```
#include <graphics.h>
void main()
{
    initgraph(Graphics driver, graphics mode,
              path of graphics file);
}
```

Graphics driver → EGA, VGA, CGA, EGAVGA
CGAVGA — DETECT
default

* Basically every graphics driver is assigned with an integer value.

Pixel :

```
#include <graphics.h>
void main()
{
    int gd = DETECT, gm = 0;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
    putpixel(100, 200, WHITE);
    getch();
    closegraph();
```

Resolution: The maximum number of pixels that can be plotted on a screen without overlapping is called as resolution.

Pixel: It is the shortest picture element.

Graphics resolution

Ex:- EGA : 0, 1, 2

VGA : 0, 1

CGA : 0, 1, 2, 3

DETECT : 0, 1, 2, 3, 4

↓

640x480 (say)

In DOS Shell

Let

C:\TURBOC3\bin

↳ not support graphics file

C:\TURBOC3\bgi

\t → is used as escape character

To overcome this ambiguity we will use \\ instead of \

C supports 16 colors

Color	Value
Black	0
Blue	1
Green	2
Cyan	3
Red	4
Magenta	5
Brown	6
Light gray	7
Dark gray	8
Light blue	9
Light green	10
Light cyan	11
Light red	12
Light magenta	13
Yellow	14
White	15

Darkest - - - - - - - - - - - Lightest

Some random frequently used functions:

To plot a line:

→ Line(int x_1, int y_1, int x_2, int y_2)

Starting Point : (x_1, y_1)

Ending Point : (x_2, y_2)

Circle (int x, int y, int r)

To plot a Circle:

→ Circle(int x, int y, int r)

(x, y) : Center of Circle

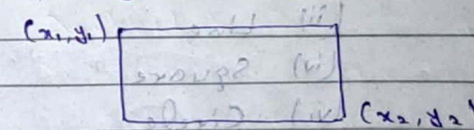
r : radius

To plot a Rectangle:

→ Rectangle (int x_1, int y_1, int x_2, int y_2)

(x_1, y_1) = Left Corner

(x_2, y_2) = Right Corner



To plot an Arc :

→ Arc (int x , int y , int r , int s_angle , int e_angle)

(x,y) : Center

r : radius

s_angle : starting angle

e_angle : ending angle

To plot an Ellipse :

→ Ellipse (int x, int y, int r1, int r2)

(x, y) : Center of Ellipse

r1 : radius in x-direction

r2 : radius in y-direction

Assignment - 1 :

Plot the following

(i) Pixel (ii) Line

(iii) Rectangle (iv) Square

(v) Triangle (vi) Circle

(vii) Circular Arc (viii) Ellipse

(ix) Elliptical Arc

Assignment 2 :

(i) set color (ii) set bk color

(iii) outtextxy

(iv) Gototext , to change cursor position

(v) Change in text styling

* setcolor (color_name) : To set a color in graphics

* In default, graphics plot in white color.

* setbkcolor (color_name) : To change background color.

* outtextxy (int x, int y, "Hi") ;

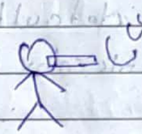
Assignment 3 :

(i) Make blinking circle
(keep increasing the radius of circle till 150 cm and then decrease it to become 0 and so on)

(ii) Collision of balls

Assignment 4 :

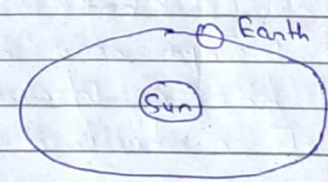
Make smoking person



Assignment 5 :

(i) Earth moving in Circular path

(ii) Earth moving in Elliptical path



Display devices :

① Cathode Ray Tube (CRT)
It is a technology which is used in traditional computer monitor and television. Cathode Ray Tube is a particular type of vacuum tube that displays images when an electron beam collides on the radiant surfaces.

* Frame buffer : frame buffer stores picture definition which is required to display on the screen.

* Picture definition : It defines that which pixels need to glow to see the picture on screen.

* Persistence : Persistance is defined as the time that it takes the emitted light from the screen to decay to one-tenth of its original intensity.
Phosphorous : persistance 10-60 microsecond.

* Refresh Rate : The number of frames draw per second on the screen is refresh rate.

* Flickers on screen : Low persistence + Refresh rate is low

(2) Color CRT Monitor :

Raster Scan display
M x N (N rows, M Columns)
Horizontal Retraces = No. of rows - 1
= N - 1
Vertical Retraces = 1

Interlacing Technique
Horizontal Retraces = No. of lines - 2
= N - 2
Vertical Retraces = 2

(i) Beam Penetration Method :

Alt to Raster Scan Method :

* Scan line : In raster scan system, electron beam is swept across the screen, one row at a time from top to bottom. Each row is referred as scan line

* Refresh Buffer / frame Buffer : Picture definition is stored in a memory area called the refresh buffer or frame buffer, where buffer frame refers to the total screen area.

* Pixel : Each screen spot that can be illuminated by the electron beam is referred to as pixel / bit.

* Color Buffer : Since the refresh buffer is used to store the set of screen color values, it is also set time called a color buffer.

Ex : Home TV, Printers

Ex : Let screen resolution : 640 x 480

$2^8 = 256$, $2^9 = 512$, $2^{10} = 1024$

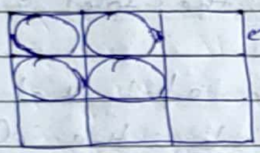
∴ X Coordinate = 10 bits, Y Coordinate = 9 bits

bits for pixel position

For black & white monitor = 1 bit.

* **Aspect ratio**: No. of pixel columns divided by the no. of scan lines is aspect ratio.

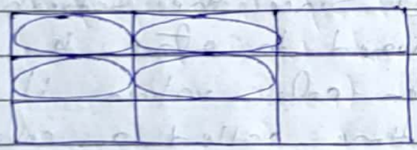
Aspect ratio = 480/640 = 0.75



← Square grid

most common definition

Aspect ratio = height / width (of pixel /Grid)
= 1 or vice versa



Aspect ratio <1

* **Depth**: The no. of bits per pixel in frame buffer is called as depth.

* **Bitmap**: A frame buffer with a bit per pixel for color is called as bitmap

* **Pixmap**: A frame buffer with multiple bits per pixel for color is called as pixmap

* **Refresh rate**: In a raster scan system no. of frame drawn per sec is called as refresh rate.

Persistence is inversely proportional to refresh rate

$$\text{Persistence} \propto \frac{1}{\text{refresh rate}}$$

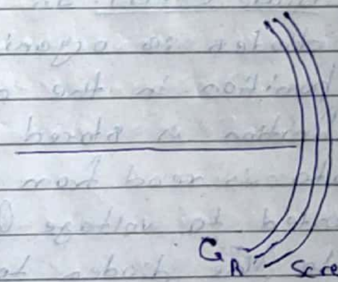
Quality of phosphorous is measured

(ii) **Shadow Mask Method**:

→ Line arrangement of electron gun

→ Triangular arrangement of electron gun.

(i) **Beam Penetration Method**:



Green: Only Green

Yellow: More Green + less red

Orange: less Green + more red

Red: Only red.

* **Plasma Technology:** A plasma display is a computer video display in which each pixel on screen is illuminated by a tiny bit of plasma or charged gas, somewhat like a tiny neon light. It is thinner than Cathode Ray tube (CRT) displays and brighter than liquid Crystal Displays (LCD).

* **Liquid Crystal Display (LCD):** LCD are the displays that produce by passing polarized light from the surrounding or from an internal light source ~~display~~ through a liquid-crystal material that transmits the light.

* **Light Emitting Diode (LED):** In an LED, a matrix of diodes is organised to form the pixel position in the display and picture definition is stored in a refresh buffer. Data is read from the refresh buffer and converted to voltage levels that are applied to the diodes to produce the light pattern in the display.

Q. How laptop is flat but not desktop and TV when Cathode Ray Tube (CRT) was used?

Soln: Both Cathode Ray tube and Monitor are parallel to each other.

* **Flat Panel devices:**
  - → Emissive
  - → Non-emissive

(i) **Emissive:** Take electral energy and convert it into graphical pattern.
Ex: Plasma, LED

(ii) **Non-Emissive:** Take light and convert it into graphical pattern.
Ex: LCD.

Numericals

① Consider 3 diff^t raster systems with resolutions of 640 x 480, 1280 x 1024 and 2560 x 2048

(i) What size frame buffer in bytes is needed in each of these systems 12 bits per pixel.

(ii) How much storage required for each of these systems 12 bits per pixel & 24 bits per pixel.

Sol^n (i) Size of frame buffer $= 12 \times 640 \times 480$ bits

$= (12 \times 640 \times 480) / 8$ bytes

(ii) Size of frame buffer $= 24 \times 640 \times 480$ bits

$= (24 \times 640 \times 480) / 8$ bytes

② Suppose an RGB raster system is to be designed using an 8 inch / 10 inch screen with resolution of 100 pixel per inch. If we want to store 6 bits / pixel in the frame buffer. How much storage in bytes do we need for the frame buffer.

Sol^n Total pixels on screen $= 8 \times 100 \times 10 \times 100$

$= 800 \times 1000$

Size of frame buffer $= (6 \times 800 \times 1000) / 8$ bytes

$= 6 \times 10^5$ bytes

③ (a) How long would it take to load a 640 x 480 frame buffer with 12 bits / sec if 10^5 bits can be transfer per se.

(b) How long would it take to load a 24 bit / pixel frame buffer with a resolution of 1280 x 1024 using this same transfer rate

Sol^n (a) Size of frame buffer $= 12 \times 640 \times 480$

Time taken $= \frac{12 \times 640 \times 480}{10^5} = \frac{3686400}{10^5} = 36.864$

(b) Size of frame buffer $= 24 \times 640 \times 480$

Time taken $= \frac{24 \times 640 \times 480}{10^5} = \frac{31457280}{10^5} = 314.5728$

④ Suppose we have a computer with 32 bits / word and transfer rate of 1 million instruction / sec (MIP). How long would it take to fill the frame buffer of a 300 dpi (dots per inch) laser printer with a page size of 8.5 inches / 11 inches

Sol^n: 300 dpi => 300 pixels per inch
∴ Size of frame buffer = $32 \times 300 \times 8.5 \times 11$
Page size = 8.5 inches × 11 inches
Total data on screen = Total pixels on screen = Resolution of screen = $300 \times 8.5 \times 11$
Size of whole page = $300 \times 8.5 \times 11 \times 32$ bits
Transfer rate = 32×10^6 sec
Time taken to print the whole page
= total data / transfer rate
= $300 \times 8.5 \times 11 / 10^6$
= 0.02805

(5) Consider two Raster System with resolution of 640×480 and 1280×1024. How many pixels could be accessed per second in each of these systems by a display controller that refreshes the screen at the rate of 60 frames/sec. What is the access time / pixel in each system.

Sol^n: Resolution = 640×480
No. of pixels accessed per second
= $640 \times 480 \times 60$
= 18432000
Access time per pixel = $\frac{1}{18432000}$

Resolution = 1280×1024
No. of pixels accessed per second
= $1280 \times 1024 \times 60$
Access time per pixel = $\frac{1}{1280 \times 1024 \times 60}$

(6) Suppose we have a video monitor with a display area measures 12 inches across 9.6 inches high. If the resolution is 1280×1024 and aspect ratio is 1. What is the diameter of each screen point.

Sol^n: 12 inches long one row include 1280 pixels
9.6 inches long one column includes 1024 pixels
Aspect ratio is 1
∴ Screen point is circular = grid is in square form

diameter of screen point = One side of square grid
= $\frac{12}{1280}$
= $\frac{9.6}{1024}$
= 0.009375 inches

(7) How much time is spent scanning across each row during refresh in raster system with resolution 1280 x 1024 and refresh rate of 60 frames/sec

Sol^n: Time taken = $\frac{1}{1024 \times 60} = 1.62 \times 10^{-5}$ sec

(8) Consider a non-interlaced raster monitor with a resolution of n x m (m is scan lines n pixels per scan line) a refresh rate of r frames/sec, a horizontal retraced time = t_{horiz} and a vertical retrace time = t_{vert}. What is the fraction of the total refresh time per frame spent in retrace of the electron beam.

Sol^n: Refresh rate = r frames/sec
time required to refresh one frame = $\frac{1}{r}$ sec

No. of retraces in plotting one frame
= (m-1) horizontal retraces + 1 vertical retraces.
Time spent in total retrace per frame
= $(m-1)\, t_{horiz} + t_{vert}$ (in sec)
fraction of time spent in retraces per frame
$= [(m-1)t_{horiz} + t_{vert}] / (1/r)$

(9) What is the fraction of the total refresh time
For a non-interlaced raster system with a resolution 1280 x 1024, a refresh rate of 60 hz, a horizontal retrace time of 5 us and a vertical retrace time of 500 us

Sol^n: fraction of time spent in retraces per frame
$= [(1024 - 1) 5 \times 10^{-6} + 500 \times 10^{-6}] / (1/60)$
$= (1023 \times 5 + 500) \times 10^{-6} \times 60$
$= (5115 + 500) \times 60 \times 10^{-6}$
$= 5615 \times 60 \times 10^{-6}$
$= 336900 \times 10^{-6}$
$= 0.3369$

Interlace Technique

Total retrace = (m-2) horizontal retraces + 2 x vertical retraces
Retrace time per frame = 1022 x 5 + 2 x 500
= 6110 usec
fraction of time in retraces = $6110 \times 10^{-6} \times 60$
$= 366600 \times 10^{-6}$
$= 0.3666$

(10) Assuming that a certain 24 bit/pixel raster system has 512 x 512. How many distinct color choices i.e. intensity level would be have available. How many colors could display at any one time.

Soln: Resolution = 512 x 512 = $2^9 \times 2^9$

No. of pixels required to identify location = 9+9 = 18

Bits left for color = 24 - 18 = 6

Max^m no. of colors available = 2^6 = 64

Colors = 64

A pixel can have one color at one time, while the picture can have 64 colors at any time.

(11) Compute the resolution per inch a 2x2 inches image that has 512 x 512 pixels.

Soln: 256 pixels per inch

Input Devices:

(1) Keyboard: text, numerical values, special characters, predefined functions, to move cursor.

(2) Button box: A set of buttons and some switches to input some predefined function.

(3) Dials: To enter scalar values. A potentiometer is used to measure dial rotation which is converted into corresponding numerical value.

(4) Mouse: A hand held device used for positioning the cursor and movement. Mouse is with wheel or roller on bottom.

Optical mouse with sensors (no ball)

Cordless Mouse: Communication with processor using digital ratio technology. Buttons at the top of mouse to invoke diff^t functions to position the cursor.

Z-Mouse: Three buttons and trackball on it.

(5) Track ball: A handheld ball, A potentiometer measures movement at cursor on screen.

(6) Space ball: A handheld device. Amount of pressure on spaceball decides input for spatial position and orientation of ball pulled / pushed decides the direction.

Ex: Useful in Virtual reality (VR).

(7) Joystick: A Vertical lever with button at top.

(8) Data Gloves: In 3D in Virtual Reality.
Gloves have sensors to sense finger motion / movement.
Electromagnetic Coupling is there b/w transmitting antennas and receiving antenna. Information provided is direction of finger and position.

(9) Digitizer: Used for drawing, painting or interactively selecting positions Input coordinate values

Ex: Tablets with stylus.

(10) Touch panel:
- Optical
- Electrical
- Acoustical

Optical: It implies a line of infrared light omitting diodes along horizontal and vertical edges. Light detectors are placed along opposite vertical and horizontal edges. These detectors record which beams are interrupted when touch the panel.

Electrical: It is constructed with two transparent plates seperated by a small distance. One of the plates is coated with a conducting material and another with resistive material. When outer plate is touched, it is forced into contact with another plate. This contact creates a voltage drop across the resistive plate that is converted to the coordinate values of the selected screen positions.

Acoustical: High frequency waves. High frequency sound waves are generated in horizontal and vertical directions across a glass plate. Touching the screen causes part of each wave to be reflected.

(11) **Image Scanner**: Optical scanning mechanism.

(12) **Light pen**: Light pen is sensitive to the light emitted from phosphor in CRT screen.

(13) **Voice System**: Speech recognizer softwares is used to input data and graphics operation. The system operates by matching an input against a predefined dictionary of words and phrases.

(14) **Webcam**: Light on Object.

Drawing Objects

* **Pixel / Point**:
Plotpixel (x coordinate, y coordinate, color/intensity)

* **Line**:
Line equation is

$$y = mx + b \quad \text{—— (1)}$$

m = slope of line
b = intercept on y-axis

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

This line is infinite passing through (x_1, y_1) & (x_2, y_2)

To draw line segment we should have starting point (x_0, y_0) and ending point (x_{end}, y_{end}) → {Line from left to right}

$$\text{Slope, } m = \frac{y_{end} - y_0}{x_{end} - x_0} = \frac{\Delta y}{\Delta x} \quad \text{—— (2)}$$

From eq^n (1)

$$b = y_0 - m x_0 \quad \text{—— (3)}$$

DDA algorithm (Digital Differential Analyzer):

It is a scan conversion line algorithm.

Case 1:

If $|m| \le 1$, $\tan\theta \le 1$ $\Rightarrow \theta \le 45°$

$\Delta x = 1$ and calculate successive y value from eq^n (2) as $m = \Delta y$

or, $m = y_{k+1} - y_k$ ——— (4)

Case 2: $|m| > 1$, $\Delta y = 1$ and calculate successive x value from eq^n (2)

$\Delta x = 1/m$

$\Rightarrow x_{k+1} - x_k = 1/m$

$\Rightarrow x_{k+1} = x_k + 1/m$ ——— (5)

If we plot right to left

for $|m| \le 1$, $\Delta x = -1$ then from eq^n (2)

$m = -\Delta y$

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

$\Rightarrow y_{k+1} = y_k - m$ ——— (6)

for $|m| > 1$, $\Delta y = -1$ then from eq^n (2)

$m = -1/\Delta x$

$\Rightarrow x_{k+1} - x_k = -1/m$

$\Rightarrow x_{k+1} = x_k - 1/m$ ——— (7)

DDA Algorithm:

```
line DDA (int x0, int y0, int xend, int yend)
{
    int dx, dy, steps, k;
    float xincr, yincr, x, y;
    dx = xend - x0;
    dy = yend - y0;
    x = x0;
    y = y0;
    if (abs(dx) > abs(dy))
        steps = abs(dx);
    else
        steps = abs(dy);
    xincr = dx / steps;
    yincr = dy / steps;
    putpixel( round(x), round(y), int color);
    for (k=0; k < steps; k++)
    {
        x = x + xincr;
        y = y + yincr;
        putpixel( round(x), round(y), int color);
    }
}
```

Ex: (10,10) , (20,18)

$dx = 20 - 10 = 10$

$dy = 18 - 10 = 8$

$x = 10,\ y = 10$ (steps = 10)

$x_{incr} = 10/10 = 1$, $y_{incr} = 8/10 = 0.8$

k	(x, y)	Plot Pixel
	10, 10	10, 10
0	11, 10.8	11, 11
1	12, 11.6	12, 12
2	13, 12.4	13, 12
3	14, 13.2	14, 13
4	15, 14	15, 14
5	16, 14.8	16, 15
6	17, 15.6	17, 16
7	18, 16.4	18, 16
8	19, 17.2	19, 17
9	20, 18	20, 18

Bresenham's Line Algorithm:

If absolute value of slope of the line is lesser than 1 i.e. $|m| < 1$

Suppose current pixel is (x_k, y_k) at any step k. Then, the next possible pixel choices are (x_{k+1}, y_k) or (x_k+1, y_k+1)

At sample step k+1

$y = m(x_k + 1) + b$ ——— (1)

Calculate $d_1 = y - y_k = m(x_k + 1) + b - y_k$ ——— (2)

Calculate $d_2 = (y_k + 1) - y$

$= y_k + 1 - m(x_k + 1) - b$ ——— (3)

Perform (2) - (3)

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

$= 2m(x_k + 1) - 2y_k + 2b - 1$ ——— (4)

put $m = \Delta y / \Delta x$, where

$\Delta y = y_{end} - y_0$

$\Delta x = x_{end} - x_0$

(x_0, y_0) and (x_{end}, y_{end}) are starting & ending points of line segment

then $d_1 - d_2 = 2(\Delta y / \Delta x)(x_k + 1) - 2y_k + 2b - 1$

$\Rightarrow \Delta x(d_1 - d_2) = 2\Delta y(x_k + 1) - 2\Delta x y_k + 2\Delta x b - \Delta x$

Put $\Delta x(d_1 - d_2) = p_k$ then

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

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

$$\Rightarrow p_k = 2\Delta y\, x_k - 2\Delta x\, y_k + c \quad \text{——— (5)}$$

At step $k+1$,

$$p_{k+1} = 2\Delta y\, x_{k+1} - 2\Delta x\, y_{k+1} + c \quad \text{——— (6)}$$

perform (6) − (5)

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

$$\Rightarrow p_{k+1} = p_k + 2\Delta y - 2\Delta x(y_{k+1} - y_k) \quad \text{——— (7)}$$

$$(\because x_{k+1} = x_k + 1)$$

At $k=0$, current pixel is (x_0, y_0)

Put $k=0$ in eq^n (5)

$$p_0 = 2\Delta y\, x_0 - 2\Delta x\, y_0 + c$$

$$\Rightarrow p_0 = 2\Delta y\, x_0 - 2\Delta x\, y_0 + 2\Delta y + 2\Delta x b - \Delta x$$

$$\Rightarrow p_0 = 2\Delta y\, x_0 - 2\Delta x((\Delta y/\Delta x)\cdot x_0 + b) + 2\Delta y + 2\Delta x b - \Delta x$$

$$\Rightarrow p_0 = 2\Delta y\, x_0 - 2\Delta y\, x_0 - 2\Delta x\, b + 2\Delta y - \Delta x$$

$$\Rightarrow p_0 = 2\Delta y - \Delta x \quad \text{——— (8)}$$

If $p_k < 0$ i.e. $d_1 - d_2 < 0$

then next pixel is $(x_k + 1, y_k)$

Put $y_{k+1} = y_k$ in eq^n (7)

$$p_{k+1} = p_k + 2\Delta y \quad \text{——— (9)}$$

Otherwise, if $(p_k \geq 0)$, then next pixel is $(x_k + 1, y_k + 1)$

Put $y_{k+1} = y_k + 1$ in eq^n (7)

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x \quad \text{——— (10)}$$

Bresenham's Line Algorithm : ($|m| < 1$)

(1) Input two end points. Store left end points in (x_0, y_0)

(2) Plot (x_0, y_0)

(3) Calculate Δx, Δy, $2\Delta x$, $2\Delta y$, $2\Delta y - 2\Delta x$ and also calculate $p_0 = 2\Delta y - \Delta x$

(4) At each step k for current (x_k, y_k), if $p_k < 0$ then plot next pixel $(x_k + 1, y_k)$
Also calculate $p_{k+1} = p_k + 2\Delta y$

Otherwise, plot next pixel $(x_k + 1, y_k + 1)$ and calculate
$p_{k+1} = p_k + 2\Delta y - 2\Delta x$

(5) Repeat step (4) Δx times.

Ex: $(20,10), (30,18)$

$(x_0, y_0) = (20,10)$

$\Delta x = 30 - 20 = 10$, $\Delta y = 18 - 10 = 8$

$2\Delta x = 20$, $2\Delta y = 16$

$p_0 = 2\Delta y - \Delta x = 16 - 10 = 6$

k	Plot pixel	p_k
0	(20,10)	6
1	(21,11)	2
2	(22,12)	-2
3	(23,12)	14
4	(24,13)	10
5	(25,14)	6
6	(26,15)	2
7	(27,16)	-2
8	(28,16)	14
9	(29,17)	10
10	(30,18)	6

If $|m| > 1$

$\Delta y > \Delta x$

If the current pixel at step k is (x_k, y_k), then the next possible pixels are $(x_k, y_k + 1)$ or $(x_k + 1, y_k + 1)$

$y_k + 1 = mx + b$

$\Rightarrow x = \Delta x(y_k + 1 - b)/\Delta y$ —— (1)

$d_1 = x - x_k$

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

$d_2 = x_k + 1 - x$

$= x_k + 1 - \Delta x(y_k + 1 - b)/\Delta y$ —— (3)

Perform (2) - (3)

$d_1 - d_2 = \left(2\Delta x(y_k + 1 - b)/\Delta y\right) - 2x_k - 1$

$\Rightarrow \Delta y(d_1 - d_2) = 2\Delta x(y_k + 1 - b) - 2\Delta y x_k - \Delta y$

$\Rightarrow p_k = 2\Delta x y_k - 2\Delta y x_k + 2\Delta x - \Delta y - 2\Delta x b$ —— (4)

Similarly,

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

Perform (5) - (4)

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

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

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

If $p_k < 0$

$x_{k+1} = x_k$

then eq^n (6) reduces to

$p_{k+1} = p_k + 2\Delta x$ ——— (7)

Otherwise, If $p_k > 0$, then

~~x_{k+1}~~

$x_{k+1} = x_k + 1$ then eq^n (6) reduces to

$p_{k+1} = p_k + 2\Delta x - 2\Delta y$ ——— (8)

If initial pixel is (x_0, y_0) then from eq^n (4)

$p_0 = 2\Delta x y_0 - 2\Delta y x_0 + 2\Delta x - \Delta y - 2\Delta x b$

$= 2\Delta x y_0 - 2\Delta y x_0 + 2\Delta x - \Delta y - 2\Delta x(y_0 - \Delta y/\Delta x \cdot x_0)$

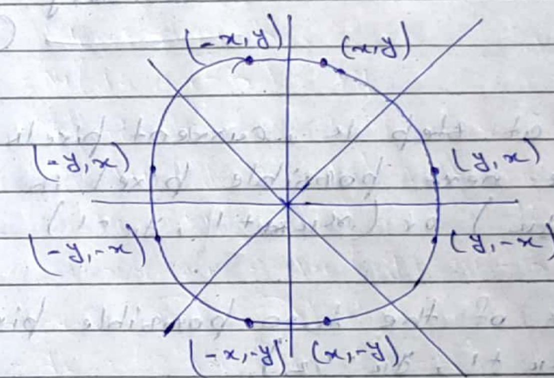
$= 2\Delta x y_0 - 2\Delta y x_0 + 2\Delta x - \Delta y - 2\Delta x y_0 + 2\Delta y x_0$

$p_0 = 2\Delta x - \Delta y$ ——— (9)

Circle

Midpoint Circle Algorithm:

Symmetry Aspect



Let us say center of circle is (x_c, y_c) and radius r, then circle eq^n is

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

If center of circle is $(0,0)$ then eq^n of circle reduces to

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

$\Rightarrow x^2 + y^2 - r^2 = 0$

$\therefore f_{circle} = x^2 + y^2 - r^2$ ——— (1)

$f_{circle}(x,y) < 0$, if (x,y) is inside the circle boundary

$= 0$, if (x,y) is on the circle boundary

> 0, if (x,y) is outside the circle boundary —— (2)

Suppose at step k, current pixel is (x_k, y_k) then the next possible pixel is either (x_k+1, y_k) or (x_k+1, y_k-1)

Midpoint of the two possible pixels is $(x_k+1, y_k-1/2)$

from eqn (1)

$p_k = f_{circle}(x_k+1, y_k-1/2)$

$\Rightarrow p_k = (x_k+1)^2 + (y_k-1/2)^2 - r^2$ —— (3)

Similarly, at step $(k+1)$

$p_{k+1} = (x_{k+1}+1)^2 + (y_{k+1}-1/2)^2 - r^2$ —— (4)

Perform (4) - (3)

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

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

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

$\Rightarrow p_{k+1} = p_k + 2x_k + 3 + (y_{k+1}^2 - y_k^2) - (y_{k+1}-y_k)$ —— (5)

If $p_k < 0$, midpoint is inside the circle i.e. (x_k+1, y_k) is closer to the boundary so pixel (x_k+1, y_k) will be plotted and put $y_{k+1} = y_k$ in eqn (5)

$p_{k+1} = p_k + 2x_k + 3$ —— (6)

$\Rightarrow p_{k+1} = p_k + 2x_{k+1} + 1$ —— (6)

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

Otherwise, midpoint is outside the circle i.e. (x_k+1, y_k-1) is closer to the boundary. The pixel (x_{k+1}, y_{k-1}) will be plotted & put $y_{k+1} = y_k - 1$ in eqn (5)

$p_{k+1} = p_k + 2x_k + 3 + \{(y_k-1)^2 - (y_k)^2\} - (y_k - 1 - y_k)$

$\Rightarrow p_{k+1} = p_k + 2x_k + 3 + 2y_k + 1 + 1$

$= p_k + 2x_k + 2y_k + 5$

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

$p_{k+1} = p_k + 2x_{k+1} - 2y_{k+1} + 1$ —— (7)

$\left\{ \begin{array}{l} x_{k+1} = x_k + 1 \\ y_{k+1} = y_k - 1 \end{array} \right\}$

Initial pixel is $(0, r)$

The next pixel may be $(1, r)$ or $(1, r-1)$

Midpoint $= \left(1, r - \frac{1}{2}\right)$

$p_0 = 1 + \left(r - \frac{1}{2}\right)^2 - r^2$

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

$= 1 - r + \frac{1}{4}$

$= \frac{5}{4} - r$

$p_0 = 1 - r$ —— (8)

Algorithm:

1. Input radius r and circle center (x_c, y_c) then set the coordinates for the first point on the circumference of a circle centered on the origin as $(x_0, y_0) = (0, r)$
2. Calculate initial decision parameter $p_0 = \left(\frac{5}{4} - r\right)$ or simply $(1 - r)$
3. At each step k (starting from 0), for current pixels (x_k, y_k) if $p_k < 0$ then next pixel along centre $(0,0)$ is $(x_k + 1, y_k)$ and $p_{k+1} = p_k + 2x_{k+1} + 1$

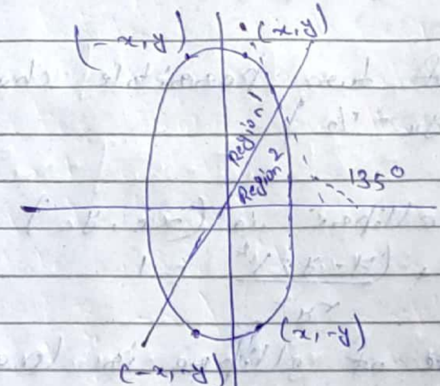
   Otherwise, the next pixel is $(x_k + 1, y_k - 1)$ and $p_{k+1} = p_k + 2x_{k+1} - 2y_{k+1} + 1$
4. Determine symmetry points in other seven octants.
5. Move each calculated pixel (x, y) considering centre at (x_c, y_c) by following eq^{n}s $x = x + x_c$ and $y = y + y_c$
6. Repeat step 3-5 until $x <= y$

eg. $(x_c, y_c) = (200, 100)$, $r = 10$, $(x_0, y_0) = (0, 10)$

$p_0 = 1 - r = 1 - 10 = -9$

k	p_k	(x_{k+1}, y_{k+1}) along (0,0)	(x_{k+1}, y_{k+1}) along (200,100)
0	-9	(1,10)	(201, 110)
1	-6	(2,10)	(202, 110)
2	-1	(3,10)	(203, 110)
3	6	(4,9)	(204, 109)
4	-3	(5,9)	(205, 109)
5	8	(6,8)	(206, 108)
6	5	(7,7)	(207, 107)

Ellipse Mid-point drawing Algorithm :



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

Region 1 :

At some ~~point~~ step k if current pixel is (x_k, y_k) then next pixels may be $(x_k + 1, y_k)$ or $(x_k + 1, y_k - 1)$

Region 2 :

(i) At some step k if current pixel is (x_k, y_k) then next pixels may be $(x_k, y_k - 1)$ or $(x_k + 1, y_k - 1)$.
Initial pixel is the last point of Region 1.

(ii) $(x_0, y_0) = (r_x, 0)$
At some step k if current pixel is (x_k, y_k)

then next pixel may be (x_k, y_k+1) or (x_k-1, y_k+1).

Midpoint of two possible choices are $(x_k+1, y_k-1/2)$

Centre of ellipse is (x_c, y_c)

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

When Centre of ellipse is $(0,0)$

$$\frac{x^2}{r_x^2} + \frac{y^2}{r_y^2} = 1 \quad \text{—— (1)}$$

Multiply whole eqn by $r_x^2 r_y^2$

$\Rightarrow r_y^2 x^2 + r_x^2 y^2 = r_x^2 r_y^2$

$\Rightarrow r_y^2 x^2 + r_x^2 y^2 - r_x^2 r_y^2 = 0$

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

$f_{ellipse}(x,y) < 0$ if (x,y) is inside ellipse boundary

$= 0$ " " " on "

> 0 " " " outside " —— (3)

Diff. eqn (1), we get

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

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

$\Rightarrow -1 = -\frac{r_y^2 x}{r_x^2 y}$ (∵ At meeting point of region 1 and region 2 tangent makes 135° angle)

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

$r_y^2 x / r_x^2 y < -1$ when we are in region 1

$\Rightarrow r_y^2 x > r_x^2 y$

Now find recursive decision parameter:

Region 1:

The decision parameter at step k is

$(p1)_k = f_{ellipse}(x_k+1, y_k-1/2)$

$= r_y^2(x_k+1)^2 + r_x^2(y_k-1/2)^2 - r_x^2 r_y^2$ —— (4)

Similarly, at step $(k+1)$, the decision parameter is

$(p1)_{k+1} = r_y^2(x_{k+1}+1)^2 + r_x^2(y_{k+1}-1/2)^2 - r_x^2 r_y^2$ —— (5)

Perform (5) − (4)

$(p1)_{k+1} - (p1)_k = r_y^2[(x_{k+1}+1)^2 - (x_k+1)^2] + r_x^2[(y_{k+1}-1/2)^2 - (y_k-1/2)^2]$

$\Rightarrow (p1)_{k+1} = (p1)_k + r_y^2[x_{k+1}^2 + 2x_{k+1} + 1 - x_k^2 - 2x_k - 1] + r_x^2[y_{k+1}^2 - y_{k+1} + \frac{1}{4} - y_k^2 + y_k - \frac{1}{4}]$

$$\Rightarrow (p_1)_{k+1} = (p_1)_k + r_y^2\left[x_{k+1}^2 + 2x_{k+1} - x_k^2 - 2x_k\right] + r_x^2\left[y_{k+1}^2 - y_{k+1} - y_k^2 + y_k\right]$$

$$\Rightarrow (p_1)_{k+1} = (p_1)_k + r_y^2\left[x_{k+1}^2 - x_k^2\right] + 2r_y^2\left[x_{k+1} - x_k\right] + r_x^2\left[y_{k+1}^2 - y_k^2\right] - r_x^2\left[y_{k+1} - y_k\right]$$

Put $(x_{k+1} = x_k + 1)$

$$(p_1)_{k+1} = (p_1)_k + r_y^2\left[x_k^2 + 2x_k + 1 - x_k^2\right] + 2r_y^2(x_k + 1 - x_k) + r_x^2(y_{k+1}^2 - y_k^2) - r_x^2(y_{k+1} - y_k)$$

$$(p_1)_{k+1} = (p_1)_k + 2r_y^2 x_k + 3r_y^2 + r_x^2(y_{k+1}^2 - y_k^2) - r_x^2(y_{k+1} - y_k)$$

$$\Rightarrow (p_1)_{k+1} = (p_1)_k + 2r_y^2 x_{k+1} + r_y^2 + r_x^2(y_{k+1}^2 - y_k^2) - r_x^2(y_{k+1} - y_k) \quad \text{—— (6)}$$

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

If $(p_1)_k < 0$, then the next pixel plotted is $(x_k + 1, y_k)$ and put it in eqn (6)

$$(p_1)_{k+1} = (p_1)_k + 2r_y^2 x_{k+1} + r_y^2 \quad \text{—— (7)}$$

Otherwise, next pixel plotted is $(x_{k+1}, y_k - 1)$ and put $y_{k+1} = y_k - 1$ in eqn (6)

$$(p_1)_{k+1} = (p_1)_k + 2r_y^2 x_{k+1} + r_y^2 + r_x^2\left((y_k - 1)^2 - y_k^2\right) - r_x^2(y_k - 1 - y_k)$$

$$= (p_1)_k + 2r_y^2 x_{k+1} + r_y^2 - 2r_x^2 y_k + r_x^2 + r_x^2$$

$$= (p_1)_k + 2r_y^2 x_{k+1} + r_y^2 - 2r_x^2 y_k + 2r_x^2$$

$$= (p_1)_k + 2r_y^2 x_{k+1} + r_y^2 - 2r_x^2(y_k - 1)$$

$$(p_1)_{k+1} = (p_1)_k + 2r_y^2 x_{k+1} + r_y^2 - 2r_x^2 y_{k+1}$$

$$(p_1)_{k+1} = (p_1)_k + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_y^2 \quad \text{—— (8)}$$

Initial pixel is $(0, r_y)$

Next possible pixels are $(1, r_y)$ and $(1, r_y - 1)$

Midpoint of two possible pixels is $(1, r - \frac{1}{2})$

So, initial decision parameter is

$$p_0 = f_{ellipse}(1, r_y - \tfrac{1}{2})$$
$$= r_y^2 \cdot 1^2 + r_x^2 (r_y - \tfrac{1}{2})^2 - r_x^2 r_y^2$$
$$= r_y^2 + r_x^2 (r_y^2 - r_y + \tfrac{1}{4}) - r_x^2 r_y^2$$
$$= r_y^2 + r_x^2 r_y^2 - r_y r_x^2 + \frac{r_x^2}{4} - r_x^2 r_y^2$$

$$p_0 = r_y^2 - r_x^2 r_y + \frac{r_x^2}{4} \quad \text{--- (9)}$$

Region 2 :

At some ~~point~~ step k if current pixel is (x_k, y_k) then next pixels may be $(x_k, y_k - 1)$ or $(x_k + 1, y_k - 1)$.

Midpoint of two possible choices $(x_k + \frac{1}{2}, y_k - 1)$.

The decision parameter at step k is

$$(p2)_k = f_{ellipse}(x_k + \tfrac{1}{2}, y_k - 1)$$

$$(p2)_k = r_y^2 (x_k + \tfrac{1}{2})^2 + r_x^2 (y_k - 1)^2 - r_x^2 r_y^2 \quad \text{--- (10)}$$

Similarly, at step k+1, the decision parameter is

$$(p2)_{k+1} = r_y^2 (x_{k+1} + \tfrac{1}{2})^2 + r_x^2 (y_{k+1} - 1)^2 - r_x^2 r_y^2 \quad \text{--- (11)}$$

Perform (11) − (10)

$$(p2)_{k+1} - (p2)_k = r_y^2 \left[(x_{k+1} + \tfrac{1}{2})^2 - (x_k + \tfrac{1}{2})^2 \right] + r_x^2 \left[(y_{k+1} - 1)^2 - (y_k - 1)^2 \right]$$

$$\Rightarrow (p2)_{k+1} = (p2)_k + r_y^2 \left[x_{k+1}^2 + x_{k+1} - x_k^2 - x_k \right] + r_x^2 \left[y_{k+1}^2 - 2y_{k+1} - y_k^2 + 2y_k \right]$$

$$\Rightarrow (p2)_{k+1} = (p2)_k + r_y^2 \left[x_{k+1}^2 - x_k^2 \right] + r_y^2 \left[x_{k+1} - x_k \right] + r_x^2 \left[y_{k+1}^2 - y_k^2 \right] - 2r_x^2 \left[y_{k+1} - y_k \right]$$

Put $(y_{k+1} = y_k - 1)$

$$\Rightarrow (p2)_{k+1} = (p2)_k + r_y^2 \left[x_{k+1}^2 - x_k^2 \right] + r_y^2 \left[x_{k+1} - x_k \right] + r_x^2 \left[y_k^2 - 2y_k + 1 - y_k^2 \right] - 2r_x^2 \left[y_k - 1 - y_k \right]$$

$$\Rightarrow (p2)_{k+1} = (p2)_k + r_y^2 \left[x_{k+1}^2 - x_k^2 \right] + r_y^2 \left[x_{k+1} - x_k \right] - 2r_x^2 y_k + r_x^2 + 2r_x^2$$

$$\Rightarrow (p2)_{k+1} = (p2)_k + r_y^2 \left[x_{k+1}^2 - x_k^2 \right] + r_y^2 \left[x_{k+1} - x_k \right] - 2r_x^2 y_k + 3r_x^2$$

$$\Rightarrow (p_2)_{k+1} = (p_2)_k + r_y^2 (x^2_{k+1} - x^2_k) + r_y^2 (x_{k+1} - x_k) - 2r_x^2 (y_k - 1) + r_x^2$$

$$\Rightarrow (p_2)_{k+1} = (p_2)_k + r_y^2 (x^2_{k+1} - x^2_k) + r_y^2 (x_{k+1} - x_k) - 2r_x^2 y_{k+1} + r_x^2$$

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

(12)

If $(p_2)_k < 0$, then next pixel plotted is $(x_k + 1, y_k - 1)$ and put $(x_{k+1} = x_k + 1)$ in eq^n (12)

$$(p_2)_{k+1} = (p_2)_k + r_y^2 (x^2_k + 2x_k + 1 - x_k^2) + r_y^2 - 2r_x^2 y_{k+1} + r_x^2$$

$$\Rightarrow (p_2)_{k+1} = (p_2)_k + 2r_y^2 (x_k + 1) - 2r_x^2 y_{k+1} + r_x^2$$

$$(p_2)_{k+1} = (p_2)_k + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_x^2$$

(13)

Otherwise, the next pixel plotted is $(x_k, y_k - 1)$ and put $x_{k+1} = x_k$ in eq^n (12), then eq^n (12) reduces to

$$(p_2)_{k+1} = (p_2)_k - 2r_x^2 y_{k+1} + r_x^2$$

(14)

Suppose
Initial pixel of Region 2 is the last pixel of Region 1. Suppose starting pixel of Region 2 is (x_0, y_0)

Then next possible pixels are $(x_0, y_0 - 1)$ or $(x_0 + 1, y_0 - 1)$
Then midpoint is $(x_0 + 1/2, y_0 - 1)$

So, Initial decision parameter of Region 2 is
$(p_2)_0 = f_{ellipse}(x_0 + 1/2, y_0 - 1)$

$$(p_2)_0 = r_y^2 (x_0 + 1/2)^2 + r_x^2 (y_0 - 1)^2 - r_x^2 r_y^2$$

(15)

Algorithm :

1. Input (r_x, r_y) and ellipse centre (x_c, y_c)
Obtain the first point on ellipse as $(0, r_y)$ along the centre $(0,0)$

2. Calculate initial decision parameter in Region 1 as
$(p1)_0 = r_y^2 - r_x^2 r_y + r_x^2/4$

3. At each x_k position in Region 1, starting at $k=0$, perform the following test :
→ If $(p1)_k < 0$, then the next pixel along centre $(0,0)$ is (x_k+1, y_k) and
$(p1)_{k+1} = (p1)_k + 2r_y^2 x_{k+1} + r_y^2$
→ Otherwise, the next pixel along centre $(0,0)$ is (x_k+1, y_k-1) and
$(p1)_{k+1} = (p1)_k + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_y^2$
with
$x_{k+1} = x_k + 1$, $y_{k+1} = y_k - 1$
Continue while $(r_y^2 x > r_x^2 y)$

4. Calculate initial decision of Region 2 as
$(p2)_0 = r_y^2 (x_0 + 1/2)^2 + r_x^2 (y_0 - 1) - r_x^2 r_y^2$

The last pixel of Region 1 is the first pixel of Region 2

5. At each x_k position in Region 2, starting at $k=0$, perform the following test :
If $(p2)_k > 0$, then the next pixel along centre $(0,0)$ is $(x_k, y_k - 1)$ and
$(p2)_{k+1} = (p2)_k - 2r_x^2 y_{k+1} + r_x^2$

Otherwise, the next pixel along center $(0,0)$ is $(x_k, y_k - 1)$ and
$(p2)_{k+1} = (p2)_k + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_x^2$

Continue until $y=0$

6. For both the regions, determine symmetry points in the other three quadrants.

7. Move each calculated pixel position along the center (x_c, y_c) and plot them
$x = x + x_c$, $y = y + y_c$