University of Vavuniya

Computer Graphics

Prepared By

Mr. V. Vinoharan

Department of Information and Communication Technology
Faculty of Technological Studies

Contents

Chapter 1

Introduction

- ❋ Computer Graphics is the creation of pictures with the help of a computer. It comprises of software techniques to create, store, modify, represents pictures.
- ❋ Interactive computer graphics work using the concept of two-way communication between computer users.

1.1 Application of Computer Graphics

- ❋ Use in Biology: Molecular biologist can display a picture of molecules and gain insight into their structure with the help of computer graphics.
- ❋ Computer-Generated Maps: Town planners and transportation engineers can use computer-generated maps which display data useful to them in their planning work.
- ❋ Architect: Architect can explore an alternative solution to design problems at an interactive graphics terminal.
- ❋ Computer Art: Used in the field of commercial arts. It is used to generate television and advertising commercial.
- ❋ Entertainment: Used in making motion pictures, music videos and television shows.
- ❋ Visualization: It is used for visualization of scientists, engineers, medical personnel, business analysts for the study of a large amount of information.
- ❋ Educational Software: Used in the development of educational software for making computer-aided instruction.
- ❋ Printing Technology: Used for printing technology and textile design.
- ❋ Example of Computer Graphics Packages: LOGO, COREL DRAW, AUTO CAD, 3D STUDIO, CORE, GKS (Graphics Kernel System), PHIGS, CAM (Computer Graphics Metafile), and CGI (Computer Graphics Interface).

1.2 Interactive and Passive Graphics

- ❋ Non-Interactive or Passive Computer Graphics:
  - ☆ The picture is produced on the monitor, and the user does not have any controlled over the image. One example of its Titles shown on T.V.
  - ☆ User can see the produced image, and he cannot make any change in the image.
- ❋ Interactive Computer Graphics:
  - ☆ In interactive Computer Graphics user have some controls over the picture. One example of it is the ping-pong game.
  - ☆ Require two-way communication between the computer and the user. A User can see the image and make any change by sending his command with an input device.

- Advantages:
  - ☆ Higher Quality
  - ☆ More precise results or products
  - ☆ Greater Productivity
  - ☆ Lower analysis and design cost
  - ☆ Significantly enhances our ability to understand data.
- Frame Buffer: A digital frame buffer is large, contiguous piece of computer memory used to hold or map the image displayed on the screen.
  - ☆ At a minimum, there is 1 memory bit for each pixel in the raster. This amount of memory is called a bit plane.
  - ☆ The picture is built up in the frame buffer one bit at a time.
  - ☆ A memory bit has only two states (binary 0 or 1), a single bit plane yields a black and white (monochrome display).
  - ☆ As frame buffer is a digital device write raster CRT is an analog device.

1.3 Properties of Video Monitor:

- Phosphorescence is the term used to characterize the light given off by a phosphor after it has been exposed to an electron beam.
- *Persistence:* Persistence is the duration of phosphorescence. i.e., how they continue to emit light after removing the electron beam.
- *Resolution:* Use to describe the number of pixels that are used on display image.
- *Aspect Ratio:* It is the ratio of width to its height.

$$Aspect\ Ratio = \frac{width\ unit}{height\ unit}$$

1.4 Display Processor

- It is interpreter or piece of hardware that converts display processor code into pictures.
- Figure 1.1 shows the block diagram of display system.

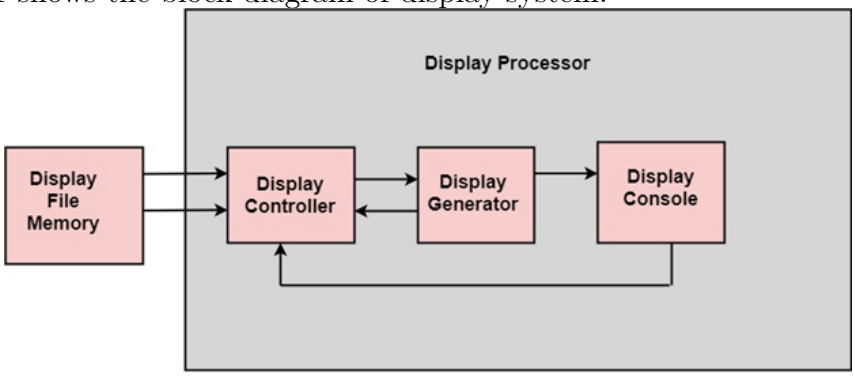


Figure 1.1: Block Diagram of Display System

- Parts of Display Processor

- ☆ *Display File Memory:* It is used for identification of graphic entities.
- ☆ *Display Controller:* It handles interruptions, maintains timings, and is used for interpretation of instruction.
- ☆ *Display Generator:* It is used for the generation of character and the generation of curves.
- ☆ *Display Console:* It contains CRT, Light Pen, and Keyboard and deflection system.

❋ It consists of the control processing unit (CPU) and a particular processor called a display controller. Display Controller controls the operation of the display device. It is also called a video controller.

❋ Working: The video controller in the output circuitry generates the horizontal and vertical drive signals so that the monitor can sweep. Its beam across the screen during raster scans.

❋ Figure 1.2 shows architecture of a raster display system with display processor

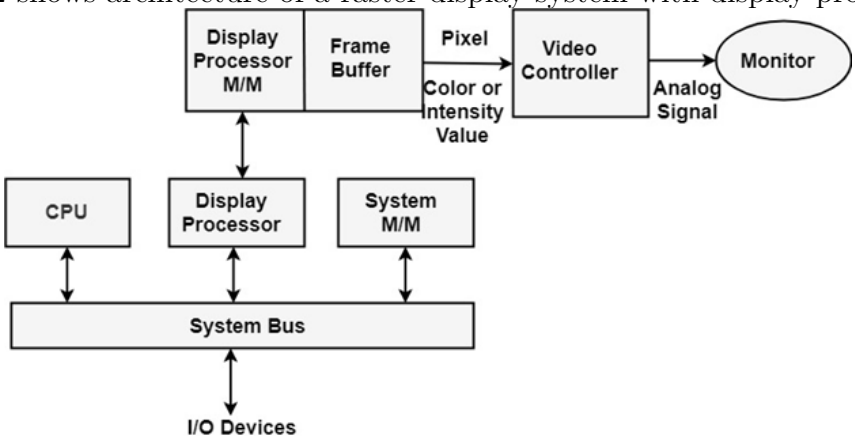


Figure 1.2: Architecture of a Raster Display System with Display Processor

❋ As figure 1.3 showing that two registers (X register and Y register) are used to store the coordinate of the screen pixels.

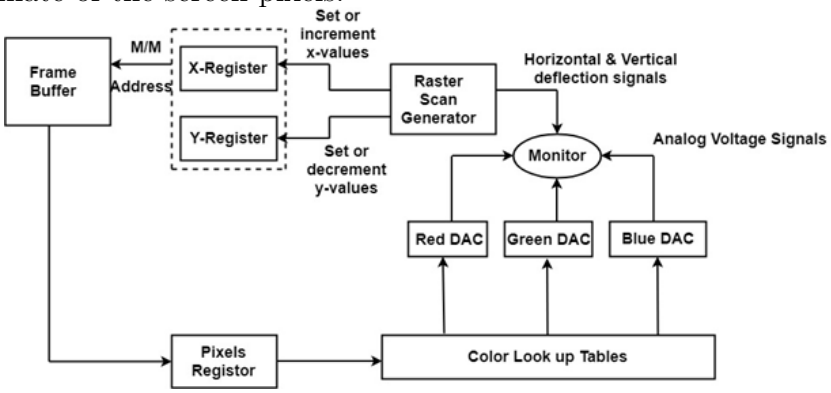


Figure 1.3: Display Processor

❋ At the start of a Refresh Cycle:

- ☆ x register is set to 0 and y register is set to ymax. This (x, y) address is translated into a memory address of frame buffer where the color value for this pixel position is stored.

- ☆ The controller receives this colour value (a binary no) from the frame buffer, breaks it up into three parts and sends each element to a separate Digital-to-Analog Converter (DAC).
- ☆ These voltages, in turn, controls the intensity of 3 e-beam that are focused at the (x, y) screen position by the horizontal and vertical drive signals.
- ☆ This process is repeated for each pixel along the top scan line, each time incrementing the x register by y.
- ☆ As pixels on the first scan line are generated, the x register is incremented throughx-max.
- ☆ Then x register is reset to 0, and y register is decremented by 1 to access the next scan line.
- ☆ Pixel along each scan line is then processed, and the procedure is repeated for each successive scan line units pixels on the last scan line (y=0) are generated.
- ☆ For a display system employing a colour look-up table frame buffer value is not directly used to control the CRT beam intensity.
- ☆ It is used as an index to find the three pixel-colour value from the look-up table. This lookup operation is done for each pixel on every display cycle.

1.5 Display Devices

❋ The following display devices are used:

- ☆ Refresh Cathode Ray Tube
- ☆ Random Scan and Raster Scan
- ☆ Color CRT Monitors
- ☆ Direct View Storage Tubes
- ☆ Flat Panel Display
- ☆ Lookup Table

1.5.1 Cathode Ray Tube (CRT)

❋ CRT is a technology used in traditional computer monitors and televisions. The image on CRT display is created by firing electrons from the back of the tube of phosphorus located towards the front of the screen.

❋ Once the electron heats the phosphorus, they light up, and they are projected on a screen. The colour you view on the screen is produced by a blend of red, blue and green light.

❋ Figure 1.4 shows the diagrammatic form of Display Processor.

Components of CRT

❋ *Electron Gun:* The electron gun creates a source of electrons which are focused into a narrow beam directed at the face of the CRT.

❋ *Control Electrode:* It is used to turn the electron beam on and off.

❋ *Focusing system:* It is used to create a clear picture by focusing the electrons into a narrow beam.

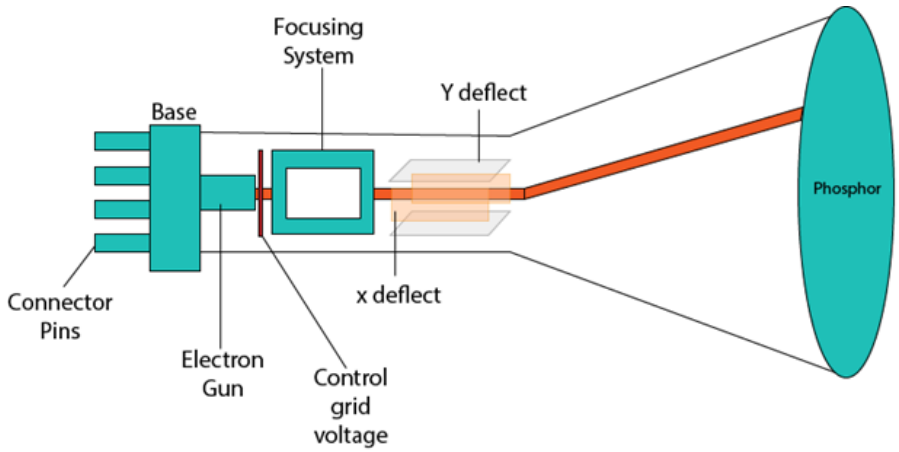


Figure 1.4: Display Processor

❋ *Deflection Yoke:* It is used to control the direction of the electron beam. It creates an electric or magnetic field which will bend the electron beam as it passes through the area.

❋ *Phosphorus-coated screen:* The inside front surface of every CRT is coated with phosphors. Phosphors glow when a high-energy electron beam hits them.

1.5.2 Random Scan and Raster Scan Display

Random Scan Display

❋ Random Scan System uses an electron beam which operates like a pencil to create a line image on the CRT screen.

❋ Each line segment is drawn on the screen by directing the beam to move from one point on the screen to the next, where its x & y coordinates define each point.

❋ The system cycles back to the first line and design all the lines of the image 30 to 60 time each second. The process is shown in figure 1.5:

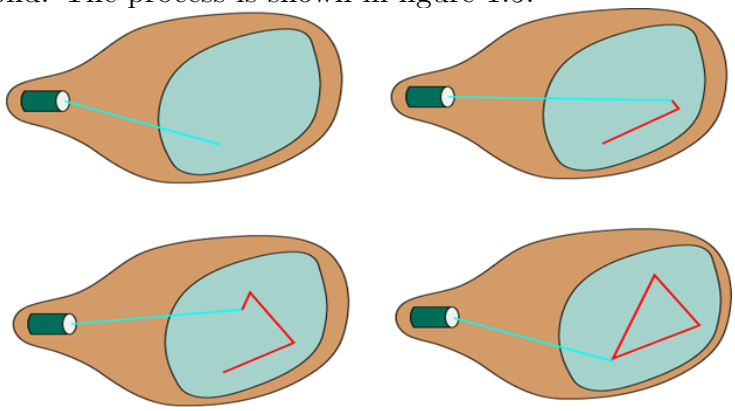


Figure 1.5: Random Scan Display

❋ Random-scan monitors are also known as vector displays or stroke-writing displays or calligraphic displays.

Raster Scan Display

- A Raster Scan Display is based on intensity control of pixels in the form of a rectangular box called Raster on the screen.
- The raster scan system can store information of each pixel position, so it is suitable for realistic display of objects. Raster Scan provides a refresh rate of 60 to 80 frames per second.
- Frame Buffer is also known as Raster or bit map. In Frame Buffer the positions are called picture elements or pixels.
- Beam refreshing is of two types: horizontal retracing and vertical retracing.
  - When the beam starts from the top left corner and reaches the bottom right scale, it will again return to the top left side called at vertical retrace.
  - Then it will again more horizontally from top to bottom call as horizontal retracing shown in figure 1.6:

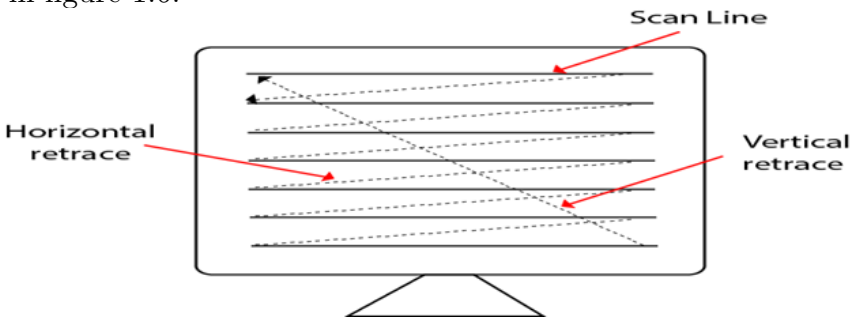


Figure 1.6: Random Scan Display

- Types of Scanning or travelling of beam in Raster Scan
  - Interlaced Scanning
  - Non-Interlaced Scanning
- Interlaced scanning is a screen display in which all odd-numbered lines are traced by an electron beam, and then in the next circle, an even number of lines are traced. Refresh rate of 60 frames per second.
- A non-interlaced scanning is a screen display in which the lines are scanned progressively from the top to the bottom of the screen. Refresh rate of 30 frames per second.
- Differentiate between Random and Raster Scan Display:

No	Random Scan	Raster Scan
1	It has high Resolution	Its resolution is low
2	It is more expensive	It is less expensive
3	Any modification if needed is easy	Modification is tough
4	Solid pattern is tough to fill	Solid pattern is easy to fill
5	Refresh rate depends or resolution	Refresh rate does not depend on the picture

6	Only screen with view on an area is displayed.	Whole screen is scanned
7	Beam Penetration technology come under it.	Shadow mark technology came under this
8	It does not use interlacing method.	It uses interlacing
9	It is restricted to line drawing applications	It is suitable for realistic display

1.5.3 Colour CRT Monitors:

❋ There are two popular approaches for producing colour displays with a CRT are:

- ☆ Beam Penetration Method
- ☆ Shadow-Mask Method

Beam Penetration Method

❋ The CRT screen is coated with two layers of phosphor, red and green and the displayed colour depends on how far the electron beam penetrates the phosphor layers.

❋ This method produces four colors only, red, green, orange and yellow. A beam of slow electrons excites the outer red layer only; hence screen shows red color only. A beam of high-speed electrons excites the inner green layer. Thus screen shows a green colour.

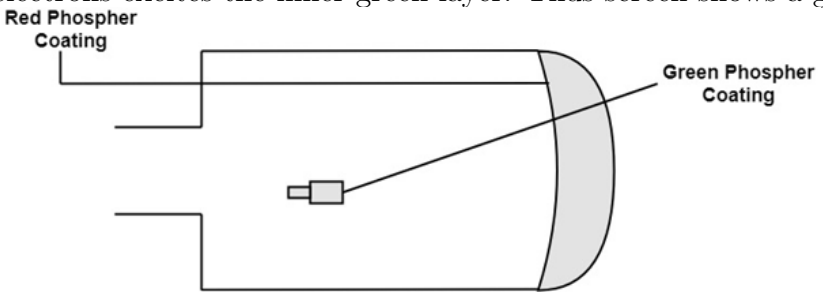


Figure 1.7: Random Scan Display

❋ Advantages:

- ☆ Inexpensive

❋ Disadvantages:

- ☆ Only four colours are possible
- ☆ Quality of pictures is not as good as with another method.

Shadow-Mask Method

❋ Shadow Mask Method is commonly used in Raster-Scan System because they produce a much wider range of colors than the beam-penetration method.

❋ It is used in the majority of colour TV sets and monitors.

❋ *Construction:* A shadow mask CRT has 3 phosphor colour dots at each pixel position.

- ☆ One phosphor dot emits: red light
- ☆ Another emits: green light
- ☆ Third emits: blue light

❋ This type of CRT has 3 electron guns, one for each colour dot and a shadow mask grid just behind the phosphor coated screen.

❋ Shadow mask grid is pierced with small round holes in a triangular pattern.

❋ Figure 1.8, 1.9 shows the delta-delta shadow mask method commonly used in colour CRT system.

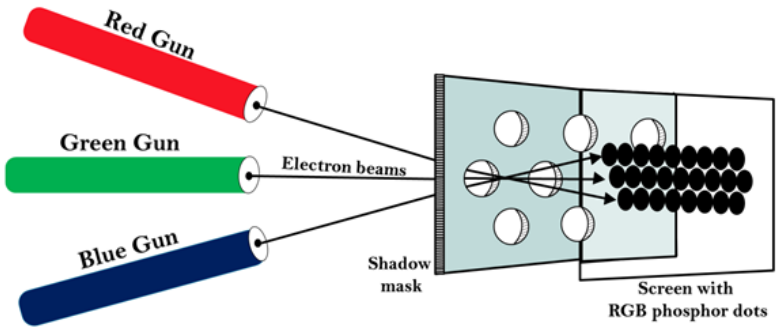


Figure 1.8: The delta-delta shadow mask method

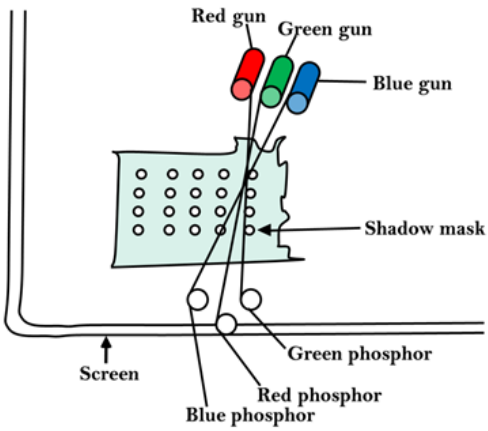


Figure 1.9: The delta-delta shadow mask method

❋ *Working:* Triad arrangement of red, green, and blue guns.

- ☆ The 3 electron beams are deflected and focused as a group onto the shadow mask.
- ☆ When the three beams pass through a hole in the shadow mask, they activate a dotted triangle, which occurs as a small colour spot on the screen.
- ☆ The phosphor dots in the triangles are organized so that each electron beam can activate only its corresponding colour dot when it passes through the shadow mask.

❋ *Inline arrangement:* Another configuration for the 3 electron guns is an Inline arrangement in which the 3 electron guns and the corresponding red-green-blue colour dots

on the screen, are aligned along one scan line rather of in a triangular pattern (see figure 1.10).

- ☆ This inline arrangement of electron guns in easier to keep in alignment and is commonly used in high-resolution color CRT's.

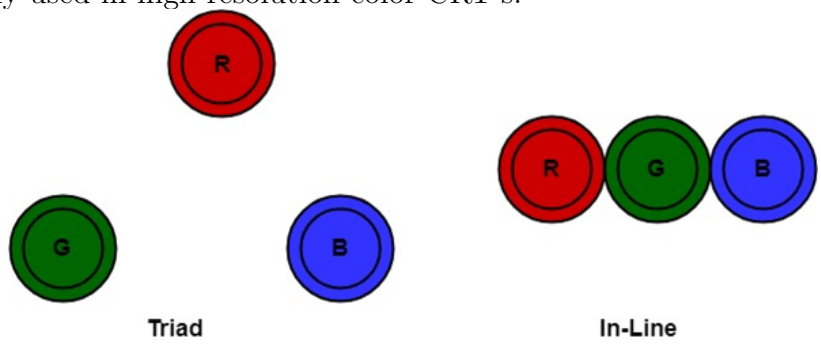


Figure 1.10: The delta-delta shadow mask method

- ❋ Advantage:
  - ☆ Realistic image
  - ☆ Million different colors to be generated
  - ☆ Shadow scenes are possible
- ❋ Disadvantage:
  - ☆ Relatively expensive compared with the monochrome CRT.
  - ☆ Relatively poor resolution
  - ☆ Convergence Problem

1.5.4 Direct View Storage Tubes (DVST)

- ❋ DVST terminals also use the random scan approach to generate the image on the CRT screen. The term "storage tube" refers to the ability of the screen to retain the image which has been projected against it, thus avoiding the need to rewrite the image constantly.
- ❋ *Function of guns:* Two guns are used in DVST
  - ☆ *Primary guns:* It is used to store the picture pattern.
  - ☆ *Flood gun or Secondary gun:* It is used to maintain picture display.
- ❋ Figure 1.11 shows the structure of the direct view storage tube.
- ❋ Advantage:
  - ☆ No refreshing is needed.
  - ☆ High Resolution
  - ☆ Cost is very less
- ❋ Disadvantage:
  - ☆ It is not possible to erase the selected part of a picture.
  - ☆ It is not suitable for dynamic graphics applications.
  - ☆ If a part of picture is to modify, then time is consumed.

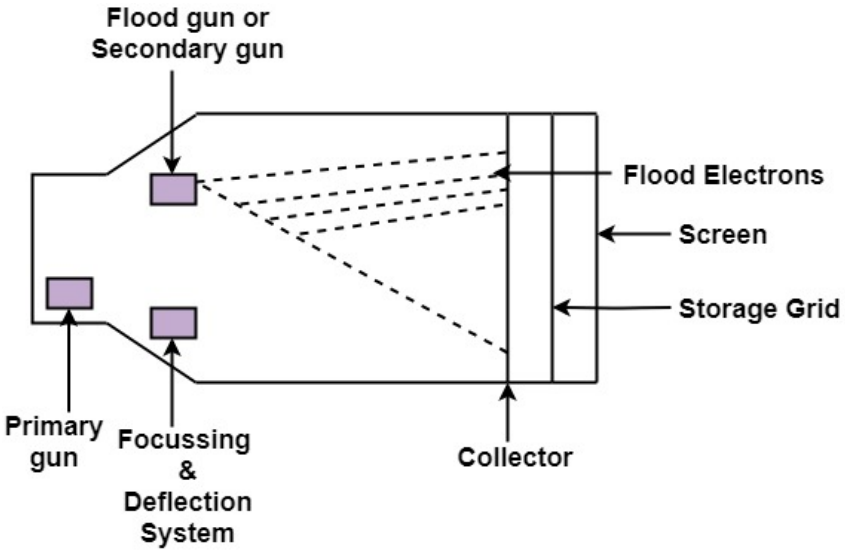


Figure 1.11: The Direct View Storage Tubes

1.5.5 Flat Panel Display

- The Flat-Panel display refers to a class of video devices that have reduced volume, weight and power requirement compare to CRT.
- *Example:* Small T.V. monitor, calculator, pocket video games, laptop computers, an advertisement board in elevator.
- Flat panel display is divided in to two categories:
  - *Emissive Display:* The emissive displays are devices that convert electrical energy into light. Examples are Plasma Panel, thin film electroluminescent display and LED (Light Emitting Diodes).
  - *Non-Emissive Display:* The Non-Emissive displays use optical effects to convert sunlight or light from some other source into graphics patterns. Examples are LCD (Liquid Crystal Device).

1.5.6 Plasma Panel Display

- Plasma-Panels are also called as Gas-Discharge Display. It consists of an array of small lights. The essential components of the plasma-panel display are:
  - *Cathode:* It consists of line wires. It delivers negative voltage to gas cells. The voltage is released along with the negative axis.
  - *Anode:* It also consists of line wires. It delivers positive voltage. The voltage is supplied along positive axis.
  - *Fluorescent cells:* It consists of small pockets of gas liquids when the voltage is applied to this liquid (neon gas) it emits light.
  - *Glass Plates:* These plates act as capacitors. The voltage will be applied, the cell will glow continuously.
- The gas will glow when there is a significant voltage difference between horizontal and vertical wires. The voltage level is kept between 90 volts to 120 volts. Plasma level

does not require refreshing. Erasing is done by reducing the voltage to 90 volts.

❋ Each cell of plasma has two states, so cell is said to be stable. Displayable point in plasma panel is made by the crossing of the horizontal and vertical grid. The resolution of the plasma panel can be up to 512 * 512 pixels.

❋ Figure 1.12 shows the state of cell in plasma panel display.

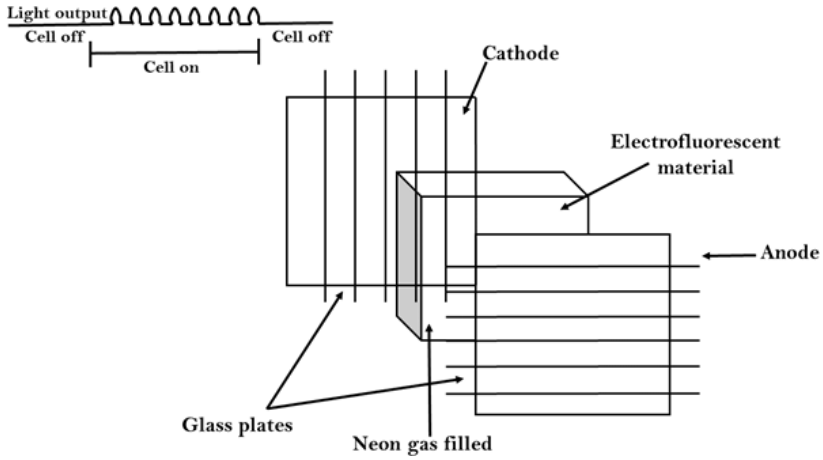


Figure 1.12: The Plasma Panel Display

❋ Advantage:

- ☆ High Resolution
- ☆ Large screen size is also possible.
- ☆ Less Volume
- ☆ Less weight
- ☆ Flicker Free Display

❋ Disadvantage:

- ☆ Poor Resolution
- ☆ Wiring requirement anode and the cathode is complex.
- ☆ Its addressing is also complex.

1.5.7 LED (Light Emitting Diode):

❋ In an LED, a matrix of diodes is organized to form the pixel positions 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.

1.5.8 LCD (Liquid Crystal Display)

❋ Liquid Crystal Displays are the devices that produce a picture by passing polarized light from the surroundings or from an internal light source through a liquid-crystal material that transmits the light.

❋ LCD uses the liquid-crystal material between two glass plates; each plate is the right angle to each other between plates liquid is filled.

❋ One glass plate consists of rows of conductors arranged in vertical direction. Another glass plate is consisting of a row of conductors arranged in horizontal direction. The pixel position is determined by the intersection of the vertical & horizontal conductor. This position is an active part of the screen.

❋ Figure 1.13 shows the structure of the LCD display.

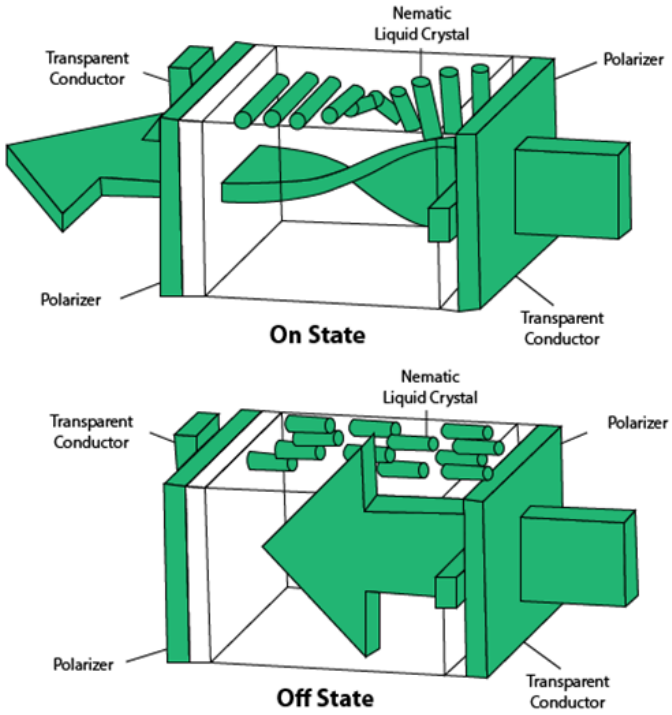


Figure 1.13: Liquid Crystal Display

❋ Advantage:

- ☆ Low power consumption.
- ☆ Small Size
- ☆ Low Cost

❋ Disadvantage:

- ☆ LCDs are temperature-dependent (0-70řC)
- ☆ LCDs do not emit light; as a result, the image has very little contrast.
- ☆ LCDs have no color capability.
- ☆ The resolution is not as good as that of a CRT.

Look-Up Table

❋ Image representation is essentially the description of pixel colors. There are three primary colors: R (red), G (green) and B (blue). Each primary color can take on intensity levels produces a variety of colors.

❋ Using direct coding, we may allocate 3 bits for each pixel, with one bit for each primary color. The 3-bit representation allows each primary to vary independently between two intensity levels: 0 (off) or 1 (on). Hence each pixel can take on one of the eight colours.

Table 1.2: Look-Up Table

Bit 1:r	Bit 2:g	Bit 3:b	Color name
0	0	0	Black
0	0	1	Blue
0	1	0	Green
0	1	1	Cyan
1	0	0	Red
1	0	1	Magenta
1	1	0	Yellow
1	1	1	White

❄ A widely accepted industry standard uses 3 bits, or 24 bytes, per pixel, with one byte for each primary colour. The way, we allow each primary colour to have 256 different intensity levels.

❄ Thus a pixel can take on a colour from 256 x 256 x 256 or 16.7 million possible choices. The 24-bit format is commonly referred to as the actual colour representation.

Chapter 2

Scan Conversion a Line

2.1 Scan Conversion

- ❋ It is a process of representing graphics objects as a collection of pixels. Each pixel can have either on or off state.
- ❋ The circuitry of the video display device of the computer is capable of converting binary values (0, 1) into a pixel on and pixel off information. 0 is represented by pixel off. 1 is represented using pixel on.
- ❋ Any model of graphics can be reproduced with a dense matrix of dots or points. Most human beings think graphics objects as points, lines, circles, ellipses. For generating graphical object, many algorithms have been developed.
- ❋ Advantage of developing algorithms for scan conversion
  - ☆ Algorithms can generate graphics objects at a faster rate.
  - ☆ Using algorithms memory can be used efficiently.
  - ☆ Algorithms can develop a higher level of graphical objects.
- ❋ Examples of objects which can be scan converted
  - ☆ Point
  - ☆ Line
  - ☆ Sector
  - ☆ Arc
  - ☆ Ellipse
  - ☆ Rectangle
  - ☆ Polygon
  - ☆ Characters
  - ☆ Filled Regions

Pixel

- ❋ The term pixel is a short form of the picture element. It is also called a point or dot. It is the smallest picture unit accepted by display devices.
- ❋ A picture is constructed from hundreds of such pixels. Lines, circle, arcs, characters; curves are drawn with closely spaced pixels.
- ❋ The closer the dots or pixels are, the better will be the quality of picture. The closer dots are, the crisper will be the picture. Picture will not appear jagged and unclear if pixels are closely spaced. So the quality of the picture is directly proportional to the density of pixels on the screen.
- ❋ Pixels are also defined as the smallest addressable unit or element of the screen. Each pixel can be assigned an address as shown in figure 2.1:
- ❋ Different graphics objects can be generated by setting the different intensity of pixels and different colors of pixels. Each pixel has some co-ordinate value. The coordinate is represented using row and column.

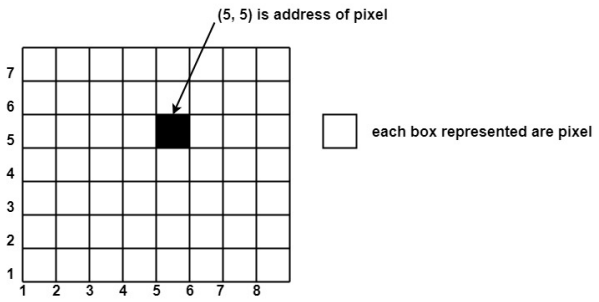


Figure 2.1: Pixel Representation

❋ P (5, 5) used to represent a pixel in the 5th row and the 5th column. Each pixel has some intensity value which is represented in memory of computer called a frame buffer.

- ☆ Frame Buffer is also called a *refresh buffer* or *digital memory*.
- ☆ Inside the buffer, image is stored as a pattern of binary digits either 0 or 1. So there is an array of 0 or 1 used to represent the picture.
- ☆ In black and white monitors, black pixels are represented using 1's and white pixels are represented using 0's.
- ☆ In case of systems having one bit per pixel frame buffer is called a *bitmap*.
- ☆ In systems with multiple bits per pixel it is called a *pixmap*.

Scan Converting a Point

❋ Each pixel on the graphics display does not represent a mathematical point. Instead, it means a region that theoretically can contain infinite points. Scan-Converting a point involves illuminating the pixel that contains the point.

❋ Example: Scan Converting a Point as shown in figure 2.2 would both be represented by pixel $(2, 1)$. In general, a point $p(x, y)$ is represented by the integer part of x & the integer part of y that is pixels $[(int(x), int(y)]$.

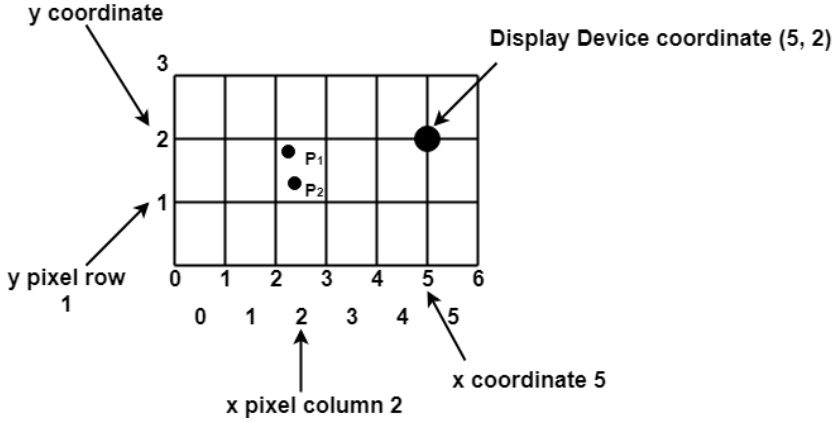


Figure 2.2: Scan Converting a Point

Scan Converting a Straight Line

❋ A straight line may be defined by two endpoints & an equation. In figure 2.3, the two endpoints are described by (x_1, y_1) and (x_2, y_2).

☆ The equation of the line is used to determine the x, y coordinates of all the points that lie between these two endpoints.

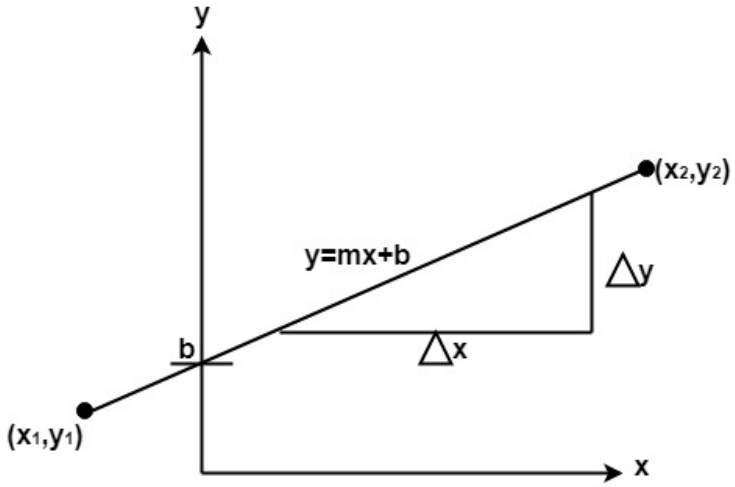


Figure 2.3: Scan Converting a Straight Line

☆ Using the equation of a straight line, $y = mx + b$ where $m = \frac{\triangle y}{\triangle x}$ & b = the y interrupt, we can find values of y by incrementing x from $x = x_1$, to $x = x_2$.

☆ By scan-converting these calculated x, y values, we represent the line as a sequence of pixels.

Algorithm for line Drawing

❋ Direct use of line equation

❋ DDA (Digital Differential Analyzer)

❋ Bresenham's Algorithm

❋ Mid-point line Algorithm

2.2 Direct use of line equation

❋ It is the simplest form of conversion. First of all scan P_1 and P_2 points. P_1 has co-ordinate (x'_1, y'_1) and P_2 has co-ordinate (x'_2, y'_2).

❋ Then $m = (y'_2, y'_1)/(x'_2, x'_1)$ and $b = y_1^1 + mx_1^1$

❋ If value of $|m| \leq 1$ for each integer value of x. But do not consider x_1^1 and x_2^2

❋ If value of $|m| > 1$ for each integer value of y. But do not consider y_1^1 and y_2^2

❋ Example: A line with starting point as $(0, 0)$ and ending point $(6, 18)$ is given. Calculate value of intermediate points and slope of line.

☆ Solution: $P1(0, 0),\ P7(6, 18)$

* $x_1 = 0,\ y_1 = 0$
* $x_2 = 6,\ y_2 = 18$
* $M = \frac{\triangle y}{\triangle x} = \frac{y_2 - y_1}{x_2 - x_1} = \frac{18-0}{6-0} = \frac{18}{6} = 3$

☆ We know equation of line is

* $y = mx + b$

* $y = 3x + b$ ..... Eq (1)

☆ put value of x from initial point in equation (1), i.e., $(0, 0) x = 0, y = 0$

* $0 = 3 \times 0 + b$
* $0 = b \Rightarrow b = 0$

☆ put $b = 0$ in equation (1)

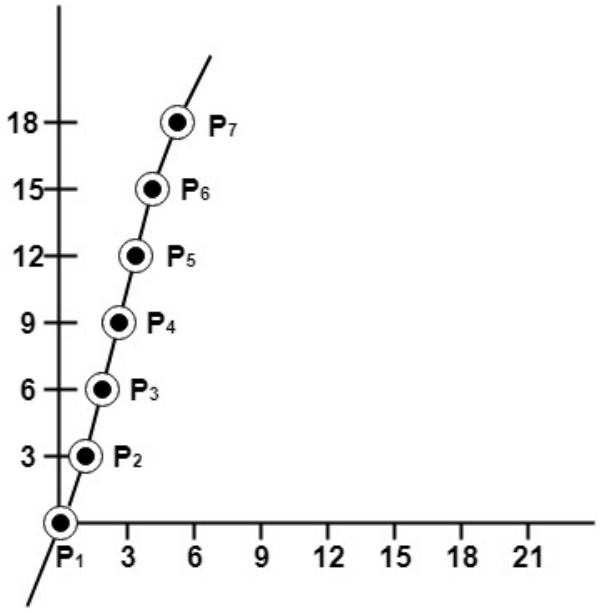
* $y = 3x + 0$
* $y = 3x$

☆ Now calculate intermediate points

* $Let\ x = 1 \Rightarrow y = 3 \times 1 \Rightarrow y = 3$
* $Let\ x = 2 \Rightarrow y = 3 \times 2 \Rightarrow y = 6$
* $Let\ x = 3 \Rightarrow y = 3 \times 3 \Rightarrow y = 9$
* $Let\ x = 4 \Rightarrow y = 3 \times 4 \Rightarrow y = 12$
* $Let\ x = 5 \Rightarrow y = 3 \times 5 \Rightarrow y = 15$
* $Let\ x = 6 \Rightarrow y = 3 \times 6 \Rightarrow y = 18$

☆ So points are $P1(0, 0)$

* $P2(1, 3)$
* $P3(2, 6)$
* $P4(3, 9)$
* $P5(4, 12)$
* $P6(5, 15)$
* $P7(6, 18)$



Algorithm 1: Algorithm for drawing line using equation

Step 1: Start Algorithm

Step 2: Declare variables $x_1,\ x_2,\ y_1,\ y_2,\ dx,\ dy,\ m,\ b,$

Step 3: Enter values of $x_1,\ x_2,\ y_1,\ y_2$.

The $(x_1,\ y_1)$ are co-ordinates of a starting point of the line.

The $(x_2,\ y_2)$ are co-ordinates of a ending point of the line.

Step 4: Calculate $dx = x_2 - x_1$, $dy = y_2 - y_1$, $m = \frac{dy}{dx}$

Step 5: Calculate $b = y_1 - m \times x_1$

Step 6: Set (x, y) equal to starting point.

if $dx < 0$ **then**
| $x = x_2$; $y = y_2$; $x_{end} = x_1$;

if $dx > 0$ **then**
| $x = x_1$; $y = y_1$; $x_{end} = x_2$;

Step 9: Check whether the complete line has been drawn if $x = x_{end}$, stop

Step 10: Plot a point at current (x, y) coordinates

Step 11: Increment value of x, i.e., $x = x + 1$

Step 12: Compute next value of y from equation $y = mx + b$

Step 13: Go to Step 9.

Code for drawing line using equation

```
INPUT: x1, x2, y1, y2
dx = x2 - x1; dy = y2 - y1; m = dy/dx;
c = y1 - (m * x1);
if(dx < 0){
    x = x2; y = y2; xend = x1;
}
else{
    x = x1; y = y1; xend = x2;
}
while(x <= xend){
    drawPoint(x,(int)y,g); \\ Graphics g
    x++; y = (x * x) + c;
}
```

2.3 DDA Algorithm

❋ DDA stands for Digital Differential Analyzer. It is an incremental method of scan conversion of line. In this method calculation is performed at each step by using results of previous steps.

❋ Suppose at step i, the pixels is (x_i, y_i)

❋ The line of equation for step i

$y_i = mx_i + b$ ..... Eq (1)

❋ Next value will be

$y_{i+1} = mx_{i+1} + b$ ..... Eq (2)

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

$y_{i+1} - y_i = \triangle y$ ..... Eq (3)

$y_{i+1} - x_i = \triangle x$ ..... Eq (4)

$y_{i+1} = y_i + \triangle y$

$\triangle y = m\triangle x$

$y_{i+1} = y_i + m\triangle x$

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

$x_{i+1} = x_i + \triangle x$

$x_{i+1} = x_i + \frac{\triangle y}{m}$

Case 1: When $|M| < 1$ then

$x = x_1,\ y = y_1 \qquad set\ \triangle x = 1$

$y_{i+1} = y_i + m, x = x + 1$

Until $x = x_2$

Case 2: When $|M| = 1$ then

$x = x_1,\ y = y_1$

$x_{i+1} = x_i + 1,\ y = y + 1$

Until $y = y_2$ or $x = x_2$

Case 2: When $|M| > 1$ then

$x = x_1,\ y = y_1 \qquad set\ \triangle y = 1$

$x_{i+1} = x_i + \frac{1}{m},\ y = y + 1$

Until $y = y_2$

❋ Advantage:

- ☆ It is a faster method than method of using direct use of line equation.
- ☆ This method does not use multiplication theorem.
- ☆ This method gives overflow indication when a point is repositioned.
- ☆ It is an easy method because each step involves just two additions.

❋ Disadvantage:

- ☆ It involves floating point additions rounding off is done. Accumulations of round off error cause accumulation of error.
- ☆ Rounding off operations and floating point operations consumes a lot of time.
- ☆ It is more suitable for generating line using the software. But it is less suited for hardware implementation.

❋ Example: If a line is drawn from $(2, 3)$ to $(6, 15)$ with use of DDA. How many points will needed to generate such line?

Solution:

$x_1 = 2,\ y_1 = 3$

$x_2 = 6,\ y_2 = 15$

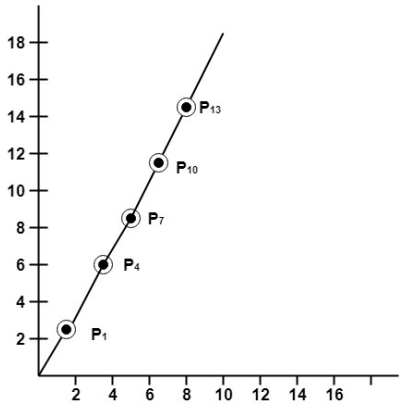
$dx = 6 - 2 = 4$

$dy = 15 - 3 = 12$

$m = \frac{\Delta y}{\Delta x} = \frac{12}{4}$

For calculating next value of x takes $x = x + \frac{1}{m}$

$P_1(2, 3)$	point plotted
$P_2(2\frac{1}{3}, 4)$	point plotted
$P_3(2\frac{2}{3}, 5)$	point not plotted
$P_4(3, 6)$	point plotted
$P_5(3\frac{1}{3}, 7)$	point not plotted
$P_6(3\frac{2}{3}, 8)$	point not plotted
$P_7(4, 9)$	point plotted
$P_8(4\frac{1}{3}, 10)$	point not plotted
$P_9(4\frac{2}{3}, 11)$	point not plotted
$P_{10}(5, 12)$	point plotted
$P_{11}(5\frac{1}{3}, 13)$	point not plotted
$P_{12}(5\frac{2}{3}, 14)$	point not plotted
$P_{13}(6, 15)$	point plotted



Algorithm 2: DDA Algorithm

Step 1: Start Algorithm

Step 2: Declare $x_1,\ y_1,\ x_2,\ y_2,\ dx,\ dy,\ x,\ y$ as integer variables.

Step 3: Enter value of $x_1,\ y_1,\ x_2,\ y_2$.

Step 4: Calculate $dx = x_2 - x_1$, $dy = y_2 - y_1$.

if *abs(dx) > abs(dy)* **then**

| $step = abs(dx)$

else

| $step = abs(dy)$

end

Step 5: Calculate x_{inc}, y_{inc}

$x_{inc} = \frac{dx}{step},\ y_{inc} = \frac{dy}{step}$

assign $x = x_1$, assign $y = y_1$

Step 6: Set pixel (x, y)

$x = x + x_{inc},\ y = y + y_{inc}$

Set pixels $(Round(x),\ Round(y))$

Step 7: Repeat step 6 until $x = x_2$

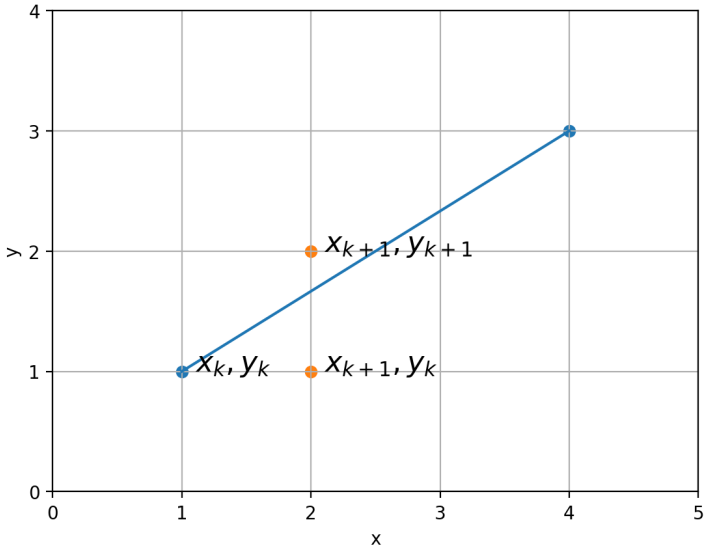
Step 8: End Algorithm

Code for drawing DDA line

```
INPUT: x1, x2, y1, y2
dx = x1 - x0; dy = y1 - y0;
if(dx>=dy){
    steps = dx;
}
else{
    steps = dy;
}
dx = dx/steps; dy = dy/steps;
x = x0; y = y0; i = 1;
while(i<= steps){
    drawPoint(x,(int)y,g); \\ Graphics g
    x += dx; y += dy; i=i+1;
}
```

2.4 Bresenham's Line Algorithm

- It was developed by Bresenham. It is an efficient method because it involves only integer addition, subtractions, and multiplication operations.
- These operations can be performed very rapidly so lines can be generated quickly.
- The next pixel selected is that one who has the least distance from true line.
- Once a pixel is chosen at any step. The next pixel is
  - Either the one to its right (lower-bound for the line)
  - One top its right and up (upper-bound for the line)
- The line is best approximated by those pixels that fall the least distance from the path between P_1', P_2'.



❋ To chooses the next one between the bottom pixel S and top pixel T.
If S is chosen, we have $x_{i+1} = x_i + 1$ and $y_{i+1} = y_i$
If T is chosen, we have $x_{i+1} = x_i + 1$ and $y_{i+1} = y_i + 1$

❋ The actual y coordinates of the line at $x = x_{i+1}$ is
$y = mx_{i+1} + b$
$y = m(x_i + 1) + b$

❋ The distance from S to the actual line in y direction $s = y - y_i$.

❋ The distance from T to the actual line in y direction $t = (y_i + 1) - y$.

❋ Now consider the difference between these 2 distance values $s - t$

❋ When $(s - t) < 0 \Rightarrow s < t$, the closest pixel is S

❋ When $(s - t) \geq 0 \Rightarrow s < t$, the closest pixel is T

❋ This difference is $s - t = (y - y_i) - [(y_i + 1) - y] = 2y - 2y_i - 1$
$s - t = 2m(x_i + 1) + 2b - 2y_i - 1$

❋ Substituting m by $\frac{\Delta y}{\Delta x}$ and introducing decision variable
$di = \Delta x(s - t)$; $(s - t)$ has m it gives a float value so we multiply by Δx
$di = \Delta x \left(2\frac{\Delta y}{\Delta x}(x_i + 1) + 2b - 2yi - 1\right) = 2\Delta y x_i - 2\Delta y + 2b\Delta x - 2y_i\Delta x - \Delta x$
$d_i = 2\Delta y.x_i - 2\Delta x.y_i + c$, where $c = 2\Delta y + \Delta x(2b - 1)$

❋ We can write the decision variable d_{i+1} for the next slip on
$d_{i+1} = 2\Delta y.x_{i+1} - 2\Delta x.y_{i+1} + c$
$d_{i+1} - d_i = 2\Delta y.(x_{i+1} - x_i) - 2\Delta x(y_{i+1} - y_i)$

❋ Since $x_{i+1} = x_i + 1$, we have $d_{i+1} - d_i = 2\Delta y.(x_{i+1} - x_i) - 2\Delta x(y_{i+1} - y_i)$

❋ Special Cases

  ☆ If chosen pixel is at the top pixel $T(d_i \geq 0) \Rightarrow y_{i+1} = y_i + 1$
  $d_{i+1} = d_i + 2\Delta y - 2\Delta x$

  ☆ If chosen pixel is at the bottom pixel $S(di < 0) \Rightarrow y_{i+1} = y_i$
  $d_{i+1} = d_i + 2\Delta y$

❋ Finally, we calculate d_1
$d_1 = \Delta x[2m(x_1 + 1) + 2b - 2y_1 - 1]$
$d_1 = \Delta x[2(mx_1 + b - y_1) + 2m - 1]$

❋ Since $mx_1 + b - yi = 0$ and $m = \frac{\Delta y}{\Delta x}$, we have $d_1 = 2\Delta y - \Delta x$

❋ Advantage:

  ☆ It involves only integer arithmetic, so it is simple.

  ☆ It avoids the generation of duplicate points.

  ☆ It can be implemented using hardware because it does not use multiplication and division.

  ☆ It is faster as compared to DDA (Digital Differential Analyzer) because it does not involve floating point calculations like DDA Algorithm.

❋ Disadvantage:

☆ It is meant for a basic line drawing.

☆ Anti-aliasing is not part of Bresenham's algorithm, so to draw smooth lines, one had wanted to look into a different algorithm.

❋ Example: Starting and Ending position of the line are $(1, 1)$ and $(8, 5)$. Find intermediate points. Solution:

$x_1 = 1$, $y_1 = 1$

$x_2 = 8$, $y_2 = 5$

$\Delta x = x_2 - x_1 = 8 - 1 = 7$

$\Delta y = y_2 - y_1 = 5 - 1 = 4$

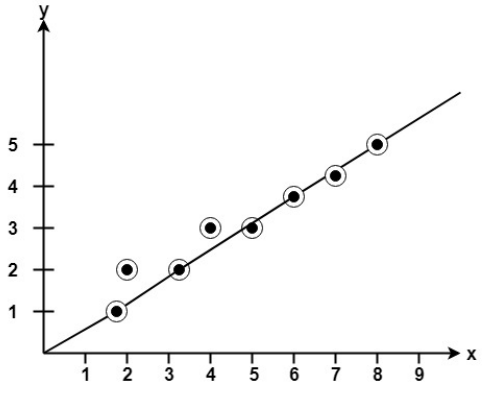
$I_1 = 2 * (\Delta y) = 2 * 4 = 8$

$I_2 = 2 * (\Delta y - \Delta x) = 2 * (4 - 7) = -6$

$d = I_1 - \Delta x = 8 - 7 = 1$

For calculating next value of x takes $x = x + \frac{1}{m}$

x	y	$d = d + I_1 or I_2$
1	1	$d + I_2 = 1 + (-6) = -5$
2	2	$d + I_1 = -5 + 8 = 3$
3	2	$d + I_2 = 3 + (-6) = -3$
4	3	$d + I_1 = -3 + 8 = 5$
5	3	$d + I_2 = 5 + (-6) = -1$
6	4	$d + I_1 = -1 + 8 = 7$
7	4	$d + I_2 = 7 + (-6) = 1$
8	5	



Algorithm 3: Bresenham's Line Algorithm

Step 1: Start Algorithm

Step 2: Declare $x_1,\ y_1,\ x_2,\ y_2,\ d,\ i_1,\ i_2\ dx\ dy$ as integer variables.

Step 3: Enter value of $x_1,\ y_1,\ x_2,\ y_2$.

Step 4: Calculate $dx = x_2 - x_1$, $dy = y_2 - y_1$, $i_1 = 2 * dy$, $i_2 = 2 * (dy - dx)$, $d = i_1 - d_x$

Step 5: Consider (x, y) as starting point and x_{end} as maximum possible value of x.

if $dx < 0$ **then**
| $x = x_2$, $y = y_2$, $x_{end} = x_1$
else
| $x = x_1$, $y = y_1$, $x_{end} = x_2$
end

Step 6: Generate point at (x, y)coordinates.

Step 7: Check if whole line is generated.

if $x >= x_{end}$ **then**
| *Stop.*

Step 8: Calculate co-ordinates of the next pixel

if $d < 0$ **then**
| $d = d + i_1$
else
| $d = d + i_2$, $y = y + 1$
end
$x = x + 1$

Step 9: Draw a point of latest (x, y) coordinates

Step 10: Go to step 7

Step 11: End Algorithm

Code for drawing bresenham's line

```
int dx, dy, p, x, y;
dx=x1-x0;  dy=y1-y0;
x=x0; y=y0;
p=2*dy-dx;
while(x<x1){
    if(p>=0){
        putpixel(x,y,7);
        y=y+1;
        p=p+2*dy-2*dx;
    }
    else{
        putpixel(x,y,7);
        p=p+2*dy;}
        x=x+1;
    }
}
```

2.4.1 Differentiate between DDA Algorithm and Bresenham's Line Algorithm:

DDA Algorithm	Bresenham's Line Algorithm
DDA Algorithm use floating point, i.e., Real Arithmetic.	Bresenham's Line Algorithm use fixed point, i.e., Integer Arithmetic
DDA Algorithms uses multiplication & division its operation	Bresenham's Line Algorithm uses only subtraction and addition its operation
DDA Algorithm is slowly than Bresenham's Line Algorithm in line drawing because it uses real arithmetic (Floating Point operation)	Bresenham's Algorithm is faster than DDA Algorithm in line because it involves only addition & subtraction in its calculation and uses only integer arithmetic.
DDA Algorithm is not accurate and efficient as Bresenham's Line Algorithm.	Bresenham's Line Algorithm is more accurate and efficient at DDA Algorithm.
DDA Algorithm can draw circle and curves but are not accurate as Bresenham's Line Algorithm	Bresenham's Line Algorithm can draw circle and curves with more accurate than DDA Algorithm.

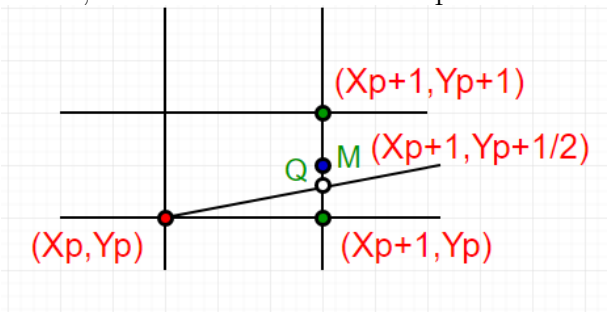
2.5 Mid-point Line Algorithm

❋ For any given/calculated previous pixel $P(X_p, Y_p)$, there are two candidates for the next pixel closest to the line, $E(X_p+1, Y_p)$ and $NE(X_p+1, Y_p+1)$ (E stands for East and NE stands for North-East).

❋ Find middle of two possible next points. Middle of $E(X_p+1, Y_p)$ and $NE(X_p+1, Y_p+1)$

is $M\left(X_p+1, \frac{Y_p+1}{2}\right)$.

- ☆ If M is above the line, then choose E as next point.
- ☆ If M is below the line, then choose NE as next point.



❋ Below are some assumptions to keep algorithm simple.

- ☆ We draw line from left to right.
- ☆ $x_1 < x_2$ and $y_1 < y_2$
- ☆ Slope of the line is between 0 and 1. We draw a line from lower left to upper right.

❋ Cases other than above assumptions can be handled using reflection.

❋ Let us consider a line $y = mx + B$.

❋ We can re-write the equation as:

- ☆ $y = \left(\frac{dy}{dx}\right)x + B$ or
- ☆ $(dy)x + B(dx) - y(dx) = 0$
- ☆ Let $F(x,y) = (dy)x - y(dx) + B(dx)$ .....(1)

❋ Let we are given two end points of a line (under above assumptions)

- ☆ For all points (x,y) on the line, the solution to $F(x,y)$ is 0.
- ☆ For all points (x,y) above the line, $F(x,y)$ result in a negative number.
- ☆ And for all points (x,y) below the line, $F(x,y)$ result in a positive number.

❋ This relationship is used to determine the relative position of $M = \left(X_p+1, \frac{Y_p+1}{2}\right)$

❋ So our decision parameter $d = F(M) = F\left(X_p+1, \frac{Y_p+1}{2}\right)$. For simplicity, let as write $F(x,y)$ as $ax + by + c$. Where, $a = dy$, $b = -dx$, $c = B \times dx$. We got these values from above equation (1)

Case 1: If E is chosen then for next point:

$dnew = F\left(X_p+2, \frac{Y_p+1}{2}\right) = a(X_p+2) + b\left(\frac{Y_p+1}{2}\right) + c$

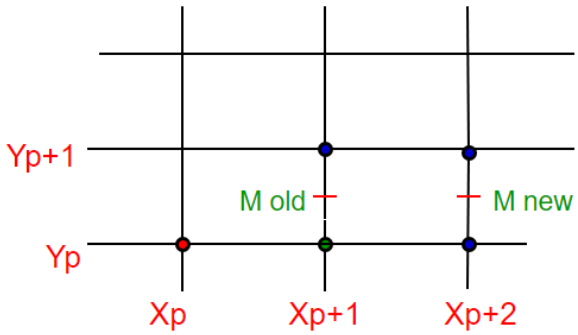
$dold = a(X_p+1) + b\left(\frac{Y_p+1}{2}\right) + c$

Difference (Or delta) of two distances: $DELd = dnew - dold$

$DELd = a(X_p+2) - a(X_p+1) + b\left(\frac{Y_p+1}{2}\right) - b\left(\frac{Y_p+1}{2}\right) + c - c$

$DELd = a(X_p) + 2a - a(X_p) - a = a$

Therefore, $dnew = dold + dy$. (as $a = dy$)



Case 2: If NE is chosen then for next point:

$dnew = F(X_p + 2, Y_p + \frac{3}{2}) = a(X_p + 2) + b(Y_p + \frac{3}{2}) + c$

$dold = a(X_p + 1) + b(Y_p + \frac{1}{2}) + c$

Difference (Or delta) of two distances: $DELd = d_{new} - d_{old}$

$DELd = a(X_p + 2) - a(X_p + 1) + b(Y_p + \frac{3}{2}) - b(Y_p + \frac{1}{2}) + c - c$

$DELd = a(X_p) + 2a - a(X_p) - a + b(Y_p) + \frac{3}{2}b - b(Y_p) - \frac{1}{2}b$

$DELd = a + b$

Therefore, $dnew = dold + dy - dx.(asa = dy, b = -dx)$

❄ Calculation For initial value of decision parameter d_0:

$d_0 = F(X_1 + 1, Y_1 + \frac{1}{2})$

$d_0 = a(X_1 + 1) + b(Y_1 + \frac{1}{2}) + c$

$d_0 = aX_1 + bY_1 + c + a + \frac{b}{2}$

$d_0 = F(X_1, Y_1) + a + \frac{b}{2}$

$d_0 = a + \frac{b}{2}(as\ F(X_1, Y_1) = 0)$

$d_0 = dy - \frac{dx}{2}(as\ a = dy,\ b = -dx)$

Code for drawing mid-point line

```
INPUT: x1, x2, y1, y2
dx = x2 - x1; dy = y2 - y1;
if(dx <= dy){
    d = dy - dx/2, x = X1, y = Y1
    drawPoint(x,(int)y,g); \\ Graphics g
    while(x < X2){
        x = x + 1
        if(d < 0)
            d = d + dy
        else{
            d = d + dy - dx, y = y + 1
            drawPoint(x,(int)y,g); \\ Graphics g
        }
    }
}
else{
    d = dx - dy/2, x = X1, y = Y1
    drawPoint(x,(int)y,g); \\ Graphics g
    while(y < y2){
        y = y + 1
        if(d < 0)
            d = d + dx
        else{
            d = d + dx - dy, x = x + 1
            drawPoint(x,(int)y,g); \\ Graphics g
        }
    }
}
```

Chapter 3

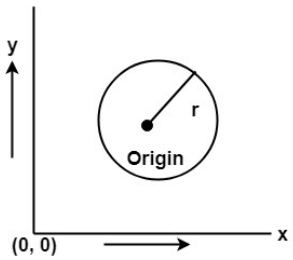
Scan Conversion a Circle

3.1 Defining a Circle

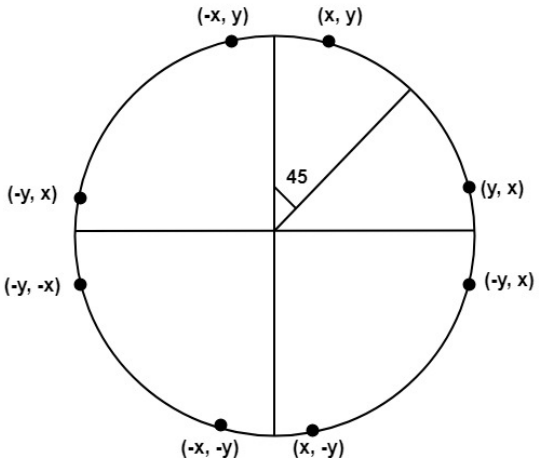
❋ Circle is an eight-way symmetric figure. The shape of circle is the same in all quadrants. In each quadrant, there are two octants.

❋ If the calculation of the point of one octant is done, then the other seven points can be calculated easily by using the concept of eight-way symmetry.

❋ For drawing, circle considers it at the origin. If a point is P1(x, y), then the other seven points will be



P2 (x, -y)
P3(-x, -y)
P4 (-x, y)
P5 (y, x)
P6 (y, -x)
P7 (-y, -x)
P8 (-y, x)



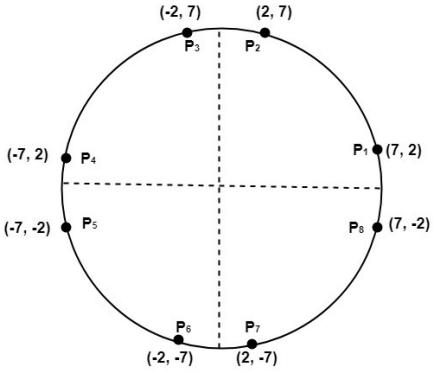
❋ So we will calculate only 45řarc. From which the whole circle can be determined easily.

❋ If we want to display circle on screen then the putpixel function is used for eight points as shown below:

- ☆ putpixel (x, y, color)
- ☆ putpixel (x, -y, color)
- ☆ putpixel (-x, y, color)
- ☆ putpixel (-x, -y, color)
- ☆ putpixel (y, x, color)
- ☆ putpixel (y, -x, color)
- ☆ putpixel (-y, x, color)
- ☆ putpixel (-y, -x, color)

❋ Example: Let we determine a point (2, 7) of the circle then other points will be (2, -7), (-2, -7), (-2, 7), (7, 2), (-7, 2), (-7, -2), (7, -2)

❋ These seven points are calculated by using the property of reflection. The reflection is accomplished in the following way:
The reflection is accomplished by reversing x, y co-ordinates.



Eight way symmetry of a Circle

❋ There are two standards methods of mathematically defining a circle centered at the origin.

☆ Defining a circle using Polynomial Method

☆ Defining a circle using Polar Co-ordinates

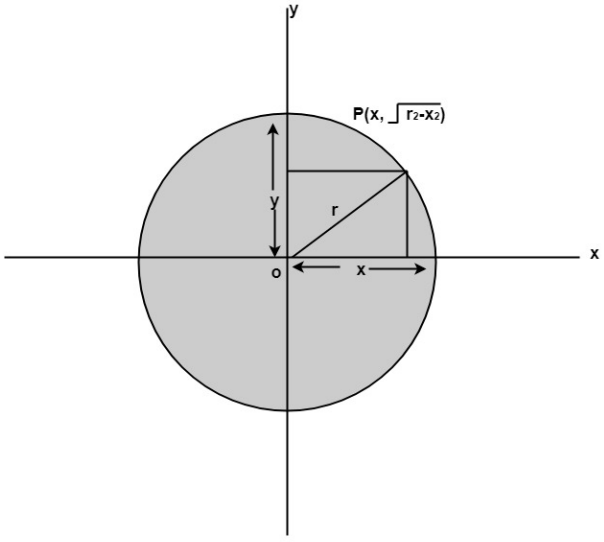
3.2 Defining a circle using Polynomial Method

❋ The first method defines a circle with the second-order polynomial equation as shown:

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

Where, x = the x coordinate, y = the y coordinate, r = the circle radius

❋ Each x coordinate in the sector, from 90ř to 45ř, is found by stepping x from 0 to $\frac{r}{\sqrt{2}}$ & each y coordinate is found by evaluating $\sqrt{r^2 - x^2}$ for each step of x.



Algorithm 4: Algorithm for drawing circle using polynomial method

Step 1: Start Algorithm

Step 2: Set the initial variables: r = circle radius,

(h, k) = coordinates of circle center, x=0, I = step size, $x_{end} = \frac{r}{\sqrt{2}}$

Step 3: Test to determine whether the entire circle has been scan-converted.

if $x > x_{end}$ **then**

| Stop

Step 4: Compute $y = \sqrt{r^2 - x^2}$

Step 5: Plot the eight points found by symmetry concerning the center (h, k) at the current (x, y) coordinates.

$Plot(x + h, y + k)$ $Plot(-x + h, -y + k)$

$Plot(y + h, x + k)$ $Plot(-y + h, -x + k)$

$Plot(-y + h, x + k)$ $Plot(y + h, -x + k)$

$Plot(-x + h, y + k)$ $Plot(x + h, -y + k)$

Step 6: Increment $x = x + I$

Step 9: Go to step (3).

Code for drawing line using equation

```
INPUT: (h,k), r;
x=0, x2 = r/sqrt(2);
while(x<=x2){
    y = sqrt(r*r-x*x);
    setPixel(floor(x), floor(y), h,k);
    x += 1;
}
setPixel(int x, int y, int h, int k){
    putpixel(x+h, y+k, RED);
    putpixel(x+h, -y+k, RED);
    putpixel(-x+h, -y+k, RED);
    putpixel(-x+h, y+k, RED);
    putpixel(y+h, x+k, RED);
    putpixel(y+h, -x+k, RED);
    putpixel(-y+h, -x+k, RED);
    putpixel(-y+h, x+k, RED);
}
```

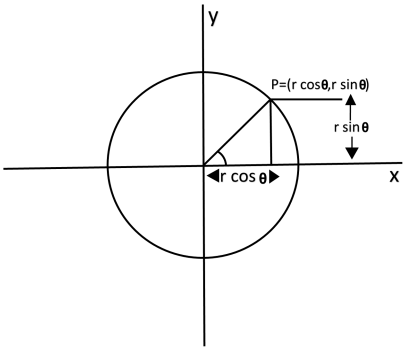
3.3 Defining a circle using Polar Co-ordinates

❄ The second method of defining a circle makes use of polar coordinates as shown:

$$x = rcos\theta,\ y = rsin\theta$$

Where θ=current angle, r = circle radius, x = x coordinate, y = y coordinate

❋ By this method, θ is stepped from 0 to $\frac{\pi}{4}$ & each value of x & y is calculated.



Algorithm 5: Algorithm for drawing circle using polynomial method

Step 1: Start Algorithm

Step 2: Set the initial variables: r = circle radius,
(h, k) = coordinates of the circle center, i = step size, $\theta_{end} = \frac{\pi}{4}$, $\theta = 0$

Step 3:

if $\theta > \theta_{end}$ **then**
| Stop

Step 4: Compute $x = r * cos\theta, y = r * sin\theta$

Step 5: Plot the eight points found by symmetry concerning the center (h, k) at the current (x, y) coordinates.

$Plot(x + h, y + k)$ $Plot(-x + h, -y + k)$

$Plot(y + h, x + k)$ $Plot(-y + h, -x + k)$

$Plot(-y + h, x + k)$ $Plot(y + h, -x + k)$

$Plot(-x + h, y + k)$ $Plot(x + h, -y + k)$

Step 6: Increment $\theta = \theta + i$

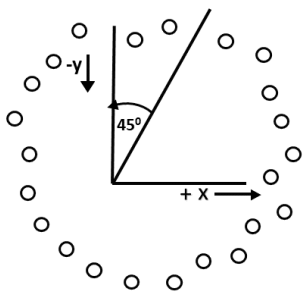
Step 9: Go to step (3).

Code for drawing line using equation

```
INPUT: (h,k), r;
theta=0; theta2=pi/4;
while(theta<=theta2){
    x=r*cos(theta);
    y=r*sin(theta);
    setPixel(floor(x), floor(y), h,k);
    theta += pi/16;
}
setPixel(int x, int y, int h, int k){
    putpixel(x+h, y+k, RED);
    putpixel(x+h, -y+k, RED);
    putpixel(-x+h, -y+k, RED);
    putpixel(-x+h, y+k, RED);
    putpixel(y+h, x+k, RED);
    putpixel(y+h, -x+k, RED);
    putpixel(-y+h, -x+k, RED);
    putpixel(-y+h, x+k, RED);
}
```

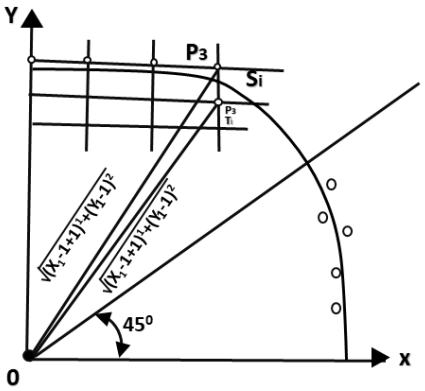
3.4 Bresenham's Circle Algorithm

❋ Scan-Converting a circle using Bresenham's algorithm works as follows: Points are generated from 90ř to 45ř, moves will be made only in the +x & -y directions as shown in fig:



❋ The best approximation of the true circle will be described by those pixels in the raster that falls the least distance from the true circle. We want to generate the points from

❋ 90ř to 45ř. Assume that the last scan-converted pixel is P1 as shown in fig. Each new point closest to the true circle can be found by taking either of two actions.



- ☆ Move in the x-direction one unit or
- ☆ Move in the x- direction one unit & move in the negative y-direction one unit.

❋ Let $D(S_i)$ is the distance from the origin to the true circle squared minus the distance to point P_3 squared. $D(T_i)$ is the distance from the origin to the true circle squared minus the distance to point P_2 squared. Therefore, the following expressions arise.

- ☆ $D(S_i) = (x_i + 1)^2 + y_i^2 - r^2$
- ☆ $D(T_i) = (x_i + 1)^2 + (y_i - 1)^2 - r^2$

❋ Since $D(Si)$ will always be $+ve$ & $D(Ti)$ will always be $-ve$, a decision variable d may be defined as follows:

❋ $d_i = D(S_i) + D(T_i)$

❋ Therefore, $d_i = (x_i + 1)^2 + y_i^2 - r^2 + (x_i + 1)^2 + (y_i - 1)^2 - r^2$

❋ From this equation, we can drive initial values of d_i as d_1 (The circle is centered at the origin so $x = 0$ *and* $y = r$.)

❋ Therefore, $d_i = (0+1)^2+r^2-r^2+(0+1)^2+(r-1)^2-r^2 = 1+1+r^2-2r+1-r^2 = 3-2r$

❋ Thereafter, if $d_i < 0$,then only x is incremented.
$x_{i+1} = x_i + 1,\ d_{i+1} = d_i + 4x_{i+1} + 6$

❋ if $d_i \geq 0$,then x *and* y are incremented
$x_{i+1} = x_{i+1},\ y_{i+1} = y_i - 1\ d_{i+1} = d_i + 4(x_{i+1} - y_{i+1}) + 10$

❋ Example: Plot 6 points of circle using Bresenham Algorithm. When radius of circle is 10 units. The circle has centre (50, 50).
Solution: Let r = 10 (Given)

- ☆ Step1: Take initial point (0, 10)
  d = 3 - 2r = 3 - 2 * 10 = -17
  d < 0 so, d = d + 4x + 6 = -17 + 4 (0) + 6 = -11

☆ Step2: Plot (1, 10)
d = d + 4x + 6 (d < 0) = -11 + 4 (1) + 6 = -1

☆ Step3: Plot (2, 10)
d = d + 4x + 6 (d < 0) = -1 + 4 x 2 + 6 = 13

☆ Step4: Plot (3, 9) d is > 0 so x = x + 1, y = y - 1
d = d + 4 (x - y) + 10 (d > 0) = 13 + 4 (3 - 9) + 10
= 13 + 4 (-6) + 10 = 23 - 24 = -1

☆ Step5: Plot (4, 9)
d = -1 + 4x + 6 = -1 + 4 (4) + 6 = 21

☆ Step6: Plot (5, 8)
d = d + 4 (x-y) + 10 (d > 0) = 21 + 4 (5-8) + 10 = 21-12 + 10 = 19

☆ P1 (0,0) ⇒ (50,50) P2 (1,10) ⇒ (51,60) P3 (2,10) ⇒ (52,60)
P4 (3,9) ⇒ (53,59) P5 (4,9) ⇒ (54,59) P6 (5,8) ⇒ (55,58)

Algorithm 6: Bresenham's Circle Algorithm

Step 1: Start Algorithm

Step 2: Declare p, q, x, y, r, d variables

p, q are coordinates of the center of the circle

r is the radius of the circle

Step 3: Enter the value of r

Step 4: Calculate d = 3 - 2r

Step 5: Initialize x = 0, y = r

Step 6: Check if the whole circle is scan converted

if $x \geq y$ **then**
| Stop

Step7: Plot eight points by using concepts of eight-way symmetry.

The center is at (p, q). Current active pixel is (x, y).

putpixel(x + p, y + q) putpixel(y + p, x + q)

putpixel(-y + p, x + q) putpixel(-x + p, y + q)

putpixel(-x + p, -y + q) putpixel(-y + p, -x + q)

putpixel(y + p, -x + q) putpixel(x + p, -y - q)

Step 8: Find location of next pixels to be scanned

if $d < 0$ **then**
| x = x + 1
| d = d + 4x + 6

if $d \geq 0$ **then**
| x = x + 1
| y = y - 1
| d = d + 4 (x - y) + 10

Step 9: Go to step (6).

Step10: Stop Algorithm

Code for drawing line using equation

```
INPUT: xc, yc, r
void BresenhamCircle(int xc,int yc,int r){
    int x=0,y=r,d=3-(2*r);
    EightWaySymmetricPlot(xc,yc,x,y);
    while(x <= y){
        if(d<=0){
            x=x+1;
            d=d+(4*x)+6;
        }
        else{
            x=x+1;
            y=y-1;
            d=d+(4*x)-(4*y)+10;
        }
        EightWaySymmetricPlot(xc,yc,x,y);
    }
}
EightWaySymmetricPlot(int xc,int yc,int x,int y) {
     putpixel(x+xc, y+yc, RED);
     putpixel(x+xc, -y+yc, YELLOW);
     putpixel(-x+xc, -y+yc, GREEN);
     putpixel(-x+xc, y+yc, YELLOW);
     putpixel(y+xc, x+yc, 12);
     putpixel(y+xc, -x+yc, 14);
     putpixel(-y+xc, -x+yc, 15);
     putpixel(-y+xc, x+yc, 6);
}
```

3.5 Mid-Point Circle Algorithm

❋ It is based on the following function for testing the spatial relationship between the arbitrary point (x, y) and a circle of radius r centered at the origin:

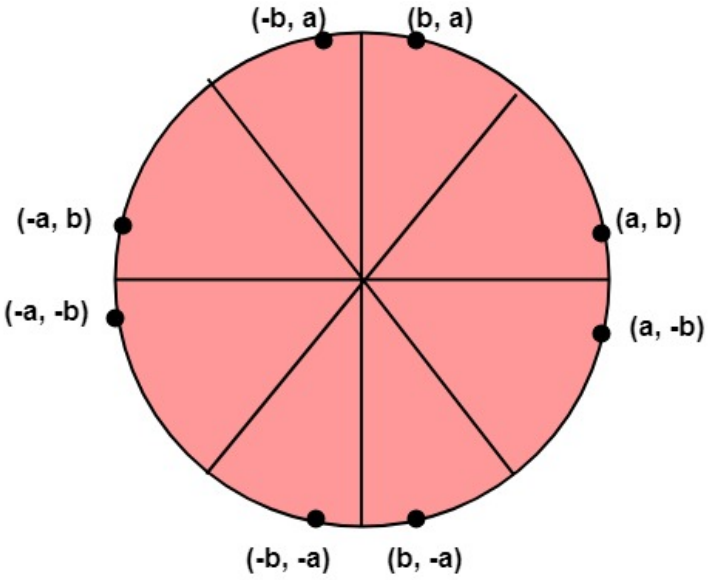
$$f(x,y) = x^2 + y^2 - r^2 \begin{bmatrix} < 0 \; for \; (x,y) \; inside \; the \; circle \\ = 0 \; for \; (x,y) on \; the \; ciecle \\ > 0 \; for \; (x,y) \; outside \; the \; circle \end{bmatrix} \quad \ldots\ldots\ldots\ldots\ldots \text{Eq (1)}$$

❋ Now, consider the coordinates of the point halfway between pixel T and pixel S

❋ This is called midpoint $(x_{i+1}, y_i - \frac{1}{2})$ and we use it to define a decision parameter:
$P_i = f(x_{i+1}, y_i - \frac{1}{2}) = (x_{i+1})^2 + (y_i - \frac{1}{2})^2 - r^2$ ..... Eq (2)

☆ If P_i is $-ve \Rightarrow$ *midpoint* is inside the circle and we choose pixel T

☆ If P_i is $+ve \Rightarrow$ *midpoint* is outside the circle (or on the circle) we choose pixel S.



❋ The decision parameter for the next step is:
$P_{i+1} = (x_{i+1}+1)^2 + (y_{i+1} - \frac{1}{2})^2 - r^2$ ..... Eq (3)

❋ Since $x_{i+1} = x_i + 1$, we have
$P_{i+1} - P_i = ((x_i+1)+1)^2 - (x_i+1)^2 + (y_{i+1} - \frac{1}{2})^2 - (y_i - \frac{1}{2})^2$
$P_{i+1} - P_i = x_i^2 + 4 + 4x_i - x_i^2 + 1 - 2x_i + y_{i+1}^2 + \frac{1}{4} - y_{i+1} - y_i^2 - \frac{1}{4} - y_i$
$P_{i+1} - P_i = 2(x_i+1) + 1 + (y_{i+1}^2 - y_i^2) - (y_{i-1} - y_i)$
$P_{i+1} = P_i + 2(x_i+1) + 1 + (y_{i+1}^2 - y_i^2) - (y_{i-1} - y_i)$ ..... Eq (4)

❋ If pixel T is chosen $\Rightarrow$ $P_i < 0$, We have $y_{i+1} = y_i$

❋ If pixel S is chosen $\Rightarrow$ $P_i \geq 0$, We have $y_{i+1} = y_i - 1$
Thus, $P_{i+1} = \begin{bmatrix} P_i + 2(x_i+1) + 1, \ if P_i < 0 \\ P_i + 2(x_i+1) + 1 - 2(y_i - 1), \ if P_i \geq 0 \end{bmatrix}$ ..... Eq (5)

❋ We can continue to simplify this in n terms of (x_i, y_i) and get
$P_{i+1} = \begin{bmatrix} P_i + 2x_i + 3, \ if P_i < 0 \\ P_i + 2(x_i - y_i + 5), \ if P_i \geq 0 \end{bmatrix}$ ..... Eq (6)

❋ Now, initial value of $P_i(0, r)$ from equation 2
$P_1 = (0+1)^2 + (r - \frac{1}{2})^2 - r^2 \ = \ 1 + \frac{1}{4} - r \ = \ \frac{5}{4} - r$
We can put $\frac{5}{4} \cong 1$, r is an integer. So, $P_1 = 1 - r$

Algorithm 7: Mid-point Circle Algorithm

Step 1: Put $x = 0$, $y = r$ in equation 2

We have $p = 1 - r$

Step 2: Repeat steps while $x \leq y$

$Plot(x, y)$

if $p < 0$ **then**

$x = x + 1$

$p = p + 2x + 3$

else

$x = x + 1$

$y = y - 1$

$p = p + 2(x - y) + 5$

end

Step 3: End

Code for drawing line using equation

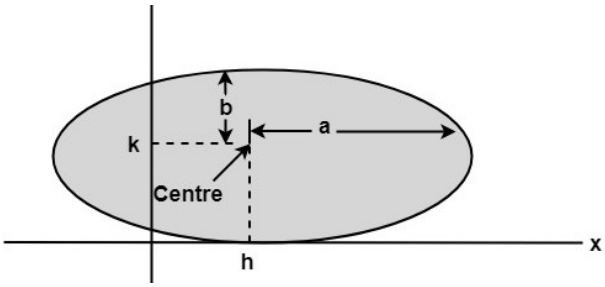
```
INPUT: x=0; y=r;
putpixel (a, b+r, RED);
putpixel (a, b-r, RED);
putpixel (a-r, b, RED);
putpixel (a+r, b, RED);
p=5/4-r;
while (x<=y){
    If (p<0)
        x = x+1;
        p = p + (4*x)+6;
    else{
        x = x+1;
        y=y-1;
        p = p + (2*(x-y))+5;
    }
    putpixel (a+x, b+y, RED);
    putpixel (a-x, b+y, RED);
    putpixel (a+x, b-y, RED);
    putpixel (a+x, b-y, RED);
    putpixel (a+x, b+y, RED);
    putpixel (a+x, b-y, RED);
    putpixel (a-x, b+y, RED);
    putpixel (a-x, b-y, RED);
}
```

Chapter 4

Scan Converting a Ellipse

4.1 Defining a Ellipse

❋ The ellipse is also a symmetric figure like a circle but is four-way symmetry rather than eight-way.



❋ The equation of an ellipse in standard form follows:

$$\frac{(x-h)^2}{a^2} + \frac{(y-k)^2}{b^2} = 1$$

❋ There two methods of defining an Ellipse:

- ☆ Polynomial Method of defining an Ellipse
- ☆ Trigonometric method of defining an Ellipse

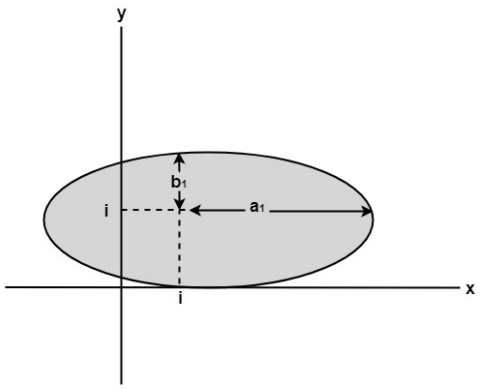
4.2 Polynomial Method

❋ The ellipse has a major and minor axis. If a_1 and b_1 are major and minor axis respectively. The centre of ellipse is (i, j). The value of x will be incremented from i to a1and value of y will be calculated using the following formula

$$y = b_1\sqrt{1 - \frac{x-i}{a_1^2}} + j$$

❋ Drawback of Polynomial Method

- ☆ It requires squaring of values. So floating point calculation is required.
- ☆ Routines developed for such calculations are very complex and slow.



Polynomial Description of Ellipse

Algorithm 8: Algorithm for Polynomial Method of defining an Ellipse

Step 1: Start

Step 2: Initialise a = length of major axis; b = length of minor axis;

(h, k) = coordinates of ellipse center; $x = 0$; $i = step$; $x_{end} = a$.

Step 2: Test to determine whether the entire ellipse has been scan-converted.

if $x > x_{end}$ **then**

| Stop

Step 3: Compute the value of the y coordinate:

$y = b\sqrt{1 - \frac{x^2}{a^2}}$

Step 4:Plot the four points, found by symmetry, at the current (x, y) coordinates:

at the current (x, y) coordinates.

$Plot(x + h, y + k)$ $Plot(-x + h, -y + k)$

$Plot(-y - h, x + k)$ $Plot(y + h, -x + k)$

Step 6: Increment $x = x + i$

Step 9: Go to step (3).

Code for drawing line using equation

```
x = 0;
xend=a;
while (x<xend){
    t = (1-((x * x)/(a * a)));
    if (t<0)
        te=-t;
    else
        te=t;
    y = b * sqrt (te);
    putpixel (h+x, k+y, RED);
    putpixel (h-x, k+y, RED);
    putpixel (h+x, y-y, RED);
    putpixel (h-x, k-y, RED);
    x = x + step;
}
```

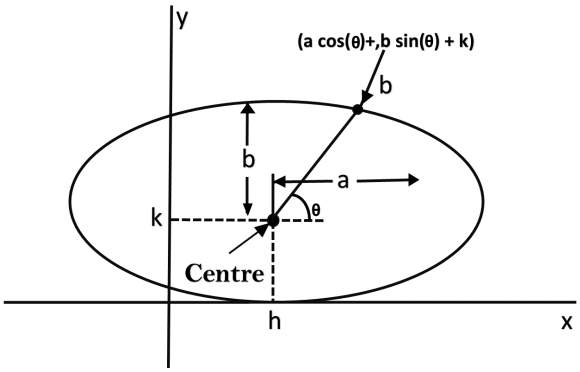
4.3 Trignometric Method

❋ The following equation defines an ellipse trigonometrically

$$x = a * cos(\theta) + h \text{ and } y = b * sin(\theta) + k,$$

Where, (x, y) = the current coordinates, a = length of major axis, b = length of minor axis, θ = current angle, (h, k) = ellipse center

❋ In this method, the value of θ is varied from 0 to $\frac{\pi}{2}$. The remaining points are found by symmetry.



❋ Drawback:

☆ This is an inefficient method.

☆ It is not an interactive method for generating ellipse.

☆ The table is required to see the trigonometric value.

☆ Memory is required to store the value of θ.

Algorithm 9: Algorithm for drawing circle using polynomial method

Step 1: Start Algorithm

Step 2: Set the initial variables: r = circle radius,
(h, k) = coordinates of the circle center, i = step size, $\theta_{end} = \frac{\pi}{4}$, $\theta = 0$

Step 3:
if $\theta > \theta_{end}$ **then**
| Stop

Step 4: Compute $x = r * cos\theta, y = r * sin\theta$

Step 5: Plot the eight points found by symmetry concerning the center (h, k)
at the current (x, y) coordinates.
$Plot(x + h, y + k)$ $Plot(-x + h, -y + k)$ $Plot(y + h, x + k)$
$Plot(-y + h, -x + k)$ $Plot(-y + h, x + k)$ $Plot(y + h, -x + k)$
$Plot(-x + h, y + k)$ $Plot(x + h, -y + k)$

Step 6: Increment $\theta = \theta + i$

Step 9: Go to step (3).

Code for drawing ellipse using Trigonometric Method

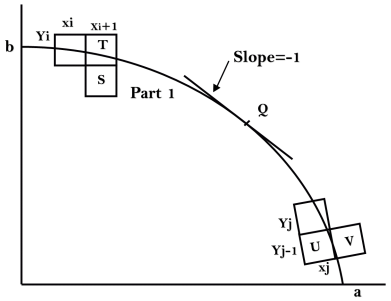
```
a, b, h, k, thetaend,step,x,y;
theta= 0;
thetaend=(pi*90)/180;
while(theta<thetaend){
    x = a * cos (theta);
    y = b * sin (theta);
    putpixel(x+h, y+k, RED);
    putpixel(-x+h, y+k, RED);
    putpixel(-x+h, -y+k, RED);
    putpixel(x+h, -y+k, RED);
    theta += step;
}
```

4.4 Midpoint Ellipse Algorithm

❋ It is very similar to the midpoint circle algorithm. Because of the four-way symmetry property we need to consider the entire elliptical curve in the first quadrant.

❋ Let's first rewrite the ellipse equation and define the function f that can be used to decide if the midpoint between two candidate pixels is inside or outside the ellipse:

$$f(x,y) = b^2x^2 + a^2y^2 - a^2b^2 = \begin{bmatrix} < 0 \; for \; (x,y) \; inside \; the \; circle \\ = 0 \; for \; (x,y) on \; the \; ciecle \\ > 0 \; for \; (x,y) \; outside \; the \; circle \end{bmatrix}$$



❋ Now divide the elliptical curve from $(0, b)$ to $(a, 0)$ into two parts at point Q where the slope of the curve is -1.

❋ Slope of the curve is defined by the $f(x, y) = 0$ is $\frac{dy}{dx} = -\frac{fx}{fy}$ where fx & fy are partial derivatives of $f(x, y)$ with respect to x & y.

❋ We have $fx = 2b^2x,\ fy = 2a^2y$ & $\frac{dy}{dx} = -\frac{2b^2x}{2a^2y}$ Hence we can monitor the slope value during the scan conversion process to detect Q. Our starting point is $(0, b)$

❋ Suppose that the coordinates of the last scan converted pixel upon entering step i are (x_i, y_i).

❋ We are to select either $T(x_{i+1}, y_i)$ or $S(x_{i+1}, y_{i-1})$ to be the next pixel. The midpoint of T & S is used to define the following decision parameter.

$$p_i = f(x_{i+1}, y_i - \tfrac{1}{2})$$
$$p_i = b^2(x_{i+1})^2 + a^2(y_i - \tfrac{1}{2})^2 - a^2b^2$$

❋ If $p_i < 0$, the midpoint is inside the curve and we choose pixel T.

❋ If $p_i > 0$, the midpoint is outside or on the curve and we choose pixel S.

❋ Decision parameter for the next step is:

$$p_{i+1} = f(x_{i+1} + 1, y_{i+1} - \tfrac{1}{2})$$
$$p_{i+1} = b^2(x_{i+1} + 1)^2 + a^2(y_{i+1} - \tfrac{1}{2})^2 - a^2b^2$$

❋ Since $x_{i+1} = x_i + 1$, we have

$$p_{i+1} - p_i = b^2(x_{i+1} + 1)^2 - b^2(x_{i+1})^2) + a^2(y_{i+1} - \tfrac{1}{2})^2 - (y_i - \tfrac{1}{2})^2]$$
$$p_{i+1} = p_i + 2b^2x_{i+1} + b^2 + a^2[(y_{i+1} - \tfrac{1}{2})^2 - (y_i - \tfrac{1}{2})^2]$$

❋ If T is chosen pixel ($p_i < 0$), we have $y_{i+1} = y_i$.

❋ If S is chosen pixel ($p_i > 0$) we have $y_{i+1} = y_{i-1}$. Thus we can express

❋ p_{i+1} in terms of p_i and (x_{i+1}, y_{i+1}):

$$p_{i+1} = p_i + 2b^2x_{i+1} + b^2 \quad \text{if } p_i < 0$$
$$p_{i+1} = p_i + 2b^2x_{i+1} + b^2 - 2a^2y_{i+1} \quad \text{if } p_i > 0$$

❋ The initial value for the recursive expression can be obtained by the evaluating the original definition of pi with $(0, b)$:

$$p_1 = (b^2 + a^2(b - \tfrac{1}{2})^2 - a^2b^2$$
$$p_1 = b^2 - a^2b + \tfrac{a^2}{4}$$

❋ Suppose the pixel (x_jy_j) has just been scan converted upon entering step j. The next pixel is either $U(x_j, y_{j-1})$ or $V(x_{j+1}, y_{j-1})$. The midpoint of the horizontal line connecting U & V is used to define the decision parameter:

$$q_j = f(x_j + \tfrac{1}{2}, y_j - 1)$$
$$q_j = b^2(x_j + \tfrac{1}{2})^2 + a^2(y_j - 1)^2 - a^2b^2$$

❋ If $q_j < 0$, the midpoint is inside the curve and we choose pixel V.

❋ If $q_j \geq 0$, the midpoint is outside the curve and we choose pixel U. Decision parameter for the next step is:

$$q_{j+1} = f(x_{j+1} + \tfrac{1}{2}, y_{j+1} - 1)$$
$$q_{j+1} = b^2(x_{j+1} + \tfrac{1}{2})^2 + a^2(y_{j+1} - 1)^2 - a^2b^2$$

❋ Since $y_{j+1} = y_j - 1$,we have

$$q_{j+1} - q_j = b^2[(x_{j+1} + \tfrac{1}{2})^2 - (x_j + \tfrac{1}{2})^2] + a^2(y_{j+1} - 1)^2 - (y_{j+1})^2]$$
$$q_{j+1} = q_j + b^2[(x_{j+1} + \tfrac{1}{2})^2 - (x_j + \tfrac{1}{2})^2] - 2a^2y_{j+1} + a^2$$

❋ If V is chosen pixel ($q_j < 0$), we have $x_{j+1} = x_j$.

❋ If U is chosen pixel ($q_j \geq 0$) we have $x_{j+1} = x_j + 1$. Thus we can express

❋ q_{j+1} in terms of q_j and (x_{j+1}, y_{j+1}):

$$q_{j+1} = q_j + 2b^2x_{j+1} - 2a^2y_{j+1} + a^2 \quad \text{if } q_j < 0$$
$$q_{j+1} = q_j - 2a^2y_{j+1} + a^2 \quad \text{if } q_j \geq 0$$

❋ The initial value for the recursive expression is computed using the original definition of q_j. And the coordinates of (x_k, y_k) of the last pixel choosen for the part 1 of the curve:

$$q_1 = f(x_k + \tfrac{1}{2}, y_{k-1}) = b^2(x_k + \tfrac{1}{2})^2 - a^2(y_{k-1})^2 - a^2b^2$$

Algorithm 10: Algorithm for Mid-point Ellipse Algorithm

Step 1: Start Algorithm

Step 2: Declare variable $x_1, y1, aa_1, bb_1, aa_2, bb_2, fx, fy, p_1, a_1, b_1$

Step 3: Initialize $x_1 = 0$ and $y_1 = b$

Step 4: Calculate $aa_1 = a_1 * a_1$, $bb_1 = b_1 * b_1$, $aa_2 = aa_1 * 2$, $bb_2 = bb_1 * 2$

Step 5: Initialize $fx = 0$, $fy = aa_2 * b_1$

Step 6: Calculate $p_1 = bb_1 - aa_1 * b_1 + 0.25 * aa_1$

Step 7:

while $fx < fy$ **do**
  $Setpixel(x_1, y_1)$
  $x = x + 1$
  $fx = fx + bb_2$
  if $p_1 < 0$ **then**
    $p_1 = p_1 + fx + bb_1$
  else
    $y = y - 1$
    $fy = fy - bb_2;$
    $p_1 = p_1 + fx + bb_1 - fy$
  end
end

Step 8: $Setpixel(x_1,\ y_1)$

Step 9: Calculate $p_1 = bb_1(x + 0.5)(x + 0.5) + aa(y - 1)(y - 1) - aa_1 * bb_1$

Step 10:

while $fx \geq fy$ **do**
  $y = y - 1$
  $fy = fx - aa_+2$
  if $p_1 \geq 0$ **then**
    $p_1 = p_1 - fx + aa_1$
  else
    $x = x + 1$
    $fx = fx + bb_2$
    $p_1 = p_1 + fx - fy - aa_1$
  end
end

Step 11: $Setpixel(x_1, y_1)$

Step 12: Stop Algorithm

Code for drawing ellipse using mid-point algorithm

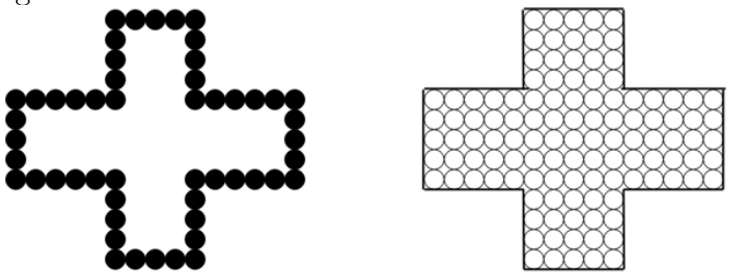
```
float dx, dy, d1, d2, x=0, y=ry, rx, ry, xc, yc;
// Initial decision parameter of region 1
d1 = (ry * ry) - (rx * rx * ry) + (0.25f * rx * rx);
dx = 2 * ry * ry * x;
dy = 2 * rx * rx * y;
while (dx < dy){
    putpixel(x+xc, y+yc, RED);
    putpixel(-x+xc, y+yc, RED);
    putpixel(-x+xc, -y+yc, RED);
    putpixel(x+xc, -y+yc, RED);
    if (d1 < 0) {
        x++;
        dx = dx + (2 * ry * ry);
        d1 = d1 + dx + (ry * ry);
    }else{
        x++;
        y--;
        dx = dx + (2 * ry * ry);
          dy = dy - (2 * rx * rx);
        d1 = d1 + dx - dy + (ry * ry);
    }
}

d2=((ry*ry)*((x+0.5f)*(x+0.5f)))+((rx*rx)*((y-1)*(y-1)))-(rx*rx*ry*ry);
while (y >= 0) {
    putpixel(x+xc, y+yc, RED);
    putpixel(-x+xc, y+yc, RED);
    putpixel(-x+xc, -y+yc, RED);
    putpixel(x+xc, -y+yc, RED);
    if (d2 > 0) {
        y--;
        dy = dy - (2 * rx * rx);
        d2 = d2 + (rx * rx) - dy;
    }else {
        y--;
        x++;
        dx = dx + (2 * ry * ry);
        dy = dy - (2 * rx * rx);
        d2 = d2 + dx - dy + (rx * rx);
    }
}
```

Chapter 5

Filled Area Primitives

❋ Region filling is the process of filling image or region. Filling can be of boundary or interior region as shown in fig. Boundary Fill algorithms are used to fill the boundary and flood-fill algorithm are used to fill the interior.

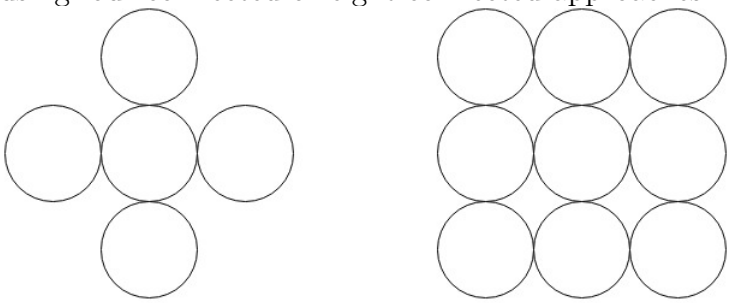


Boundary Filled Region Interior or Flood Filled Region

5.1 Boundary Filled Algorithm

❋ This algorithm uses the recursive method. First of all, a starting pixel called as the seed is considered. The algorithm checks boundary pixel or adjacent pixels are coloured or not.

❋ If the adjacent pixel is already filled or coloured then leave it, otherwise fill it. The filling is done using four connected or eight connected approaches.



Four Connected Eight Connected

❋ Four connected approaches is more suitable than the eight connected approaches.

1. Four connected approaches: In this approach, left, right, above, below pixels are tested.
2. Eight connected approaches: In this approach, left, right, above, below and four diagonals are selected.

❋ Boundary can be checked by seeing pixels from left and right first. Then pixels are checked by seeing pixels from top to bottom. The algorithm takes time and memory because some recursive calls are needed.

❋ **Problem with recursive boundary fill algorithm:** It may not fill regions sometimes correctly when some interior pixel is already filled with color. The algorithm will

check this boundary pixel for filling and will found already filled so recursive process will terminate. This may vary because of another interior pixel unfilled.

❋ So check all pixels color before applying the algorithm.

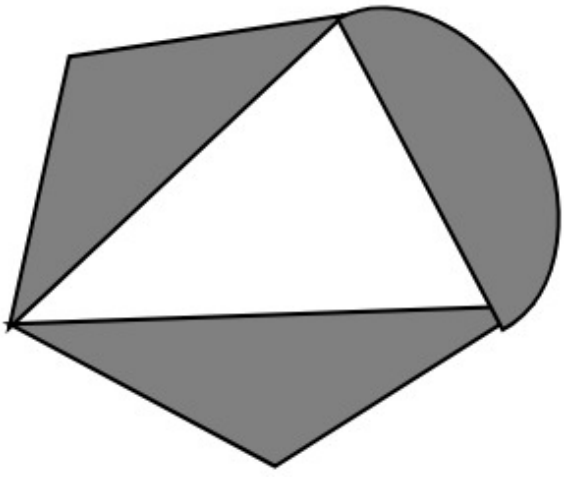
Code for boundary filling algorithm

```
Procedure fill (x, y, color, color1)
int c;
c=getpixel (x, y);
if (c!=color) (c!=color1){
    setpixel (x, y, color)
    fill (x+1, y, color, color 1);
    fill (x-1, y, color, color 1);
    fill (x, y+1, color, color 1);
    fill (x, y-1, color, color 1);
}
```

5.2 Flood Fill Algorithm

❋ In this method, a point or seed which is inside region is selected. This point is called a seed point. Then four connected approaches or eight connected approaches is used to fill with specified colour.

❋ The flood fill algorithm has many characters similar to boundary fill. But this method is more suitable for filling multiple colours boundary. When boundary is of many colours and interior is to be filled with one colour we use this algorithm.



❋ In fill algorithm, we start from a specified interior point (x, y) and reassign all pixel values are currently set to a given interior colour with the desired colour.

❋ Using either a 4-connected or 8-connected approaches, we then step through pixel positions until all interior points have been repainted.

❋ Disadvantage:

- ❋ Very slow algorithm
- ❋ May be fail for large polygons
- ❋ Initial pixel required more knowledge about surrounding pixels.

❋ So check all pixels colour before applying the algorithm.

Code for filling algorithm

```
Procedure floodfill (x, y,fill_ color, old_color: integer){
    If (getpixel (x, y)=old_color){
        setpixel (x, y, fill_color);
        fill (x+1, y, fill_color, old_color);
        fill (x-1, y, fill_color, old_color);
        fill (x, y+1, fill_color, old_color);
        fill (x, y-1, fill_color, old_color);
    }
}
```

5.3 Scan Line Polygon Fill Algorithm

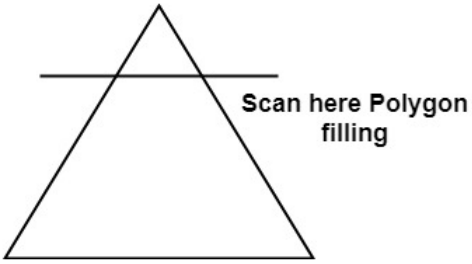
❋ This algorithm lines interior points of a polygon on the scan line and these points are done on or off according to requirement. The polygon is filled with various colours by colouring various pixels.

❋ In above figure polygon and a line cutting polygon in shown. First of all, scanning is done. Scanning is done using raster scanning concept on display device.

❋ The beam starts scanning from the top left corner of the screen and goes toward the bottom right corner as the endpoint.

❋ The algorithms find points of intersection of the line with polygon while moving from left to right and top to bottom. The various points of intersection are stored in the frame buffer.

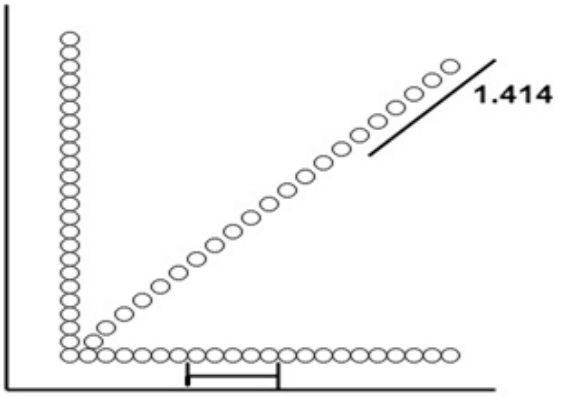
❋ The intensities of such points is keep high. Concept of coherence property is used. According to this property if a pixel is inside the polygon, then its next pixel will be inside the polygon.



❋ Side effects of Scan Conversion:

1. Staircase or Jagged: Staircase like appearance is seen while the scan was converting line or circle.

2. Unequal Intensity: It deals with unequal appearance of the brightness of different lines. An inclined line appears less bright as compared to the horizontal and vertical line.



Pixels along with horizontal line are 1 unit apart and vertical.
Pixels along diagonal line are 1.414 units.

Chapter 6

2D Transformation

6.1 Introduction of Transformations

❋ Computer Graphics provide the facility of viewing object from different angles. The architect can study building from different angles i.e.

1. Front Evaluation
2. Side elevation
3. Top plan

❋ A Cartographer can change the size of charts and topographical maps. So if graphics images are coded as numbers, the numbers can be stored in memory. These numbers are modified by mathematical operations called as Transformation.

❋ The purpose of using computers for drawing is to provide facility to user to view the object from different angles, enlarging or reducing the scale or shape of object called as Transformation.

❋ Two essential aspects of transformation are given below:

1. Each transformation is a single entity. It can be denoted by a unique name or symbol.
2. It is possible to combine two transformations, after connecting a single transformation is obtained, e.g., A is a transformation for translation. The B transformation performs scaling. The combination of two is C=AB. So C is obtained by concatenation property.

❋ There are two complementary points of view for describing object transformation.

1. Geometric Transformation: The object itself is transformed relative to the coordinate system or background. The mathematical statement of this viewpoint is defined by geometric transformations applied to each point of the object.
2. Coordinate Transformation: The object is held stationary while the coordinate system is transformed relative to the object. This effect is attained through the application of coordinate transformations.

❋ An example that helps to distinguish these two viewpoints: The movement of an automobile against a scenic background we can simulate this by

☆ Moving the automobile while keeping the background fixed-(Geometric Transformation)

- ☆ We can keep the car fixed while moving the background scenery- (Coordinate Transformation)

❋ Types of Transformations:

1. Translation
2. Scaling
3. Rotating
4. Reflection
5. Shearing
6. Translation

6.2 Translation

❋ It is the straight line movement of an object from one position to another is called Translation. Here the object is positioned from one coordinate location to another.

6.2.1 Translation of point

❋ To translate a point from coordinate position (x, y) to another (x_1y_1), we add algebraically the translation distances T_x and T_y to original coordinate.

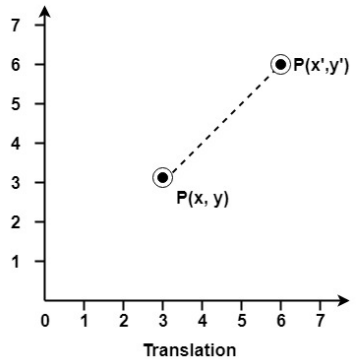
$$x_1 = x + T_x \; and \; y_1 = y + T_y$$

❋ The translation pair (Tx,Ty) is called as shift vector.

❋ Translation is a movement of objects without deformation. Every position or point is translated by the same amount. When the straight line is translated, then it will be drawn using endpoints.

❋ For translating polygon, each vertex of the polygon is converted to a new position. Similarly, curved objects are translated. To change the position of the circle or ellipse its center coordinates are transformed, then the object is drawn using new coordinates.

❋ Let P is a point with coordinates (x, y). It will be translated as (x^1y^1).

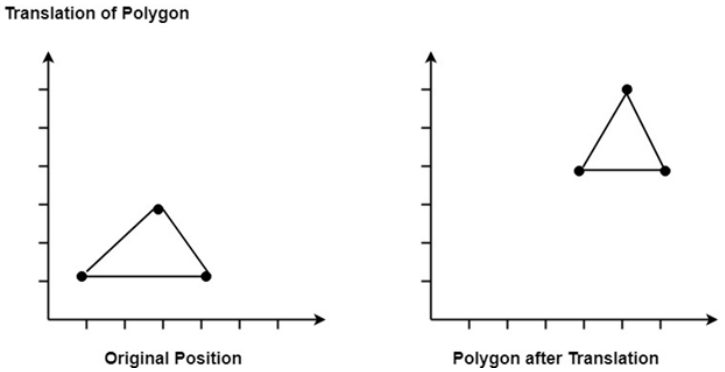


❋ Matrix for homogeneous co-ordinate translation

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix} \text{ or } \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

6.3 Scaling

❋ It is used to alter or change the size of objects. The change is done using scaling factors. There are two scaling factors, i.e. S_x in x direction Sy in y-direction. If the



original position is x and y. Scaling factors are S_x and S_y then the value of coordinates after scaling will be x_1 and y_1.

❋ If the picture to be enlarged to twice its original size then $S_x = S_y = 2$. If Sx and Sy are not equal then scaling will occur but it will elongate or distort the picture.

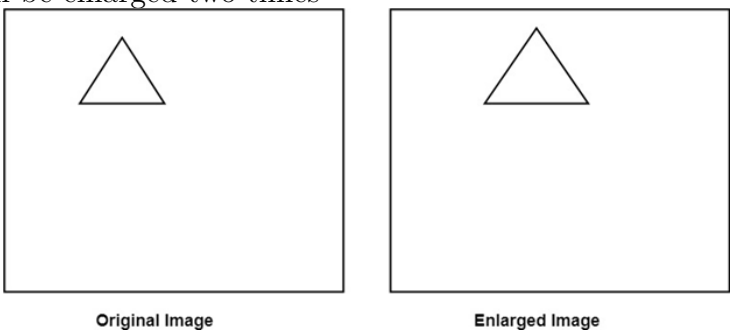
❋ If scaling factors are less than one, then the size of the object will be reduced. If scaling factors are higher than one, then the size of the object will be enlarged.

❋ If S_x and S_y are equal it is also called as Uniform Scaling. If not equal then called as Differential Scaling. If scaling factors with values less than one will move the object closer to coordinate origin, while a value higher than one will move coordinate position farther from origin.

❋ Enlargement: If $T_1 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$, If (x_1, y_1) is original position and T_1 is translation vector then (x_2, y_2) are coordinated after scaling

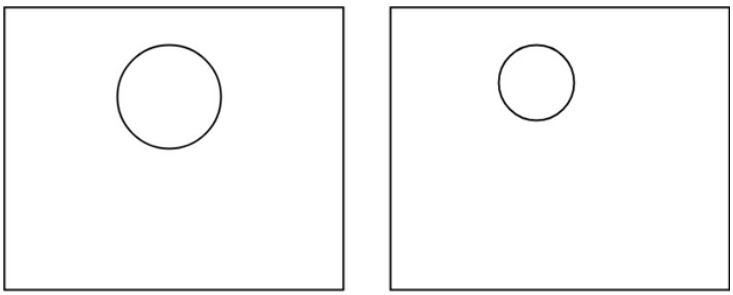
$$[x_2, y_2] = [x_1, y_1] \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = [2x_1, 2y_1]$$

❋ The image will be enlarged two times



❋ Enlargement: If $T_1 = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$, If (x_1, y_1) is original position and T_1 is translation vector then (x_2, y_2) are coordinated after scaling

$$[x_2, y_2] = [x_1, y_1] \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} = [0.5x_1, 0.5y_1]$$



❋ Matrix for homogeneous co-ordinate scaling:

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

❋ Example: Prove that 2D Scaling transformations are commutative i.e, $S_1 S_2 = S_2 S_1$.

❋ Solution: S_1 and S_2 are scaling matrices

$$S_1 = \begin{bmatrix} Sx_1 & 0 & 0 \\ 0 & Sx_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S_1 = \begin{bmatrix} Sx_2 & 0 & 0 \\ 0 & Sx_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S_1 * S_2 = \begin{bmatrix} Sx_1 & 0 & 0 \\ 0 & Sx_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Sx_2 & 0 & 0 \\ 0 & Sx_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S_1 * S_2 = \begin{bmatrix} Sx_1Sx_2 & 0 & 0 \\ 0 & Sx_1Sx_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \dots \text{Eq 1}$$

$$S_2 * S_1 = \begin{bmatrix} Sx_2 & 0 & 0 \\ 0 & Sx_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Sx_1 & 0 & 0 \\ 0 & Sx_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S_1 * S_2 = \begin{bmatrix} Sx_2Sx_1 & 0 & 0 \\ 0 & Sx_2Sx_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \dots \text{Eq 2}$$

❋ From equation 1 and equation 2 $S_1 * S_2 = S_2 * S_1$ Hence proved.

6.4 Rotation

❋ It is a process of changing the angle of the object. Rotation can be clockwise or anticlockwise.

❋ For rotation, we have to specify the angle of rotation and rotation point. Rotation point is also called a pivot point. It is print about which object is rotated.

- Types of Rotation
  - Anticlockwise
  - Counterclockwise
- The positive value of the pivot point (rotation angle) rotates an object in a counter-clockwise (anti-clockwise) direction.
- The negative value of the pivot point (rotation angle) rotates an object in a clockwise direction.
- When the object is rotated, then every point of the object is rotated by the same angle.
  - **Straight Line:** Straight Line is rotated by the endpoints with the same angle and redrawing the line between new endpoints.
  - **Polygon:** Polygon is rotated by shifting every vertex using the same rotational angle.
  - **Curved Lines:** Curved Lines are rotated by repositioning of all points and drawing of the curve at new positions.
  - **Circle:** It can be obtained by center position by the specified angle.
  - **Ellipse:** Its rotation can be obtained by rotating major and minor axis of an ellipse by the desired angle.
- Matrix for rotation is a clockwise direction.

$$T_1 = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$$

- Matrix for rotation is an anticlockwise direction.

$$T_1 = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}$$

- Matrix for homogeneous co-ordinate rotation (clockwise)

$$T_1 = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Matrix for homogeneous co-ordinate rotation (anticlockwise)

$$T_1 = \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- **Rotation about an arbitrary point:** If we want to rotate an object or point about an arbitrary point, first of all, we translate the point about which we want to rotate to the origin. Then rotate point or object about the origin, and at the end, we again translate it to the original place. We get rotation about an arbitrary point.
- **Example1:** Prove that 2D rotations about the origin are commutative i.e. $R_1R_2 = R_2R_1$.

❋ **Solution:** R_1 and R_2 are rotation matrices

$$R_1 = \begin{bmatrix} cos\theta_1 & sin\theta_1 & 0 \\ -sin\theta_1 & cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_2 = \begin{bmatrix} cos\theta_2 & sin\theta_2 & 0 \\ -sin\theta_2 & cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_1 * R_2 = \begin{bmatrix} cos\theta_1 & sin\theta_1 & 0 \\ -sin\theta_1 & cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} cos\theta_2 & sin\theta_2 & 0 \\ -sin\theta_2 & cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_1 * R_2 = \begin{bmatrix} cos\theta_1 cos\theta_2 - sin\theta_1 sin\theta_2 & cos\theta_1 sin\theta_2 + sin\theta_1 cos\theta_2 & 0 \\ -sin\theta_1 cos\theta_2 - cos\theta_1 sin\theta_2 & -sin\theta_1 sin\theta_2 + cos\theta_1 cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{Qq 1}$$

$$R_2 * R_1 = \begin{bmatrix} cos\theta_2 & sin\theta_2 & 0 \\ -sin\theta_2 & cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} cos\theta_1 & sin\theta_1 & 0 \\ -sin\theta_1 & cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_1 * R_2 = \begin{bmatrix} cos\theta_2 cos\theta_1 - sin\theta_2 sin\theta_1 & cos\theta_2 sin\theta_1 + sin\theta_2 cos\theta_1 & 0 \\ -sin\theta_2 cos\theta_1 - cos\theta_2 sin\theta_1 & -sin\theta_2 sin\theta_1 + cos\theta_2 cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{Qq 2}$$

from Eq 1 and Eq 2 $R_1R_2 = R_2R_1$ Hence proved

❋ Example2: Rotate a line CD whose endpoints are $(3, 4)$ and $(12, 15)$ about origin through a 45° anticlockwise direction.

❋ Solution: The point $C(3, 4)$

$$R = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}, \theta = 45°, R = \begin{bmatrix} cos45° & sin45° \\ -sin45° & cos45° \end{bmatrix} = \begin{bmatrix} 0.707 & 0.707 \\ -0.707 & 0.707 \end{bmatrix}$$

❋ The point A(3,4) after the rotation will be

$$[x, y] = [3, 4] \begin{bmatrix} 0.707 & 0.707 \\ -0.707 & 0.707 \end{bmatrix}$$

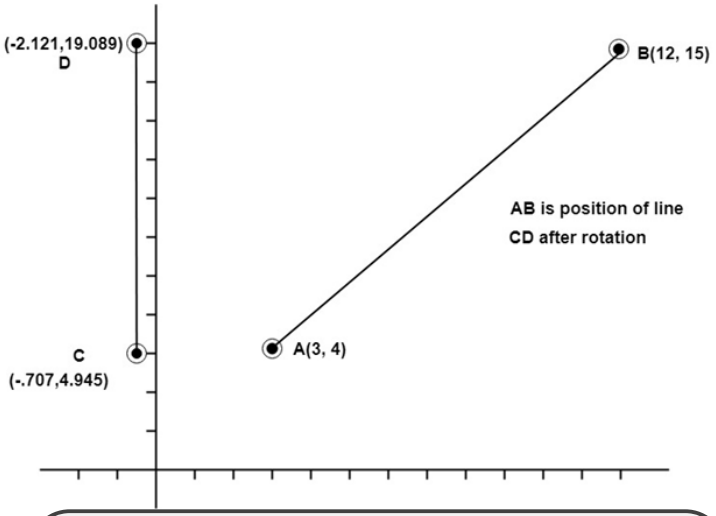
$$[x, y] = \begin{bmatrix} 3 * 0.707 - 4 * 0.707 & 3 * 0.707 + 4 * 0.707 \end{bmatrix} = \begin{bmatrix} -0.707 & 4.949 \end{bmatrix}$$

❋ The point B(12,15) after the rotation will be

$$[x, y] = [12, 15] \begin{bmatrix} 0.707 & 0.707 \\ -0.707 & 0.707 \end{bmatrix}$$

$$[x, y] = \begin{bmatrix} 12 * 0.707 - 15 * 0.707 & 12 * 0.707 + 15 * 0.707 \end{bmatrix} = \begin{bmatrix} -2.121 & 19.089 \end{bmatrix}$$

❋ Example3: Rotate line AB whose endpoints are A (2, 5) and B (6, 12) about origin through a 30ř clockwise direction.



Code for Scaling

```
int x1,y1,x2,y2;
double s,c, angle;
line(x1,y1,x2,y2);
c = cos(angle*M_PI/180);
s = sin(angle*M_PI/180);
x1 = floor(x1 * c + y1 * s);
y1 = floor(-x1 * s + y1 * c);
x2 = floor(x2 * c + y2 * s);
y2 = floor(-x2 * s + y2 * c);
x3 = floor(x3 * c + y3 * s);
y3 = floor(-x3 * s + y3 * c);
line(x1, y1 ,x2, y2);
line(x2,y2, x3,y3);
line(x3, y3, x1, y1)
```

6.5 Reflection

❄ It is a transformation which produces a mirror image of an object. The mirror image can be either about x-axis or y-axis. The object is rotated by180ř.

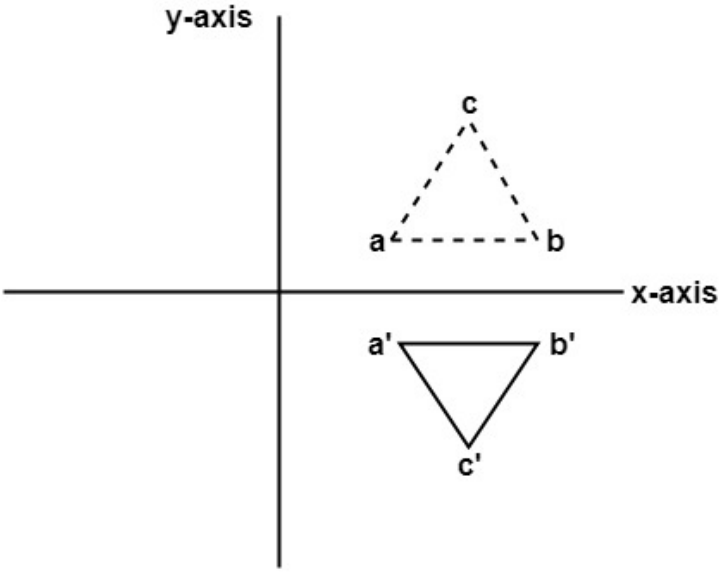
❄ Types of Reflection:

- ☆ Reflection about the x-axis
- ☆ Reflection about the y-axis
- ☆ Reflection about an axis perpendicular to xy plane and passing through the origin
- ☆ Reflection about line $y = x$

❄ **Reflection about x-axis:** The object can be reflected about x-axis with the help of the following matrix

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

❋ In this transformation value of x will remain same whereas the value of y will become negative. Following figures shows the reflection of the object axis. The object will lie another side of the x-axis.

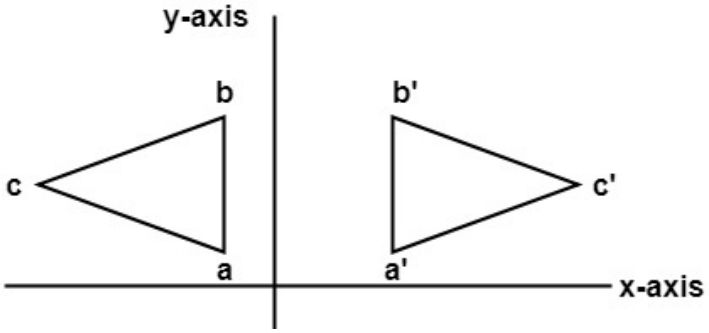


❋ **Reflection about y-axis:** The object can be reflected about y-axis with the help of following transformation matrix

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

❋ Here the values of x will be reversed, whereas the value of y will remain the same. The object will lie another side of the y-axis.

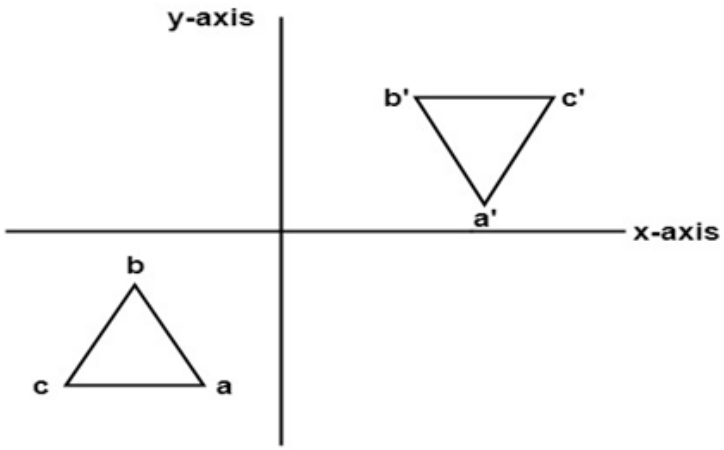
❋ The following figure shows the reflection about the y-axis



❋ **Reflection about an axis perpendicular to xy plane and passing through origin:** In the matrix of this transformation is given below

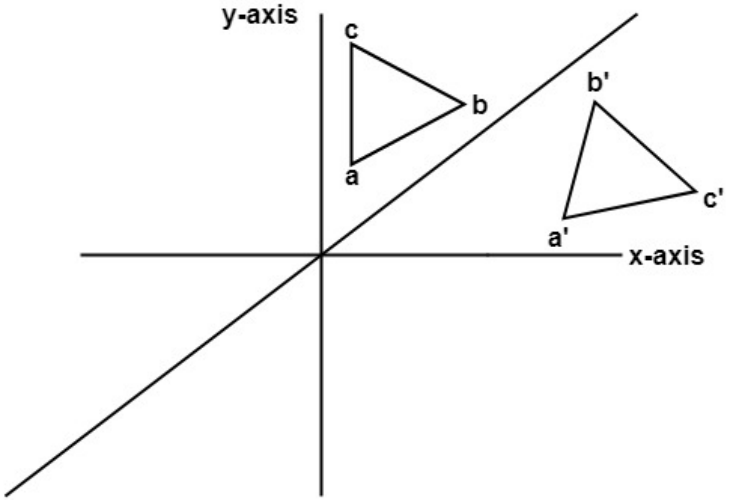
$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

❋ In this value of x and y both will be reversed. This is also called as half revolution about the origin.



❋ **Reflection about line y=x:** The object may be reflected about line y = x with the help of following transformation matrix

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

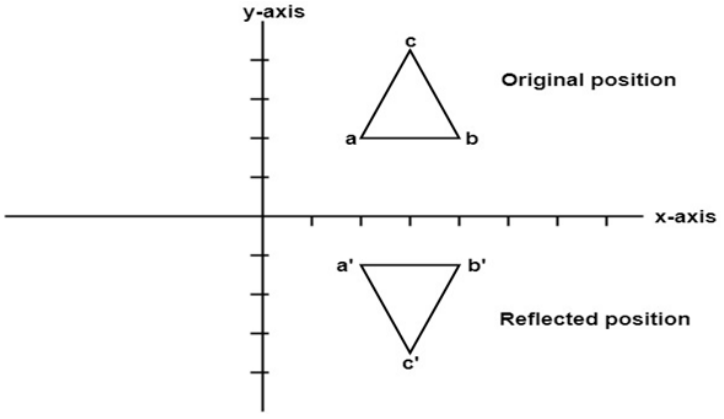


❋ First of all, the object is rotated at 45°. The direction of rotation is clockwise. After it reflection is done concerning x-axis. The last step is the rotation of $y = x$ back to its original position that is counterclockwise at 45°.

❋ Example: A triangle ABC is given. The coordinates of A, B, C are given as A(3,4), B(6,4), C(4,8)

❋ Find reflected position of triangle i.e., to the x-axis.

❋ Solution:



❋ The matrix for the reflection about x axis $\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

❋ The a point coordinates after reflection $[x, y] = [3, 4] \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [3, -4]$

❋ The b point coordinates after reflection $[x, y] = [6, 4] \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [6, -4]$

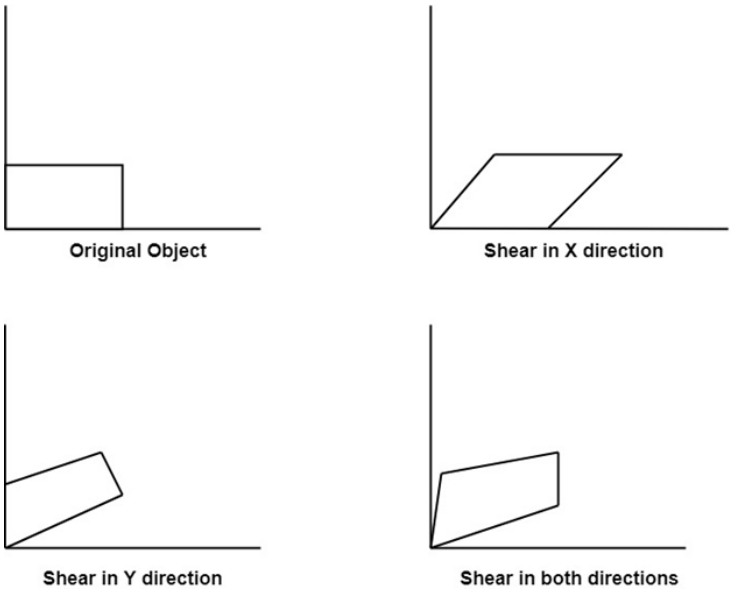
❋ The coordinate of point c after reflection $[x, y] = [4, 8] \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [4, -8]$

6.6 Shearing

❋ It is transformation which changes the shape of object. The sliding of layers of object occur. The shear can be in one direction or in two directions.

❋ **Shearing in the X-direction:** In this horizontal shearing sliding of layers occur. The homogeneous matrix for shearing in the x-direction is shown below:

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



❋ **Shearing in the Y-direction:** Here shearing is done by sliding along vertical or y-axis.

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

❋ **Shearing in X-Y directions:** Here layers will be slided in both x as well as y direction. The sliding will be in horizontal as well as vertical direction. The shape of

the object will be distorted. The matrix of shear in both directions is given by:

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