels (0,1) & (1,1).
- out of these two pixels we have to select one pixel. This selection of pixel will depend on the slope of the line. if the ~~pixel~~ slope of line is greater than 0.5 then we are selecting upper pixel i.e (1,1) & if slope is less than 0.5 then we are selecting next pixel as (1,0)

e.g

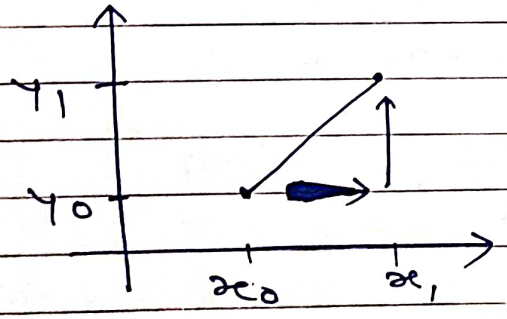


- Suppose slope of line is 0.4. after displaying 1st point as (0,0), we will add slope of line to error factor G:
- so that G will become 0.4. we are considering gentle slope case.
- we are moving along x-axis so next pixel will be x = 1. & at that time y will be decided by G
if G > 0.5 then y = 1 else y = 0.
but here as G = 0.4 we have to select next point as (1,0)

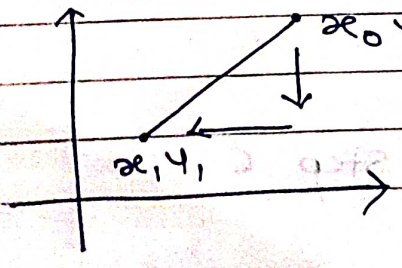
— Again for next point x is increased by 1 and it becomes x=2, for that G will become G = G+m i.e 0.8,

Steps – for Gentle slope (m<1)

1) Accept two endpoints from user & store the left end point in (x_0, y_0) as starting point.
2) plot the point (x_0, y_0)
3) Calculate all constants from two endpoints such as Dx, Dy, 2Dy, 2Dy−2Dx & find the starting value for G as G = 2Dy − Dx
4) for each column increment x & decide y-value by checking G>0 condition. if it is true then increment y-value & add (2Dy−2Dx) to current value of G. otherwise add (2Dy) to G & don't increment y-value.
plot next point.



5) Repeat step 4 till Dx times.



for steep slope cases just interchange the roles of x & y i.e we step along y-direction in unit steps & calculate successive x-values nearest to line path.

— if the initial position for a line with +ve slope is right endpoint. like in fig. then we have to decrease both x & y as.

algo :-

1) Read the line end points (x_1, y_1) & (x_2, y_2) such that they are not equal.

2) $\Delta x = |x_2 - x_1|$ $\Delta y = |y_2 - y_1|$

3) $x = x_1$
   $y = y_1$
   plot (x, y)

4) $e = 2 * \Delta y - \Delta x$
5) $i = 1$
6) while $(e \geq 0)$
   {
   $y = y + 1$
   $e = e - 2 * \Delta x$
   }

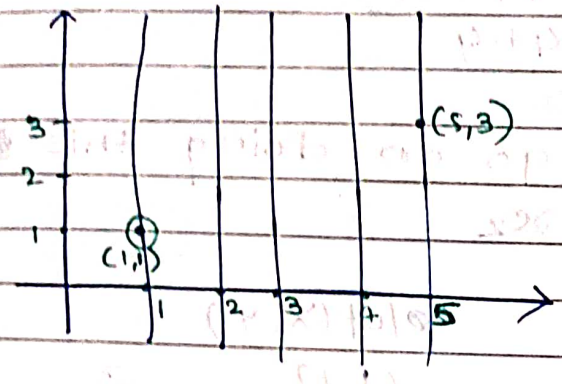
   $x = x + 1$
   $e = e + 2 * \Delta y$
7) plot (x, y)
8) $i = i + 1$
9) if $(i < \Delta x)$ then go to step 6
10) stop

Example -

plot a line from (1,1) to (5,3)

$x_1 = 1$ $\quad y_1 = 1$

$x_2 = 5$ $\quad y_2 = 3$



$\therefore D_x = x_2 - x_1$

$= 5 - 1 = 4$

$\boxed{D_x = 4}$

$\therefore D_y = y_2 - y_1$

$= 3 - 1$

$\boxed{D_y = 2}$

$e = 2D_y - D_x$

$= 2(2) - (4)$

$\boxed{e = 0}$

1) plot 1st point (1,1)
as $D_x > D_y$ i.e $4 > 2$ i.e Gentle slope. so we have to move on x till x_2 i.e 5

2) after ploting 1st point as (1,1) increase x by 1

$\therefore x = 2$

$e = 0$

we have to increment y by 1 & update e as

$e = e + 2(D_y - D_x)$

$= 0 + 2(2-4)$

$= -4$

— So, plot next point as (2,2) then again increase x by 1

$\therefore x = 3, \quad G = -4$

So don't Increase y just update G only as

$G = G + 2Dy$

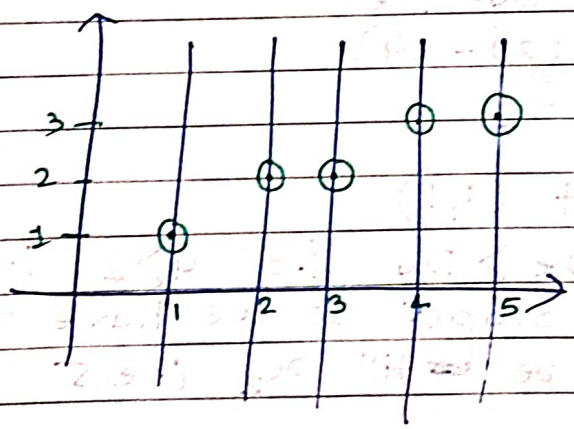
$= -4 + 2(2)$

$= -4 + 4$

$= 0$

So, plot (3,2) go on doing this till x reaches to x_2

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



— it is much more efficient than DDA.

— it requires only integers & requires only addition & subtraction operations.

[Reference : TB1]

Line Styles :—

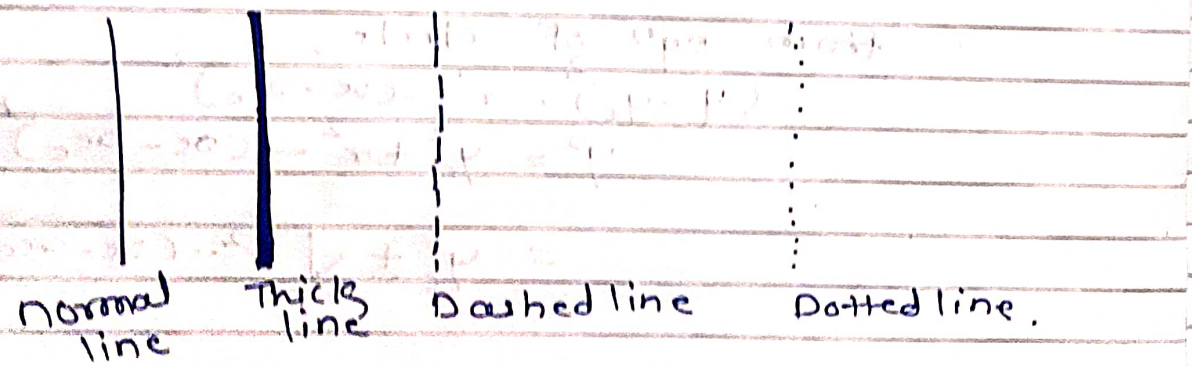
Characteristics of line Drawing Algo :—

1) line should appear straight
2) line should terminate Accurately
3) line should have constant Density.

4. line Density should be independent of line length & angle.
5. lines should be Drawn Rapidly.

There are 3 ways to draw line
- Dotted lines
- Dashed lines
- Thick lines

- The appearance of a line can be controlled by specification of different attributes, like line style, line width, color & pen style.
  - can change line style by using SetLineStyle(CONTINUOUS/DASHED/DOTTED/..
  - can set width of line by SetLineWidth(int WIDTH)
- width of a line is measured in pixel by default line style is CONTINUOUS & width is 1, if we set the pixel value as 1 then the pixel is displayed with foreground color & if we set the pixel value as zero then that pixel is displayed with background color.



solid line - 0
dotted line - 1
centre line - 2
dashed line - 3
user bit line - 4

SetLineStyle(int linestyle, unsigned upattern, int thickness)

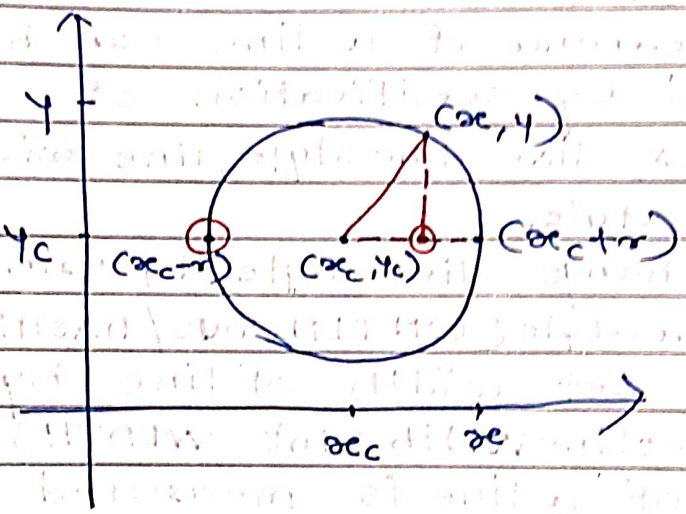
SetLineStyle(1, 0, 1);

[Reference : TBI]

Circle Drawing Algorithm :-

— A circle is a set of points that are placed at a given dist 'r' from center (x_c, y_c). The distance relationship is expressed by pythagorean theorem

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



— To calculate the position of points on circumference of circle by stepping along x-axis from $(x_c - r)$ to $(x_c + r)$ & calculating the corresponding y-values at each position by using

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

from eqn of circle,

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

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

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

[Reference : TBI]

DDA Circle Drawing Algorithm

center of circle

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

$$2x\,dx+2y\,dy=0 \qquad r \text{ is constant}$$

$$\therefore x\,dx+y\,dy=0$$

$$\therefore y\,dy=-x\,dx$$

$$\therefore \frac{dy}{dx}=-\frac{x}{y}$$

We can construct the circle by using incremental x value $\Delta x=\varepsilon y$ & incremental y value $\Delta y=-\varepsilon x$ where ε is calculated from the radius of the circle.

$$2^{n-1}\leq r\leq 2^n \qquad r: \text{radius of circle}$$

$$\varepsilon=2^{-n}$$

e.g. if $r=50$ then $n=6$

So that

$$2^{n-1}\leq r<2^n$$

$$2^5\leq 50<2^6$$

$$\therefore 32\leq 50\leq 64$$

$$\therefore \varepsilon=2^{-6}$$

$$=0.0156$$

Algorithm :-

1] Read the radius (r) of the circle & calculate value of ε

2] start_x = 0
start_y = r

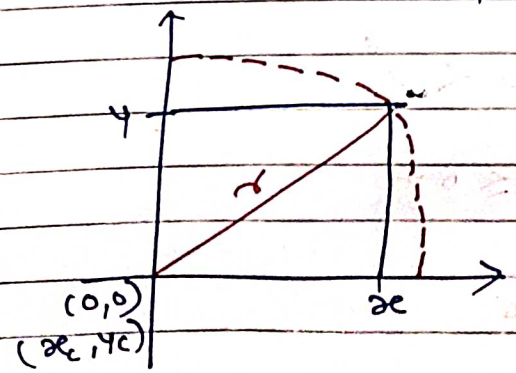
3] x_1 = start_x
y_1 = start_y

4] do
{
$x_2 = x_1 + \varepsilon y_1$
$y_2 = y_1 - \varepsilon x_2$

plot (int x_2, int y_2)
$x_1 = x_2$
$y_1 = y_2$
} while((y_1 - start_y) < ε || (start_x - x_1) > ε);

Midpoint Circle Generation Algorithm :-

- it is used to determine the closest pixel position to the specified along circle's path at each step.
- for this we have to accept radius 'r' & center point x_c & y_c



- center point as origin (0,0) then each calculated point is to be moved to proper position by adding x_c & y_c to corresponding x & y-value.

In 1st Quadrant we are moving in x-direction with starting point as x=0 to ending point x=y.

Midpoint circle Drawing Algorithm -

1] Read the radius (r) of the circle.

2] x = 0
y = r

3] Calculate initial value of decision parameter

$p = 1.25 - r$ $\quad p = 1 - r$

4] do
{
plot (x, y)
if (p < 0)
{
x = x + 1
y = y
p = p + 2x + 1
}
else
{
x = x + 1
y = y - 1
p = p + 2x - 2y + 1
}
while (x < y)

5] Determine symmetry points

6] stop.

Page No.
Date

Steps:-

1) Accept radius 'r' & center of circle (x_c, y_c) from user & plot 1st point on circumference of circle.
$(x_0, y_0) = (0, r)$

2) Calculate the initial value of the decision parameter
$P_0 = 1 - r$

3) if we are using octant symmetry property to plot the pixels, then until $(x < y)$ we have to perform following steps
if P_K is less than 0
modify P_K as $P_K = P_K + 2K + 1$ &
then increase x by 1
otherwise
modify P_K as $P_K = P_K + 2(x - y) + 1$ &
increase x by 1
& decrease y by 1

4) Determine the symmetry points in other octants also &

5) Since we have derived all formulas by considering center point as origin. move each calculated pixel position (x, y) on to the circular path centered on (x_c, y_c) & plot the co-ordinate values as

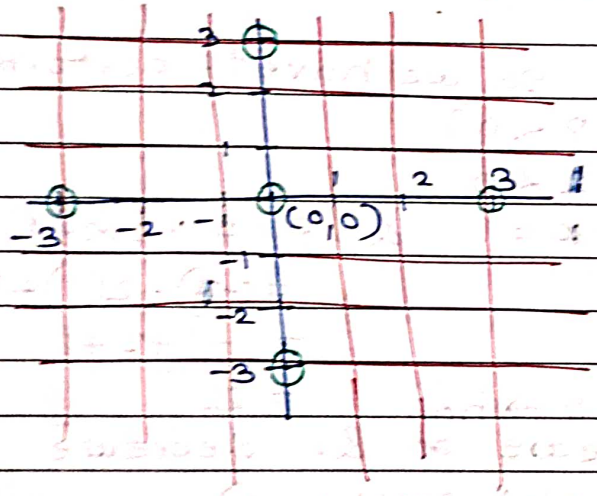
$x = x + x_c$ & $y = y + y_c$

the center point may be anything, we have to add that center co-ordinates to midpoint x & y to plot the print.

e.g. circle radius is 3 & whose center co-ordinates are (0,0).

→ $r = 3$
$x_c = 0$
$y_c = 0$.
plot 1st point as $(0, r)$
i.e $y = r$
∴ plot $(x, y) = (0, 3)$ —— (1)

– find out other points from (0,3) by using symmetry property
So plot (3, 0)
plot (0, −3)
plot (−3, 0)



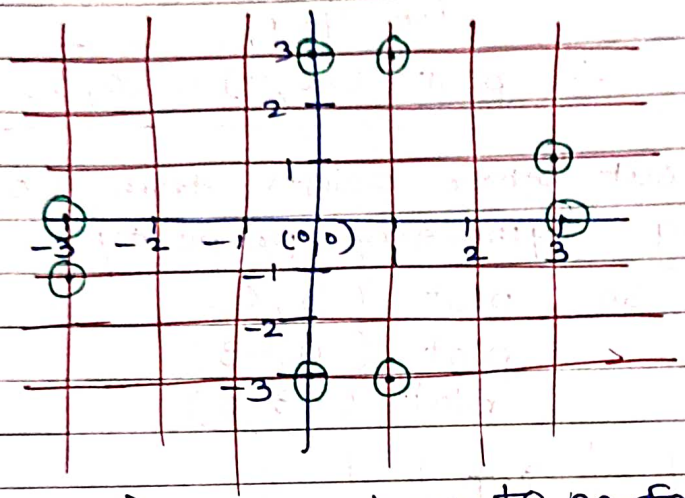
find $p = 1 - r$ ∴ $p = 1 - 3 = -2$
from (I)
Till $(x < y)$, perform following
if $p < 0$
as $p = -2$
So increase x by 1 & modify p
as $p = p + 2x + 1$
i.e $x = 0$ & $p = -2 + 2(0) + 1$
$= -2 + 1 = -1$
here we are not increasing y
∴ plot $(x+1, y)$
i.e (1, 3)

plot points from (1,3) by using symmetry property.

i.e plot (3,1)

plot (1,-3)

plot (-3,-1)



Till ($x < y$) so we have to perform

if ($p < 0$)

$-1 < 0$

So $x = 1$ & $p = p + 2x + 1$

$= (-1) + 2(1) + 1$

$= -1 + 2 + 1$

$= 2$

$p < 0 = 2 < 0 \rightarrow$ no

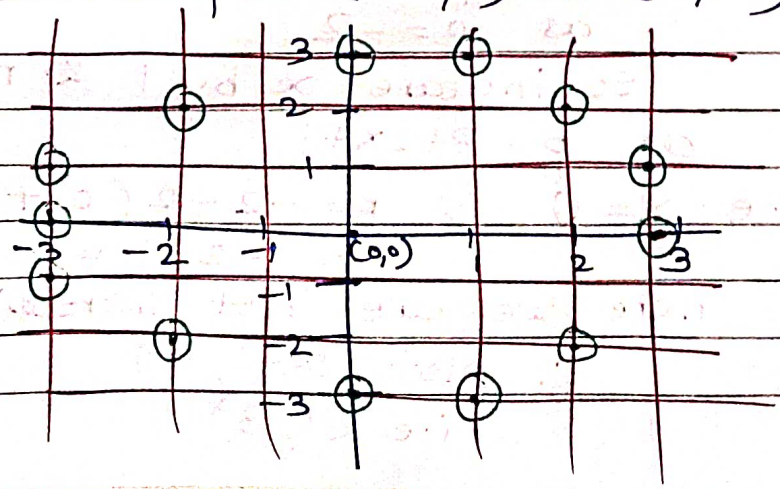
So increase x & decrease y.

$\therefore$ plot $(x+1, y-1)$

i.e $(2,2)$

$\therefore$ plot points from $(2,2)$ by using symmetry property

i.e plot $(-2,2)$ & $(2,-2)$, $(-2,-2)$



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

i.e $(x < y)$ no

so increase x & decrease y

$(x+1, y-1)$ i.e $(3, 1)$

plot $(3, 1)$

plot $(-3, 1)$

plot $(1, 3)$

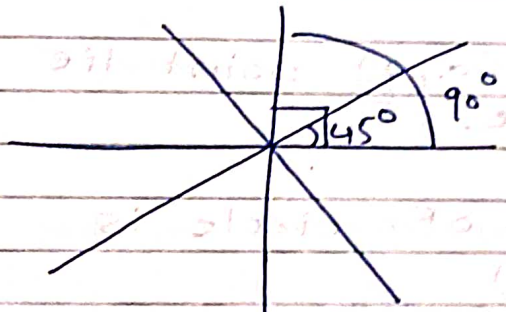
plot $(-1, 3)$

[Reference : TB1]

Bresenham Circle Generation –

– this algo does only integer arithmetic which makes it faster than floating point.

– it only creating one octant of the circle i.e from 90° to 45°.

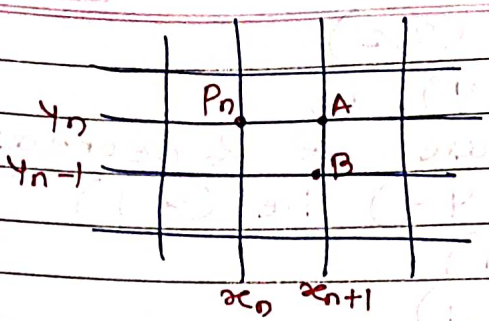


– we derive other seven octants of the circle by symmetry property.

– The circle is generated by considering center point as origin with radius 'r'

– algo calculates one new pixel per step. from any point (x_n, y_n) on the circle, the next point (x_{n+1}, y_{n+1}) must be either one on right or one to the right & down.

– If P_n is a current point with co-ordinate (x_n, y_n) then we come to know that next point could be either A or B.



point A will be having co-ordinates (x_n+1, y_n) & B will be (x_{n+1}, y_{n-1}) to select one of these two candidate pixels we have to perform some test.

— fun^n of a circle is

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

if $f(x,y) = 0$ then (x,y) point is on circle
if $f(x,y) < 0$ then (x,y) point lie inside the circle
if $f(x,y) > 0$ then (x,y) point lie outside the circle.

i.e for point A eqn of circle is

$$f(A) = f(x_n+1, y_n)$$
$$= (x_n+1)^2 + y_n^2 - r^2$$

for point B

$$f(B) = f(x_n+1, y_n-1)$$
$$= (x_n+1)^2 + (y_n-1)^2 - r^2$$

Steps for Bresenham's circle generating Algo :-

1) Accept radius & center co-ordinates from user & plot first point on circumference of circle..
$(x, y) = (0, r)$

2) Calculate the initial value of decision parameter

$$p = 3 - 2r$$

3) if we are using octant symmetry property to plot the pixels then until ($x \leq y$) we have to perform following steps :-

if ($p <= 0$)
update p by $p = p + 4x + 6$ & increase x by 1
else
update p by $p = p + 4(x - y) + 10$
& increase x by 1 & decrease y by 1.

4] determine the symmetry points in other octants also.

5] move each calculated pixel position (x, y) on to the circular path centered on (x_c, y_c) & plot the co-ordinate values as

$$X = x + x_c \quad \& \quad Y = y + y_c$$

Algorithm to plot 1/8 of the circle -

1) Read the radius (r) of the circle.
2] $p = 3 - 2r$
3] $x = 0$, $y = r$
4] do
{
plot (x, y)
if ($p <= 0$) then
{
$p = p + 4x + 6$
}
else
{
$p = p + 4(x - y) + 10$
$y = y - 1$
}

∴ $x = x+1$

} while $(x < y)$

5) stop

Example :-

plot a circle by Bresenham's algo whose radius is 3 & center co-ordinates are (0,0)

→ $r = 3,\ x_c = 0,\ y_c = 0$

plot 1st point as $(0, r)$ i.e $(0, 3)$

here $x = 0,\ y = r = 3$

→ find out rest 3 points from this by using symmetry property

plot $(0,-3)$ $(-3,0)$, $(3,0)$

→ Then find initial value of decision parameter

$$p = 3 - 2r$$

$$\therefore p = 3 - 2(3) = -3$$

As p is negative i.e $p < 0$

$x = x+1$ i.e $x = (0+1) = 1$

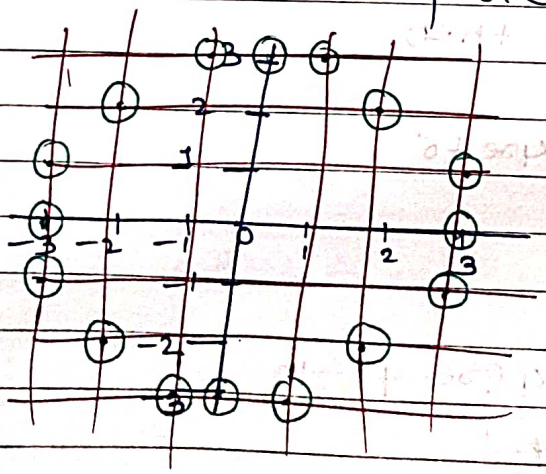
$y = 3$ &

$$p = p + 4x + 6$$

$$= -3 + 4(0) + 6$$

$$\boxed{p = +3}$$

now plot new x & y i.e $(1, 3)$ similarly continue this process till $(x < y)$



x	y
0	3
1	3
2	2
3	1

[Reference : TB1]

Character Generating Methods

- generally characters are generated by Hardware but can generate / prepare characters by software also.
- 3 primary methods for character generation.
  1) stroke / vector character generation
  2) dot matrix / bitmap method.
  3) starbust method.

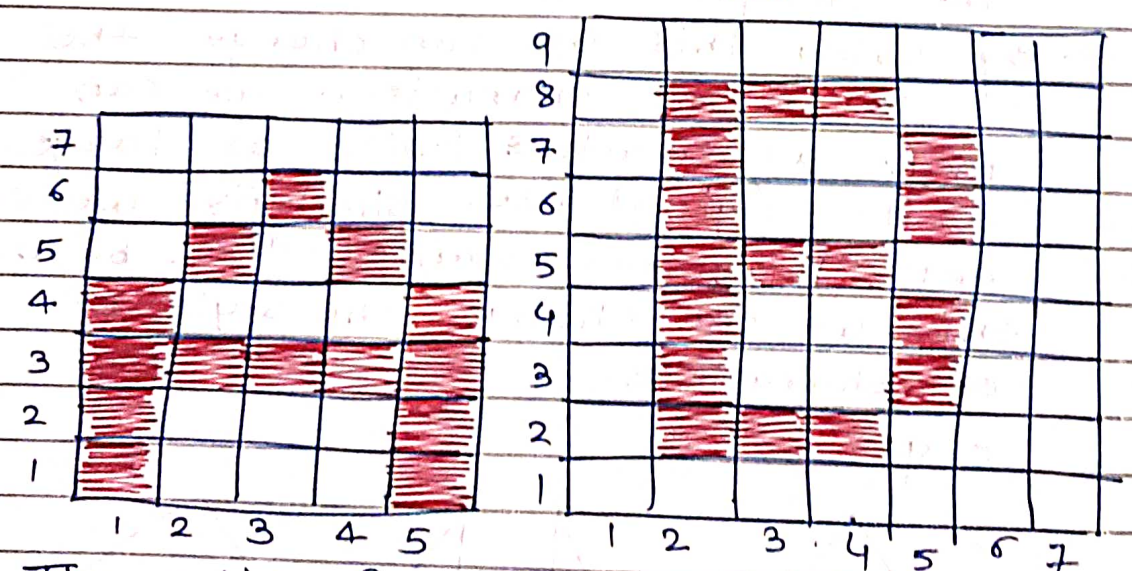
1] Stroke Method / vector character generation Method -

- This method creates characters by using a set of line segments.
- we can build our own stroke method by vector generation algo or by using any line generation method.
- To produce a character we will give a sequence of commands that defines the start point & end points of the straight lines.
- By using this we can change the scale of the characters. we can make a characters twice as large as its original size. similarly we can get characters slanted also. by using this we can change the style of of characteres.

e.g.

2) Dot-Matrix or bit Map method :-

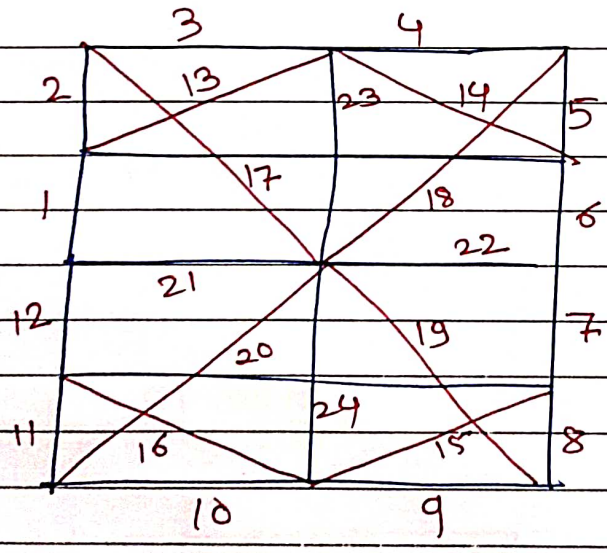
- In this method characters are represented by an array of dots, The size of this array may vary. an array of 5 dots wide & 7 dots high is generally used, but 7×9 & 9×13/14 are also used. we can select any size of array.
- This array is like a small buffer, just big enough to hold a single character.
- ~~This array is~~ The dots are the pixels of the small array.
- placing the character on the screen then becomes a matter of copying pixel values from small characters array into some portion of the screens frame buffer.
- A bitmap font uses a rectangular pattern of pixels to define each character.



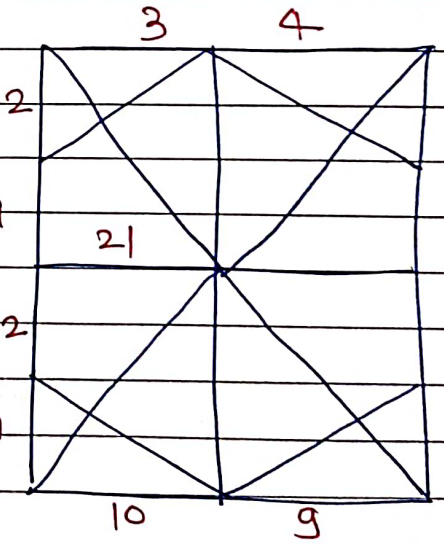
- The entire font can be loaded into an area of memory called font cache displaying character means then copying characters image from font cache into the frame buffer

at the desired position.
- Bitmap fonts require more space because each variation (size or format) must be stored.

3] Starbust Method :-
- In this method a fix pattern of line segments are wed to generate characters.
- There are 24 line segments & out of these 24 line segments, segments required to display for particular are highlighted. This method of character generation is called starbust method because of its characteristic appearance.
- The pattern for particular characters are stored in the form of 24 bit code. each bit representing one line segment.
- The bit is set to one to highlight the line segment. otherwise it is set to zero



starbust pattern of 24 line segments



starbust pattern for character E

24 bit code for character E is

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0	0	0	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1

Disadvantage –
- it requires more memory & requires additional code conversion programs to display characters from the 24 bit code.

Display files –