Introduction of Computer Graphics

Computer Graphics involves technology to access. The Process transforms and presents information in a visual form. The role of computer graphics insensible. In today life, computer graphics has now become a common element in user interfaces, T.V. commercial motion pictures.

Computer Graphics is the creation of pictures with the help of a computer. The end product of the computer graphics is a picture it may be a business graph, drawing, and engineering.

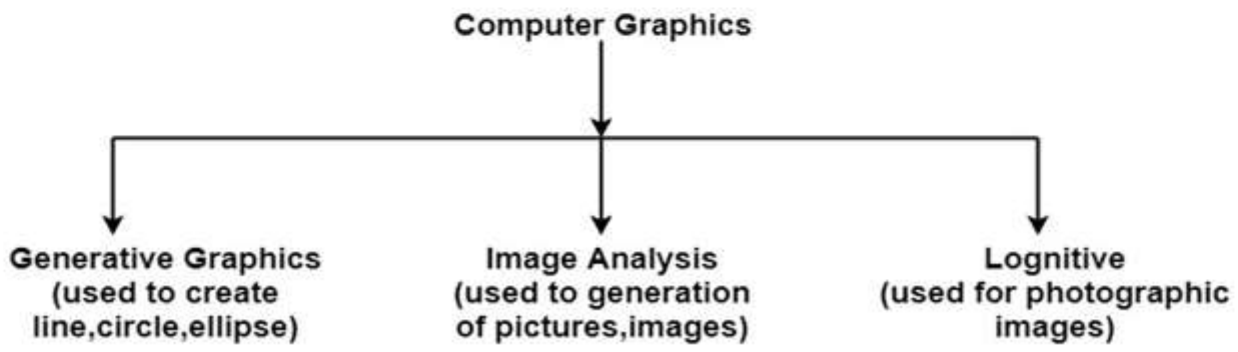
In computer graphics, two or three-dimensional pictures can be created that are used for research. Many hardware devices algorithm has been developing for improving the speed of picture generation with the passes of time. It includes the creation storage of models and image of objects. These models for various fields like engineering, mathematical and so on.

It is the use of computers to create and manipulate pictures on a display device. It comprises of software techniques to create, store, modify, represents pictures.

Why computer graphics used?

Suppose a shoe manufacturing company want to show the sale of shoes for five years. For this vast amount of information is to store. So a lot of time and memory will be needed. This method will be tough to understand by a common man. In this situation graphics is a better alternative. Graphics tools are charts and graphs. Using graphs, data can be represented in pictorial form. A picture can be understood easily just with a single look.

Interactive computer graphics work using the concept of two-way communication between computer users. The computer will receive signals from the input device, and the picture is modified accordingly. Picture will be changed quickly when we apply command.


Computer Graphics
Generative Graphics (used to create line,circle,ellipse)
Image Analysis (used to generation of pictures,images)
Lognitive (used for photographic images)

Application of Computer Graphics

1. Education and Training: Computer-generated model of the physical, financial and economic system is often used as educational aids. Model of physical systems, physiological system, population trends or equipment can help trainees to understand the operation of the system.

For some training applications, particular systems are designed. For example Flight Simulator.

Flight Simulator: It helps in giving training to the pilots of airplanes. These pilots spend much of their training not in a real aircraft but on the ground at the controls of a Flight Simulator.

Advantages:

1. Fuel Saving
2. Safety
3. Ability to familiarize the training with a large number of the world's airports.

2. Use in Biology: Molecular biologist can display a picture of molecules and gain insight into their structure with the help of computer graphics.

3. Computer-Generated Maps: Town planners and transportation engineers can use computer-generated maps which display data useful to them in their planning work.

4. Architect: Architect can explore an alternative solution to design problems at an interactive graphics terminal. In this way, they can test many more solutions that would not be possible without the computer.

5. Presentation Graphics: Example of presentation Graphics are bar charts, line graphs, pie charts and other displays showing relationships between multiple parameters. Presentation Graphics is commonly used to summarize

- Financial Reports
- Statistical Reports
- Mathematical Reports
- Scientific Reports
- Economic Data for research reports
- Managerial Reports
- Consumer Information Bulletins

- And other types of reports

6. Computer Art: Computer Graphics are also used in the field of commercial arts. It is used to generate television and advertising commercial.

7. Entertainment: Computer Graphics are now commonly used in making motion pictures, music videos and television shows.

8. Visualization: It is used for visualization of scientists, engineers, medical personnel, business analysts for the study of a large amount of information.

9. Educational Software: Computer Graphics is used in the development of educational software for making computer-aided instruction.

10. Printing Technology: Computer Graphics is used for printing technology and textile design.

Example of Computer Graphics Packages:

1. LOGO
2. COREL DRAW
3. AUTO CAD
4. 3D STUDIO
5. CORE
6. GKS (Graphics Kernel System)
7. PHIGS
8. CAM (Computer Graphics Metafile)
9. CGI (Computer Graphics Interface)

Interactive and Passive Graphics

(a) Non-Interactive or Passive Computer Graphics:

In non-interactive computer graphics, the picture is produced on the monitor, and the user does not have any controlled over the image, i.e., the user cannot make any change in the rendered image. One example of its Titles shown on T.V.

Non-interactive Graphics involves only one-way communication between the computer and the user, User can see the produced image, and he cannot make any change in the image.

(b) Interactive Computer Graphics:

In interactive Computer Graphics user have some controls over the picture, i.e., the user can make any change in the produced image. One example of it is the ping-pong game.

Interactive Computer Graphics 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.

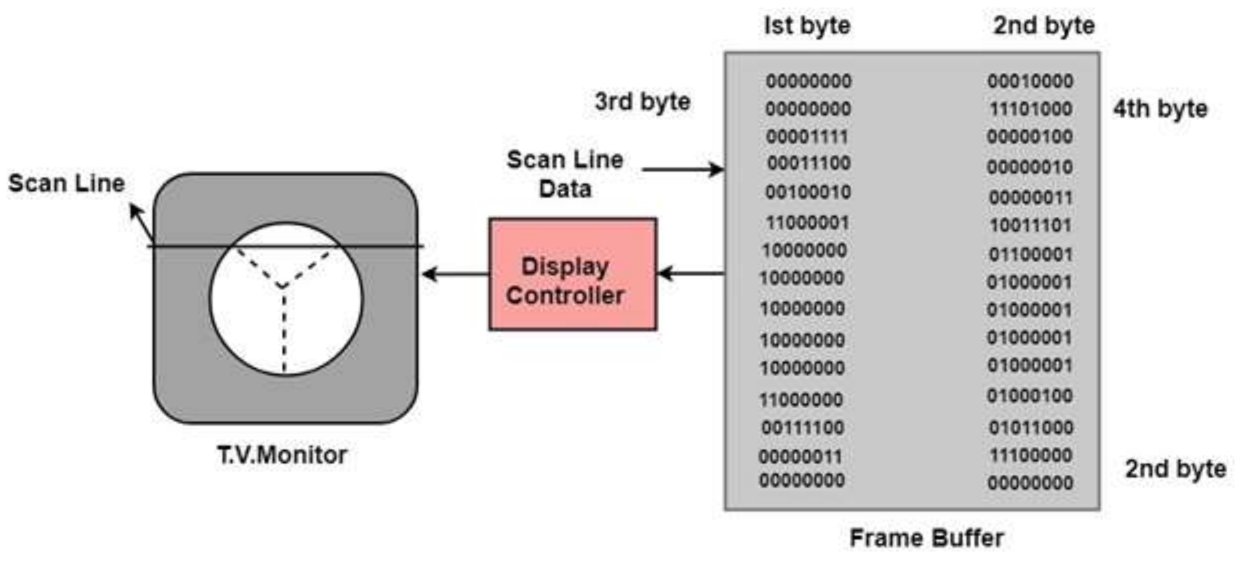
Advantages:

1. Higher Quality
2. More precise results or products
3. Greater Productivity
4. Lower analysis and design cost
5. Significantly enhances our ability to understand data and to perceive trends.

Working of Interactive Computer Graphics:

The modern graphics display is very simple in construction. It consists of three components:

1. Frame Buffer or Digital Memory
2. A Monitor likes a home T.V. set without the tuning and receiving electronics.
3. **Display Controller or Video Controller:** It passes the contents of the frame buffer to the monitor.



Frame Buffer

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.
- A 1024 x 1024 element requires 2^{20} (2^{10}=1024;2^{20}=1024 x 1024)sq.raster or 1,048,576 memory bits in a single 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.

Properties of Video Monitor:

1. Persistence: Persistence is the duration of phosphorescence. Different kinds of phosphors are available for use in CRT. Besides color, a major difference between phosphor in their persistence how they continue to emit light after the electron beam is removed.

2. Resolution: Use to describe the number of pixels that are used on display image.

3. Aspect Ratio: It is the ratio of width to its height. Its measure is unit in length or number of pixels.

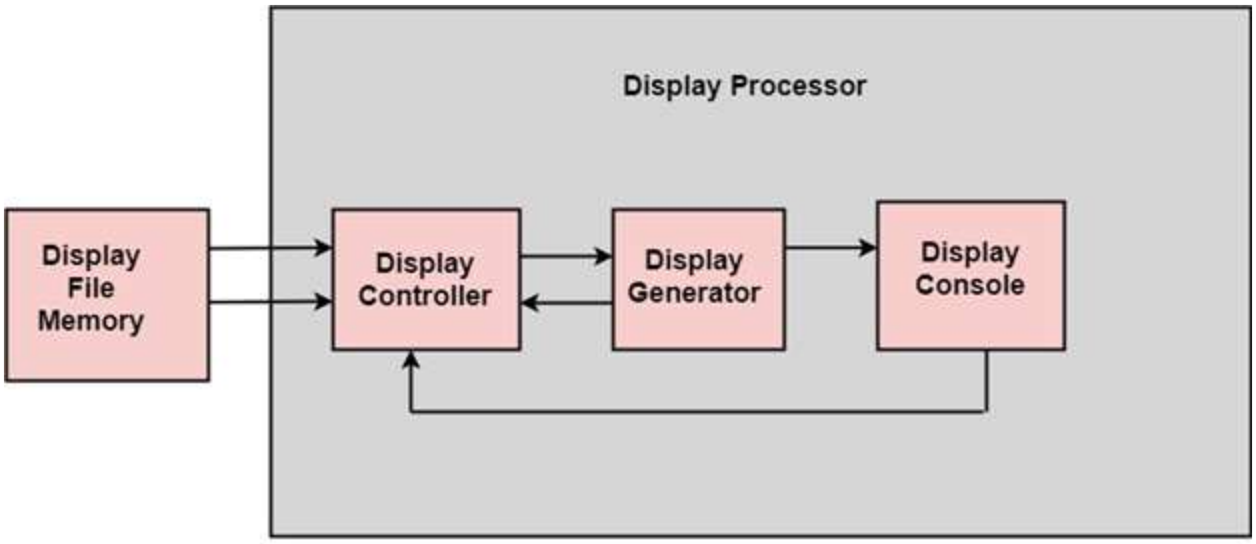
Aspect Ratio = $\frac{\text{width unit}}{\text{height unit}}$

Display Processor:

It is interpreter or piece of hardware that converts display processor code into pictures. It is one of the four main parts of the display processor

Parts of Display Processor

1. Display File Memory
2. Display Processor
3. Display Generator
4. Display Console



Block diagram of Display System

Display File Memory: It is used for generation of the picture. It is used for identification of graphic entities.

Display Controller:

1. It handles interrupt
2. It maintains timings
3. It is used for interpretation of instruction.

Display Generator:

1. It is used for the generation of character.
2. It is used for the generation of curves.

Display Console: It contains CRT, Light Pen, and Keyboard and deflection system.

The raster scan system is a combination of some processing units. 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.

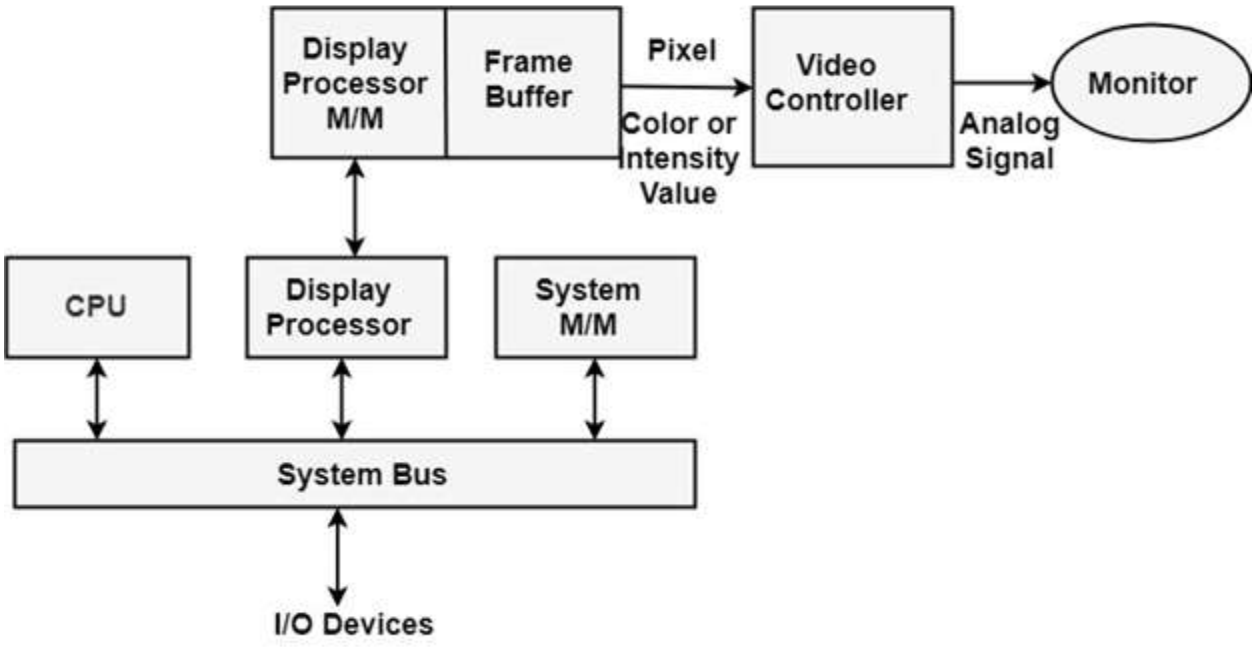


Fig: Architecture of a Raster Display System with a Display Processor

As fig showing that 2 registers (X register and Y register) are used to store the coordinate of the screen pixels. Assume that y values of the adjacent scan lines increased by 1 in an upward direction starting from 0 at the bottom of the screen to y_{max} at the top and along each scan line the screen pixel positions or x values are incremented by 1 from 0 at the leftmost position to x_{max} at the rightmost position.

The origin is at the lowest left corner of the screen as in a standard Cartesian coordinate system.

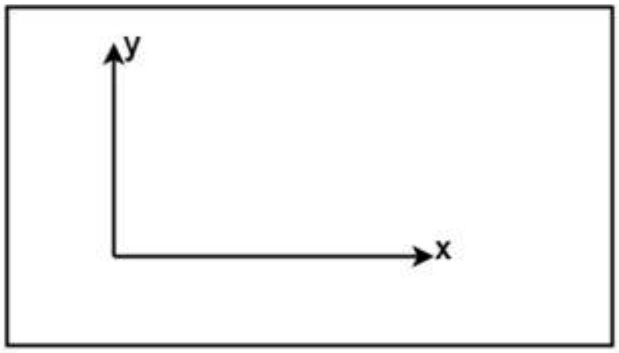


Fig: The origin of the coordinate system for identifying screen positions is usually specified in the lower-left corner.

At the start of a **Refresh Cycle**:

X register is set to 0 and y register is set to y_{max}. 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 color 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}.

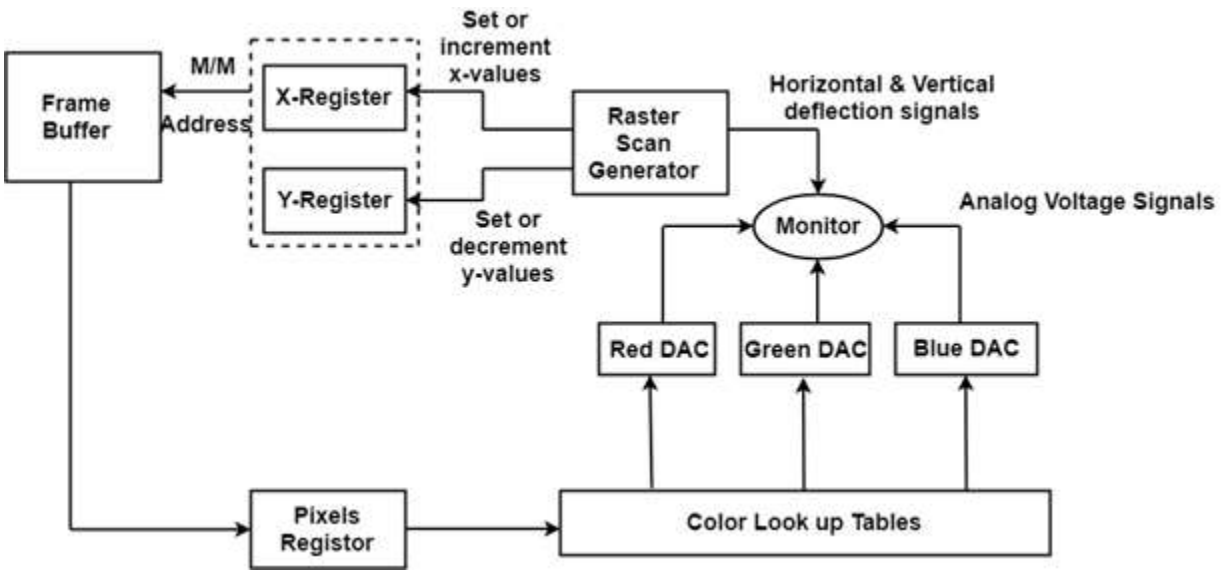
Then x register is reset to 0, and y register is decremented by 1 to access the next scan line.

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 color 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-color value from the look-up table. This lookup operation is done for each pixel on every display cycle.

As the time available to display or refresh a single pixel in the screen is too less, accessing the frame buffer every time for reading each pixel intensity value would consume more time what is allowed:



Multiple adjacent pixel values are fetched to the frame buffer in single access and stored in the register.

After every allowable time gap, the one-pixel value is shifted out from the register to control the warm intensity for that pixel.

The procedure is repeated with the next block of pixels,and so on, thus the whole group of pixels will be processed.

Display Devices:

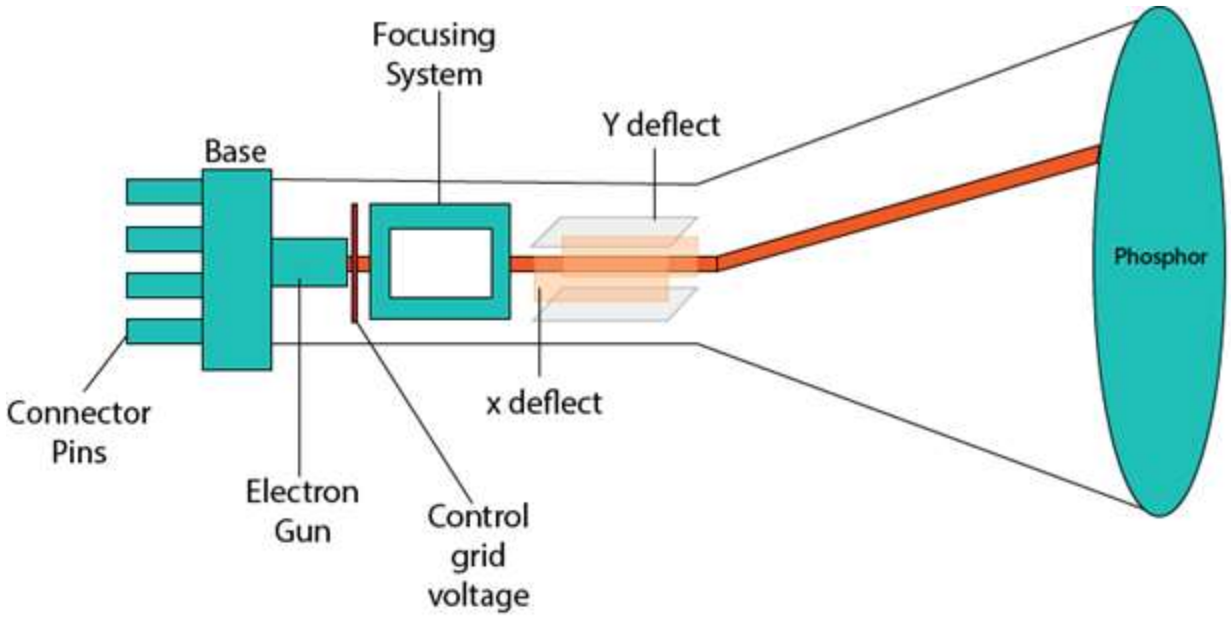
The most commonly used display device is a video monitor. The operation of most video monitors based on CRT (Cathode Ray Tube). The following display devices are used:

1. Refresh Cathode Ray Tube
2. Random Scan and Raster Scan
3. Color CRT Monitors
4. Direct View Storage Tubes
5. Flat Panel Display
6. Lookup Table

Cathode Ray Tube (CRT):

CRT stands for Cathode Ray Tube. 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 color you view on the screen is produced by a blend of red, blue and green light.



Cathode Ray Tube

Components of CRT:

Main Components of CRT are:

1. Electron Gun: Electron gun consisting of a series of elements, primarily a heating filament (heater) and a cathode. The electron gun creates a source of electrons which are focused into a narrow beam directed at the face of the CRT.

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

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

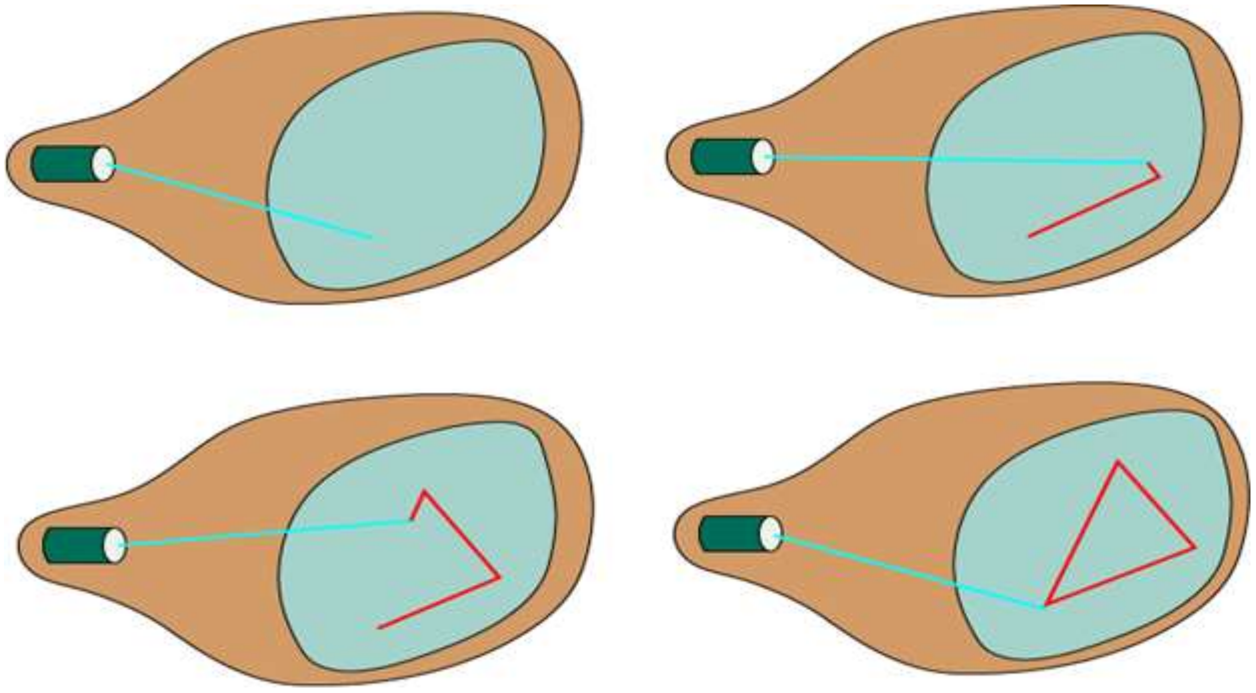
4. Deflection Yoke: It is used to control the direction of the electron beam. It creates an electric or magnetic field which will bend the electron beam as it passes through the area. In a conventional CRT, the yoke is linked to a sweep or scan generator. The deflection yoke which is connected to the sweep generator creates a fluctuating electric or magnetic potential.

5. Phosphorus-coated screen: The inside front surface of every CRT is coated with phosphors. Phosphors glow when a high-energy electron beam hits them. Phosphorescence is the term used to characterize the light given off by a phosphor after it has been exposed to an electron beam.

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. The picture is constructed out of a sequence of straight-line segments. 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. After drawing the picture. 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 fig:



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

Advantages:

1. A CRT has the electron beam directed only to the parts of the screen where an image is to be drawn.
2. Produce smooth line drawings.
3. High Resolution

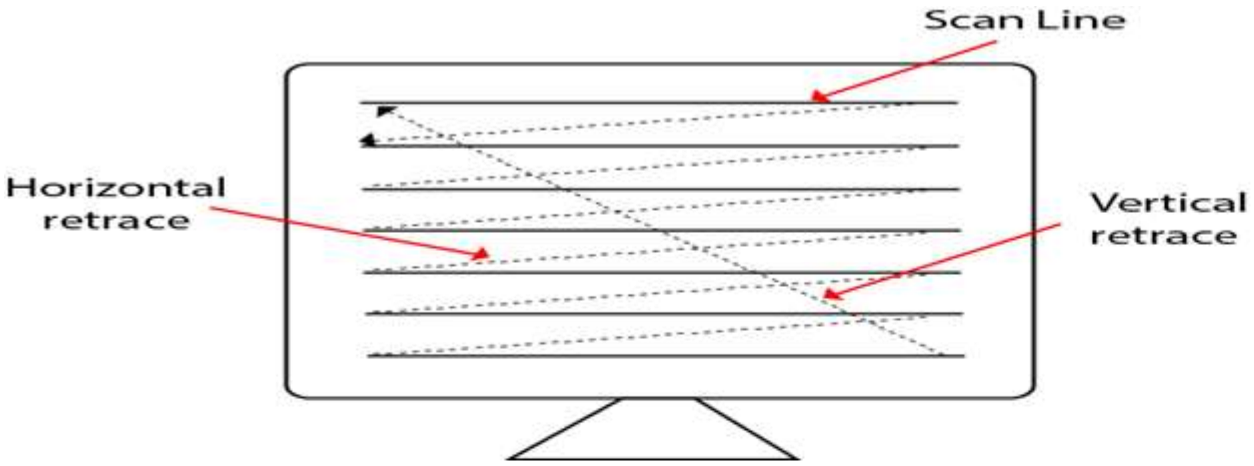
Disadvantages:

1. Random-Scan monitors cannot display realistic shades scenes.

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. Information of on and off pixels is stored in refresh buffer or Frame buffer. Televisions in our house are based on Raster Scan Method. 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. First is horizontal retracing and second is 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 fig:



Types of Scanning or travelling of beam in Raster Scan

1. Interlaced Scanning
2. Non-Interlaced Scanning

In Interlaced scanning, each horizontal line of the screen is traced from top to bottom. Due to which fading of display of object may occur. This problem can be solved by Non-Interlaced scanning. In this first of all odd numbered lines are traced or visited by an electron beam, then in the next circle, even number of lines are located.

For non-interlaced display refresh rate of 30 frames per second used. But it gives flickers. For interlaced display refresh rate of 60 frames per second is used.

Advantages:

1. Realistic image
2. Million Different colors to be generated
3. Shadow Scenes are possible.

Disadvantages:

1. Low Resolution
2. Expensive

Differentiate between Random and Raster Scan Display:

Random Scan	**Raster Scan**
1. It has high Resolution	1. Its resolution is low.
2. It is more expensive	2. It is less expensive
3. Any modification if needed is easy	3.Modification is tough
4. Solid pattern is tough to fill	4.Solid pattern is easy to fill
5. Refresh rate depends or resolution	5. Refresh rate does not depend on the picture.

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

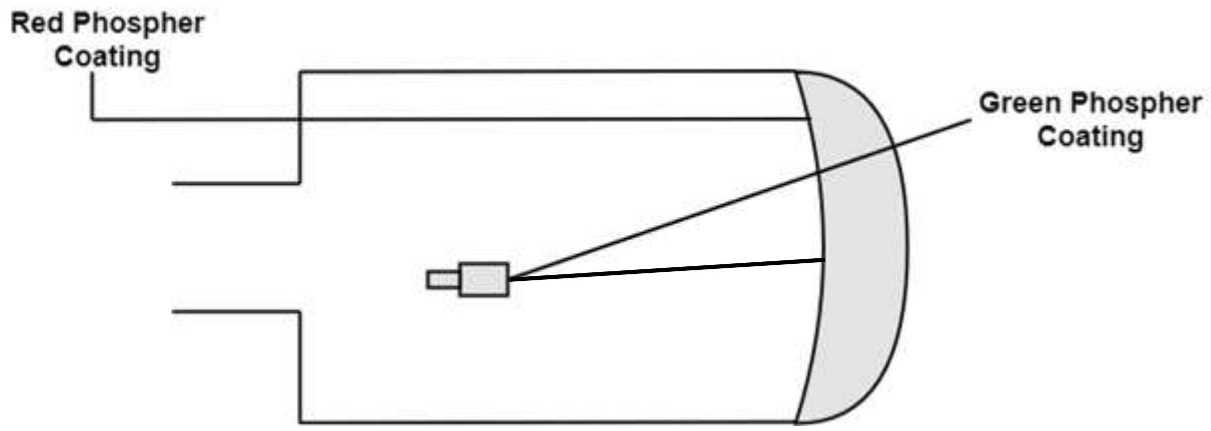
Color CRT Monitors:

The CRT Monitor display by using a combination of phosphors. The phosphors are different colors. There are two popular approaches for producing color displays with a CRT are:

1. Beam Penetration Method
2. Shadow-Mask Method

1. Beam Penetration Method:

The Beam-Penetration method has been used with random-scan monitors. In this method, the CRT screen is coated with two layers of phosphor, red and green and the displayed color 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 color.



Advantages:

1. Inexpensive

Disadvantages:

1. Only four colors are possible
2. Quality of pictures is not as good as with another method.

2. 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 color TV sets and monitors.

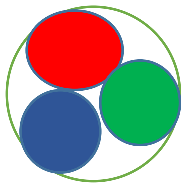
Construction: A shadow mask CRT has 3 phosphor color dots at each pixel position.

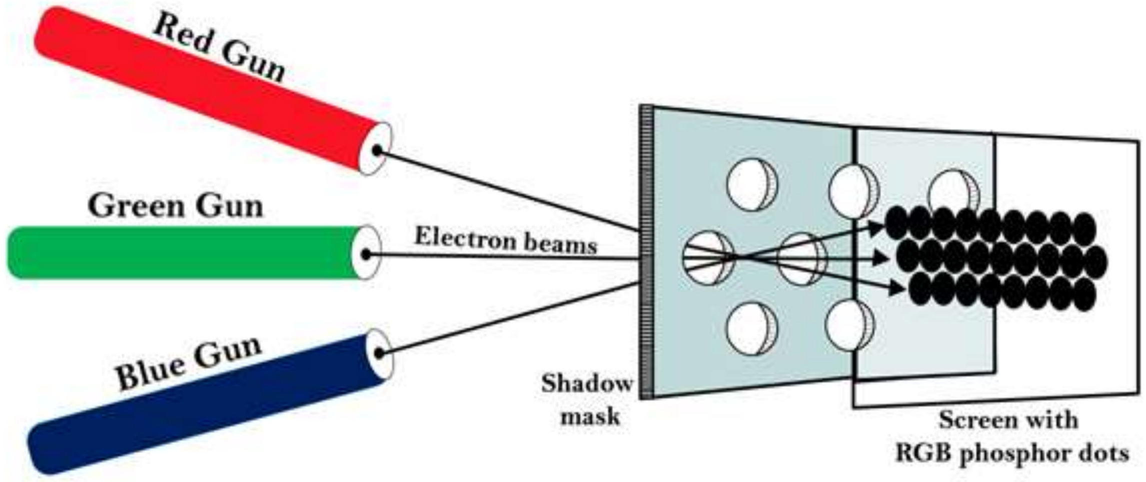
One phosphor dot emits:	red light
Another emits:	green light
Third emits:	blue light

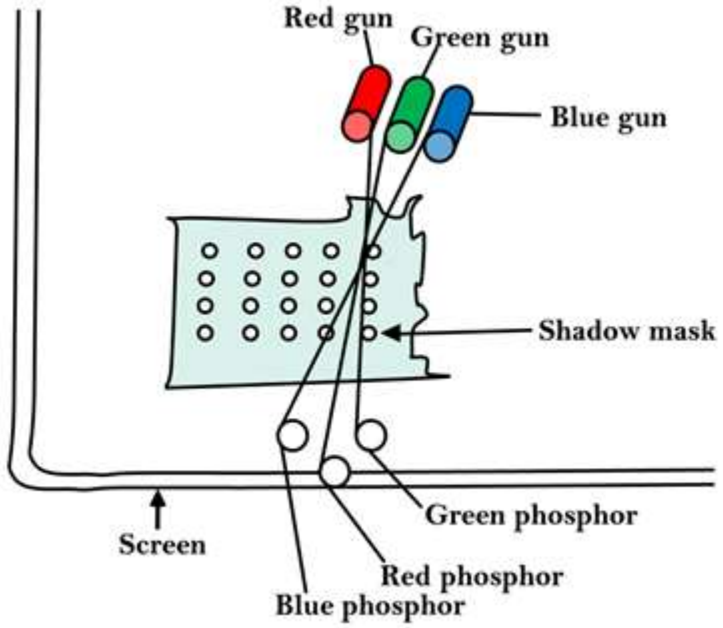
This type of CRT has 3 electron guns, one for each color dot and a shadow mask grid just behind the phosphor coated screen.

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

Figure shows the delta-delta shadow mask method commonly used in color CRT system.







The Shadow mask CRT

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

The deflection system of the CRT operates on all 3 electron beams simultaneously; the 3 electron beams are deflected and focused as a group onto the shadow mask, which contains a sequence of holes aligned with the phosphor- dot patterns.

When the three beams pass through a hole in the shadow mask, they activate a dotted triangle, which occurs as a small color spot on the screen.

The phosphor dots in the triangles are organized so that each electron beam can activate only its corresponding color 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 color dots on the screen, are aligned along one scan line rather of in a triangular pattern.

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

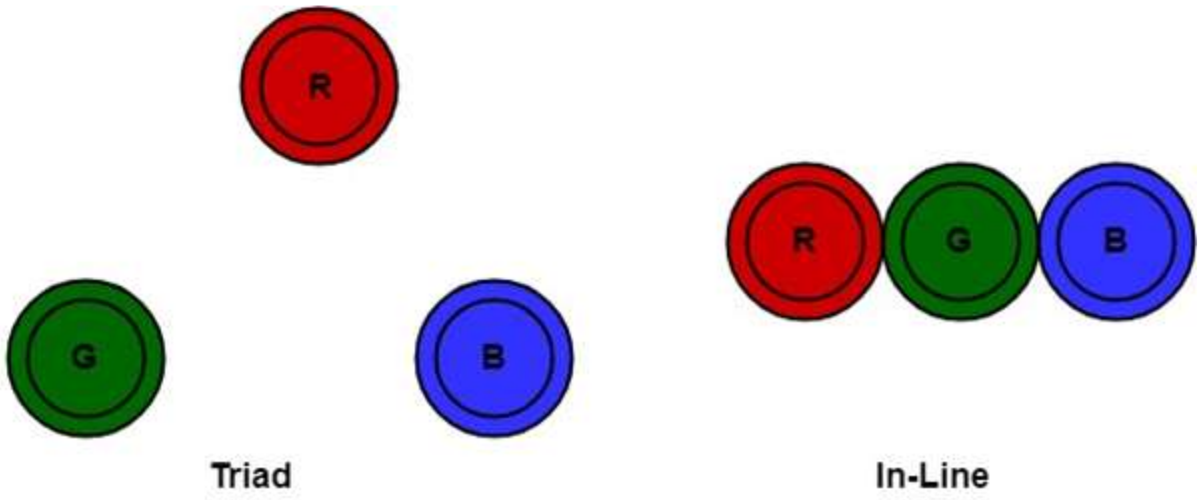


Fig: Triad-and -in-line arrangements of red, green and blue electron guns of CRT for color monitors.

Advantage:

1. Realistic image
2. Million different colors to be generated
3. Shadow scenes are possible

Disadvantage:

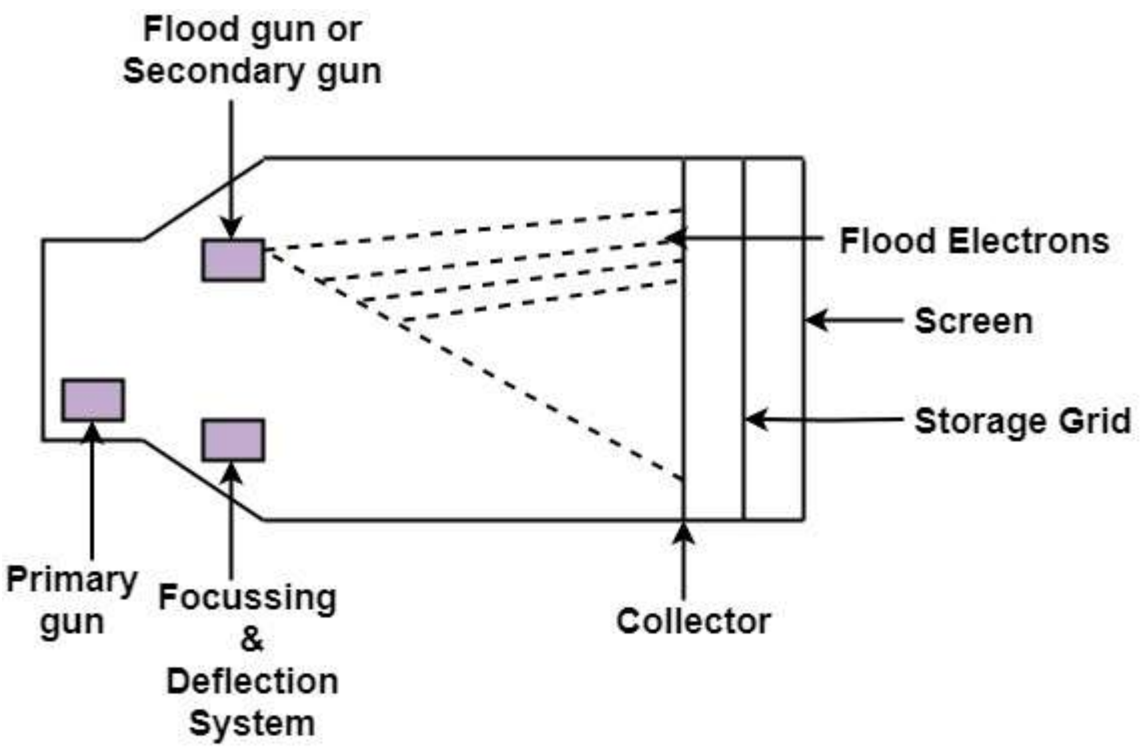
1. Relatively expensive compared with the monochrome CRT.
2. Relatively poor resolution
3. Convergence Problem

Direct View Storage Tubes:

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

1. **Primary guns:** It is used to store the picture pattern.
2. **Flood gun or Secondary gun:** It is used to maintain picture display.



Direct View Storage Tube

Advantage:

1. No refreshing is needed.
2. High Resolution
3. Cost is very less

Disadvantage:

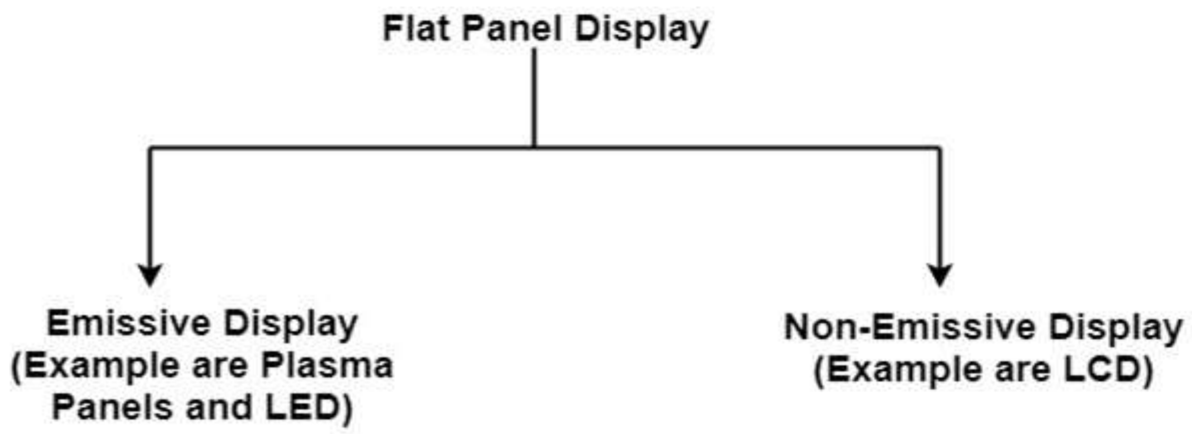
1. It is not possible to erase the selected part of a picture.

2. It is not suitable for dynamic graphics applications.
3. If a part of picture is to modify, then time is consumed.

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
Emissive Display (Example are Plasma Panels and LED)
Non-Emissive Display (Example are LCD)

1. 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).

2. 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).

Plasma Panel Display:

Plasma-Panels are also called as Gas-Discharge Display. It consists of an array of small lights. Lights are fluorescent in nature. The essential components of the plasma-panel display are:

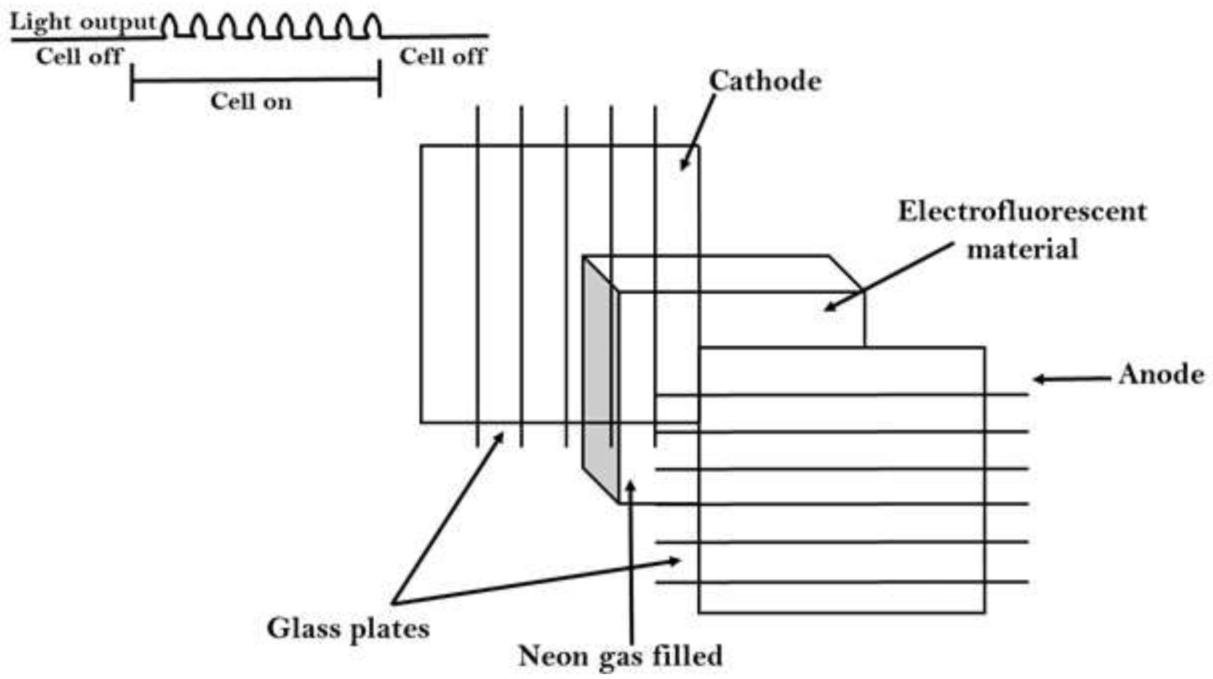
1. **Cathode:** It consists of fine wires. It delivers negative voltage to gas cells. The voltage is released along with the negative axis.
2. **Anode:** It also consists of line wires. It delivers positive voltage. The voltage is supplied along positive axis.

3. **Fluorescent cells:** It consists of small pockets of gas liquids when the voltage is applied to this liquid (neon gas) it emits light.
4. **Glass Plates:** These plates act as capacitors. The voltage will be applied, the cell will glow continuously.

The gas will slow 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 shows the state of cell in plasma panel display:



Advantage:

1. High Resolution
2. Large screen size is also possible.
3. Less Volume
4. Less weight
5. Flicker Free Display

Disadvantage:

1. Poor Resolution
2. Wiring requirement anode and the cathode is complex.
3. Its addressing is also complex.

LED (Light Emitting Diode):

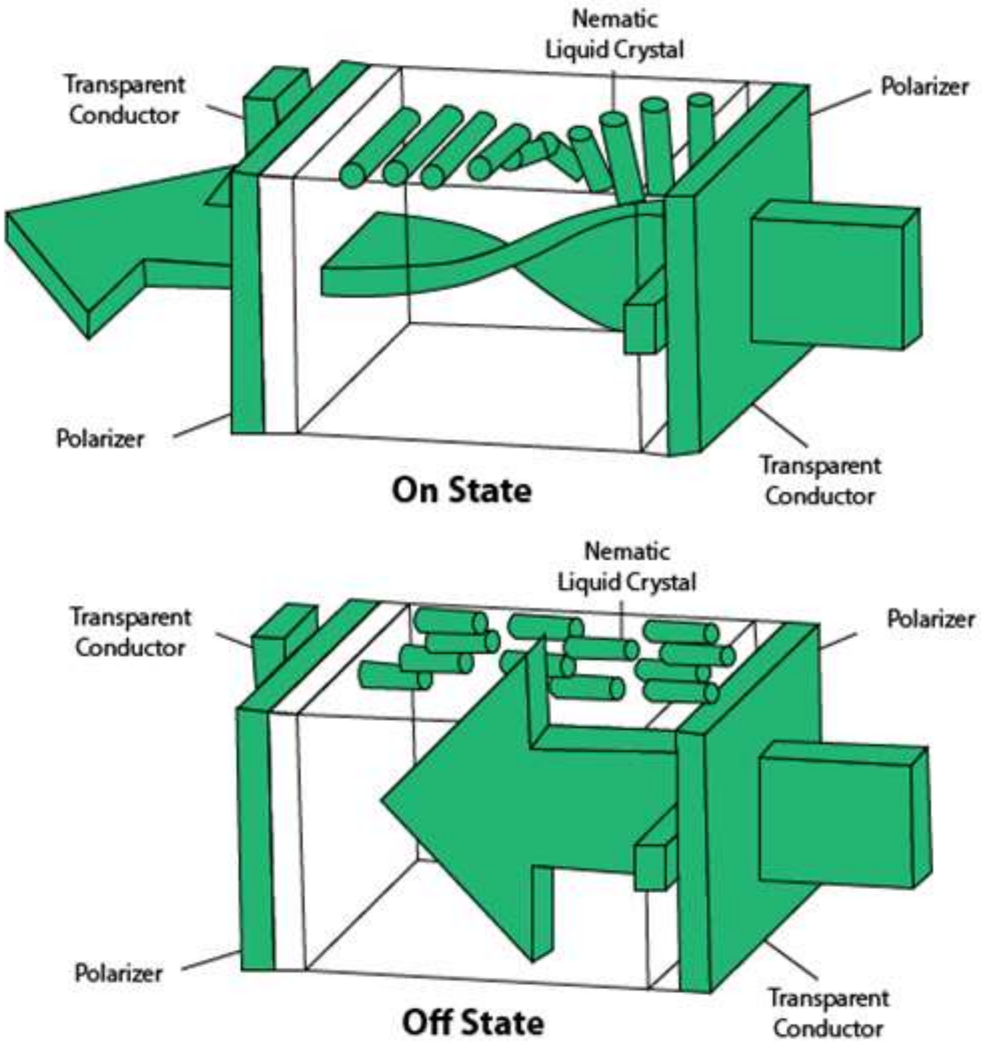
In an LED, a matrix of diodes is organized to form the pixel positions in the display and picture definition is stored in a refresh buffer. Data is read from the refresh buffer and converted to voltage levels that are applied to the diodes to produce the light pattern in the display.

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.

Liquid crystal display is temperature dependent. It is between zero to seventy degree Celsius. It is flat and requires very little power to operate.



Liquid Crystal Display

Advantage:

1. Low power consumption.
2. Small Size
3. Low Cost

Disadvantage:

1. LCDs are temperature-dependent (0-70°C)

2. LCDs do not emit light; as a result, the image has very little contrast.
3. LCDs have no color capability.
4. The resolution is not as good as that of a CRT.

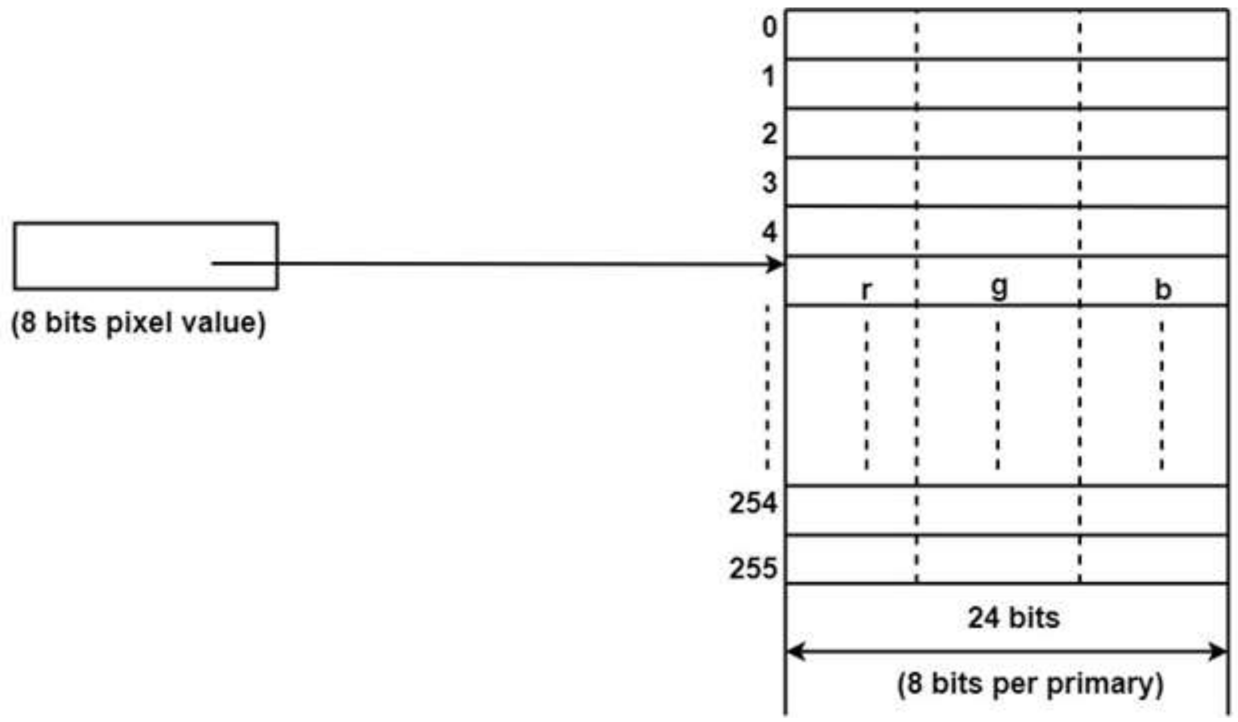
Look-Up Table:

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 colors.

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

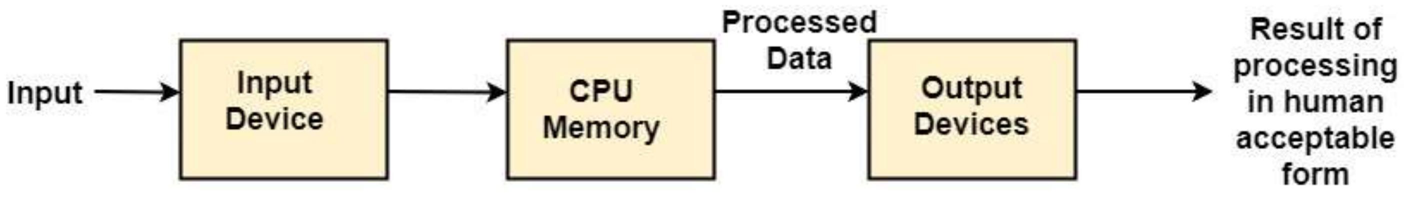
A widely accepted industry standard uses 3 bytes, or 24 bytes, per pixel, with one byte for each primary color. The way, we allow each primary color to have 256 different intensity levels. Thus a pixel can take on a color from 256 x 256 x 256 or 16.7 million possible choices. The 24-bit format is commonly referred to as the actual color representation.

Lookup Table approach reduces the storage requirement. In this approach pixel values do not code colors directly. Alternatively, they are addresses or indices into a table of color values. The color of a particular pixel is determined by the color value in the table entry that the value of the pixel references. Figure shows a look-up table with 256 entries. The entries have addresses 0 through 255. Each entry contains a 24-bit RGB color value. Pixel values are now 1-byte. The color of a pixel whose value is i, where 0 <i<255, is persistence by the color value in the table entry whose address is i. It reduces the storage requirement of a 1000 x 1000 image to one million bytes plus 768 bytes for the color values in the look-up table.



Input Devices

The Input Devices are the hardware that is used to transfer transfers input to the computer. The data can be in the form of text, graphics, sound, and text. Output device display data from the memory of the computer. Output can be text, numeric data, line, polygon, and other objects.



These Devices include:

1. Keyboard
2. Mouse
3. Trackball
4. Spaceball
5. Joystick
6. Light Pen
7. Digitizer
8. Touch Panels
9. Voice Recognition
10. Image Scanner

Keyboard:

The most commonly used input device is a keyboard. The data is entered by pressing the set of keys. All keys are labeled. A keyboard with 101 keys is called a QWERTY keyboard.

The keyboard has alphabetic as well as numeric keys. Some special keys are also available.

1. **Numeric Keys:** 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
2. **Alphabetic keys:** a to z (lower case), A to Z (upper case)
3. **Special Control keys:** Ctrl, Shift, Alt

4. **Special Symbol Keys:** ; , " ? @ ~ ? :
5. **Cursor Control Keys:** ↑ → ← ↓
6. **Function Keys:** F1 F2 F3....F9.
7. **Numeric Keyboard:** It is on the right-hand side of the keyboard and used for fast entry of numeric data.

Function of Keyboard:

1. Alphanumeric Keyboards are used in CAD. (Computer Aided Drafting)
2. Keyboards are available with special features line screen co-ordinates entry, Menu selection or graphics functions, etc.
3. Special purpose keyboards are available having buttons, dials, and switches. Dials are used to enter scalar values. Dials also enter real numbers. Buttons and switches are used to enter predefined function values.

Advantage:

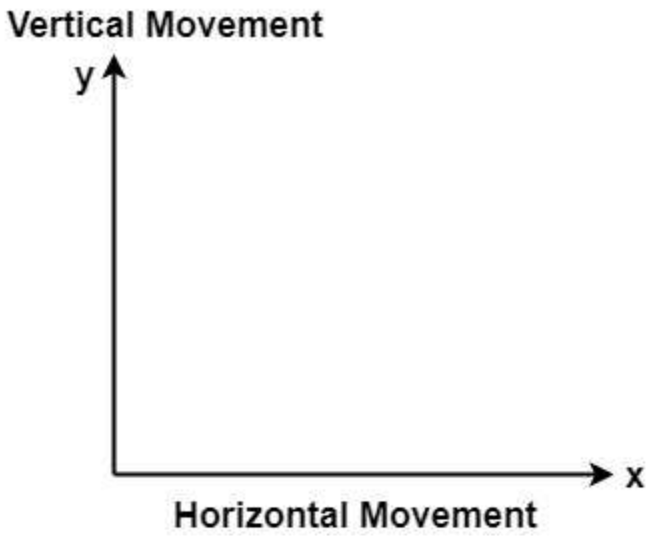
1. Suitable for entering numeric data.
2. Function keys are a fast and effective method of using commands, with fewer errors.

Disadvantage:

1. Keyboard is not suitable for graphics input.

Mouse:

A Mouse is a pointing device and used to position the pointer on the screen. It is a small palm size box. There are two or three depression switches on the top. The movement of the mouse along the x-axis helps in the horizontal movement of the cursor and the movement along the y-axis helps in the vertical movement of the cursor on the screen. The mouse cannot be used to enter text. Therefore, they are used in conjunction with a keyboard.

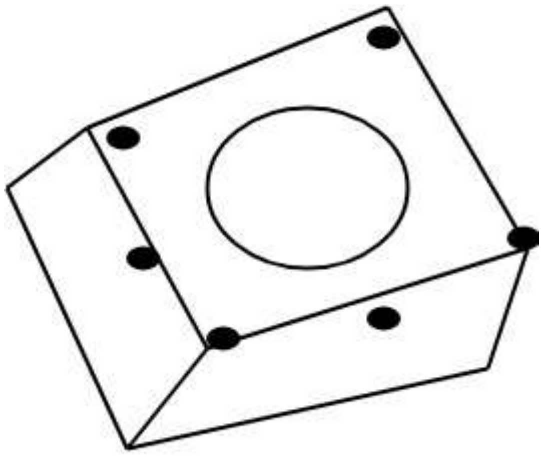


Advantage:

1. Easy to use
2. Not very expensive

Trackball

It is a pointing device. It is similar to a mouse. This is mainly used in notebook or laptop computer, instead of a mouse. This is a ball which is half inserted, and by changing fingers on the ball, the pointer can be moved.



TrackBall

Advantage:

1. Trackball is stationary, so it does not require much space to use it.
2. Compact Size

Spaceball:

It is similar to trackball, but it can move in six directions where trackball can move in two directions only. The movement is recorded by the strain gauge. Strain gauge is applied with pressure. It can be pushed and pulled in various directions. The ball has a diameter around 7.5 cm. The ball is mounted in the base using rollers. One-third of the ball is an inside box, the rest is outside.

Applications:

1. It is used for three-dimensional positioning of the object.
2. It is used to select various functions in the field of virtual reality.
3. It is applicable in CAD applications.
4. Animation is also done using spaceball.
5. It is used in the area of simulation and modeling.

Joystick:

A Joystick is also a pointing device which is used to change cursor position on a monitor screen. Joystick is a stick having a spherical ball as its both lower and upper ends as shown in fig. The lower spherical ball moves in a socket. The joystick can be changed in all four directions. The function of a joystick is similar to that of the mouse. It is mainly used in Computer Aided Designing (CAD) and playing computer games.

Voice Systems (Voice Recognition):

Voice Recognition is one of the newest, most complex input techniques used to interact with the computer. The user inputs data by speaking into a microphone. The simplest form of voice recognition is a one-word command spoken by one person. Each command is isolated with pauses between the words.

Voice Recognition is used in some graphics workstations as input devices to accept voice commands. The voice-system input can be used to initiate graphics operations or to enter data. These systems operate by matching an input against a predefined dictionary of words and phrases.

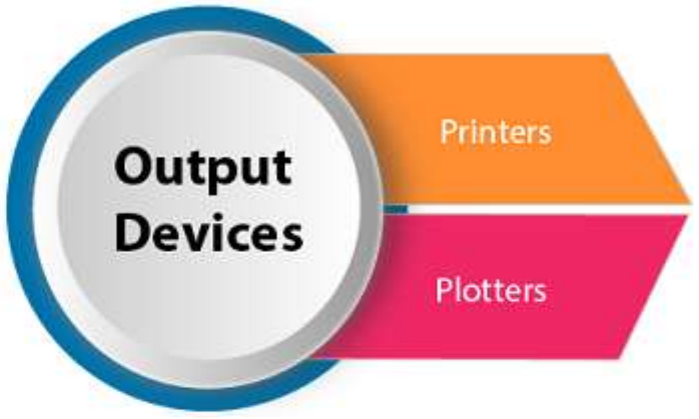
Advantage:

1. More efficient device.
2. Easy to use
3. Unauthorized speakers can be identified

Disadvantages:

1. Very limited vocabulary
2. Voice of different operators can't be distinguished.

Output Devices



It is an electromechanical device, which accepts data from a computer and translates them into form understand by users.

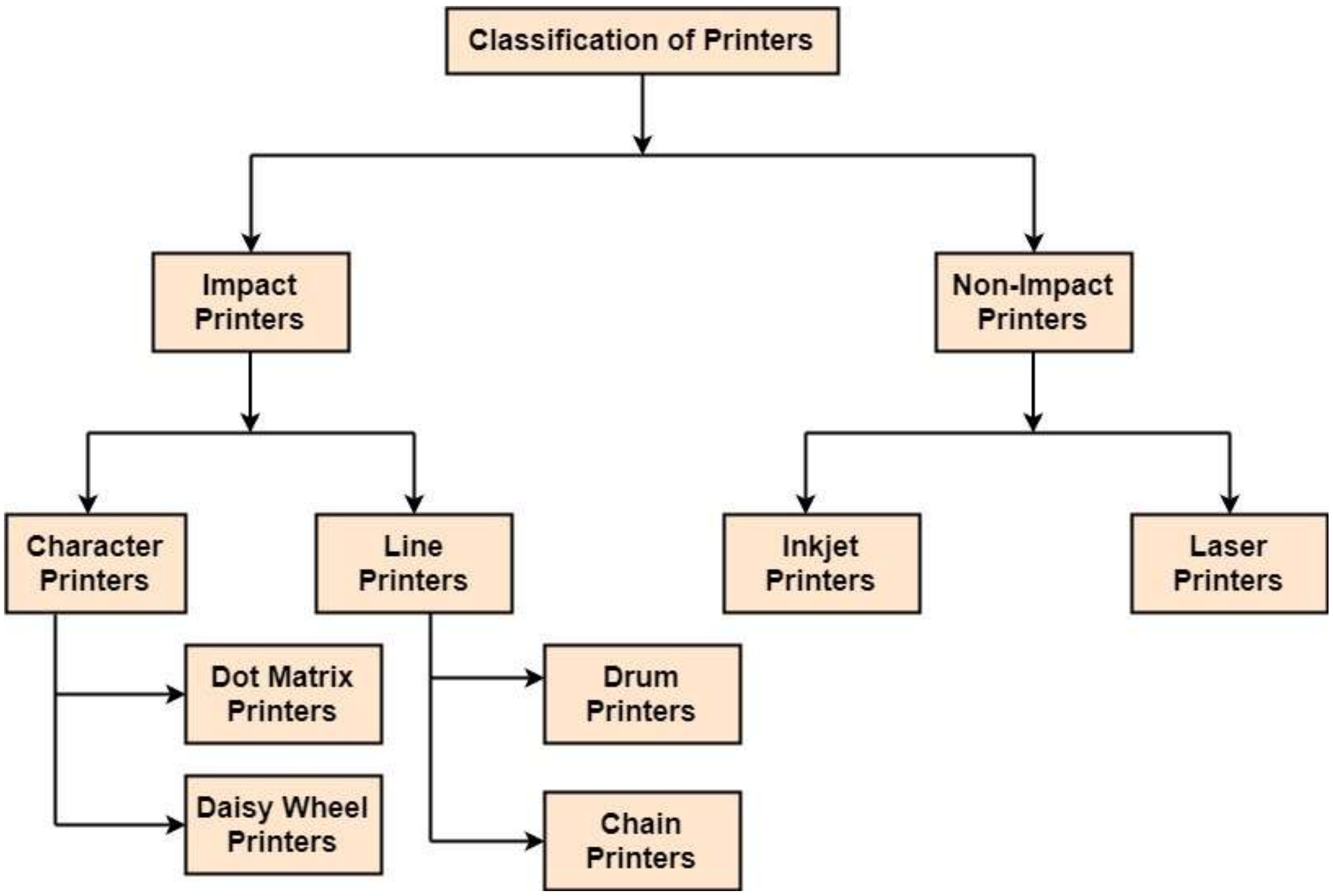
Following are Output Devices:

1. Printers
2. Plotters

Printers:

Printer is the most important output device, which is used to print data on paper.

Types of Printers: There are many types of printers which are classified on various criteria as shown in fig:



1. Impact Printers: The printers that print the characters by striking against the ribbon and onto the papers are known as Impact Printers.

These Printers are of two types:

1. Character Printers
2. Line Printers

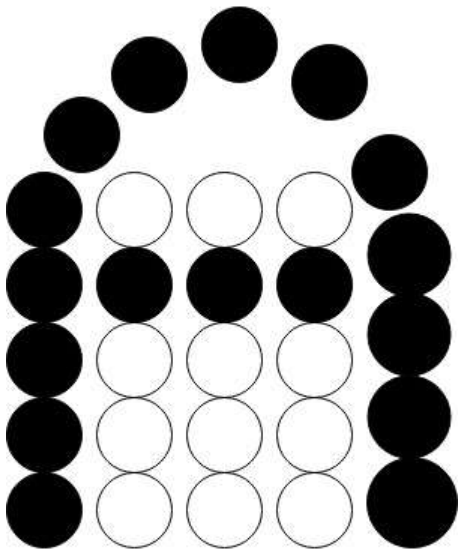
2. Non-Impact Printers: The printers that print the characters without striking against the ribbon and onto the papers are called Non-Impact Printers. These printers print a complete page at a time, therefore, also known as Page Printers.

Page Printers are of two types:

1. Laser Printers
2. Inkjet Printers

Dot Matrix Printers:

Dot matrix has printed in the form of dots. A printer has a head which contains nine pins. The nine pins are arranged one below other. Each pin can be activated independently. All or only the same needles are activated at a time. When needless is not activated, and then the tip of needle stay in the head. When pin work, it comes out of the print head. In nine pin printer, pins are arranged in 5 * 7 matrixes.



Dot Matrix Printer

Advantage:

1. Dot Matrix Printers prints output as dots, so it can print any shape of the character. This allows the printer to print special character, charts, graphs, etc.
2. Dot Matrix Printers come under the category of impact printers. The printing is done when the hammer pin strikes the inked ribbon. The impressions are printed on paper. By placing multiple copies of carbon, multiple copies of output can be produced.
3. It is suitable for printing of invoices of companies.

Daisy Wheel Printers:

Head is lying on a wheel and Pins corresponding to characters are like petals of Daisy, that's why called Daisy wheel printer.

Daisy Wheel Printer

Advantage:

1. More reliable than DMPs
2. Better Quality

Disadvantage:

1. Slower than DMPs

Drum Printers:

These are line printers, which prints one line at a time. It consists of a drum. The shape of the drum is cylindrical. The drum is solid and has characters embossed on it in the form of vertical bands. The characters are in circular form. Each band consists of some characters. Each line on drum consists of 132 characters. Because there are 96 lines so total characters are (132 * 95) = 12, 672.

Drum contains a number of hammers also.

Chain Printers:

These are called as line printers. These are used to print one line at a line. Basically, chain consists of links. Each link contains one character. Printers can follow any character set style, i.e., 48, 64 or 96 characters. Printer consists of a number of hammers also.

Advantages:

1. Chain or Band if damaged can be changed easily.
2. It allows printing of different form.
3. Different Scripts can be printed using this printer.

Disadvantages:

1. It cannot print charts and graphs.
2. It cannot print characters of any shape.
3. Chain Printers is impact printer, hammer strikes so it is noisy.

Non-Impact Printers:

Inkjet Printers:

These printers use a special link called electrostatic ink. The printer head has a special nozzle. Nozzle drops ink on paper. Head contains up to 64 nozzles. The ink dropped is deflected by the electrostatic plate. The plate is fixed outside the nozzle. The deflected ink settles on paper.

Inkjet Printer

Advantages:

1. These produce high quality of output as compared to the dot matrix.
2. A high-quality output can be produced using 64 nozzles printed.
3. Inkjet can print characters in a variety of shapes.
4. Inkjet can print special characters.
5. The printer can print graphs and charts.

Disadvantages:

1. Inkjet Printers are slower than dot matrix printers.
2. The cost of inkjet is more than a dot matrix printer.

Laser Printers:

These are non-impact page printers. They use laser lights to produces the dots needed to form the characters to be printed on a page & hence the name laser printers.

The output is generated in the following steps:

Step1: The bits of data sent by processing unit act as triggers to turn the laser beam on & off.

Step2: The output device has a drum which is cleared & is given a positive electric charge. To print a page the modulated laser beam passing from the laser scans back & forth the surface of the drum. The positive electric charge on the drum is stored on just those parts of the drum surface which are exposed to the laser beam create the difference in electric which charges on the exposed drum surface.

Laser Printer

Step3: The laser exposed parts of the drum attract an ink powder known as toner.

Step4: The attracted ink powder is transferred to paper.

Step5: The ink particles are permanently fixed to the paper by using either heat or pressure technique.

Step6: The drum rotates back to the cleaner where a rubber blade cleans off the excess ink & prepares the drum to print the next page.

Plotters

Plotters are a special type of output device. It is suitable for applications:

1. Architectural plan of the building.
2. CAD applications like the design of mechanical components of aircraft.
3. Many engineering applications.

Advantage:

1. It can produce high-quality output on large sheets.
2. It is used to provide the high precision drawing.
3. It can produce graphics of various sizes.
4. The speed of producing output is high.

Drum Plotter:

It consists of a drum. Paper on which design is made is kept on the drum. The drum can rotate in both directions. Plotters comprised of one or more pen and penholders. The holders are mounted perpendicular to drum surface. The pens are kept in the holder, which can move left to the right as well as right to the left. The graph plotting program controls the movement of pen and drum.

Drum Plotter

Flatbed Plotter:

It is used to draw complex design and graphs, charts. The Flatbed plotter can be kept over the table. The plotter consists of pen and holder. The pen can draw characters of various sizes. There can be one or more pens and pen holding mechanism. Each pen has ink of different color. Different colors help to produce multicolor design of document. The area of plotting is also variable. It can vary A4 to 21'*52'.

Flatbed Plotter

It is used to draw

1. Cars
2. Ships
3. Airplanes
4. Shoe and dress designing
5. Road and highway design

Graphics Software:

There are two types of Graphics Software.

1. General Purpose Packages: Basic Functions in a general package include those for generating picture components (straight lines, polygons, circles and other figures), setting color and intensity values, selecting views, and applying transformations.

Example of general purpose package is the GL (Graphics Library), GKS, PHIGS, PHIGS+ etc.

2. Special Purpose Packages: These packages are designed for non programmers, so that these users can use the graphics packages, without knowing the inner details.

Example of special purpose package is

1. Painting programs
2. Package used for business purpose
3. Package used for medical systems.
4. CAD packages

Scan Conversion Definition

It is a process of representing graphics objects a collection of pixels. The graphics objects are continuous. The pixels used are discrete. 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. Using this ability graphics computer represent picture having discrete dots.

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

1. Algorithms can generate graphics objects at a faster rate.
2. Using algorithms memory can be used efficiently.
3. Algorithms can develop a higher level of graphical objects.

Examples of objects which can be scan converted

1. Point
2. Line
3. Sector
4. Arc

5. Ellipse
6. Rectangle
7. Polygon
8. Characters
9. Filled Regions

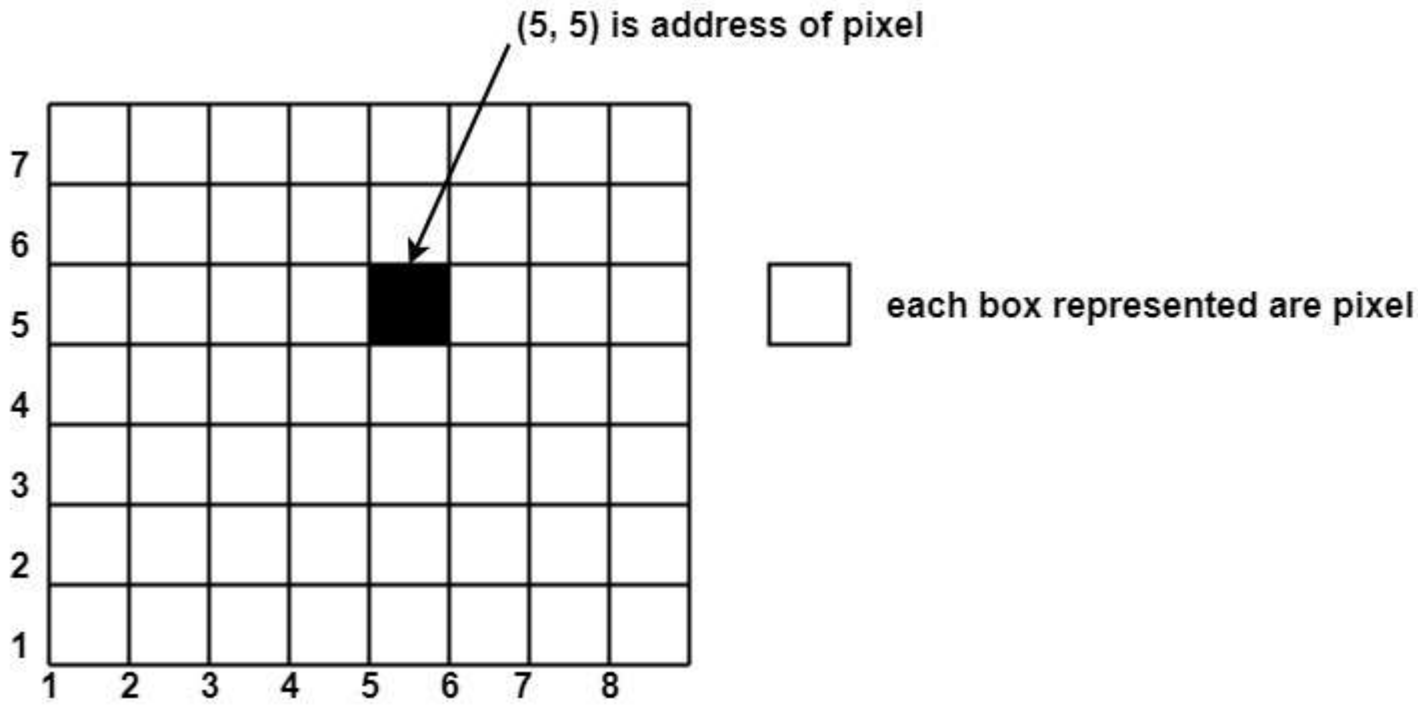
The process of converting is also called as rasterization. The algorithms implementation varies from one computer system to another computer system. Some algorithms are implemented using the software. Some are performed using hardware or firmware. Some are performed using various combinations of hardware, firmware, and software.

Pixel or Pel:

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. Pixels are generated using commands. Lines, circle, arcs, characters; curves are drawn with closely spaced pixels. To display the digit or letter matrix of pixels is used.

The closer the dots or pixels are, the better will be the quality of picture. Closer the dots are, 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 fig:



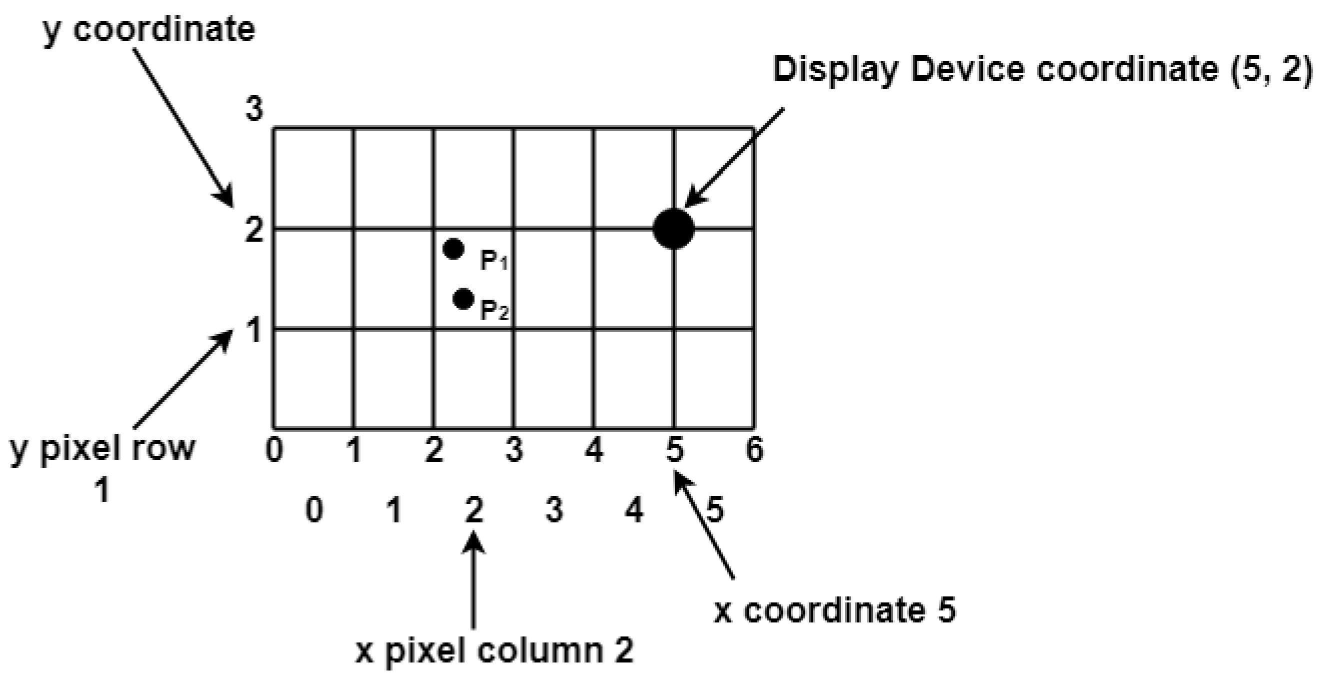
Different graphics objects can be generated by setting the different intensity of pixels and different colors of pixels. Each pixel has some co-ordinate value. The coordinate is represented using row and column.

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. This memory is a storage area for storing pixels values using which pictures are displayed. It is also called as 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

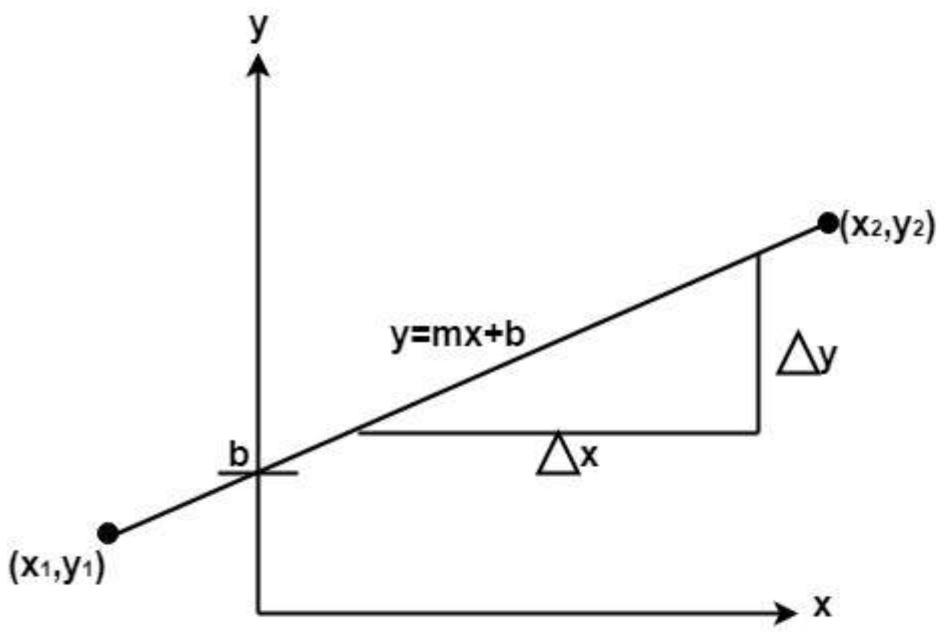
Each pixel on the graphics display does not represent a mathematical point. Instead, it means a region which theoretically can contain an infinite number of points. Scan-Converting a point involves illuminating the pixel that contains the point.

Example: Display coordinates points $P_1\ (2\frac{1}{4}, 1\frac{3}{4})$& $P_2(2\frac{2}{3}, 1\frac{1}{4})$as shown in fig 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).



Scan Converting a Straight Line

A straight line may be defined by two endpoints & an equation. In fig 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.



Using the equation of a straight line, y = mx + b where m = $\frac{\Delta y}{\Delta x}$ & 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.

Properties of Good Line Drawing Algorithm:

1. Line should appear Straight: We must appropriate the line by choosing addressable points close to it. If we choose well, the line will appear straight, if not, we shall produce crossed lines.

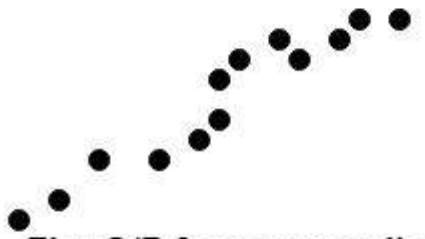


Fig: O/P from a poor line generating algorithm

The lines must be generated parallel or at 45° to the x and y-axes. Other lines cause a problem: a line segment through it starts and finishes at addressable points, may happen to pass through no another addressable points in between.

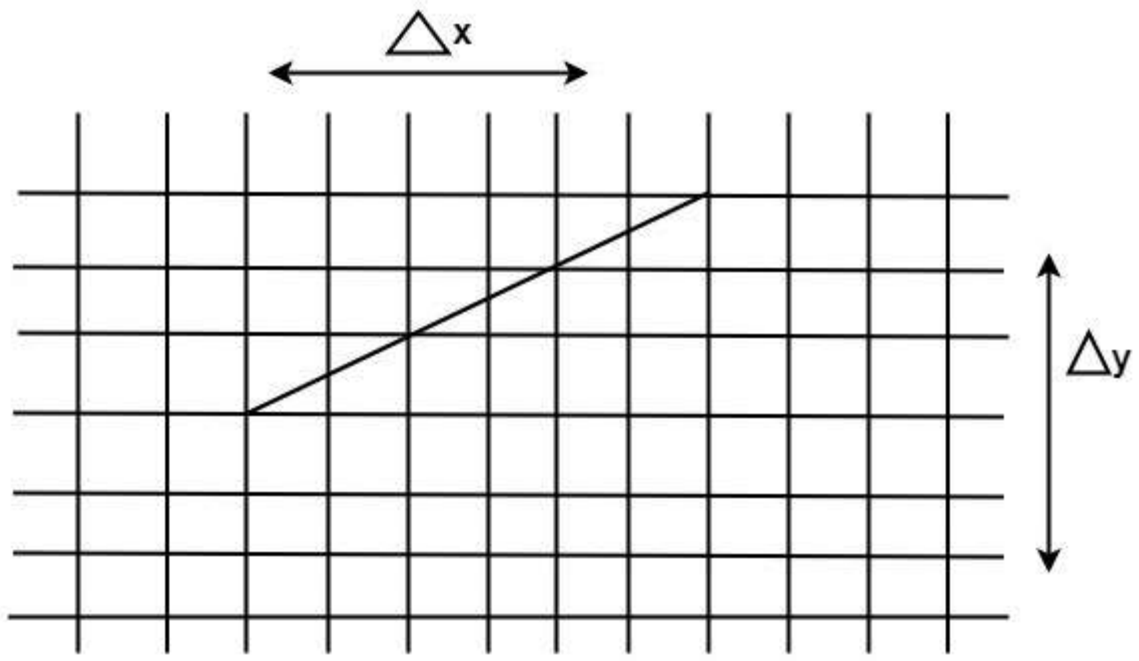


Fig: A straight line segment connecting 2 grid intersection may fail to pass through any other grid intersections.

2. Lines should terminate accurately: Unless lines are plotted accurately, they may terminate at the wrong place.

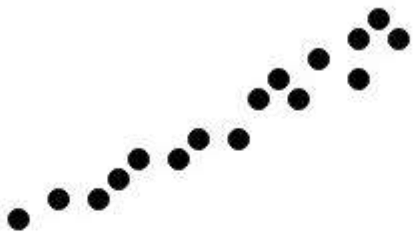


Fig: Uneven line density caused by bunching of dots.

3. Lines should have constant density: Line density is proportional to the no. of dots displayed divided by the length of the line.

To maintain constant density, dots should be equally spaced.

4. Line density should be independent of line length and angle: This can be done by computing an approximating line-length estimate and to use a line-generation algorithm that keeps line density constant to within the accuracy of this estimate.

5. Line should be drawn rapidly: This computation should be performed by special-purpose hardware.

Algorithm for line Drawing:

1. Direct use of line equation
2. DDA (Digital Differential Analyzer)
3. Bresenham's Algorithm

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-ordinates (x_1',y_1') and $(x_2'\ y_2')$.

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: P_1 (0,0) P_7 (6,18)

$x_1=0$
$y_1=0$
$x_2=6$
$y_2=18$

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

We know equation of line is
$y = m x + b$
$y = 3x + b$..............equation (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)

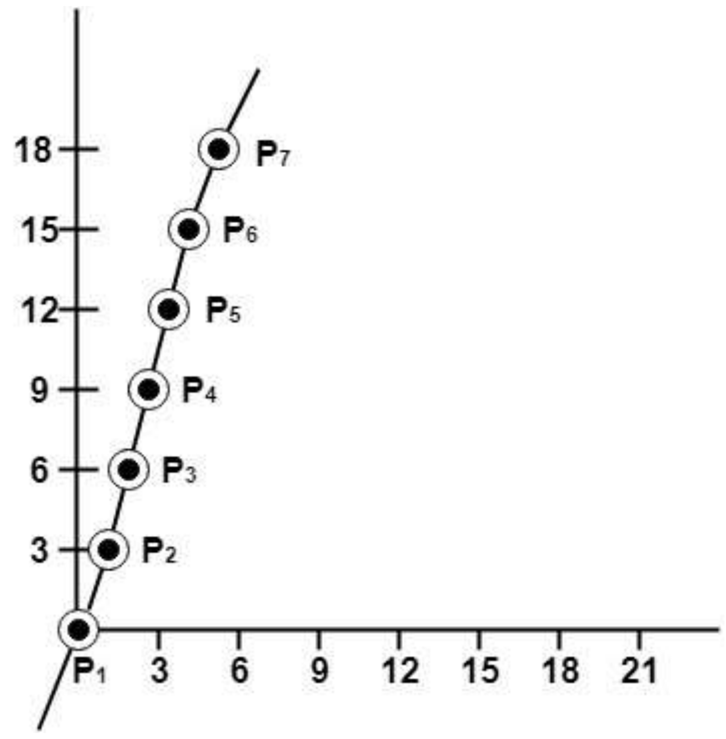
$y = 3x + 0$

$y = 3x$

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 P_1 (0,0)
P_2 (1,3)
P_3 (2,6)
P_4 (3,9)
P_5 (4,12)
P_6 (5,15)
P_7 (6,18)



Algorithm for drawing line using equation:

Step1: Start Algorithm

Step2: Declare variables $x_1, x_2, y_1, y_2, dx, dy, m, b,$

Step3: 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.

Step4: Calculate $dx = x_2 - x_1$

Step5: Calculate $dy = y_2 - y_1$

Step6: Calculate $m = \frac{dy}{dx}$

Step7: Calculate $b = y_1 - m * x_1$

Step8: Set (x, y) equal to starting point, i.e., lowest point and x_{end}equal to largest value of x.

```
    If $dx < 0$
        then $x = x_2$
    $y = y_2$
            $x_{end} = x_1$
If $dx > 0$
    then $x = x_1$
        $y = y_1$
            $x_{end} = x_2$
```

Step9: Check whether the complete line has been drawn if $x=x_{end}$, stop

Step10: Plot a point at current (x, y) coordinates

Step11: Increment value of x, i.e., $x = x+1$

Step12: Compute next value of y from equation $y = mx + b$

Step13: Go to Step9.

Program to draw a line using LineSlope Method

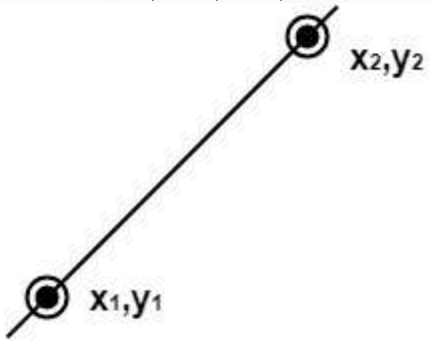
```
#include <graphics.h>
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include <conio.h>
#include <iostream.h>

```

```
class bresen
{
   float x, y, x1, y1, x2, y2, dx, dy, m, c, xend;
   public:
   void get ();
   void cal ();
};
   void main ()
   {
   bresen b;
   b.get ();
   b.cal ();
   getch ();
  }
   Void bresen :: get ()
  {
   print ("Enter start & end points");
   print ("enter x1, y1, x2, y2");
   scanf ("%f%f%f%f",sx1, sx2, sx3, sx4)
}
void bresen ::cal ()
{
   /* request auto detection */
   int gdriver = DETECT,gmode, errorcode;
   /* initialize graphics and local variables */
   initgraph (&gdriver, &gmode, " ");
   /* read result of initialization */
   errorcode = graphresult ();
   if (errorcode ! = grOK)    /*an error occurred */
   {
      printf("Graphics error: %s \n", grapherrormsg (errorcode);
      printf ("Press any key to halt:");
      getch ();
      exit (1); /* terminate with an error code */
```

```
    }
    dx = x2-x1;
    dy=y2-2y1;
    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)
    {
        putpixel (x, y, RED);
        y++;
        y=(x*x) +c;
    }
}
```

OUTPUT:

```
Enter Starting and End Points
Enter (X1, Y1, X2, Y2) 200 100 300 200
```



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 but by using results of previous steps.

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

The line of equation for step i

$y_i=mx_{i+b}$......................equation 1

Next value will be

$y_{i+1}=m_{xi+1}+b$.................equation 2

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

$y_{i+1}-y_i=\Delta y$.......................equation 3

$y_{i+1}-x_i=\Delta x$......................equation 4

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

$\Delta y=m\Delta x$

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

$\Delta x=\Delta y/m$

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

$x_{i+1}=x_i+\Delta y/m$

Case1: When |M|<1 then (assume that $x_1<x_2$)

x= x1,y=y1 set Δx=1

yi+1=y1+m,   x=x+1

Until x = x2</x

Case2: When |M|<1 then (assume that y1<y2)

x= x1,y=y1 set Δy=1

xi+1=$\frac{1}{m}$,   y=y+1

Until y → y2</y

Advantage:

1. It is a faster method than method of using direct use of line equation.
2. This method does not use multiplication theorem.
3. It allows us to detect the change in the value of x and y ,so plotting of same point twice is not possible.
4. This method gives overflow indication when a point is repositioned.
5. It is an easy method because each step involves just two additions.

Disadvantage:

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

DDA Algorithm:

```
Step1: Start Algorithm

Step2: Declare x1,y1,x2,y2,dx,dy,x,y as integer variables.

Step3: Enter value of x1,y1,x2,y2.

Step4: Calculate dx = x2-x1

Step5: Calculate dy = y2-y1

Step6: If ABS (dx) > ABS (dy)
        Then step = abs (dx)
        Else

Step7: xinc=dx/step
        yinc=dy/step
        assign x = x1
        assign y = y1

Step8: Set pixel (x, y)

Step9: x = x + xinc
        y = y + yinc
        Set pixels (Round (x), Round (y))

Step10: Repeat step 9 until x = x2

Step11: End Algorithm
```

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: P1 (2,3)   P11 (6,15)

x1=2

y1=3

x2= 6

y2=15

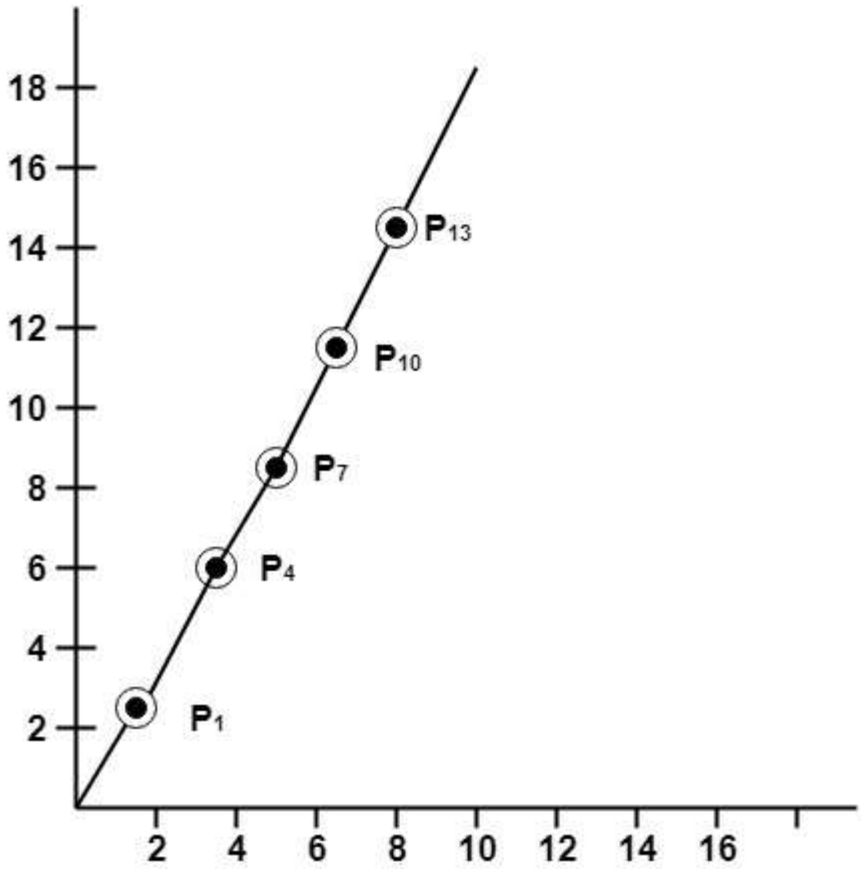
dx = 6 - 2 = 4

dy = 15 - 3 = 12

m = $\frac{dy}{dx} = \frac{12}{4}$

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

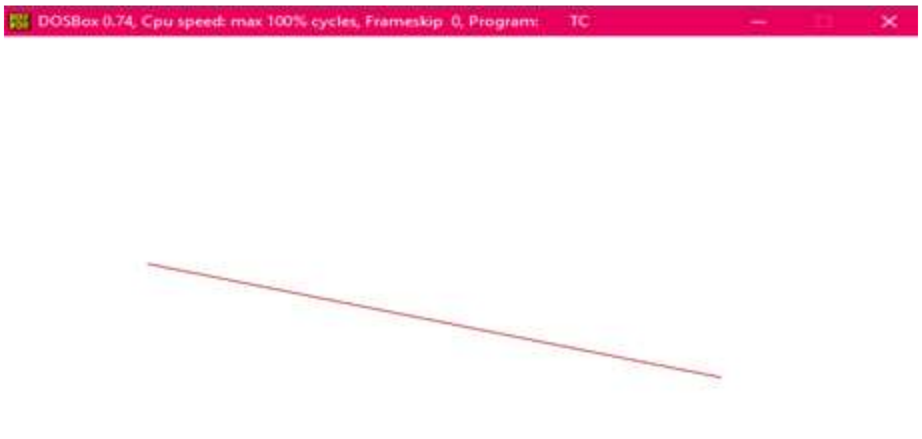


Program to implement DDA Line Drawing Algorithm:

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
void main()
{
   intgd = DETECT ,gm, i;
   float x, y,dx,dy,steps;
   int x0, x1, y0, y1;
   initgraph(&gd, &gm, "C:\\TC\\BGI");
   setbkcolor(WHITE);
   x0 = 100 , y0 = 200, x1 = 500, y1 = 300;
   dx = (float)(x1 - x0);
   dy = (float)(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)
    {
        putpixel(x, y, RED);
        x += dx;
        y += dy;
        i=i+1;
    }
    getch();
    closegraph();
}
```

Output:


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

Symmetrical DDA:

The Digital Differential Analyzer (DDA) generates lines from their differential equations. The equation of a straight line is

$$\frac{dy}{dx} = \frac{\Delta y}{\Delta x}$$

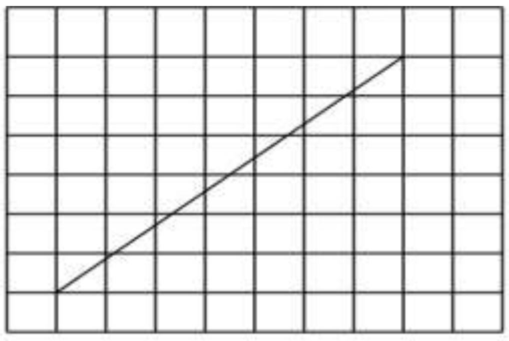
The DDA works on the principle that we simultaneously increment x and y by small steps proportional to the first derivatives of x and y. In this case of a straight line, the first derivatives are constant and are proportional to Δx and Δy. Therefore, we could generate a line by incrementing x and y by $\epsilon\ \Delta x$ and $\epsilon\ \Delta y$, where ϵ is some small quantity. There are two ways to generate points

1. By rounding to the nearest integer after each incremental step, after rounding we display dots at the resultant x and y.

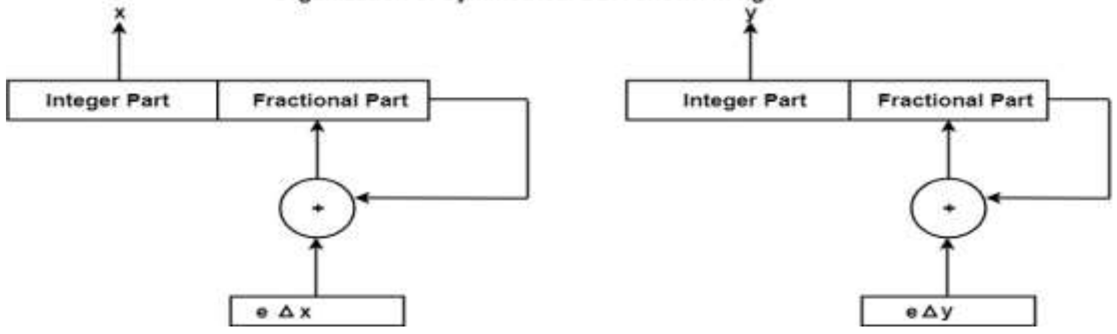
2. An alternative to rounding the use of arithmetic overflow: x and y are kept in registers that have two parts, integer and fractional. The incrementing values, which are both less than unity, are repeatedly added to the fractional parts and whenever the results overflows, the corresponding integer part is incremented. The integer parts of the x and y registers are used in plotting the line. In the case of the symmetrical DDA, we choose $\varepsilon=2^{-n}$,

where $2^{n-1} \leq \max (|\Delta x|, |\Delta y|) < 2^{n}$

A line drawn with the symmetrical DDA is shown in fig:



Organization of Symmetrical DDA shown in fig:



Example: If a line is drawn from (0, 0) to (10, 5) with a symmetrical DDA

1. How many iterations are performed?

2. How many different points will be generated?

Solutions: Given: P^1 (0,0) P^2 (10,5)

$x^1=0$

$y^1=0$

$x^2=10$

$y^2=5$

dx = 10 - 0 = 10

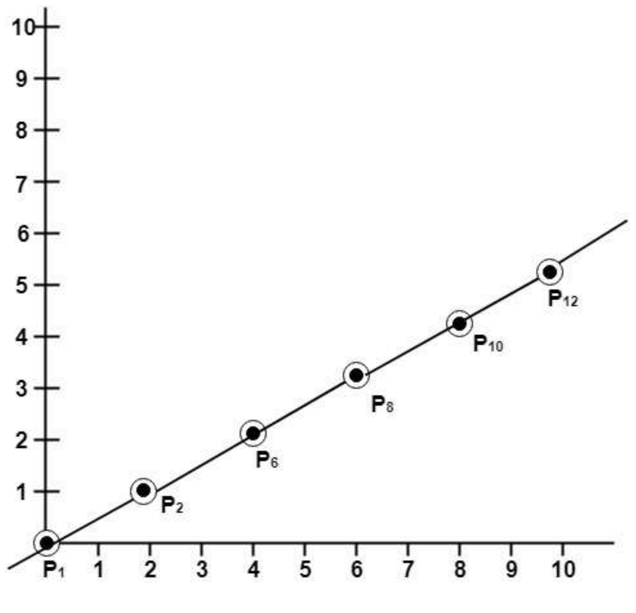
dy = 5 - 0 = 0

$$m = \frac{y_2-y_1}{x_2-x_1} = \frac{5-0}{10-0} = \frac{5}{10} = 0.5$$

P^1 (0,0) will be considered starting points

P^3 (1,0.5)	point not plotted
P^4 (2, 1)	point plotted
P^5 (3, 1.5)	point not plotted
P^6 (4,2)	point plotted
P^7 (5,2.5)	point not plotted
P^8 (6,3)	point plotted
P^9 (7,3.5)	point not plotted
P^{10} (8, 4)	point plotted
P^{11} (9,4.5)	point not plotted
P^{12} (10,5)	point plotted

Following Figure show line plotted using these points.



Bresenham's Line Algorithm

This algorithm is used for scan converting a line. 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.

In this method, next pixel selected is that one who has the least distance from true line.

The method works as follows:

Assume a pixel $P_1'(x_1', y_1')$,then select subsequent pixels as we work our may to the night, one pixel position at a time in the horizontal direction toward $P_2'(x_2', y_2')$.

Once a pixel in choose at any step

The next pixel is

1. Either the one to its right (lower-bound for the line)
2. 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'.

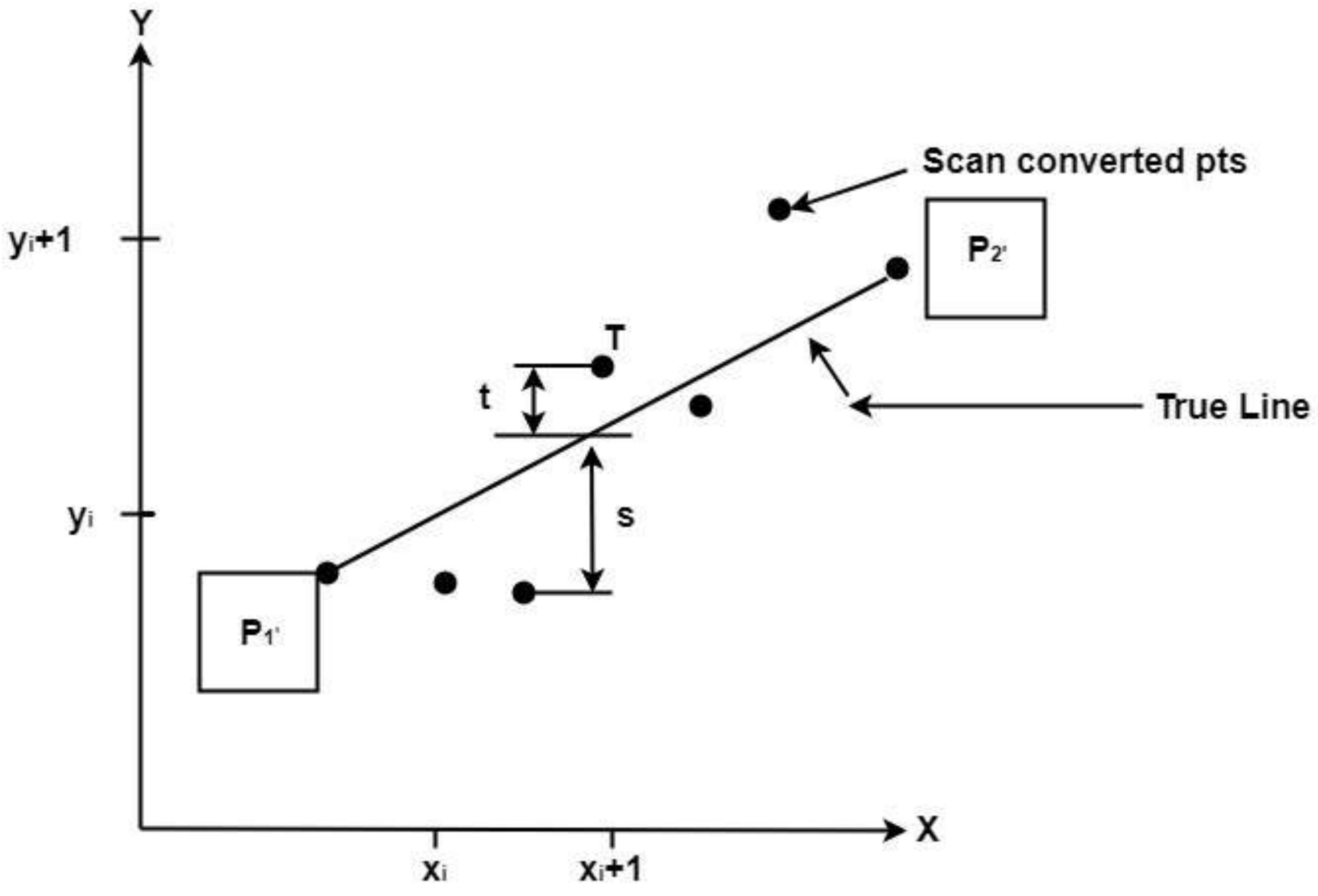


Fig: Scan Converting a line.

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

$y=mx_{i+1}+b$

$$\boxed{y = m(x_i + 1) + b}$$

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

$s-t = (y-yi)-[(yi+1)-y]$

$= 2y - 2yi - 1$

$$\boxed{s - t = 2m(x_i + 1) + 2b - 2y_i - 1}$$ [Putting the value of (1)]

Substituting m by $\frac{\triangle y}{\triangle x}$ and introducing decision variable

$d_i = \triangle x\ (s-t)$

$d_i = \triangle x\ (2\ \frac{\triangle y}{\triangle x}\ (x_i+1)+2b-2y_i-1)$

$= 2\triangle x y_i - 2\triangle y - 1\triangle x.2b - 2y_i \triangle x - \triangle x$

$d_i = 2\triangle y.x_i - 2\triangle x.y_i + c$

Where $c = 2\triangle y + \triangle x\ (2b-1)$

We can write the decision variable d_{i+1} for the next slip on

$d_{i+1} = 2\triangle y.x_{i+1} - 2\triangle x.y_{i+1} + c$

$d_{i+1} - d_i = 2\triangle y.(x_{i+1} - x_i) - 2\triangle x(y_{i+1} - y_i)$

Since x_(i+1)=x_i+1,we have

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

Special Cases

If chosen pixel is at the top pixel T (i.e., $d_i \geq 0$)$\Rightarrow y_{i+1} = y_i + 1$

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

If chosen pixel is at the bottom pixel T (i.e., $d_i < 0$)$\Rightarrow y_{i+1} = y_i$

$d_{i+1} = d_i + 2\triangle y$

Finally, we calculate d_1

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

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

Since $mx_1+b-y_i=0$ and $m = \frac{\triangle y}{\triangle x}$, we have

$$d_1=2\triangle y-\triangle x$$

Advantage:

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

2. It avoids the generation of duplicate points.

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

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

Disadvantage:

1. This algorithm is meant for basic line drawing only Initializing is not a part of Bresenham's line algorithm. So to draw smooth lines, you should want to look into a different algorithm.

Bresenham's Line Algorithm:

```
Step1: Start Algorithm

Step2: Declare variable $x_1,x_2,y_1,y_2,d,i_1,i_2,dx,dy$

Step3: Enter value of $x_1,y_1,x_2,y_2$
        Where $x_1,y_1$are coordinates of starting point
        And $x_2,y_2$ are coordinates of Ending point

Step4: Calculate $dx = x_2-x_1$
        Calculate $dy = y_2-y_1$
        Calculate $i_1=2*dy$
        Calculate $i_2=2*(dy-dx)$
        Calculate $d=i_1-dx$

Step5: Consider (x, y) as starting point and $x_{end}$as maximum possible value of x.
        If $dx < 0$
            Then $x = x_2$
            $y = y_2$
             $x_{end}=x_1$
        If $dx > 0$
          Then $x = x_1$
        $y = y_1$
            $x_{end}=x_2$
```

Step6: Generate point at (x,y)coordinates.

Step7: Check if whole line is generated.

```
        If x > = $x_{end}$
        Stop.
```

Step8: Calculate co-ordinates of the next pixel

```
        If d < 0
            Then d = d + $i_1$
        If d ≥ 0
    Then d = d + $i_2$
        Increment y = y + 1
```

Step9: Increment x = x + 1

Step10: Draw a point of latest (x, y) coordinates

Step11: Go to step 7

Step12: End of Algorithm

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

Solution: $x_1=1$

$y_1=1$

$x_2=8$

$y_2=5$

$dx= x_2-x_1=8-1=7$

$dy=y_2-y_1=5-1=4$

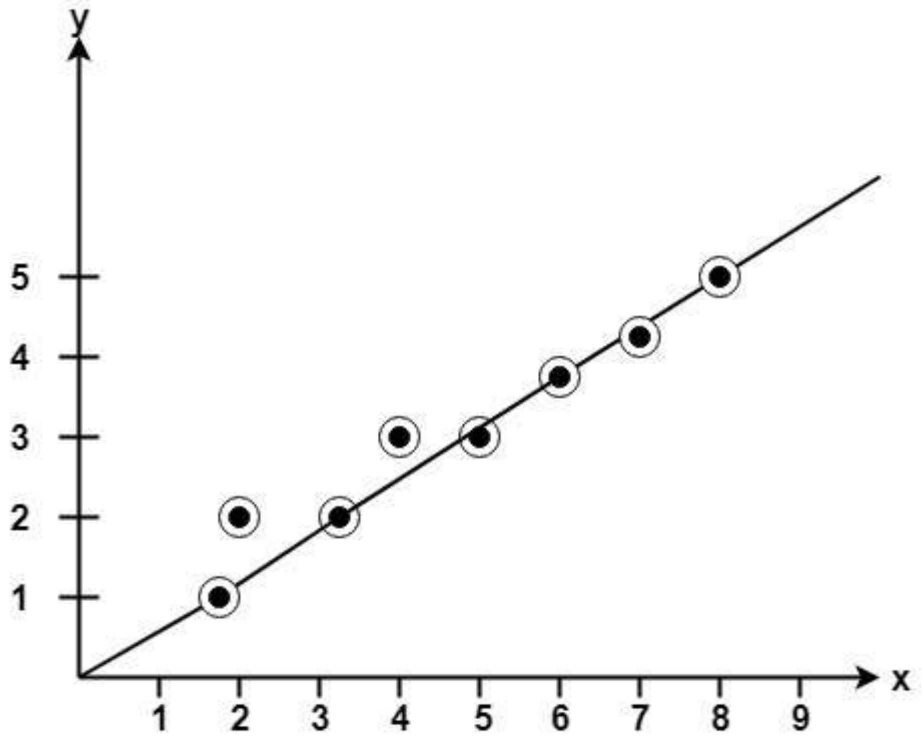
$I_1=2* \Delta y=2*4=8$

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

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

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	$d+I_1=-3+8=5$

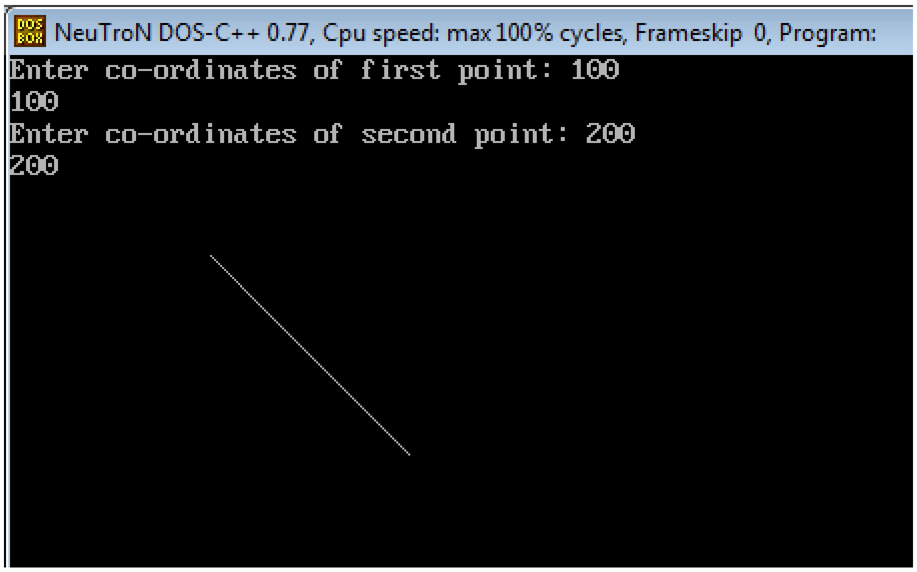
5	3	$d+I_2=5+(-6)=-1$
6	4	$d+I_1=-1+8=7$
7	4	$d+I_2=7+(-6)=1$
8	5	



Program to implement Bresenham's Line Drawing Algorithm:

```
#include<stdio.h>
#include<graphics.h>
void drawline(int x0, int y0, int x1, int y1)
{
    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;
        }
}
int main()
{
    int gdriver=DETECT, gmode, error, x0, y0, x1, y1;
    initgraph(&gdriver, &gmode, "c:\\turboc3\\bgi");
    printf("Enter co-ordinates of first point: ");
    scanf("%d%d", &x0, &y0);
    printf("Enter co-ordinates of second point: ");
    scanf("%d%d", &x1, &y1);
    drawline(x0, y0, x1, y1);
    return 0;
}
```

Output:



```
Enter co-ordinates of first point: 100
100
Enter co-ordinates of second point: 200
200
```

Differentiate between DDA Algorithm and Bresenham's Line Algorithm:

DDA Algorithm	Bresenham's Line Algorithm
1. DDA Algorithm use floating point, i.e., Real Arithmetic.	1. Bresenham's Line Algorithm use fixed point, i.e., Integer Arithmetic
2. DDA Algorithms uses multiplication & division its operation	2.Bresenham's Line Algorithm uses only subtraction and addition its operation
3. DDA Algorithm is slowly than Bresenham's Line Algorithm in line drawing because it uses real arithmetic (Floating Point operation)	3. Bresenham's Algorithm is faster than DDA Algorithm in line because it involves only addition & subtraction in its calculation and uses only integer arithmetic.

4. DDA Algorithm is not accurate and efficient as Bresenham's Line Algorithm.	4. Bresenham's Line Algorithm is more accurate and efficient at DDA Algorithm.
5.DDA Algorithm can draw circle and curves but are not accurate as Bresenham's Line Algorithm	5. Bresenham's Line Algorithm can draw circle and curves with more accurate than DDA Algorithm.

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

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.

Translation of point:

To translate a point from coordinate position (x, y) to another (x_1 y_1), we add algebraically the translation distances T_x and T_y to original coordinate.

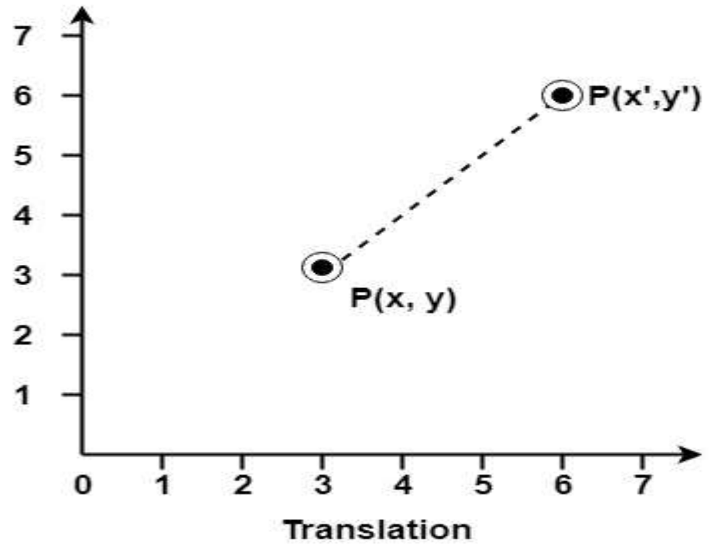
$$x_1=x+T_x$$
$$y_1=y+T_y$$

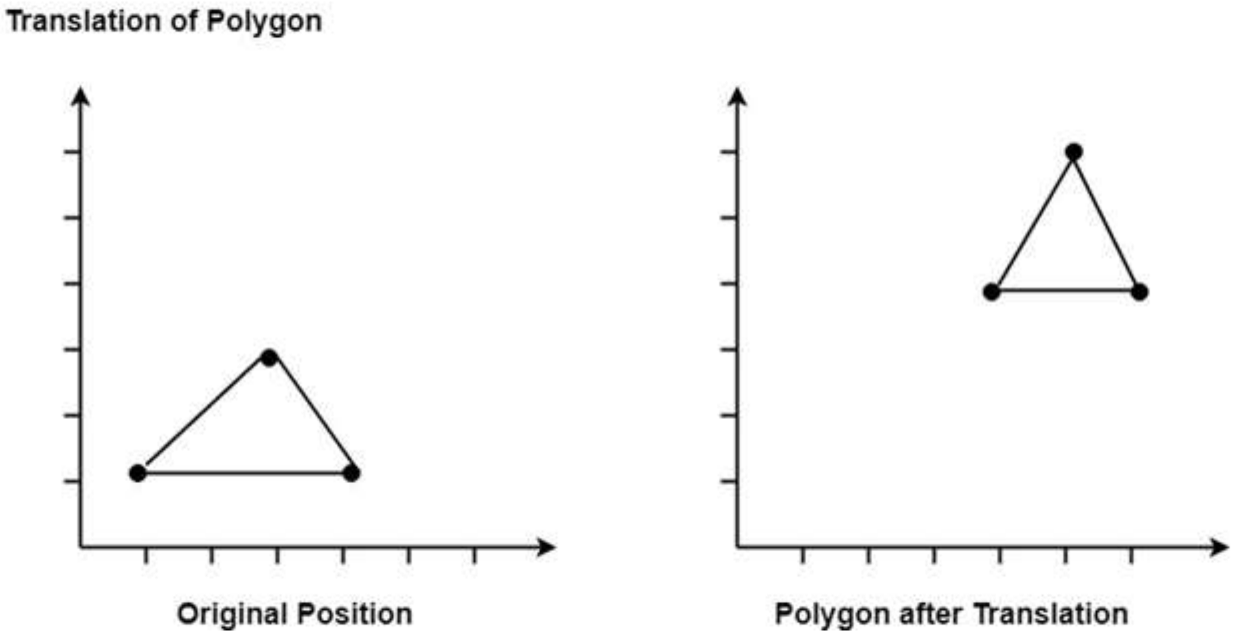
The translation pair (T_x,T_y) is called as shift vector.

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^1 y^1).





Matrix for Translation:

$$\begin{vmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{vmatrix} \text{ Or } \begin{vmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{vmatrix}$$

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 S_y 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 $S_x$and S_y are not equal then scaling will occur but it will elongate or distort the picture.

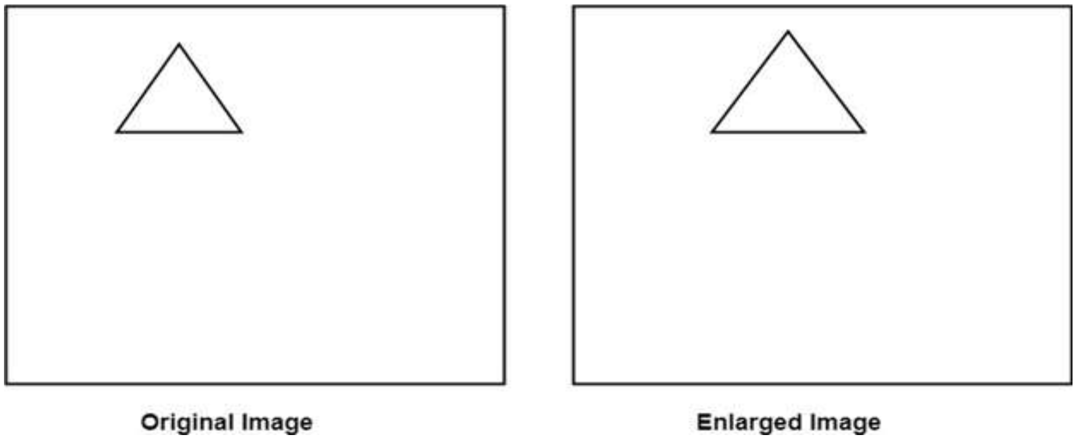
If scaling factors are less than one, then the size of the object will be reduced. If scaling factors are higher than one, then the size of the object will be enlarged.

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

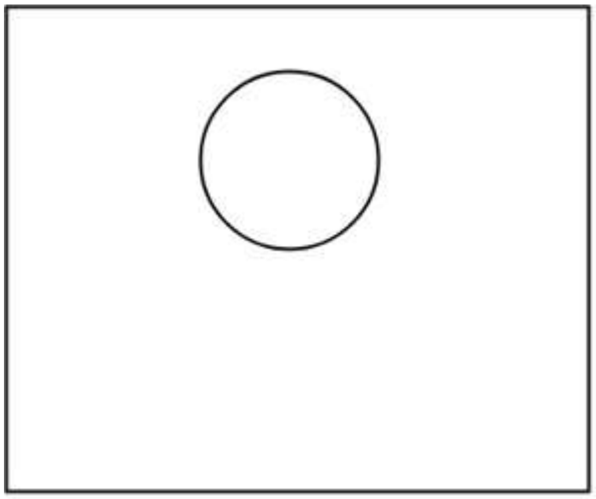
$$[x_2 y_2] = [\ x_1 y_1] \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = [2x_1 2y_1]$$

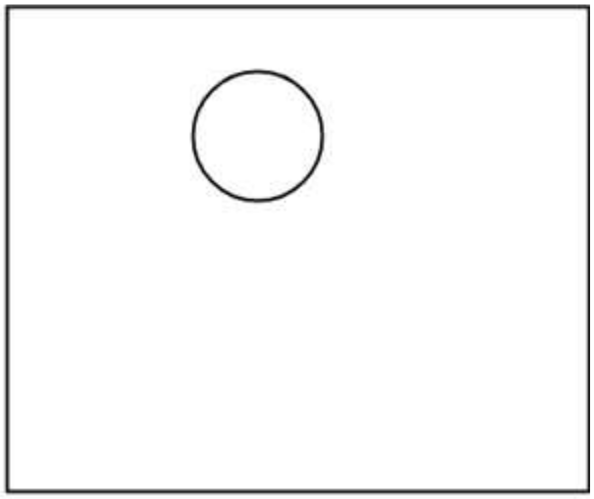
The image will be enlarged two times

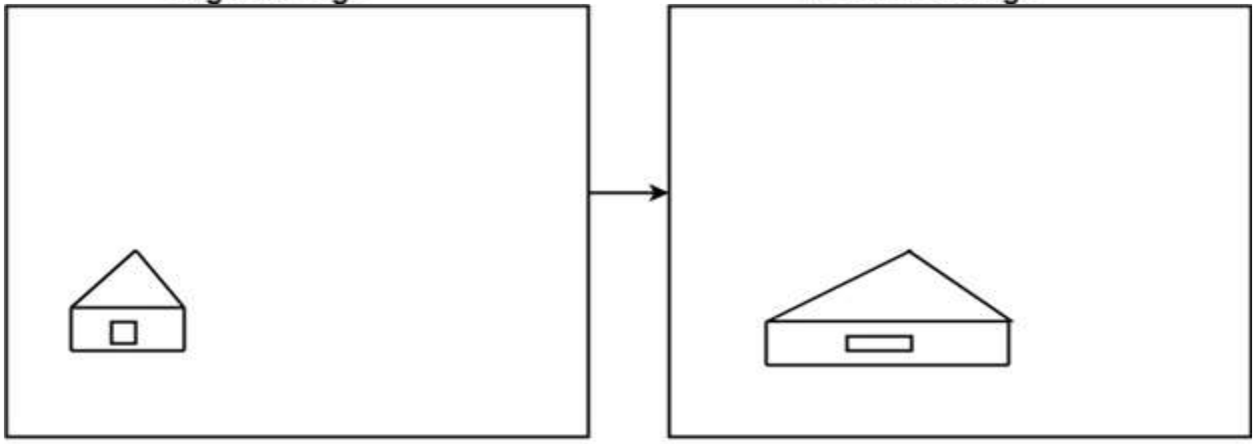


Reduction: If $T_1=\begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$. If ($x_1$ y_1) is original position and T_1 is translation vector, then (x_2 y_2) are coordinates after scaling

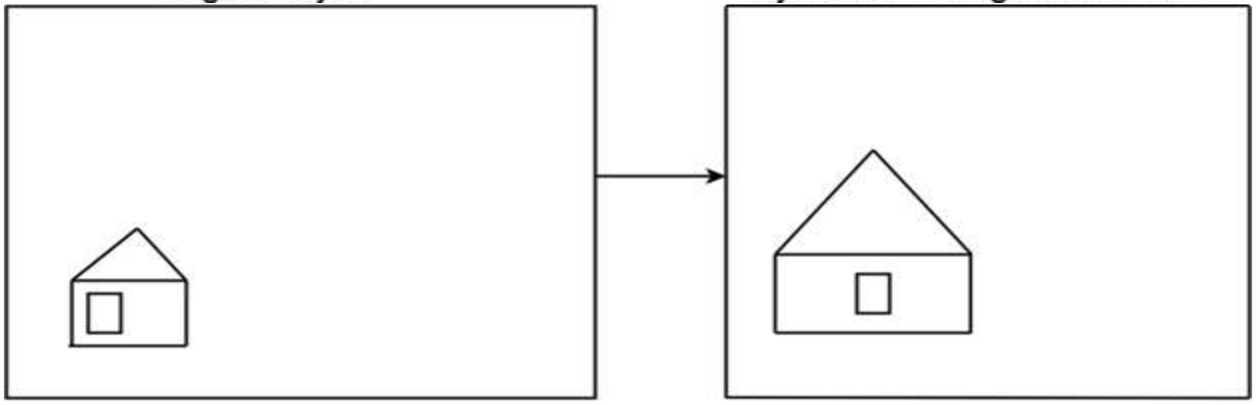
$$[\,x_2 y_2] = [\,x_1 y_1]\begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} = [.5x_1 . 5y_1]$$


Original Image


Reduction Image


Original Object

Object after scaling in X direction


Original Object

Object after scaling in Y direction

Matrix for Scaling:

$$S = \begin{vmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

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

$$S_1 = \begin{bmatrix} Sx_1 & 0 & 0 \\ 0 & Sy_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S_2 = \begin{bmatrix} Sx_2 & 0 & 0 \\ 0 & Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S_1 * S_2 = \begin{bmatrix} Sx_1 & 0 & 0 \\ 0 & Sy_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Sx_2 & 0 & 0 \\ 0 & Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} Sx_1Sx_2 & 0 & 0 \\ 0 & Sy_1Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{........................equation 1}$$

$$S_2 * S_1 = \begin{bmatrix} Sx_2 & 0 & 0 \\ 0 & Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Sx_1 & 0 & 0 \\ 0 & Sy_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} Sx_2Sx_1 & 0 & 0 \\ 0 & Sy_2Sy_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{..................................equation 2}$$

From equation 1 and 2

$$S_1S_2 = S_2S_1. \text{Hence Proved}$$

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:

1. Anticlockwise
2. 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.

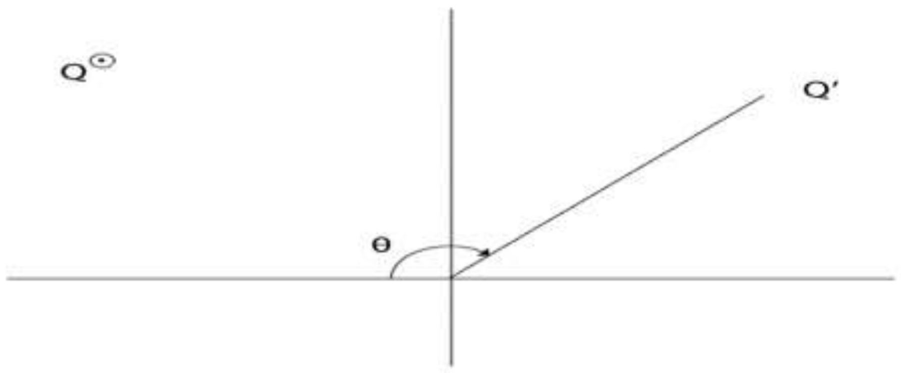
Polygon: Polygon is rotated by shifting every vertex using the same rotational angle.

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.

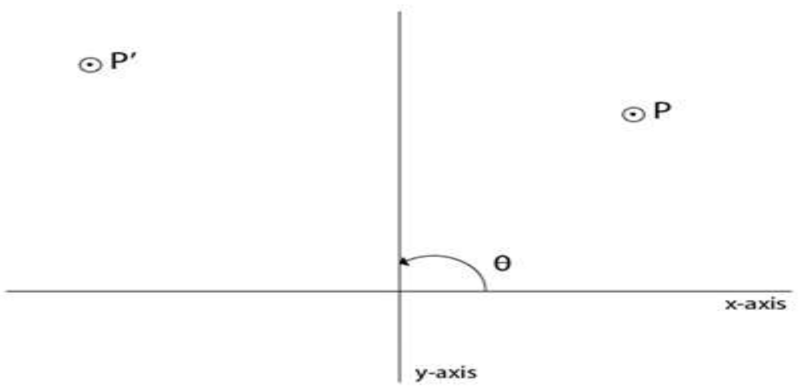
Ellipse: Its rotation can be obtained by rotating major and minor axis of an ellipse by the desired angle.

Rotation in anticlockwise direction



Q is original position
Q ' is final rotated position

Rotation of P in clockwise direction



P is original Position
P' is final position or position after rotation
where θ is angle of rotation

Matrix for homogeneous co-ordinate rotation (clockwise)

$$R = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Matrix for homogeneous co-ordinate rotation (anticlockwise)

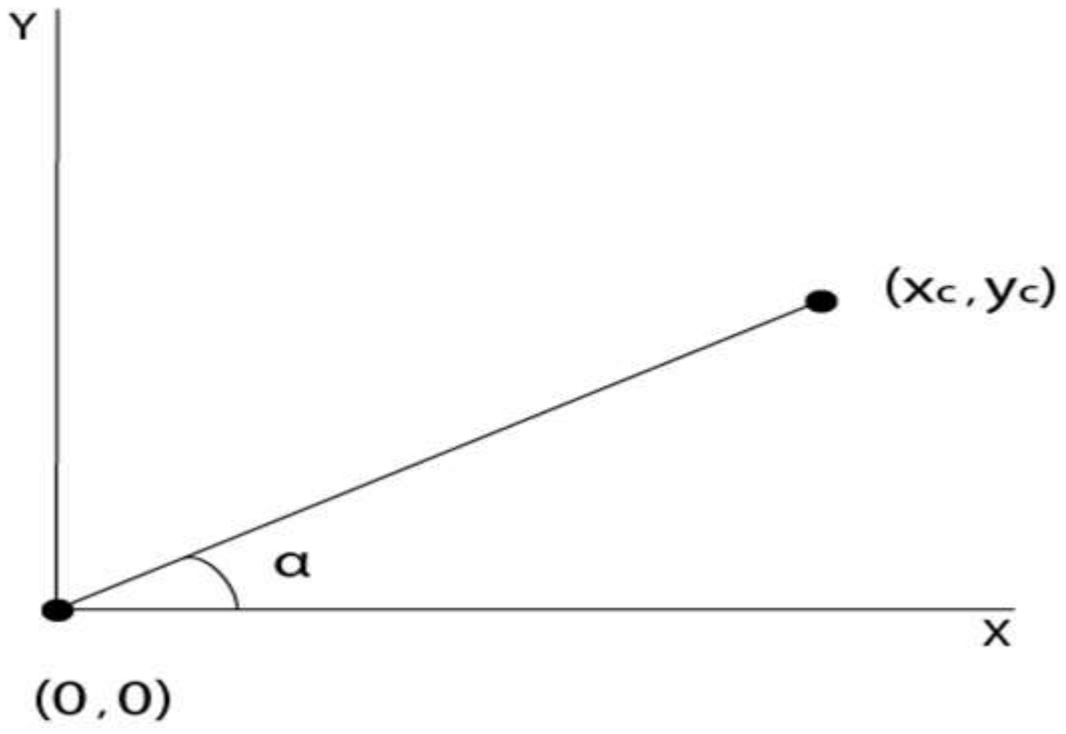
$$R = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

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.

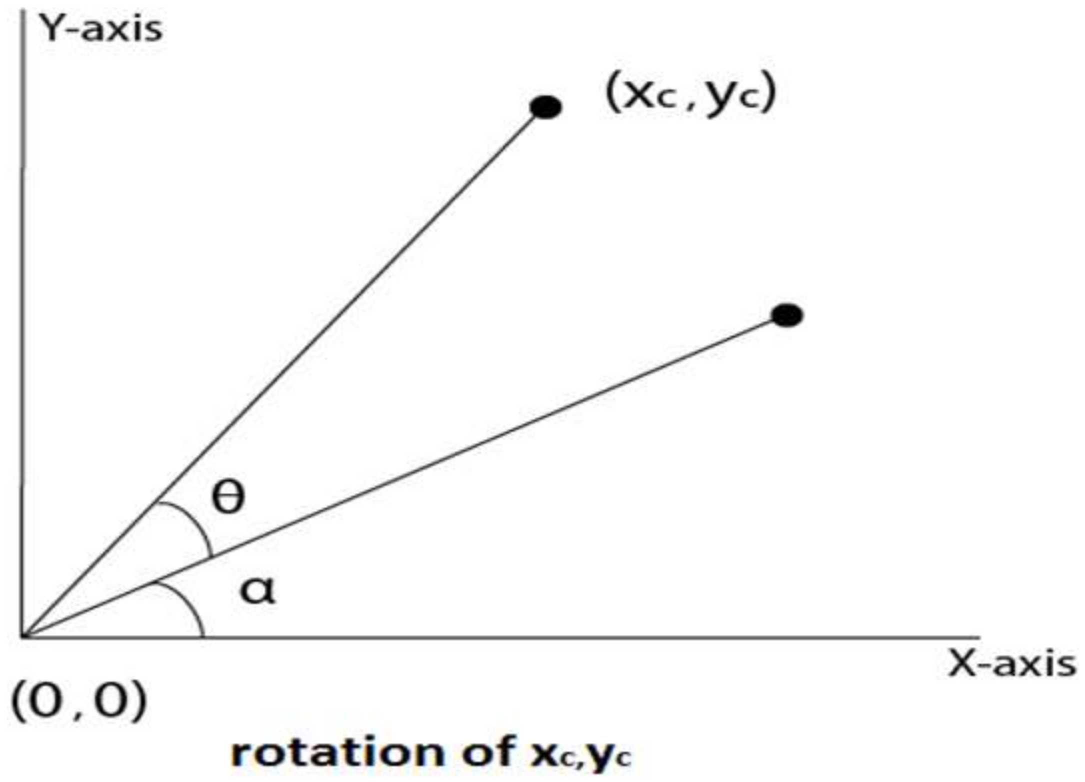
Example: The point (x, y) is to be rotated

The (x_c y_c) is a point about which counterclockwise rotation is done

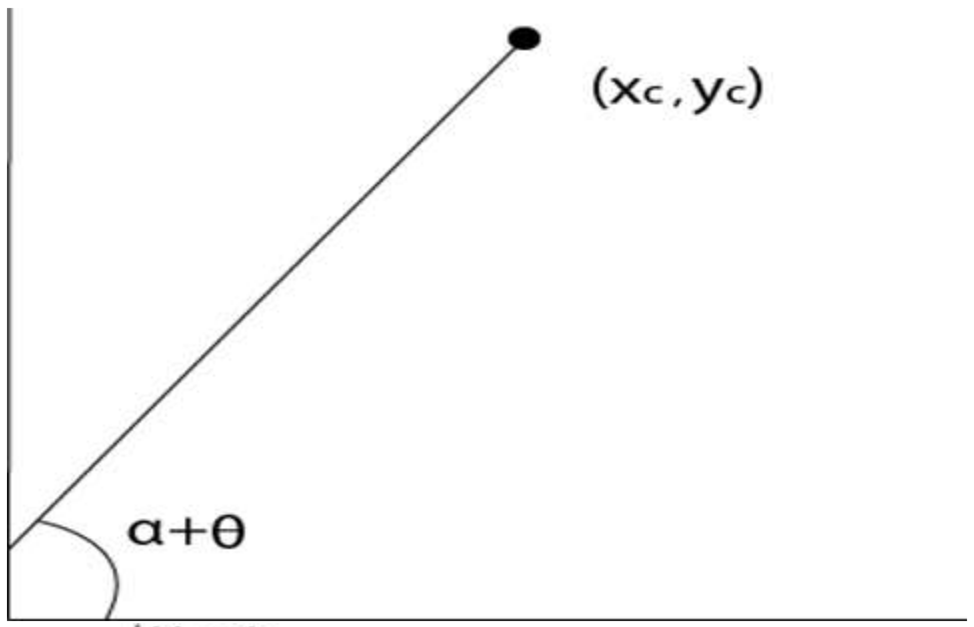
Step1: Translate point (x_c y_c) to origin

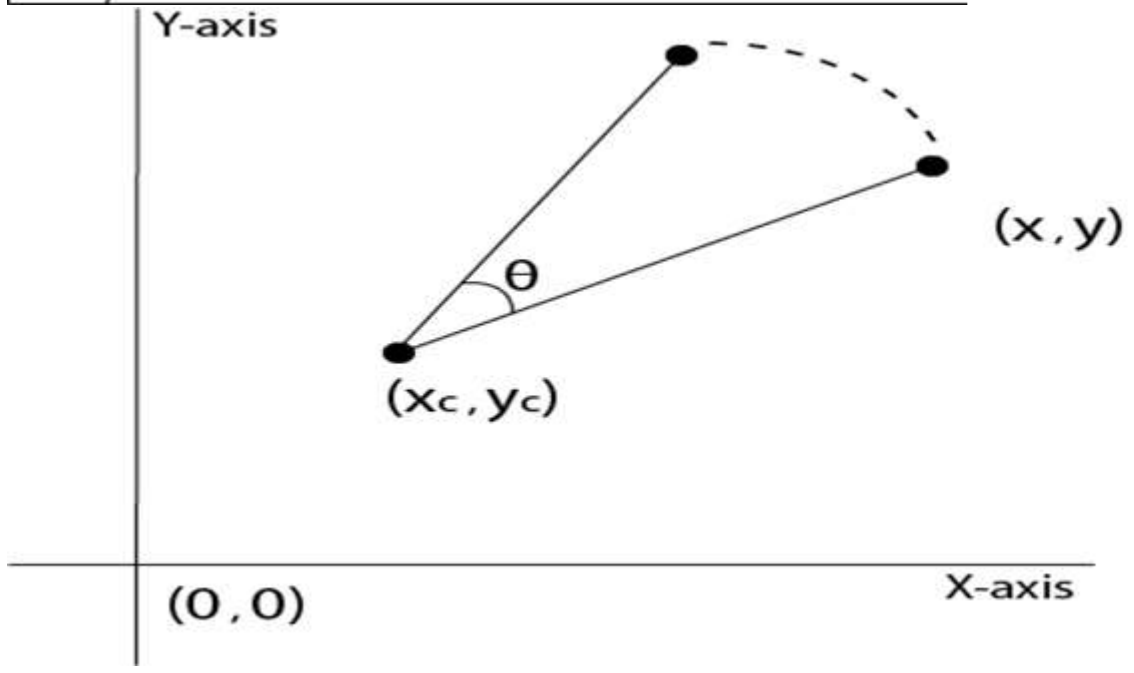


Step2: Rotation of (x, y) about the origin



Step3: Translation of center of rotation back to its original position





Example1: Prove that 2D rotations about the origin are commutative i.e. R_1 R_2=R_2 R_1.

Solution: R_1 and $R_2$are rotation matrices

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

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

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

$$= \begin{bmatrix} \cos\theta_1 \cos\theta_2 \; - \sin\;\theta_1 \sin\theta_2 & \cos\theta_1 \sin\theta_2 + \sin\theta_1 \cos\theta_2 & 0 \\ -\sin\theta_1 \cos\theta_2 \; - \; \cos\theta_1 \sin\theta_2 & -\sin\theta_1 \sin\theta_2 + \cos\theta_1 \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \ldots \text{eq 1}$$

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

$$= \begin{bmatrix} \cos\theta_2 \cos\theta_1 \; - \sin\;\theta_2 \sin\theta_2 & \cos\theta_2 \sin\theta_1 + \sin\theta_2 \cos\theta_1 & 0 \\ -\sin\theta_2 \cos\theta_1 \; - \; \cos\theta_2 \sin\theta_1 & -\sin\theta_2 \sin\theta_2 + \cos\theta_2 \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \ldots \text{eq 2}$$

From eq 1 & eq 2

$R_1 R_2 = R_2 R_1$. 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 = $\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$

$\theta = 45°$

Let

R = $\begin{bmatrix} \cos 45° & \sin 45° \\ -\sin 45° & \cos 45° \end{bmatrix}$

R = $\begin{bmatrix} 0.707 & 0.707 \\ -0.707 & -0.707 \end{bmatrix}$

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

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

= [3 * 0.707 – 4 * 0.707 3 * 0.707 + 4 * 0.707]

[2.121-2.8282.21+2.828]

[-.707 4.949]

The rotation of point B (12, 15)

$R_1 = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$

= $\begin{bmatrix} 0.707 & 0.707 \\ -0.707 & -0.707 \end{bmatrix}$

The point B (12, 15) will be

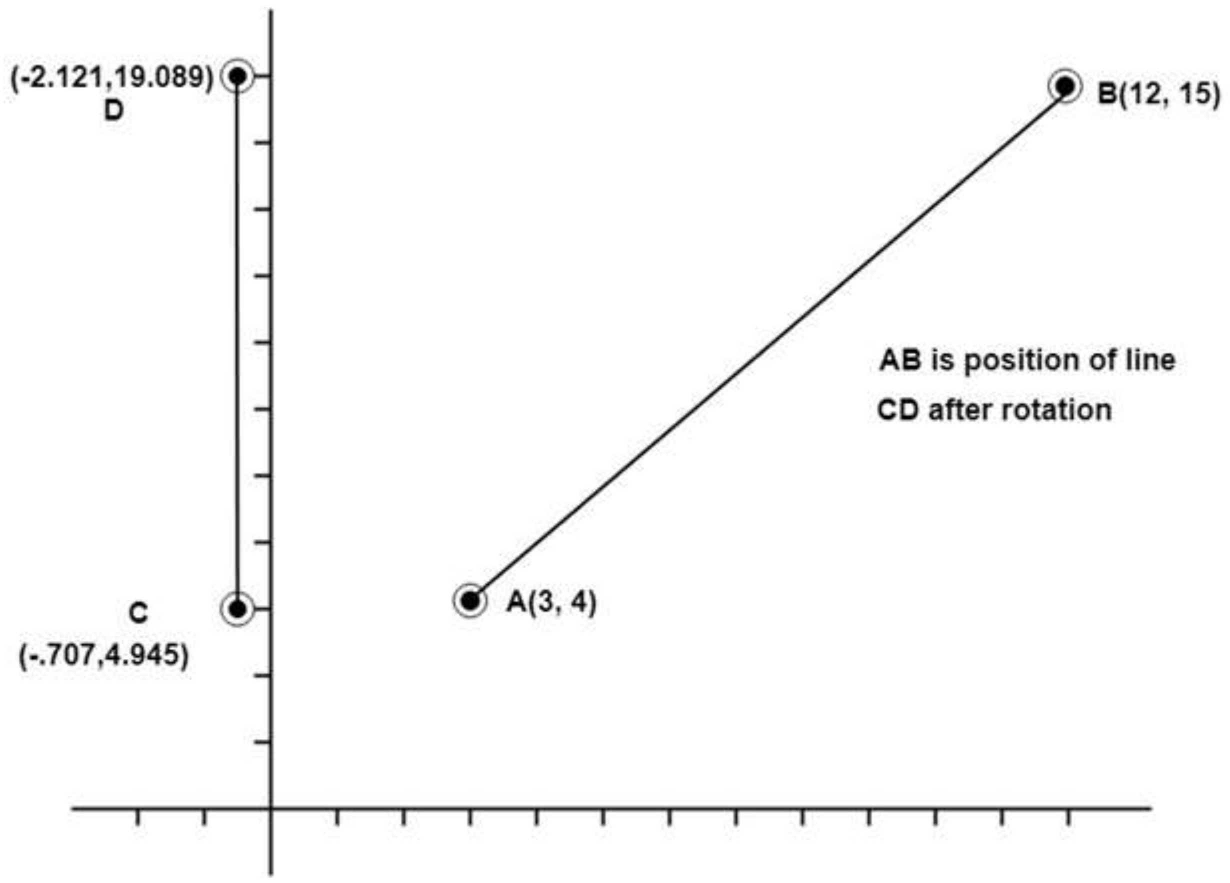
[x, y] = [12, 15] $\begin{bmatrix} 0.707 & 0.707 \\ -0.707 & -0.707 \end{bmatrix}$

= [12 * 0.707 + 15 * 0.707 12 * 0.707 + 15 * 0.707]

= [(8.484-10.605) (8.484 + 10.605)]

= [-2.121 19.089]

So line AB after rotation at $45°$become [.707, 4.945] and [-2.121, 19.089]



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

Solution: For rotation in the clockwise direction. The matrix is

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

Step1: Rotation of point A (2, 5). Take angle 30°

$$R = \begin{bmatrix} \cos 30° & -\sin 30° \\ \sin 30° & \cos 30° \end{bmatrix}$$

$$= [2, 5] \begin{bmatrix} \cos 30° & -\sin 30° \\ -\sin 30° & \cos 30° \end{bmatrix}$$

$$= [2, 5] \begin{bmatrix} .866 & -0.5 \\ .5 & .866 \end{bmatrix}$$

$$= [2 * .866 + 5 * .5 \qquad 2 * (-.5) + 5\ (.866)]$$

$$= [(1.732 + 2.5 \qquad -1 + 4.33]$$

$$= [4.232 \qquad 3.33]$$

A point (2, 5) become (4.232, 3.33)

Step2: Rotation of point B (6, 12)

$$= [6\ \ 12] \begin{bmatrix} \cos 30° & -\sin 30° \\ \sin 30° & \cos 30° \end{bmatrix}$$

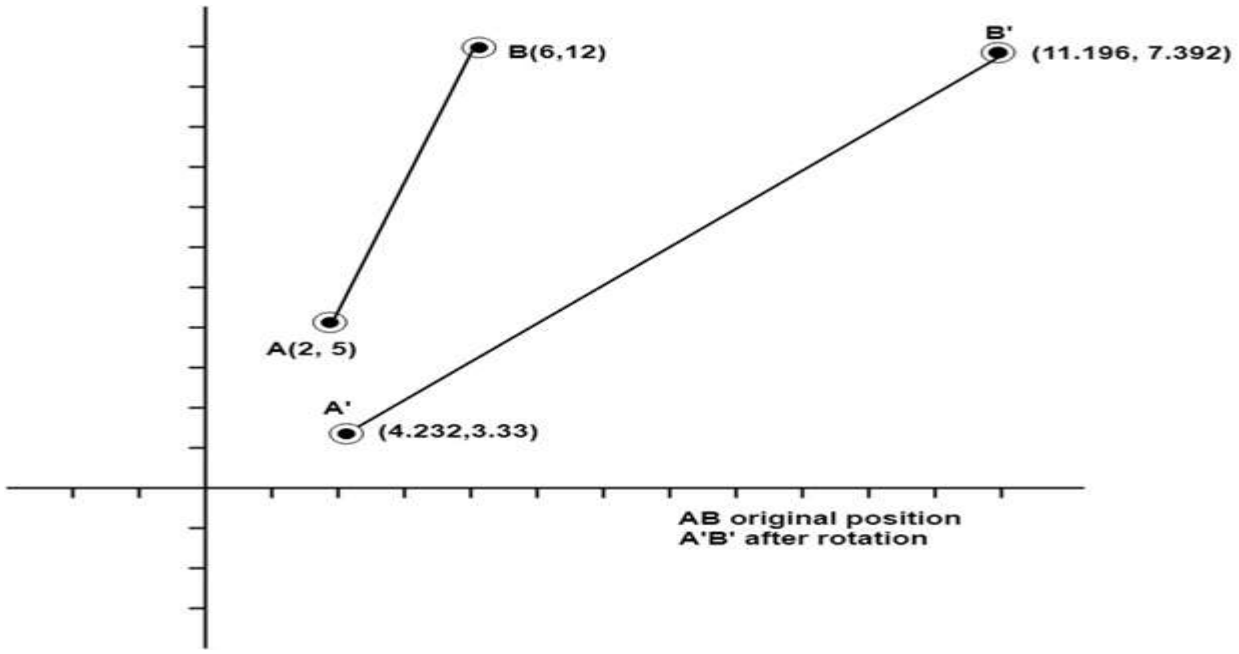
$$= [6\ 12] \begin{bmatrix} .866 & -0.5 \\ .5 & .866 \end{bmatrix}$$

$$= [6 * .866 + 12 * .5 \qquad 6 * (-.5) + 12 * (.866)]$$

$$= [5.196 + 6 \qquad -3 + 10.392]$$

$$= [11.196 \qquad 7.392]$$

B point (6, 12) becomes (11.46, 7.312) after rotation 30° in clockwise direction.

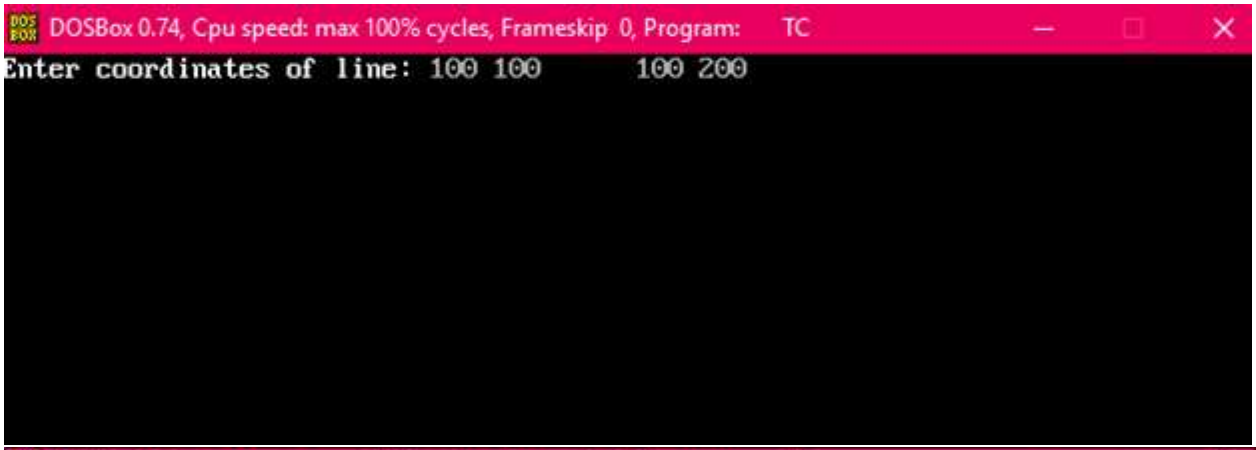


Program to rotate a line:

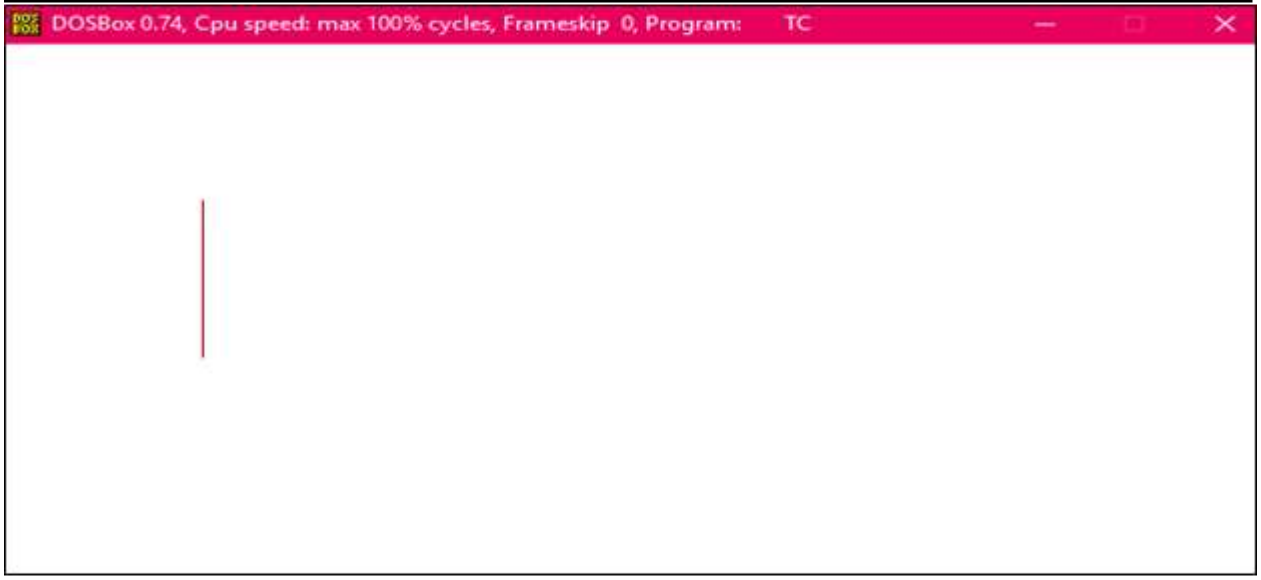
```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
int main()
{
    intgd=0,gm,x1,y1,x2,y2;
    double s,c, angle;
    initgraph(&gd, &gm, "C:\\TC\\BGI");
    setcolor(RED);
    printf("Enter coordinates of line: ");
    scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
    cleardevice();
    setbkcolor(WHITE);
    line(x1,y1,x2,y2);
    getch();
    setbkcolor(BLACK);
    printf("Enter rotation angle: ");
    scanf("%lf", &angle);
    setbkcolor(WHITE);
    c = cos(angle *3.14/180);
    s = sin(angle *3.14/180);
    x1 = floor(x1 * c + y1 * s);
    y1 = floor(-x1 * s + y1 * c);
    x2 = floor(x2 * c + y2 * s);
    y2 = floor(-x2 * s + y2 * c);
    cleardevice();
    line(x1, y1 ,x2, y2);
    getch();
    closegraph();
return 0;
}
```

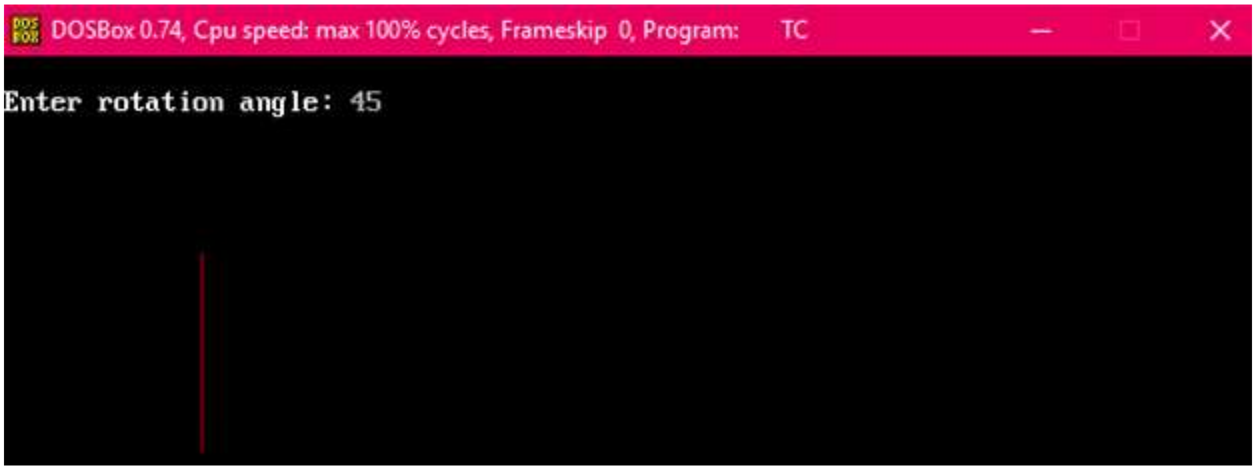
Output:

Before rotation



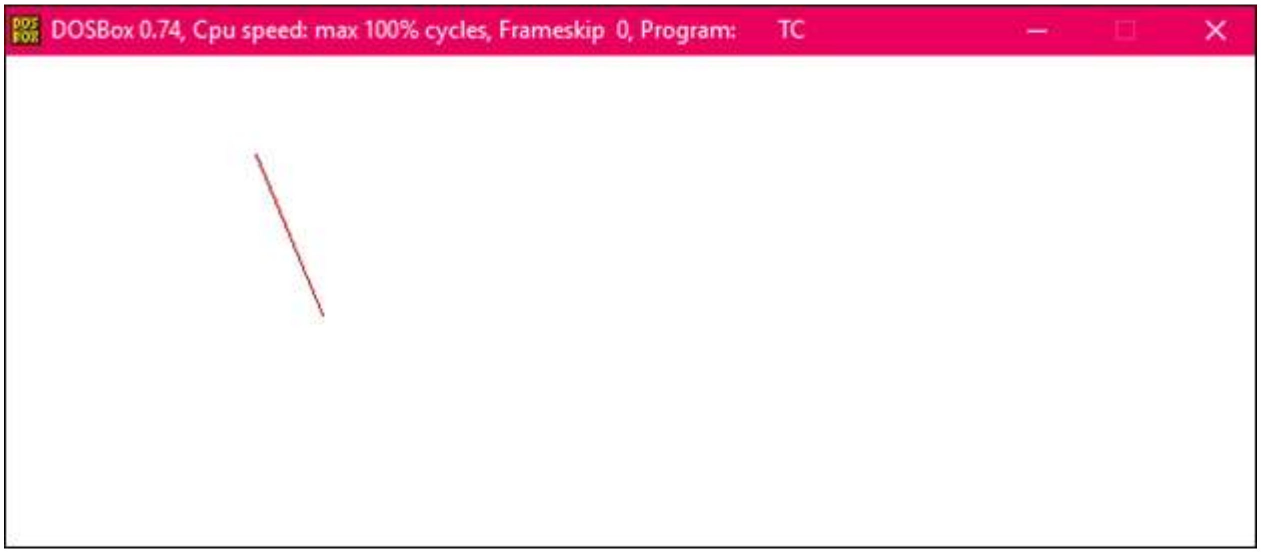
```
Enter coordinates of line: 100 100      100 200
```





```
Enter rotation angle: 45
```

After rotation

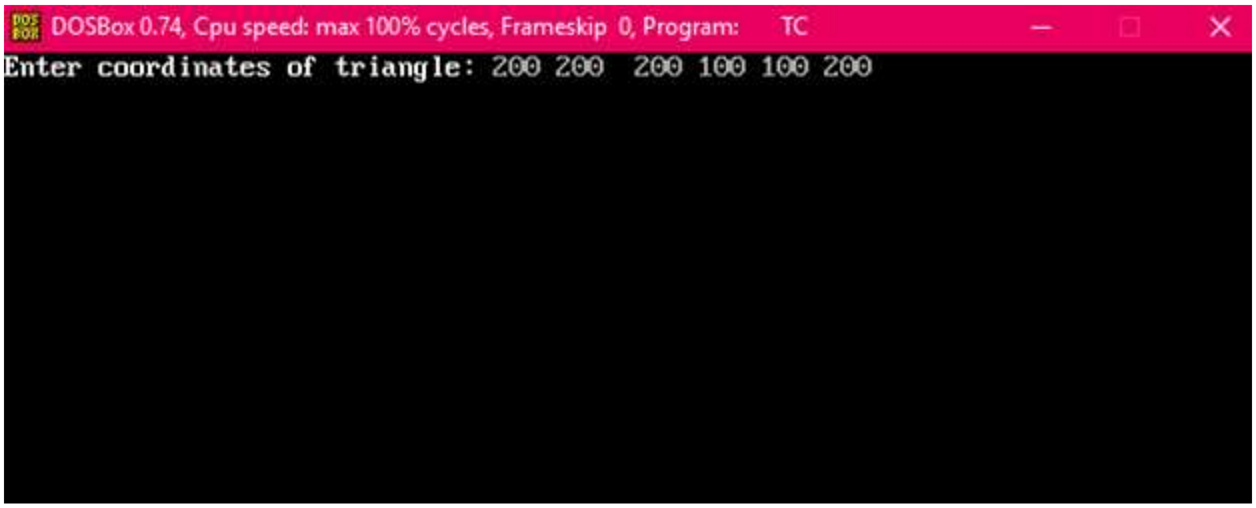

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

Program to rotate a Triangle:

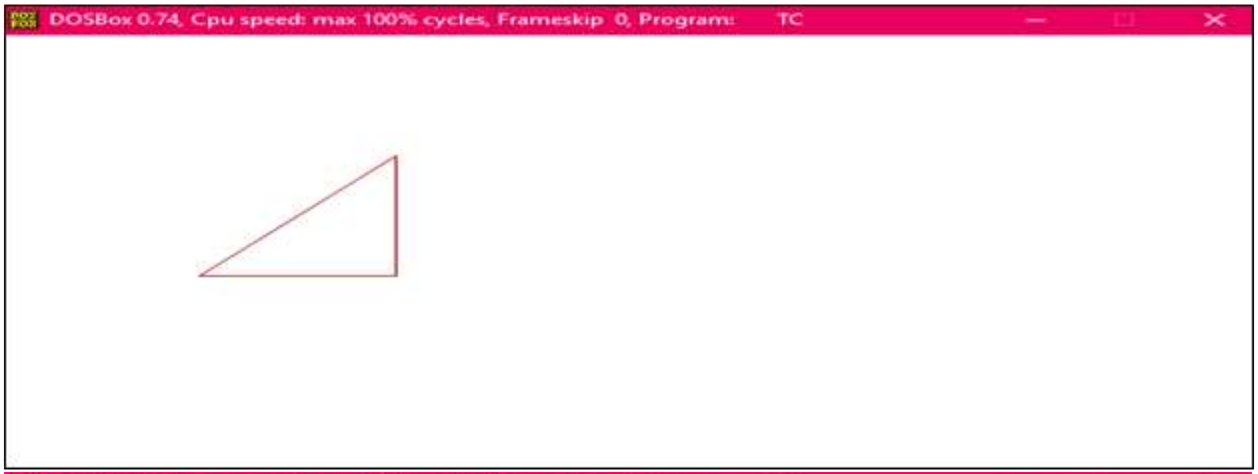
```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
main()
{
    intgd=0,gm,x1,y1,x2,y2,x3,y3;
    double s,c, angle;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
    setcolor(RED);
    printf("Enter coordinates of triangle: ");
    scanf("%d%d%d%d%d%d",&x1,&y1,&x2,&y2, &x3, &y3);
    setbkcolor(WHITE);
    cleardevice();
    line(x1,y1,x2,y2);
    line(x2,y2, x3,y3);
    line(x3, y3, x1, y1);
    getch();
    setbkcolor(BLACK);
    printf("Enter rotation angle: ");
    scanf("%lf", &angle);
    setbkcolor(WHITE);
    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);
    cleardevice();
    line(x1, y1 ,x2, y2);
    line(x2,y2, x3,y3);
    line(x3, y3, x1, y1);
    getch();
    closegraph();
    return 0;
}
```

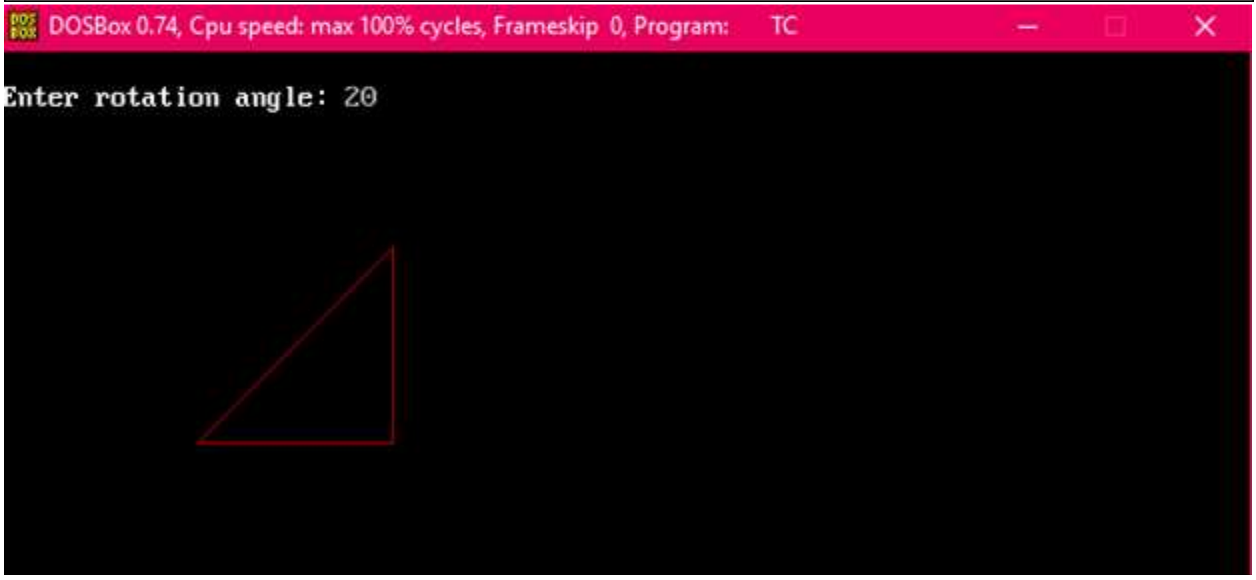
Output:

Before rotation


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
Enter coordinates of triangle: 200 200  200 100 100 200
```


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
Enter rotation angle: 20
```

After rotation

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

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 by180°.

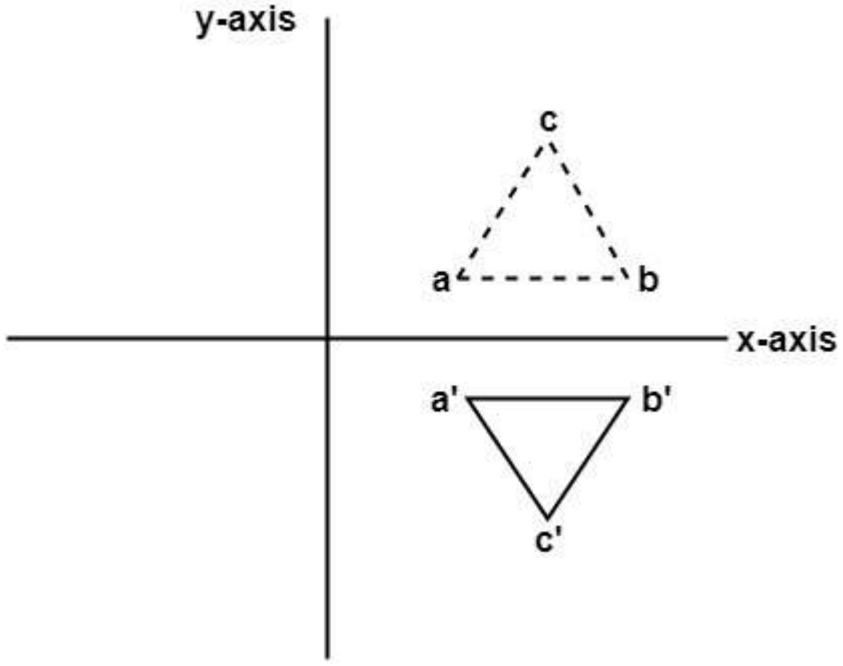
Types of Reflection:

1. Reflection about the x-axis
2. Reflection about the y-axis
3. Reflection about an axis perpendicular to xy plane and passing through the origin
4. Reflection about line y=x

1. 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.

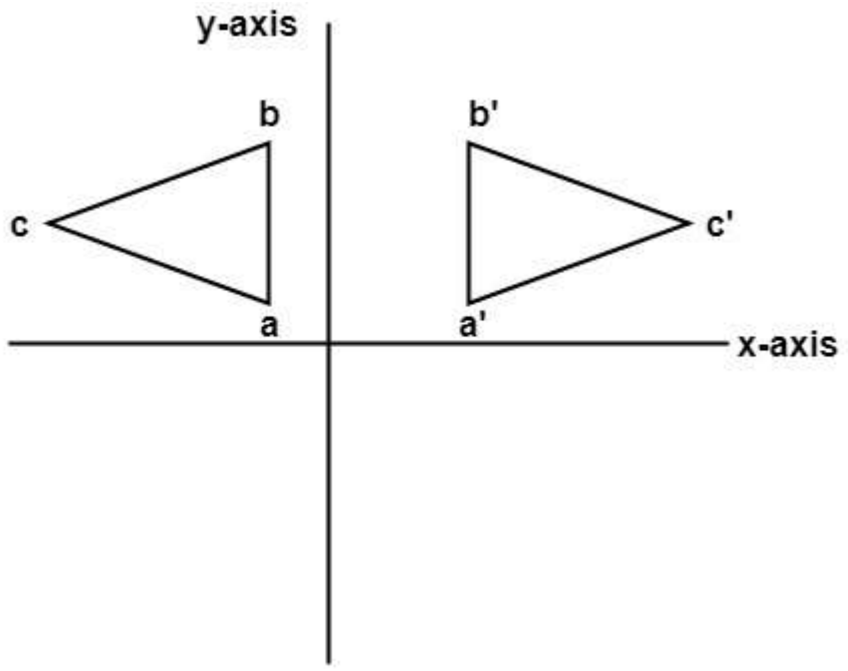


2. Reflection about y-axis: The object can be reflected about y-axis with the help of following transformation matrix

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

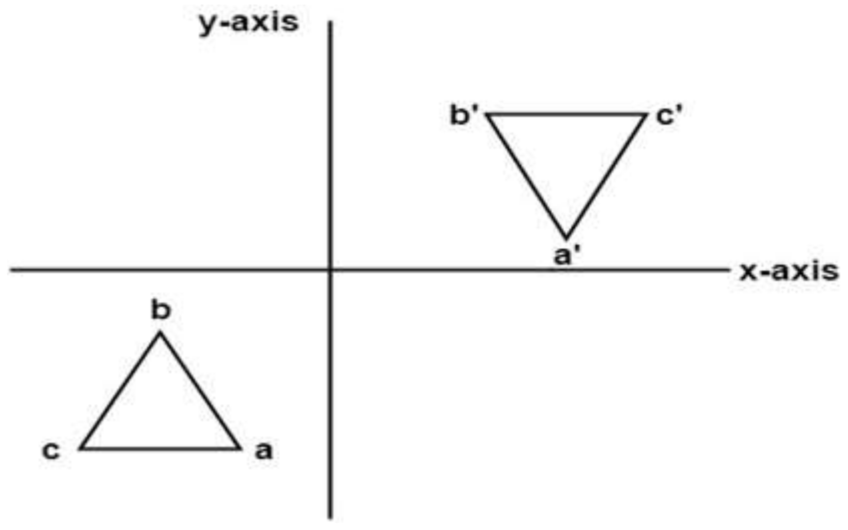
Here the values of x will be reversed, whereas the value of y will remain the same. The object will lie another side of the y-axis.

The following figure shows the reflection about the y-axis



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

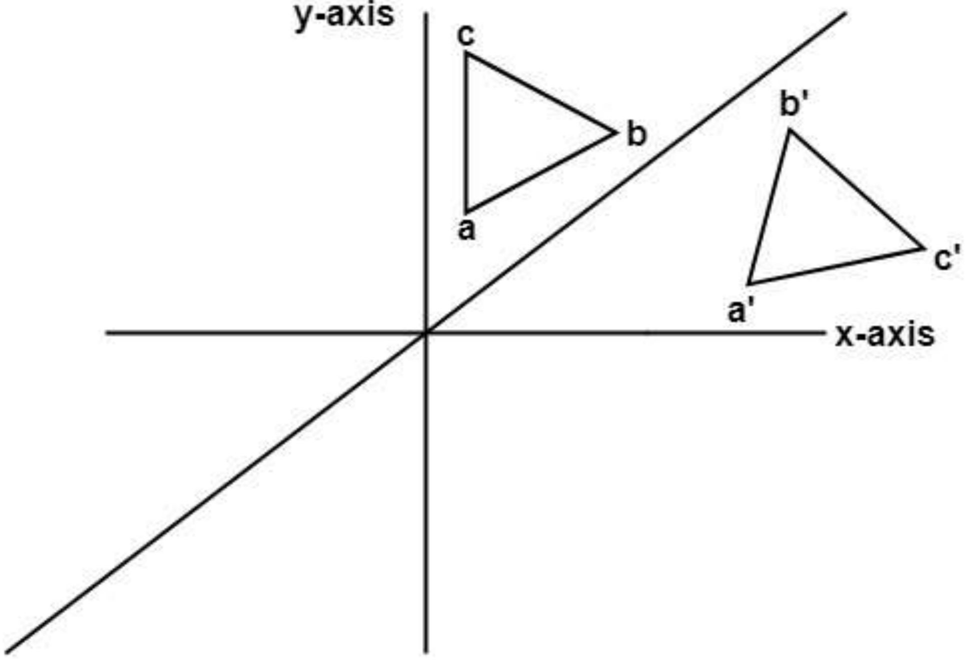
$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



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

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

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



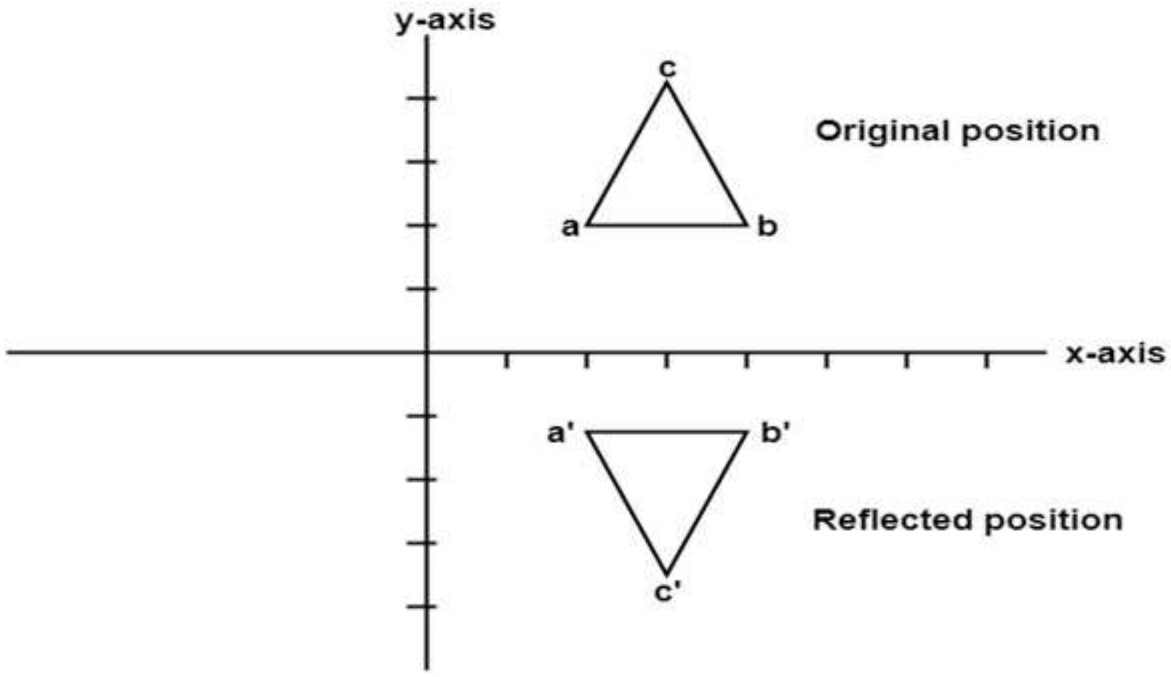
First of all, the object is rotated at 45°. The direction of rotation is clockwise. After it reflection is done concerning x-axis. The last step is the rotation of y=x back to its original position that is counterclockwise at 45°.

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 reflection about x axis $\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

The a point coordinates after reflection

$$(x, y) = (3, 4) \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$(x, y) = [3, -4]$$

The b point coordinates after reflection

$$(x, y) = (6, 4) \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$(x, y) = [6, -4]$$

The coordinate of point c after reflection

$$(x, y) = (4, 8)\begin{bmatrix}1 & 0 & 0\\ 0 & -1 & 0\\ 0 & 0 & 1\end{bmatrix}$$

$$(x, y) = [4, -8]$$

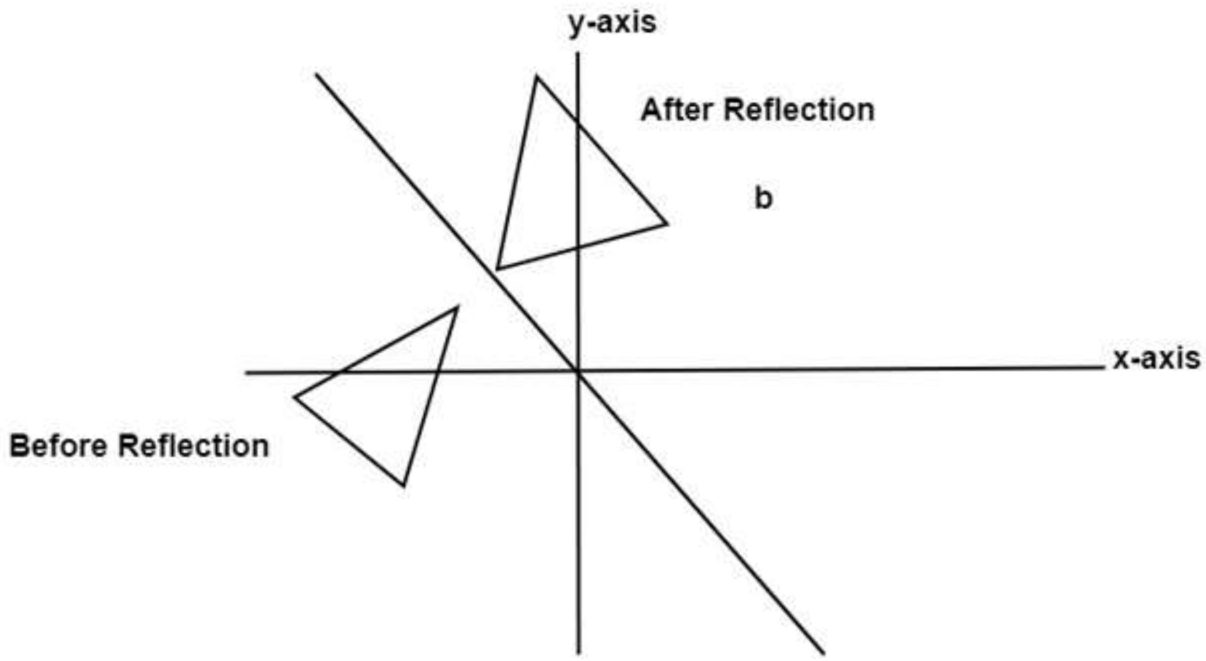
a (3, 4) becomes a^1 (3, -4)
b (6, 4) becomes b^1 (6, -4)
c (4, 8) becomes c^1 (4, -8)

Program to perform Mirror Reflection about a line:

```
#include <iostream.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>
#include <stdlib.h>
#define pi 3.14
class arc
{
    float x[10],y[10],theta,ref[10][10],ang;
        float p[10][10],p1[10][10],x1[10],y1[10],xm,ym;
    int i,k,j,n;
    public:
    void get();
    void cal ();
    void map ();
    void graph ();
    void plot ();
    void plot1();
};
void arc::get ()
{
    cout<<"\n ENTER ANGLE OF LINE INCLINATION AND Y INTERCEPT";
    cin>> ang >> b;
    cout <<"\n ENTER NO OF VERTICES";
    cin >> n;
    cout <<"\n ENTER";
    for (i=0; i<n; i++)
    {
        cout<<"\n x["<<i<<"] and y["<<i<<"]";
    }
    theta =(ang * pi)/ 180;
    ref [0] [0] = cos (2 * theta);
    ref [0] [1] = sin (2 * theta);
    ref [0] [2] = -b *sin (2 * theta);
    ref [1] [0] = sin (2 * theta);
    ref [1] [1] = -cos (2 * theta);
    ref [1] [2] = b * (cos (2 * theta)+1);
    ref [2] [0]=0;
    ref [2] [1]=0;
    ref [2] [2] = 1;
}
void arc :: cal ()
{
    for (i=0; i < n; i++)
    {
        p[0] [i] = x [i];
        p [1] [i] = y [i];
        p [2] [i] = 1;
    }
    for (i=0; i<3;i++)
    {
        for (j=0; j<n; j++)
        {
            p1 [i] [j]=0;
            for (k=0;k<3; k++)
        }
        p1 [i] [j] + = ref [i] [k] * p [k] [j];
             }
for (i=0; i<n; i++)
   {
    x1 [i]=p1[0] [i];
    y1 [i] = p1 [1] [i];
    }
}
void arc :: map ()
{
    int gd = DETECT,gm;
    initgraph (&gd, &gm, " ");
            int errorcode = graphresult ();
    /* an error occurred */
    if (errorcode ! = grOK)
    {
        printf ("Graphics error: %s \n", grapherrormsg (errorcode));
        printf ("Press any key to halt:");
        getch ();
        exit (1); /* terminate with an error code */
    }
}
void arc :: graph ()
{
    xm=getmaxx ()/2;
    ym=getmaxy ()/2;
    line (xm, 0, xmm 2*ym);
}
void arc :: plot 1 ()
{
    for (i=0; i <n-1; i++)
    {
        circle (x1[i]+xm, (-y1[i]+ym), 2);
        line (x1[i]+xm, (-y1[i]+ym), x1[i+1]+xm, (-y1[i+1]+ym));
    }
        line (x1[n-1)+xm, (-y1[n-1]+ym), x1[0]+xm, (-y1[0]+ym));
        getch();
}
```

```
void arc :: plot ()
{
   for (i=0; i <n-1; i++)
   {
      circle (x1[i]+xm, (-y1[i]+ym, 2);
             line (x1[i]+xm, (-y1[i]+ym), x[i+1]+xm, (-y1[i+1]+ym));
         }
             line (x[n-1]+xm, (-y1[n-1]+ym), x[0]+xm, (-y[0]+ym));
             getch();
      }
      void main ()
      {
         class arc a;
         clrscr();
         a.map();
         a.graph();
         a.get();
         a.cal();
         a.plot();
         a.plot1();
         getch();
      }
```

Output:



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

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:

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

Matrix Representation of 2D Transformation

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

2. Rotation (clockwise) $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

3. Rotation (anti-clock) $\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$

4. Translation $\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ t_x & t_y \end{bmatrix}$

5. Reflection $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

(about x axis)

6. Reflection $\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$

(about y axis)

7. Reflection $\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$

(about origin)

8. Reflection about Y=X $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

9. Reflection about Y= –X $\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$

10. Shearing in X direction $\begin{bmatrix} 1 & 0 \\ Sh_x & 1 \end{bmatrix}$

11. Shearing in Y direction $\begin{bmatrix} 1 & Sh_y \\ 0 & 1 \end{bmatrix}$

12. Shearing in both x and y direction $\begin{bmatrix} 1 & Sh_y \\ Sh_x & 1 \end{bmatrix}$

Homogeneous Coordinates

The rotation of a point, straight line or an entire image on the screen, about a point other than origin, is achieved by first moving the image until the point of rotation occupies the origin, then performing rotation, then finally moving the image to its original position.

The moving of an image from one place to another in a straight line is called a translation. A translation may be done by adding or subtracting to each point, the amount, by which picture is required to be shifted.

Translation of point by the change of coordinate cannot be combined with other transformation by using simple matrix application. Such a combination is essential if we wish to rotate an image about a point other than origin by translation, rotation again translation.

To combine these three transformations into a single transformation, homogeneous coordinates are used. In homogeneous coordinate system, two-dimensional coordinate positions (x, y) are represented by triple-coordinates.

Homogeneous coordinates are generally used in design and construction applications. Here we perform translations, rotations, scaling to fit the picture into proper position.

Example of representing coordinates into a homogeneous coordinate system: For two-dimensional geometric transformation, we can choose homogeneous parameter h to any non-zero value. For our convenience take it as one. Each two-dimensional position is then represented with homogeneous coordinates (x, y, 1).

Following are matrix for two-dimensional transformation in homogeneous coordinate:

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

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

3. Rotation (clockwise) $\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

4. Rotation (anti-clock) $\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

5. Reflection against X axis $\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

6. Reflection against Y axis $\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

7. Reflection against origin $\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

8. Reflection against line Y=X $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

9. Reflection against Y= –X $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

10. Shearing in X direction $\begin{bmatrix} 1 & 0 & 0 \\ Sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

11. Shearing in Y direction $\begin{bmatrix} 1 & Sh_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

12. Shearing in both x and y direction $\begin{bmatrix} 1 & Sh_y & 0 \\ Sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Composite Transformation:

A number of transformations or sequence of transformations can be combined into single one called as composition. The resulting matrix is called as composite matrix. The process of combining is called as concatenation.

Suppose we want to perform rotation about an arbitrary point, then we can perform it by the sequence of three transformations

1. Translation
2. Rotation
3. Reverse Translation

The ordering sequence of these numbers of transformations must not be changed. If a matrix is represented in column form, then the composite transformation is performed by multiplying matrix in order from right to left side. The output obtained from the previous matrix is multiplied with the new coming matrix.

Example showing composite transformations:

The enlargement is with respect to center. For this following sequence of transformations will be performed and all will be combined to a single one

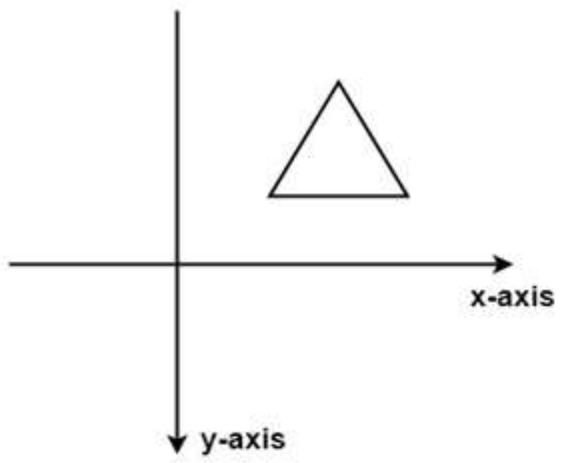
Step1: The object is kept at its position as in fig (a)

Step2: The object is translated so that its center coincides with the origin as in fig (b)

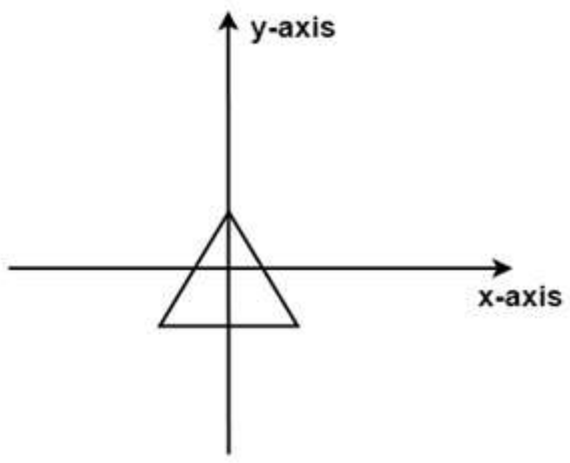
Step3: Scaling of an object by keeping the object at origin is done in fig (c)

Step4: Again translation is done. This second translation is called a reverse translation. It will position the object at the origin location.

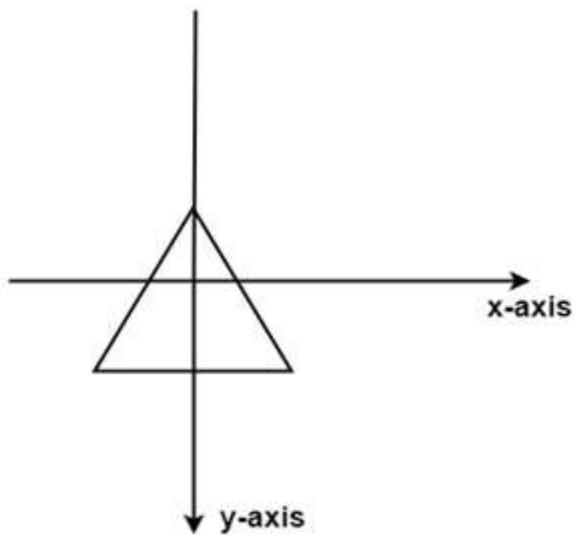
Above transformation can be represented as $T_V.ST_V^{-1}$



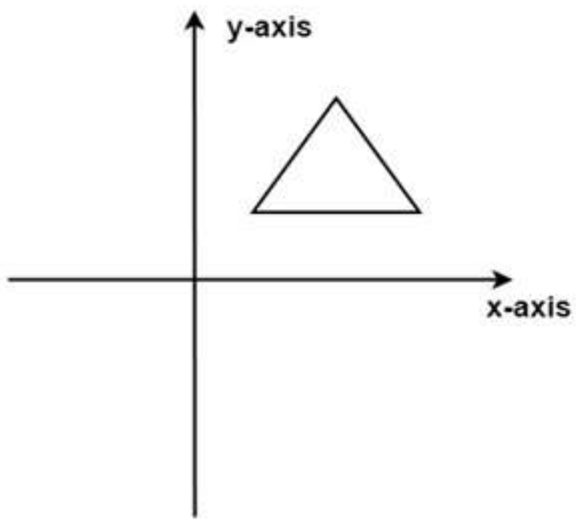
(original position of object)
Fig:(a)



(object is translated to origin)
Fig:(b)



(object is enlarged)
Fig:(c)



(object is retranslated to original position)
Fig:(d)

Row Method

1. Translation

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

2. Scaling

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

Column Method

1.Translation

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

2. Scaling

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

Advantage of composition or concatenation of matrix:

1. It transformations become compact.
2. The number of operations will be reduced.
3. Rules used for defining transformation in form of equations are complex as compared to matrix.

Composition of two translations:

Let t_1 t_2 t_3 $t_4$are translation vectors. They are two translations P_1 and P_2. The matrix of P_1 and P_2 given below. The P_1 and $P_2$are represented using Homogeneous matrices and P will be the final transformation matrix obtained after multiplication.

$$P_1 = \begin{bmatrix} 1 & 0 & t_1 \\ 0 & 1 & t_2 \\ 0 & 0 & 1 \end{bmatrix} \qquad P_2 = \begin{bmatrix} 1 & 0 & t_3 \\ 0 & 1 & t_4 \\ 0 & 0 & 1 \end{bmatrix}$$

$$P = \begin{bmatrix} 1 & 0 & t_1 + t_2 \\ 0 & 1 & t_3 + t_4 \\ 0 & 0 & 1 \end{bmatrix}$$

Above resultant matrix show that two successive translations are additive.

Composition of two Rotations: Two Rotations are also additive

Composition of two Scaling: The composition of two scaling is multiplicative. Let S_{11} and $S_{12}$are matrix to be multiplied.

$$S_{11} = \begin{bmatrix} S_1 & 0 & 0 \\ 0 & S_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

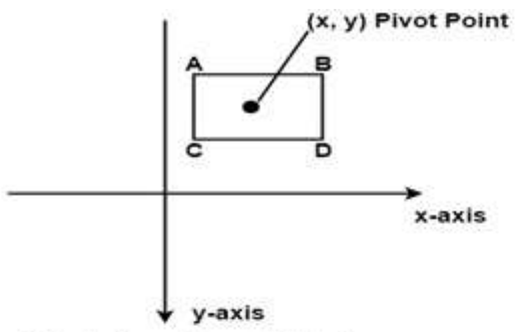
$$S_{12} = \begin{bmatrix} S_3 & 0 & 0 \\ 0 & S_4 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S = S_{11} * S_{12} = S_{11}(S_1.S_2).S_{12}(S_3.S_4) = S(S_1.S_2.S_3.S_4)$$

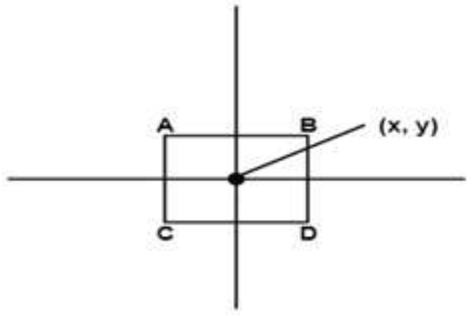
General Pivot Point Rotation or Rotation about Fixed Point:

For it first of all rotate function is used. Sequences of steps are given below for rotating an object about origin.

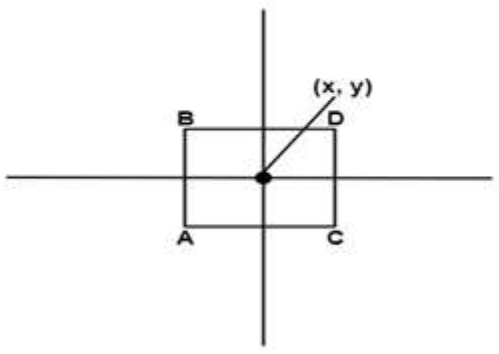
1. Translate object to origin from its original position as shown in fig (b)
2. Rotate the object about the origin as shown in fig (c).
3. Translate the object to its original position from origin. It is called as reverse translation as shown in fig (d).



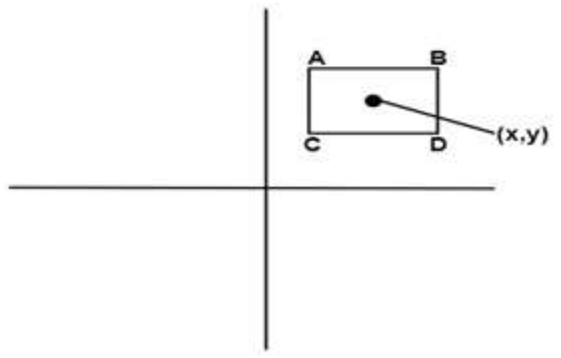
(Original position of object)
Fig (a)



(Translation of object at pivot point)
Fig (b)



(Rotation of object about origin)
Fig (c)



(Retranslation of object to its original position)
Fig (d)

The matrix multiplication of above 3 steps is given below

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

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

Scaling relative to fixed point:

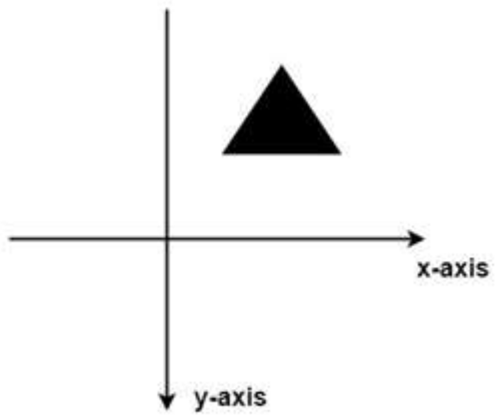
For this following steps are performed:

Step1: The object is kept at desired location as shown in fig (a)

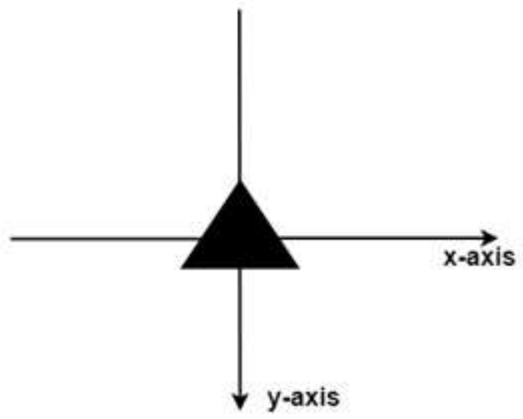
Step2: The object is translated so that its center coincides with origin as shown in fig (b)

Step3: Scaling of object by keeping object at origin is done as shown in fig (c)

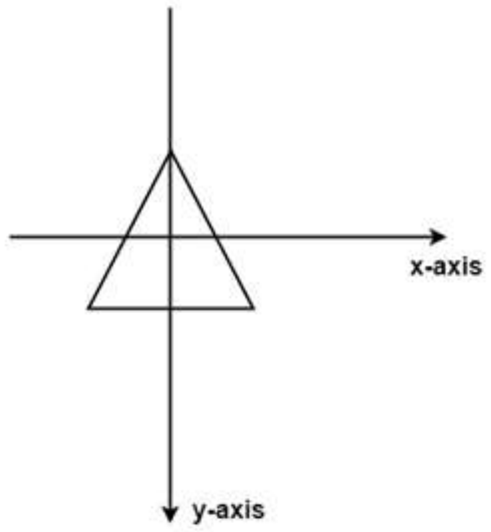
Step4: Again translation is done. This translation is called as reverse translation.



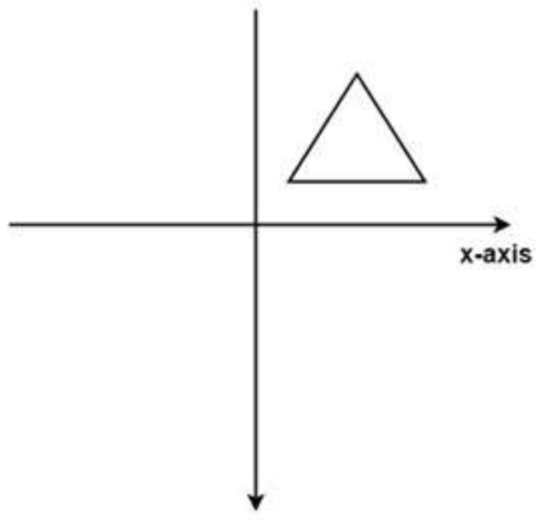
(Original position of object)
Fig (a)



(Object is translated to origin)
Fig (b)



(Object is enlarged by process of scaling)
Fig (c)



(Object is retranslated to original position)
Fig (d)

Computer Graphics Window:

The method of selecting and enlarging a portion of a drawing is called windowing. The area chosen for this display is called a window. The window is selected by world-coordinate.

Sometimes we are interested in some portion of the object and not in full object. So we will decide on an imaginary box. This box will enclose desired or interested area of the object. Such an imaginary box is called a window.

Viewport: An area on display device to which a window is mapped [where it is to displayed].

Basically, the window is an area in object space. It encloses the object. After the user selects this, space is mapped on the whole area of the viewport. Almost all 2D and 3D graphics packages provide means of defining viewport size on the screen. It is possible to determine many viewports on different areas of display and view the same object in a different angle in each viewport.

The size of the window is (0, 0) coordinate which is a bottom-left corner and toward right side until window encloses the desired area. Once the window is defined data outside the window is clipped before representing to screen coordinates. This process reduces the amount of data displaying signals.

The window size of the Tektronix 4.14 tube in Imperial College contains 4.96 points horizontally and 3072 points vertically.

Viewing transformation or window to viewport transformation or windowing transformation: The mapping of a part of a world-coordinate scene to device coordinates is referred to as a viewing transformation etc.

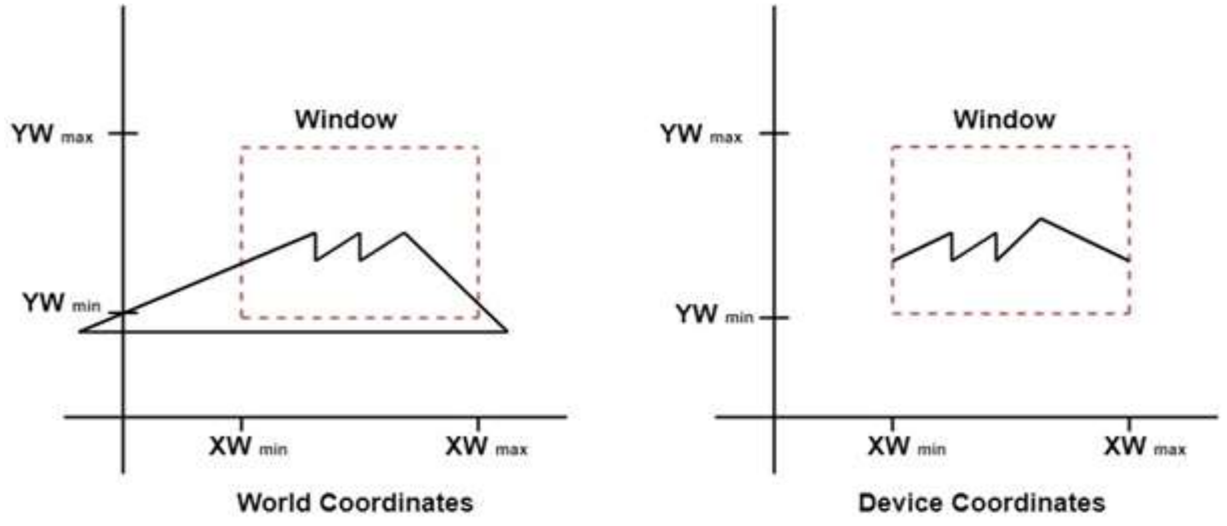


Fig: A viewing transformation using standard rectangles for the window and viewport.

Viewing transformation in several steps:

First, we construct the scene in world coordinate using the output primitives and attributes.

To obtain a particular orientation, we can set up a 2-D viewing coordinate system in the window coordinate plane and define a window in viewing coordinates system.

Once the viewing frame is established, are then transform description in world coordinates to viewing coordinates.

Then, we define viewport in normalized coordinates (range from 0 to 1) and map the viewing coordinates description of the scene to normalized coordinates.

At the final step, all parts of the picture that (i.e., outside the viewport are dipped, and the contents are transferred to device coordinates).

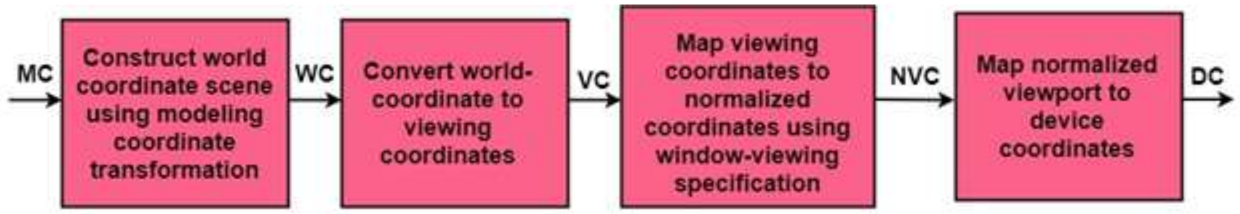

MC
Construct world coordinate scene using modeling coordinate transformation
WC
Convert world-coordinate to viewing coordinates
VC
Map viewing coordinates to normalized coordinates using window-viewing specification
NVC
Map normalized viewport to device coordinates
DC

Fig: The two-dimensional viewing-transformation.

By changing the position of the viewport: We can view objects at different locations on the display area of an output device as shown in fig:

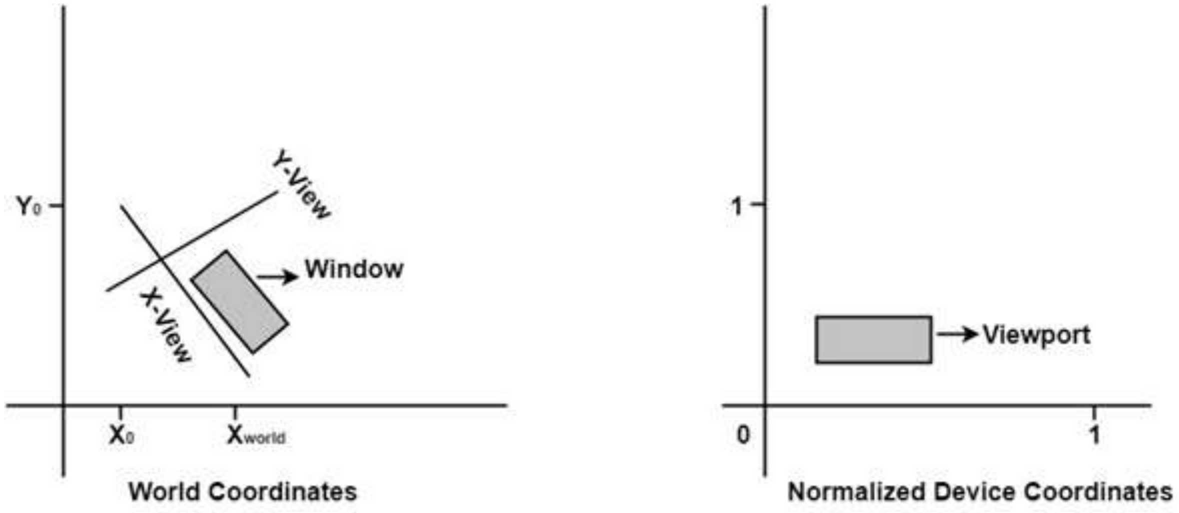


Fig:Setting up a rotated world window and corresponding normalized coordinate viewport.

By varying the size of viewports: We can change the size and proportions of displayed objects. We can achieve zooming effects by successively mapping different-sized windows on a fixed-size viewport.

As the windows are made smaller, we zoom in on some part of a scene to view details that are not shown with larger windows.

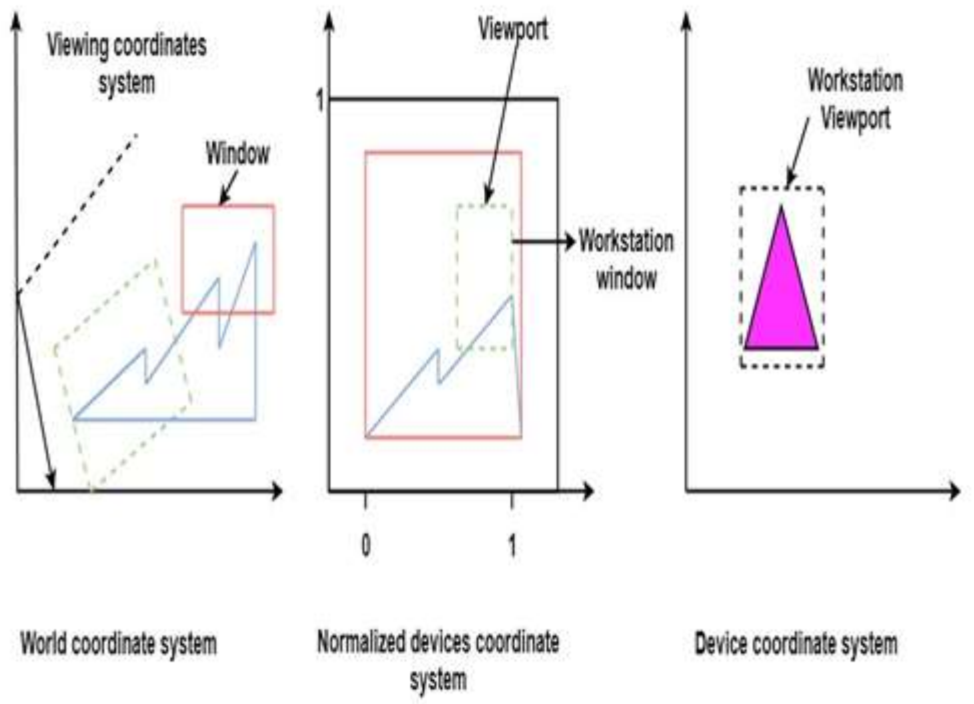


Fig: Zooming effects by mapping different-sized windows on a fixed-size viewport.

Computer Graphics Window to Viewport Co-ordinate Transformation

Once object description has been transmitted to the viewing reference frame, we choose the window extends in viewing coordinates and selects the viewport limits in normalized coordinates.

Object descriptions are then transferred to normalized device coordinates:

We do this thing using a transformation that maintains the same relative placement of an object in normalized space as they had in viewing coordinates.

If a coordinate position is at the center of the viewing window:

It will display at the center of the viewport.

Fig shows the window to viewport mapping. A point at position (xw, yw) in window mapped into position (xv, yv) in the associated viewport.

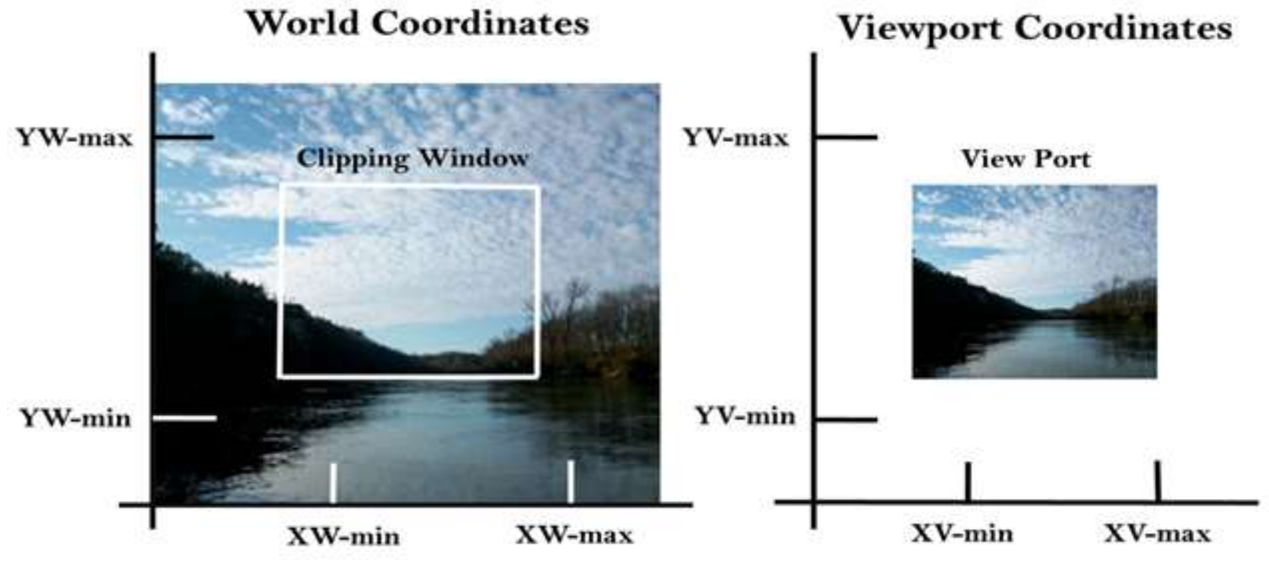


Fig: Window to viewport mapping

In order to maintain the same relative placement of the point in the viewport as in the window, we require:

$$\frac{xv - xv_{min}}{xv_{max} - xv_{min}} = \frac{xw - xw_{min}}{xw_{max} - xw_{min}} \text{ ....equation 1}$$

$$\frac{yv - yv_{min}}{yv_{max} - yv_{min}} = \frac{yw - yw_{min}}{yw_{max} - yw_{min}}$$

Solving these impressions for the viewport position (xv, yv), we have

$xv = xv_{min} + (xw - xw_{min})sx$
$yv = yv_{min} + (yw - yw_{min})sy$ ....equation 2

Where scaling factors are

$$sx = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}}$$

$$sy = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}}$$

Equation (1) and Equation (2) can also be derived with a set of transformation that converts the window or world coordinate area into the viewport or screen coordinate area. This conversation is performed with the following sequence of transformations:

1. Perform a scaling transformation using a fixed point position (xw_{min},yw_{min}) that scales the window area to the size of the viewport.
2. Translate the scaled window area to the position of the viewport. Relative proportions of objects are maintained if the scaling factors are the same (sx=sy).

From normalized coordinates, object descriptions are mapped to the various display devices.

Any number of output devices can we open in a particular app, and three windows to viewport transformation can be performed for each open output device.

This mapping called workstation transformation (It is accomplished by selecting a window area in normalized space and a viewport area in the coordinates of the display device).

As in fig, workstation transformation to partition a view so that different parts of normalized space can be displayed on various output devices).

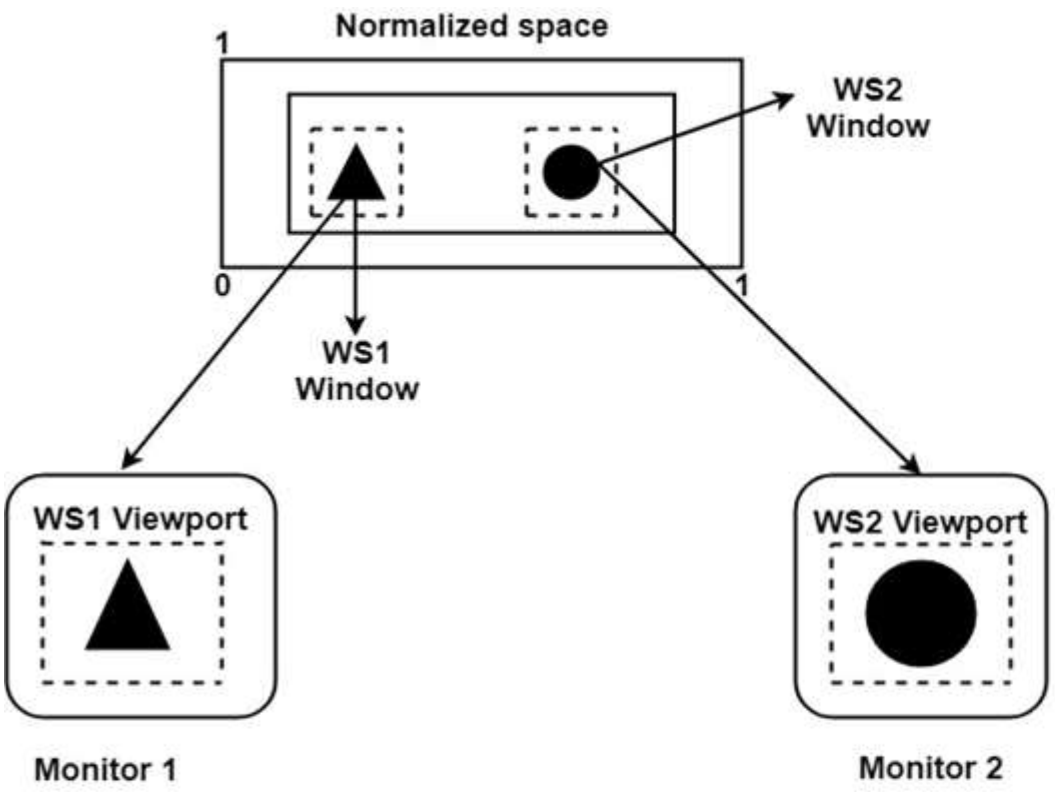
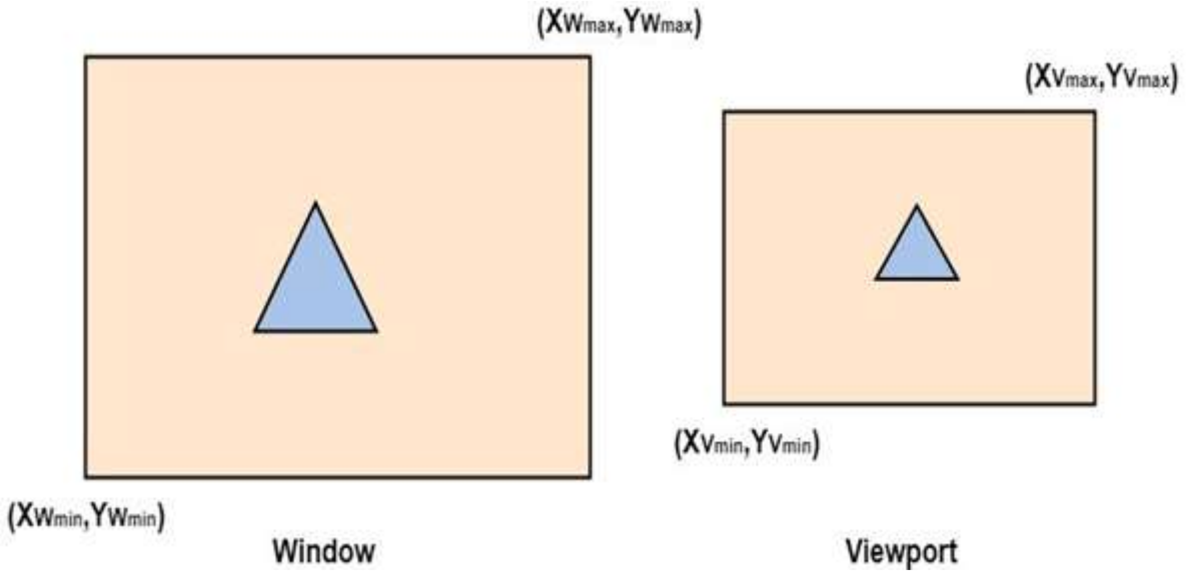


Fig:Mapping selected parts of a scene in normalized coordinates to different video monitors with workstation transformation.

Matrix Representation of the above three steps of Transformation:



Step1: Translate window to origin 1

$T_x=-Xw_{min}$ $T_y=-Yw_{min}$

Step2: Scaling of the window to match its size to the viewport

$S_x=(Xy_{max}-Xv_{min})/(Xw_{max}-Xw_{min})$

$S_y=(Yv_{max}-Yv_{min})/(Yw_{max}-Yw_{min})$

Step3: Again translate viewport to its correct position on screen.

$T_x=Xv_{min}$

$T_y=Yv_{min}$

Above three steps can be represented in matrix form:

$VT=T * S * T_1$

T = Translate window to the origin

S=Scaling of the window to viewport size

T_1=Translating viewport on screen.

$$T = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -Xw_{min} & -Yw_{min} & 1 \end{vmatrix}$$

$$S = \begin{vmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

$$T_1 = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Xv_{min} & Yv_{min} & 1 \end{vmatrix}$$

Viewing Transformation= $T * S * T_1$

Advantage of Viewing Transformation:

We can display picture at device or display system according to our need and choice.

Note:

- World coordinate system is selected suits according to the application program.
- Screen coordinate system is chosen according to the need of design.
- Viewing transformation is selected as a bridge between the world and screen coordinate.

Computer Graphics Zooming

Zooming is a transformation often provided with an imaginary software. The transformation effectively scales down or blows up a pixel map or a portion of it with the instructions from the user. Such scaling is commonly implemented at the pixel level rather than at the coordinates level. A video display or an image is necessarily a pixel map, i.e., a collection of pixels which are the smallest addressable elements of a picture. The process of zooming replicates pixels along successive scan lines.

Example: for a zoom factor of two

Each pixel value is used four times twice on each of the two successive scan lines.

Figure shows the effect of zooming by a factor of 2.

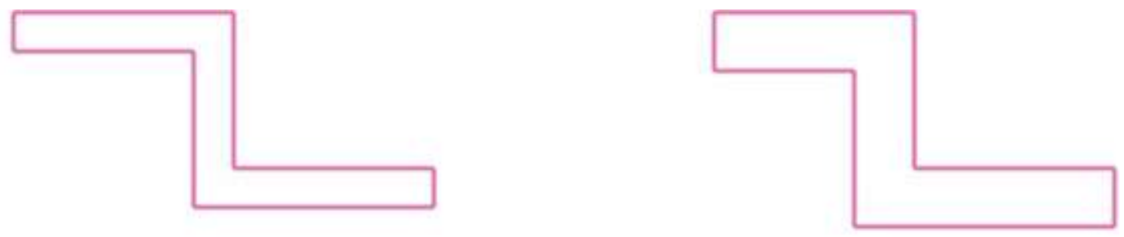


Fig:Effect of zooming by a factor of two.

Such integration of pixels sometimes involves replication using a set of ordered patterns, commonly known as Dithering.

The two most common dither types are:

- Ordered dither.
- Random dither.

There are widely used, especially when the grey levels (share of brightness) are synthetically generated.

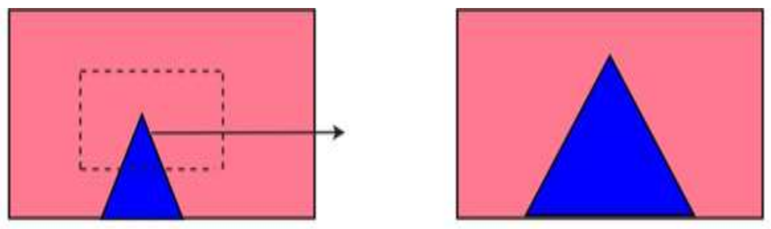


Fig:The zooming of the selected portion of an image (In the case of panning,the blown up part is shifted to the center of the screen)

Clipping:

When we have to display a large portion of the picture, then not only scaling & translation is necessary, the visible part of picture is also identified. This process is not easy. Certain parts of the image are inside, while others are partially inside. The lines or elements which are partially visible will be omitted.

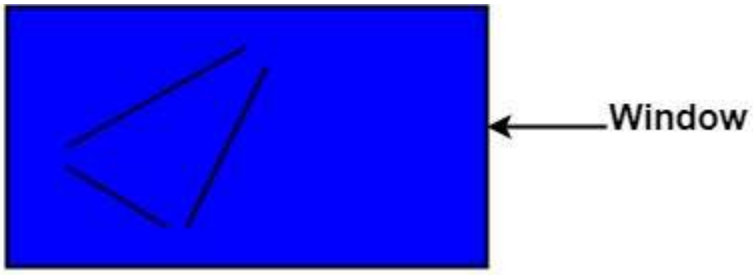
For deciding the visible and invisible portion, a particular process called clipping is used. Clipping determines each element into the visible and invisible portion. Visible portion is selected. An invisible portion is discarded.

Types of Lines:

Lines are of three types:

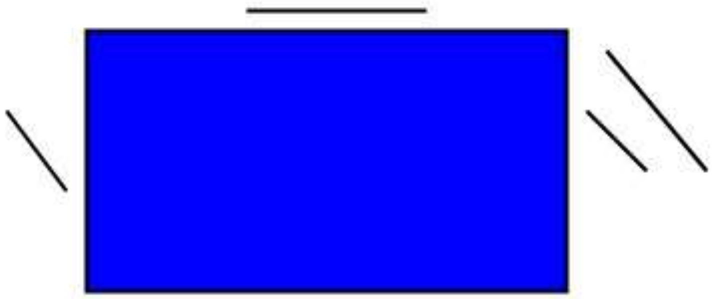
1. **Visible:** A line or lines entirely inside the window is considered visible
2. **Invisible:** A line entirely outside the window is considered invisible
3. **Clipped:** A line partially inside the window and partially outside is clipped. For clipping point of intersection of a line with the window is determined.

Case1:



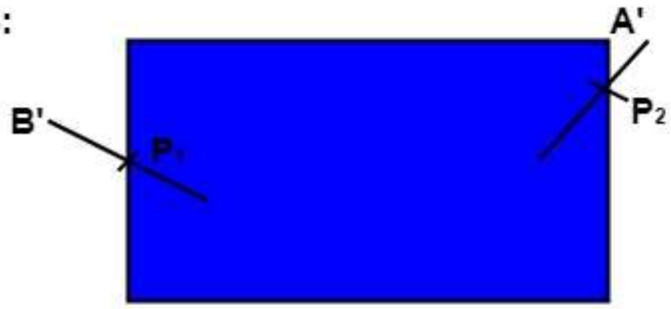
Visible Lines

Case2:



Invisible Lines

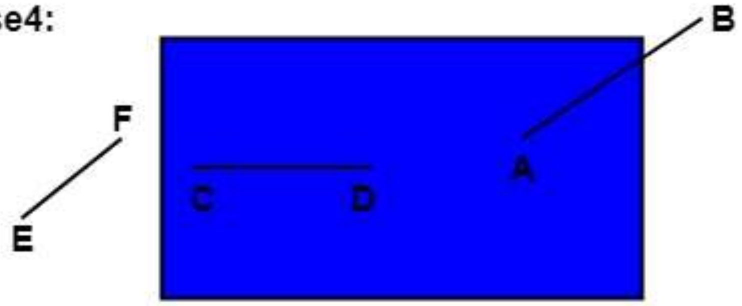
Case3:



In this figure P_1 and P_2 are point of intersection. The line P_2 to A' and P to B' is discarded or clipped.

Clipped Lines

Case4:

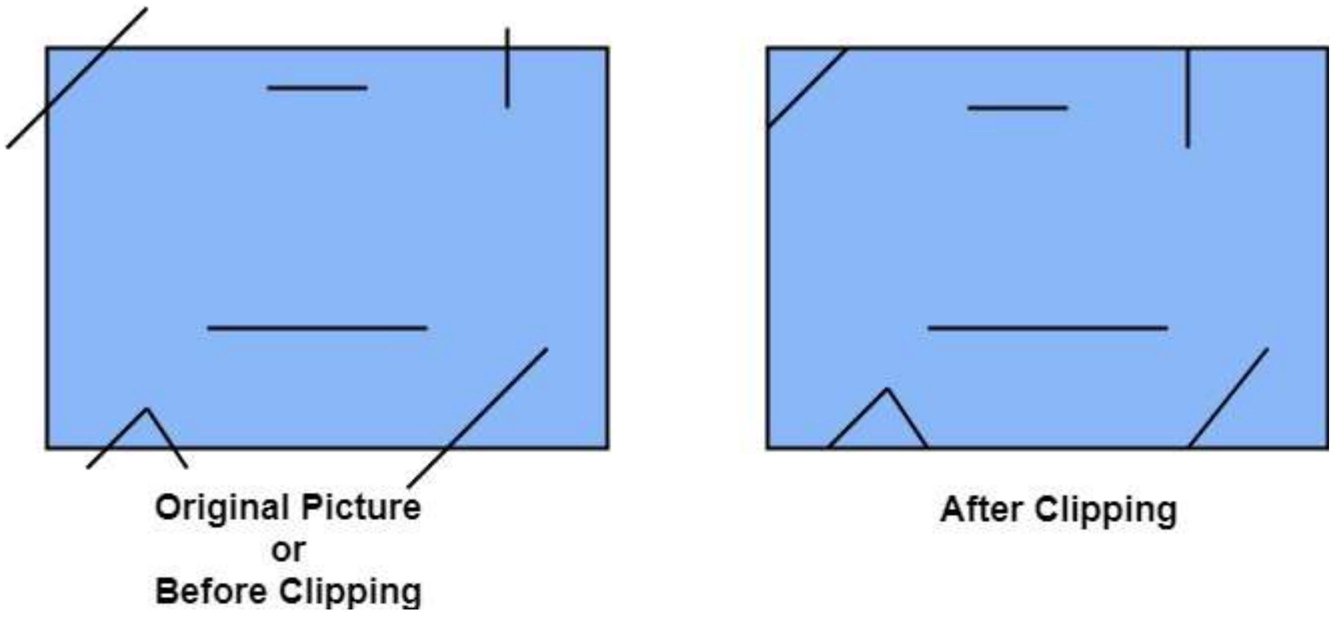


In this figure AB is clipped case
CD is visible line.
EF is invisible line.

Mixed Lines

Clipping can be applied through hardware as well as software. In some computers, hardware devices automatically do work of clipping. In a system where hardware clipping is not available software clipping applied.

Following figure show before and after clipping



The window against which object is clipped called a clip window. It can be curved or rectangle in shape.

Applications of clipping:

1. It will extract part we desire.
2. For identifying the visible and invisible area in the 3D object.
3. For creating objects using solid modeling.
4. For drawing operations.
5. Operations related to the pointing of an object.
6. For deleting, copying, moving part of an object.

Clipping can be applied to world co-ordinates. The contents inside the window will be mapped to device co-ordinates. Another alternative is a complete world co-ordinates picture is assigned to device co-ordinates, and then clipping of viewport boundaries is done.

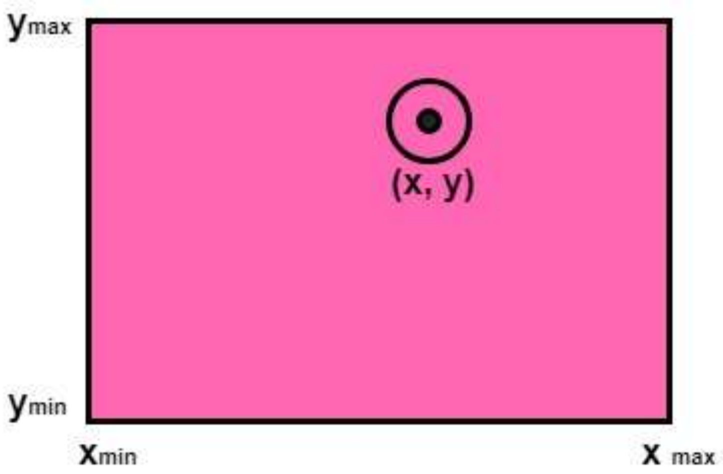
Types of Clipping:

1. Point Clipping
2. Line Clipping
3. Area Clipping (Polygon)
4. Curve Clipping
5. Text Clipping
6. Exterior Clipping

Point Clipping:

Point Clipping is used to determining, whether the point is inside the window or not. For this following conditions are checked.

1. $x \leq x_{max}$
2. $x \geq x_{min}$
3. $y \leq y_{max}$
4. $y \geq y_{min}$



The (x, y) is coordinate of the point. If anyone from the above inequalities is false, then the point will fall outside the window and will not be considered to be visible.

Line Clipping:

It is performed by using the line clipping algorithm. The line clipping algorithms are:

1. Cohen Sutherland Line Clipping Algorithm
2. Midpoint Subdivision Line Clipping Algorithm
3. Liang-Barsky Line Clipping Algorithm

Cohen Sutherland Line Clipping Algorithm:

In the algorithm, first of all, it is detected whether line lies inside the screen or it is outside the screen. All lines come under any one of the following categories:

1. Visible
2. Not Visible
3. Clipping Case

1. Visible: If a line lies within the window, i.e., both endpoints of the line lies within the window. A line is visible and will be displayed as it is.

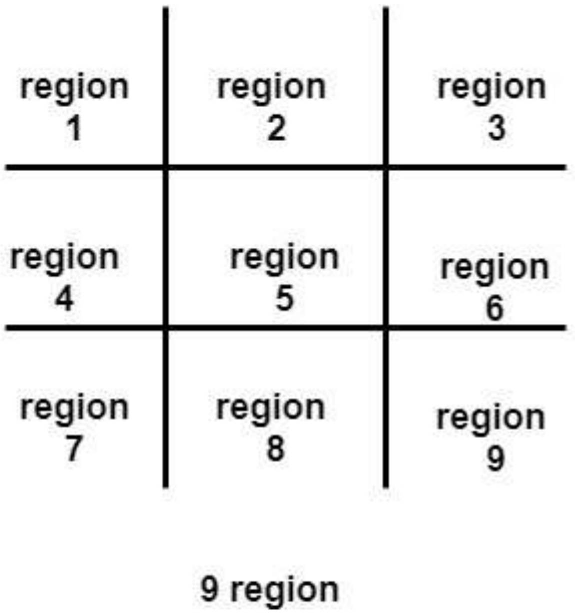
2. Not Visible: If a line lies outside the window it will be invisible and rejected. Such lines will not display. If any one of the following inequalities is satisfied, then the line is considered invisible. Let A (x_1,y_2) and B (x_2,y_2) are endpoints of line.

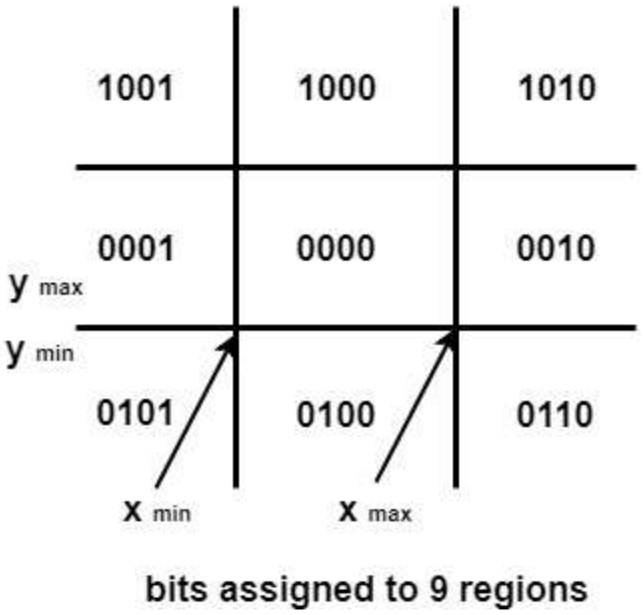
x_{min},x_{max} are coordinates of the window.

y_{min},y_{max} are also coordinates of the window.

$x_1>x_{max}$
$x_2>x_{max}$
$y_1>y_{max}$
$y_2>y_{max}$
$x_1<x_{min}$
$x_2<x_{min}$
$y_1<y_{min}$
$y_2<y_{min}$

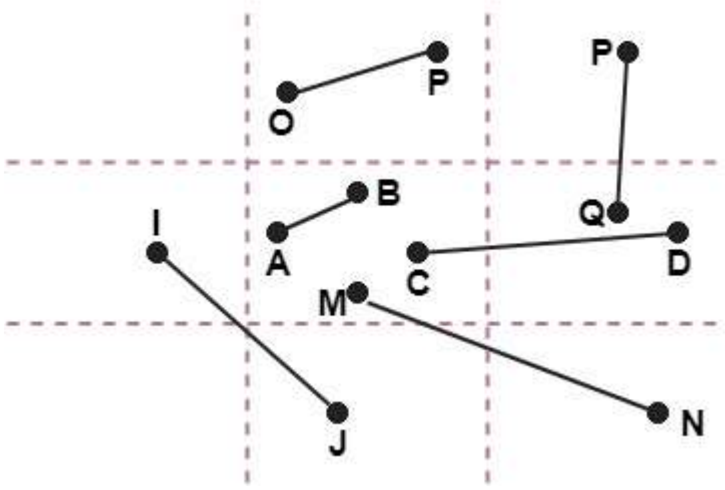
3. Clipping Case: If the line is neither visible case nor invisible case. It is considered to be clipped case. First of all, the category of a line is found based on nine regions given below. All nine regions are assigned codes. Each code is of 4 bits. If both endpoints of the line have end bits zero, then the line is considered to be visible.





The center area is having the code, 0000, i.e., region 5 is considered a rectangle window.

Following figure show lines of various types



Line AB is the visible case
Line OP is an invisible case
Line PQ is an invisible line
Line IJ are clipping candidates
Line MN are clipping candidate
Line CD are clipping candidate

Advantage of Cohen Sutherland Line Clipping:

1. It calculates end-points very quickly and rejects and accepts lines quickly.
2. It can clip pictures much large than screen size.

Algorithm of Cohen Sutherland Line Clipping:

Step1: Calculate positions of both endpoints of the line

Step2: Perform OR operation on both of these end-points

```
Step3:If the OR operation gives 0000
       Then
                line is considered to be visible
       else
          Perform AND operation on both endpoints
      If And ≠ 0000
          then the line is invisible
        else
      And=0000
    Line is considered the clipped case.
```

Step4: If a line is clipped case, find an intersection with boundaries of the window
$m=(y_2-y_1)(x_2-x_1)$

(a) If bit 1 is "1" line intersects with left boundary of rectangle window
$y_3=y_1+m(x-X_1)$
where $X = X_{wmin}$
where $X_{wmin}$is the minimum value of X co-ordinate of window

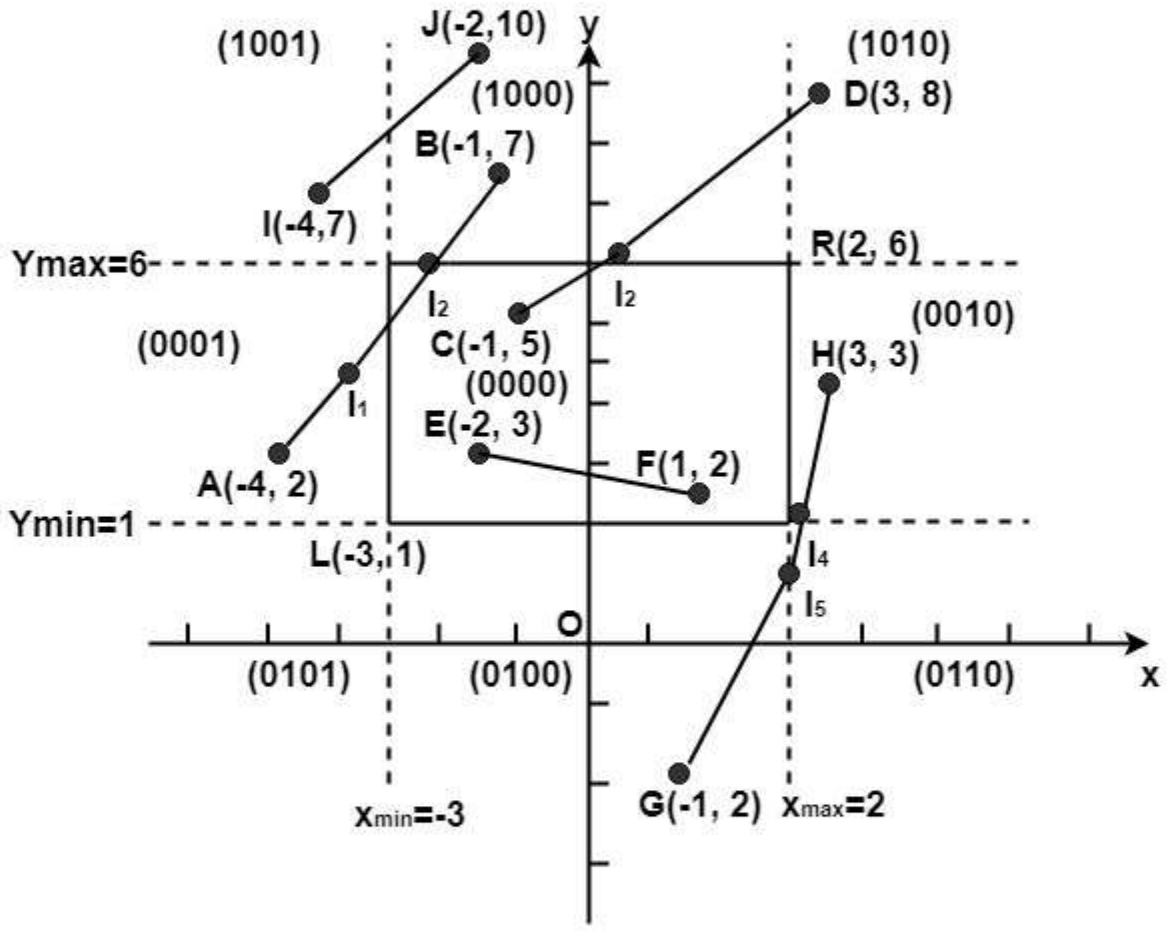
(b) If bit 2 is "1" line intersect with right boundary
$y_3=y_1+m(X-X_1)$
where $X = X_{wmax}$
where X more is maximum value of X co-ordinate of the window

(c) If bit 3 is "1" line intersects with bottom boundary
$X_3=X_1+(y-y_1)/m$
where $y = y_{wmin}$
y_{wmin} is the minimum value of Y co-ordinate of the window

(d) If bit 4 is "1" line intersects with the top boundary
$X_{3=X}1+(y-y_1)/m$
where $y = y_{wmax}$
y_{wmax} is the maximum value of Y co-ordinate of the window

Example of Cohen-Sutherland Line Clipping Algorithm:

Let R be the rectangular window whose lower left-hand corner is at L (-3, 1) and upper right-hand corner is at R (2, 6). Find the region codes for the endpoints in fig:



The region code for point (x, y) is set according to the scheme

Bit 1 = sign $(y-y_{max})$=sign (y-6)   Bit 3 = sign $(x-x_{max})$= sign (x-2)

Bit 2 = sign $(y_{min}-y)$=sign(1-y)   Bit 4 = sign $(x_{min}-x)$=sign(-3-x)

Here

$$\text{sign}(a) = \begin{cases} 1 & \text{if a is positive} \\ 0 & \text{otherwise} \end{cases}$$

So

A (-4, 2)→ 0001	F (1, 2)→ 0000
B (-1, 7) → 1000	G (1, -2) →0100
C (-1, 5)→ 0000	H (3, 3) → 0100
D (3, 8) → 1010	I (-4, 7) → 1001
E (-2, 3) → 0000	J (-2, 10) → 1000

We place the line segments in their appropriate categories by testing the region codes found in the problem.

Category1 (visible): $\boxed{EF}$ since the region code for both endpoints is 0000.

Category2 (not visible): $\boxed{IJ}$ since (1001) AND (1000) =1000 (which is not 0000).

Category 3 (candidate for clipping): $\boxed{AB}$ since (0001) AND (1000) = 0000, $\boxed{CD}$ since (0000) AND (1010) =0000, and $\boxed{GH.}$ since (0100) AND (0010) =0000.

The candidates for clipping are $\boxed{AB,}$ $\boxed{CD,}$ and $\boxed{GH.}$

In clipping $\boxed{AB}$, the code for A is 0001. To push the 1 to 0, we clip against the boundary line x_{min}=-3. The resulting intersection point is I_1 $(-3,3\frac{2}{3})$. We clip (do not display) $\boxed{AI}_1$ and I_1 B. The code for $I_1$is 1001. The clipping category for $\boxed{I}_1$ B is 3 since (0000) AND (1000) is (0000). Now B is outside the window (i.e., its code is 1000), so we push the 1 to a 0 by clipping against the line y_{max}=6. The resulting intersection is I_2 $(-1\frac{3}{5},6)$. Thus $\boxed{I_2\ B}$ is clipped. The code for I_2 is 0000. The remaining segment $\boxed{I_1\ I_2}$ is displayed since both endpoints lie in the window (i.e., their codes are 0000).

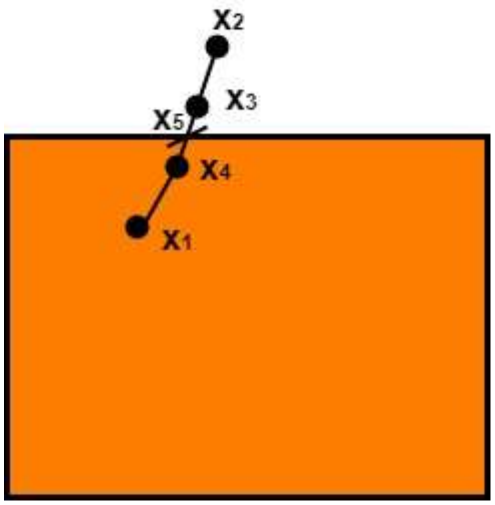
For clipping $\boxed{CD}$, we start with D since it is outside the window. Its code is 1010. We push the first 1 to a 0 by clipping against the line y_{max}=6. The resulting intersection I_3 is $(\frac{3}{5},6)$,and its code is 0000. Thus $\boxed{I_3\ D}$ is clipped and the remaining segment $\boxed{CI}_3$ has both endpoints coded 0000 and so it is displayed.

For clipping $\boxed{GH}$, we can start with either G or H since both are outside the window. The code for G is 0100, and we push the 1 to a 0 by clipping against the line y_{min}=1.The resulting intersection point is I_4 $(2\frac{1}{5},1)$ and its code is 0010. We clip $\boxed{GI}_4$ and work on $\boxed{I}_4$ H. Segment $\boxed{I}_4$ H is not displaying since (0010) AND (0010) =0010.

Mid Point Subdivision Line Clipping Algorithm:

It is used for clipping line. The line is divided in two parts. Mid points of line is obtained by dividing it in two short segments. Again division is done, by finding midpoint. This process is continued until line of visible and invisible category is obtained. Let (x_i,y_i) are midpoint

$$x_m = \frac{x_1+x_2}{2} \quad y_m = \frac{y_1+y_2}{2}$$



Step1: Find $\frac{x_2+x_1}{2}$ i. e. $x_3 = \frac{x_2+x_1}{2}$

Step2: Find $x_4 = \frac{x_3+x_1}{2}$

Step3: Find $x_5 = \frac{x_4+x_3}{2}$

$x_5$lie on point of intersection of boundary of window.

Advantage of midpoint subdivision Line Clipping:

It is suitable for machines in which multiplication and division operation is not possible. Because it can be performed by introducing clipping divides in hardware.

Algorithm of midpoint subdivision Line Clipping:

Step1: Calculate the position of both endpoints of the line

Step2: Perform OR operation on both of these endpoints

Step3: If the OR operation gives 0000

```
      then
            Line is guaranteed to be visible
    else
          Perform AND operation on both endpoints.
          If AND ≠ 0000
      then the line is invisible
  else
      AND=6000
      then the line is clipped case.
```

Step4: For the line to be clipped. Find midpoint

$X_m=(x_1+x_2)/2$

$Y_m=(y_1+y_2)/2$

$X_m$is midpoint of X coordinate.

$Y_m$is midpoint of Y coordinate.

Step5: Check each midpoint, whether it nearest to the boundary of a window or not.

Step6: If the line is totally visible or totally rejected not found then repeat step 1 to 5.

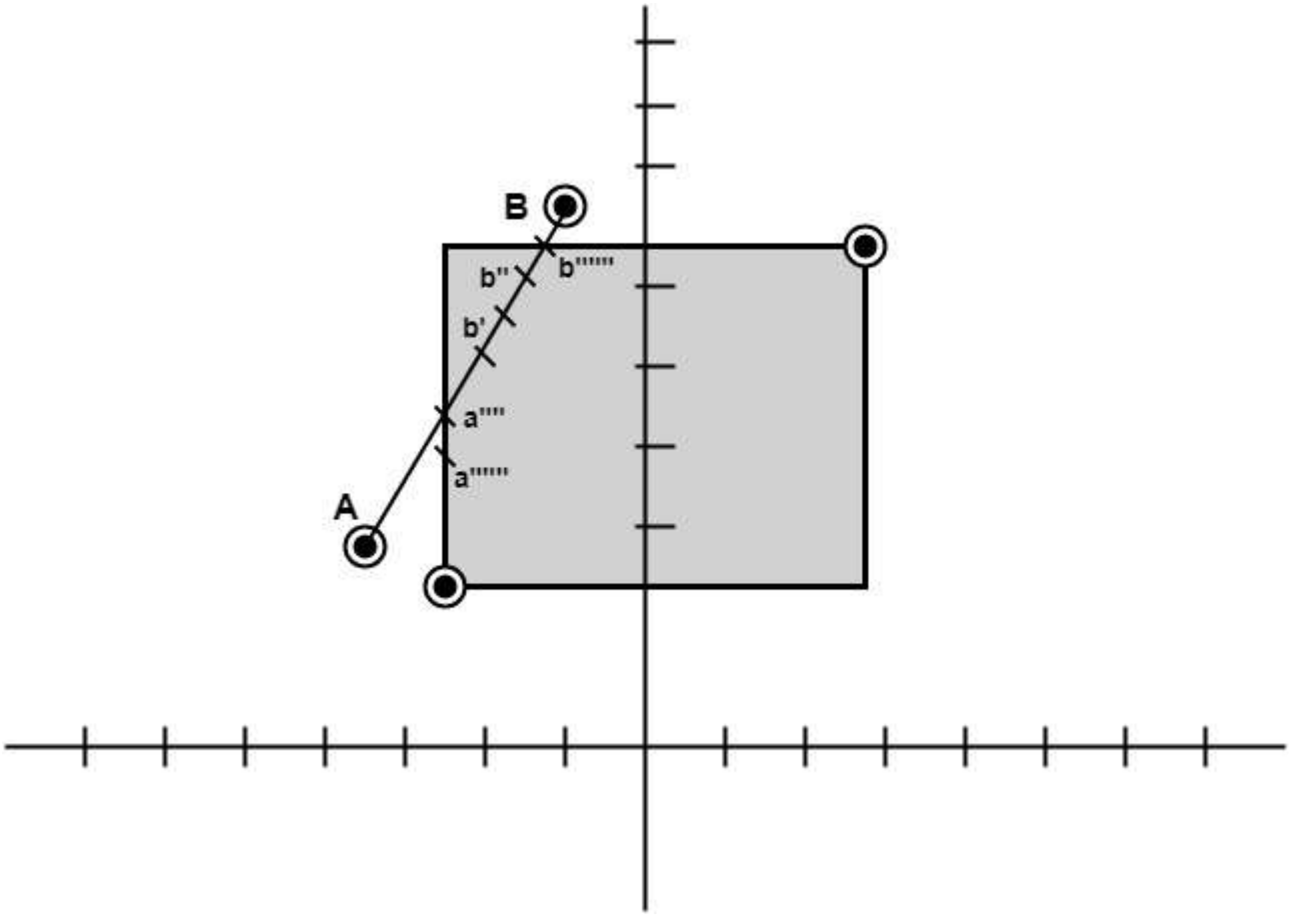
Step7: Stop algorithm.

Example: Window size is (-3, 1) to (2, 6). A line AB is given having co-ordinates of A (-4, 2) and B (-1, 7). Does this line visible. Find the visible portion of the line using midpoint subdivision?

Solution:

Step1: Fix point A (-4, 2)

$b = \left(\frac{-4+(-1)}{2}, \frac{2+7}{2}\right) = \frac{-5}{2}, \frac{9}{2} = (-2,4)$



Step2: Find b''=mid of b'and b

$b'' = \left(\frac{-2+(-1)}{2}, \frac{4+7}{2}\right)$

b''= (-1, 5)

So (-1, 5) is better than (2, 4)
Find b''&bb''(-1, 5) b (-1, 7)

$b'''' = \left(\frac{-1+(-1)}{2}, \frac{5+7}{2}\right)$

$b = (-1,6)$

So B''''to B length of line will be clipped from upper side

Now considered left-hand side portion.

A and B""are now endpoints

Find mid of A and B""

A (-4, 2) B ""(-1, 6)

$$a' = \left(\frac{-4+(-1)}{2}, \frac{2+6}{2}\right)$$

$$a' = (-2.5, 4)$$

$$a' = (-2, 4)$$

Now good a (-4, 2) and $a'(-2,4)$

$$a'' = \left(\frac{-4+(-2)}{2}, \frac{2+4}{2}\right)$$

$$a'' = (-3, 3)$$

Now find mid of a'' and a

a"""=a (-4, 2) and $a''(-3, 3)$

$$= \left(\frac{-4+(-3)}{2}, \frac{2+3}{2}\right)$$

$$= \left(\frac{-7}{2}, \frac{5}{2}\right)$$

$$= (-3.5, 2.5)$$

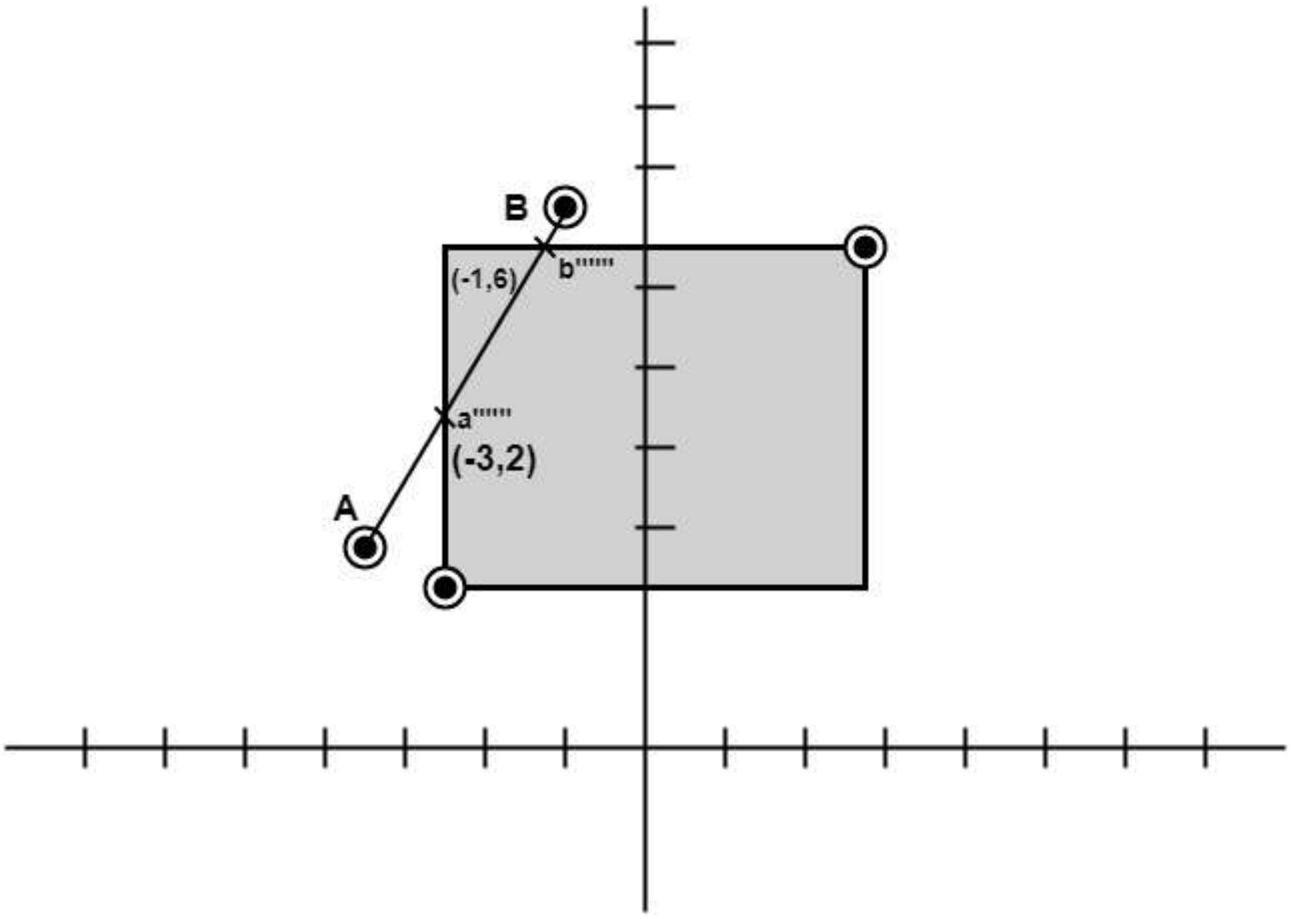
a"""= (-3, 2)

So line from A to a"""will clipped

Line after clipping from both sides will be a""to b""

a"""= (-1, 6)

b"""= (-3, 2)



Polygon Clipping:

Polygon clipping is applied to the polygons. The term polygon is used to define objects having outline of solid. These objects should maintain property and shape of polygon after clipping.

Sutherland-Hodgeman Polygon Clipping:

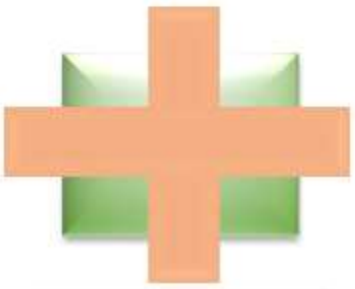
It is performed by processing the boundary of polygon against each window corner or edge. First of all entire polygon is clipped against one edge, then resulting polygon is considered, then the polygon is considered against the second edge, so on for all four edges.

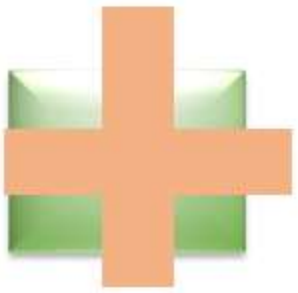
Four possible situations while processing

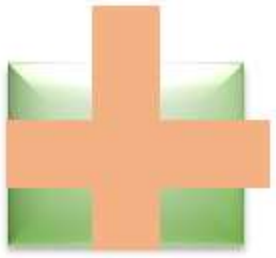
1. If the first vertex is an outside the window, the second vertex is inside the window. Then second vertex is added to the output list. The point of intersection of window boundary and polygon side (edge) is also added to the output line.

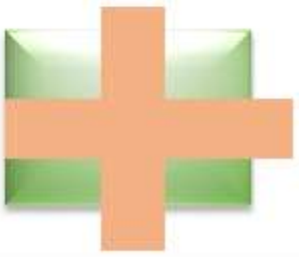
2. If both vertexes are inside window boundary. Then only second vertex is added to the output list.
3. If the first vertex is inside the window and second is an outside window. The edge which intersects with window is added to output list.
4. If both vertices are the outside window, then nothing is added to output list.

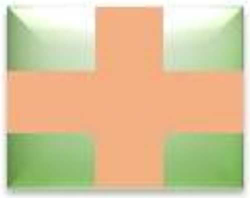
Following figures shows original polygon and clipping of polygon against four windows.

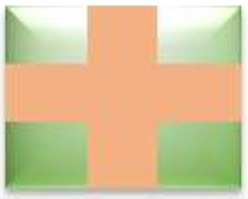

Original polygon


Clipping against left edge of window


Clipping against bottom corner of window


Clipping against Top of window


Clipping against right edge of window


Final polygon after Clipping

Disadvantage of Cohen Hodgmen Algorithm:

This method requires a considerable amount of memory. The first of all polygons are stored in original form. Then clipping against left edge done and output is stored. Then clipping against right edge done, then top edge. Finally, the bottom edge is clipped.

Results of all these operations are stored in memory. So wastage of memory for storing intermediate polygons.

Pointing and Positioning Techniques

Pointing technique refers to look at the items already on the screen whereas the positioning technique refers to position the item on the screen to a new position, i.e., the old current position. The user indicates a position on the screen with an input device, and this position is used to insert a symbol.

There are various pointing and positioning devices which are discussed below:

1. Light Pen
2. Mouse
3. Tablet
4. Joystick
5. Trackball and spaceball

1. Light Pen: It is a pointing device. When light pen is pointed at an item on the screen, it generates information from which the item can be identified by the program. It does not have any associated tracking hardware instead tracking is performed by software, making use of the output function of the display. All light pen programs depend on a rapid response from the pen when it is pointed at the screen fast response light pens can be build by using a highly sensitive photocell such as a photomultiplier tube.

Mouse: It is a positioning device which consists of a small plastic box resting on two metal wheels whose axes are at right angles. Each wheel of the mouse is linked to a shaft encoder that delivers an electrical pulse for every incremental rotation of the wheel. As the mouse is rolled around on a flat surface, its movement in two orthogonal directions is translated into rotation of the wheels. These rotations can be measured by counting the pulses received from the shaft encoders. The connected values may be held in registers accessible to the computer on written directly into the computer memory. It is simple and low cost, and there is no need to pick it up to use it. The mouse sits on the table surface. But the mouse cannot be used for tracing data from paper since a small rotation of the mouse will cause an error in all the reading and it is complicated handprint character for recognition by the computer.

3. Tablet: It is also a positioning device and is used to describe a flat surface separate from the display, on which the user draws with a stylus. There are two types of tablets:

1. **Acoustic Tablet:** It depends on the use of strip microphones which are mounted along two adjacent edges of the tablet. The styles have a small piece of ceramic mounted close to its tip, and at regular intervals, a small spark is generated

across the surface of the ceramic between two electrodes. The microphones pick up the pulse of sound produced by the spark and two counters record the delay between creating the spark and receiving the sound. These two delays are proportional to the stylus distance from the two edges of the tablet where the microphones are mounted.

2. **Electro-acoustic Tablet:** In this technique, the writing surface is a sheet of magnetostrictive material acting as a row of delay lines. An electric pulse travels through the sheet first horizontally and then vertically and is detected by a sensor in the stylus. A counter is used to determine the delay from the time the pulse is issued to the time it is detected; from this value, the position of the stylus can be determined. The electro-acoustic tablet is quieter in operation than its acoustic counterpart and is less affected by noise or air movement.

4. Joystick: A joystick consists of a small that is used to steer the screen cursor around. The distance that the stick is moved in any direction from its center position corresponds to the screen-cursor movement in that direction. Pressure sensitive joysticks have a non-moveable stick. Pressure on the stick is measured with strain gauges and converted to the movement of the cursor in the direction specified.

5. Trackball and spaceball: Trackball is a ball that can be rotated with the fingers to produces screen-cursor movement potentiometers, attached to the ball, measure the amount and direction of rotation. Trackballs are after mounted on keyboards, whereas space-ball provides six degrees of freedom. Spaceballs is used for three-dimensional positioning and selection operation in virtual reality system, modeling, animation, CAD and other applications.

Elastic or Rubber Band Techniques

Rubber banding is a popular technique of drawing geometric primitives such as line, polylines, rectangle, circle and ellipse on the computer screen.

It becomes an integral part and de facto standard with the graphical user interface (GUI) for drawing and is almost universally accepted by all windows based applications.

The user specifies the line in the usual way by positioning its two endpoints. As we move from the first endpoint to the second, the program displays a line from the first endpoint to the cursor position, thus he can see the lie of the line before he finishes positioning it.

The effect is of an elastic line stretched between the first endpoint and the cursor; hence the name for these techniques.

Consider the different linear structures in fig (a) and fig (d), depending on the position of the cross-hair cursor. The user may move the cursor to generate more possibilities and select the one which suits him for a specific application.

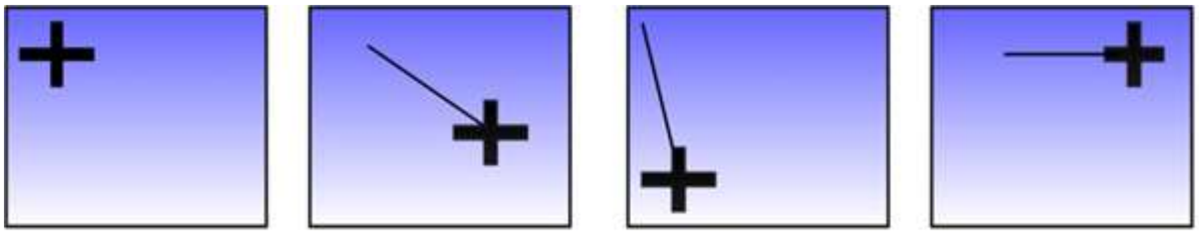


Fig:Demonstration of rubber banding: look at the cross-hair cursor(a) The start point of the line to be drawn is selected (b) 3 different lines are shown depending on the position of the cursor representing the end-point the user selects the desired line.

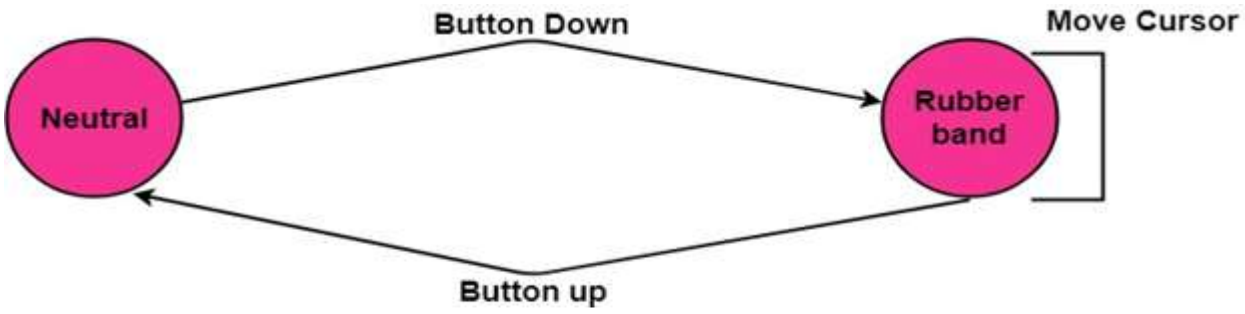
Selection of Terminal Point of the Line:

The user moves the cursor to the appropriate position and selects.

Then, as the cursor is moved, the line changes taking the latest positions of the cursors as the end-point.

As long as the button is held down, the state of the rubber band is active.

The process is explained with the state transition diagram of rubber banding in fig:



When the user is happy with the final position, the pressed button is released, and the line is drawn between the start and the last position of the cursor.

Example: This is widely followed in MS-Window based Applications like in the case of a paintbrush drawing package.

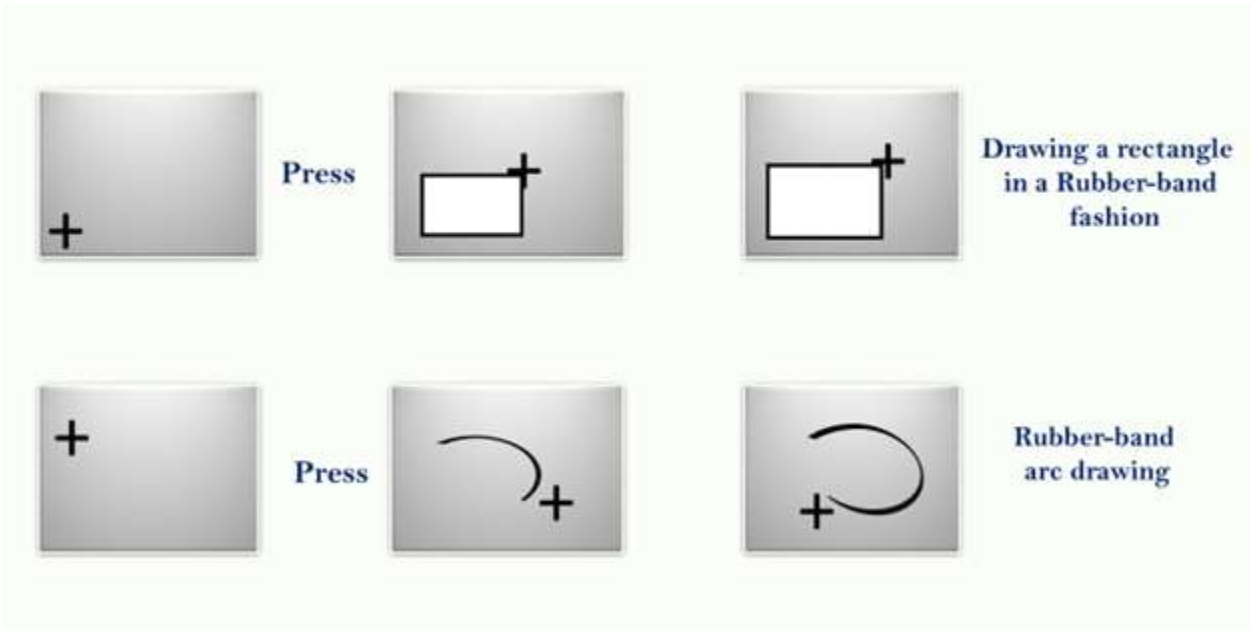
Other geometric entities can be drawn in a rubber-band fashion:

- Horizontally or vertically constructed lines

- Rectangles

- Arcs of circles

This technique is very helpful in drawing relatively complex entities such as rectangles and arcs.



Advantage:

1. It is used for drawing all geometric entities such as line, polygon, circle, rectangle, ellipse, and other curves.
2. It is easy to understand and implement.

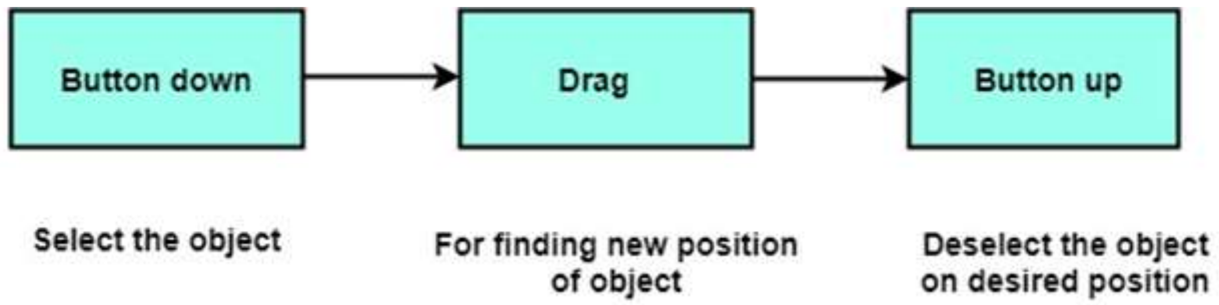
Disadvantage:

1. It requires computational resources like software and CPU speed.
2. Expensive

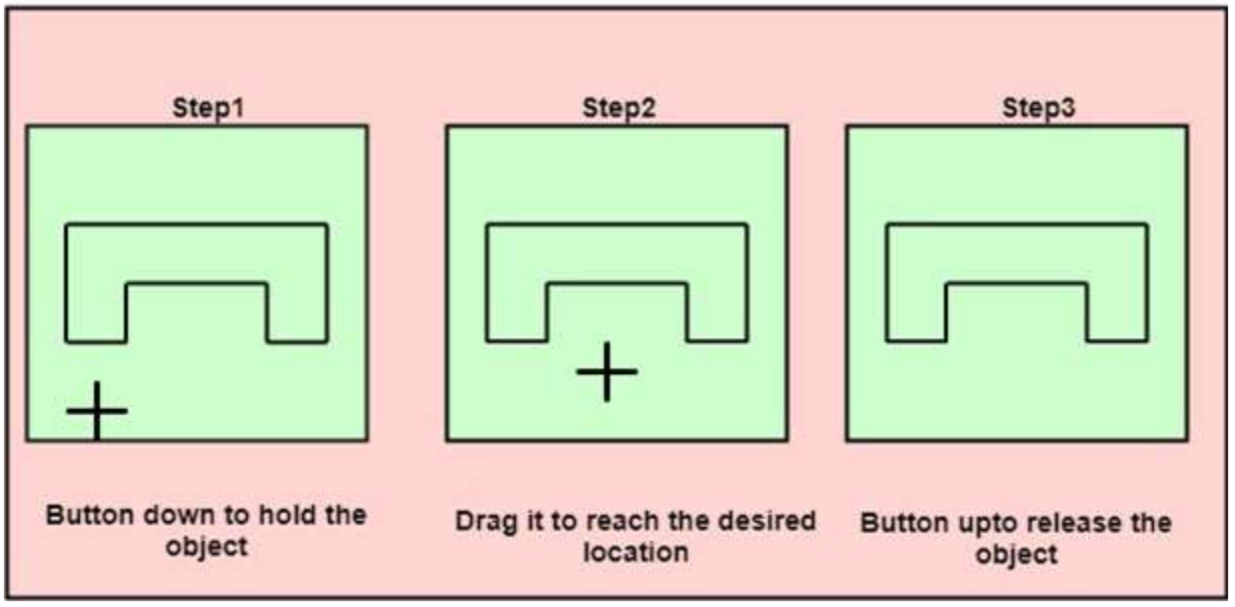
Dragging

Dragging is used to move an object from one position to another position on the computer screen. To drag any other object, first, we have to select the object that we want to move on the screen by holding the mouse button down. As cursor moved on the screen, the object is also moved with the cursor position. When the cursor reached the desired position, the button is released.

The following diagram represents the dragging procedure:


Button down
Drag
Button up
Select the object
For finding new position of object
Deselect the object on desired position

Basic Diagram of Dragging


Step1
Step2
Step3
Button down to hold the object
Drag it to reach the desired location
Button upto release the object

Inverse Transformations

These are also called as opposite transformations. If T is a translation matrix than inverse translation is representing using T^{-1}. The inverse matrix is achieved using the opposite sign.

Example1: Translation and its inverse matrix

Translation matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{pmatrix}$$

Inverse translation matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -T_x & -T_y & -T_z & 1 \end{pmatrix}$$

Example2: Rotation and its inverse matrix

$$\begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Inverse Rotation Matrix

$$\begin{pmatrix} -\cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & -\cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$