Computer Graphics Notes

Chapter 1 : Introduction

Computer Graphics

- Computer graphics is the science and art of communicating visually via a computer's display and its interaction devices. The visual aspect of the communication is usually in the computer-to-human direction, with the human-to-computer direction being mediated by devices like the mouse, keyboard, joystick, game controller, or touch-sensitive overlay.
- Computer graphics is a cross-disciplinary field in which physics, mathematics, human perception, human-computer interaction, engineering, graphic design, and art all play important roles.
- In the field of computer graphics, the word "model" can refer to a geometric model or a mathematical model. A **geometric model** is a model of something we plan to have appear in a picture: We make a model of a car, or a house, or an armadillo. The geometric model is enhanced with various other attributes that describe the color or texture or reflectance of the materials involved in the model. Starting from nothing and creating such a model is called **modeling**, and the geometric-plus-other-information description that is the result is called a **model.**
- A **mathematical** is a model of a physical or computational process.
- **Computer Graphics** are defined as any sketch or a drawing or a special network that pictorially represents some meaningful information.
- **Computer Graphics** is used where a set of images needs to be manipulated or the creation of the image in the form of pixels and is drawn on the computer.
- **Computer Graphics** can be used in digital photography, film, entertainment, electronic gadgets, and all other core technologies which are required. It is a vast subject and area in the field of computer science.
- **Computer Graphics** can be used in UI design, rendering, geometric objects, animation, and many more. In most areas, computer graphics is an abbreviation of **CG**.
- There are several tools used for the implementation of Computer Graphics. The basic is the <graphics.h> header file in Turbo-C, Unity for advanced and even OpenGL can be used for its Implementation.

- It was invented in 1960 by great researchers Verne Hudson and William Fetter from Boeing.

Computer Graphics refers to several things:

- The manipulation and the representation of the image or the data in a graphical manner.
- Various technology is required for the creation and manipulation.
- Digital synthesis and its manipulation.

Types of Computer Graphics

- **Raster Graphics:** In raster, graphics pixels are used for an image to be drawn. It is also known as a bitmap image in which a sequence of images is into smaller pixels. Basically, a bitmap indicates a large number of pixels together.
- **Vector Graphics:** In vector graphics, mathematical formulae are used to draw different types of shapes, lines, objects, and so on.

Elements

The **seven** basic elements of graphic design are line, shape, color, texture, type, space and image.

1. **Lines** are always more than just points that are strung together. Depending on their form, weight, length and context, lines can help organize information, define shapes, imply movement, and convey emotions.

   When it comes to selecting the appropriate lines for projects, designers have plenty of options. **Lines** can:
   - …be **horizontal**, vertical or diagonal.
   - …be **straight**, curved or freeform.
   - …**zigzag** or create other patterns.
   - …be **solid**, broken or implied.
2. **Shapes** are best understood as areas, forms or figures contained by a boundary or closed outline. There are **two** types of shapes that every graphic designer should understand: **geometric** and **organic** (or "free-flowing").
   - **Geometric shapes** can include either two-dimensional or three-dimensional forms. Geometric shapes can include

triangles, pyramids, squares, cubes, rectangles, pentagons, hexagons, octagons, decagons, circles, ellipses and spheres.
  - **Organic shapes** are far less uniform, proportional and well-defined. They can be symmetrical or asymmetrical. They might include natural shapes, such as leaves, crystals, and vines, or abstract shapes, such as blobs and squiggles.

3. **Color** can be a useful tool for communicating a mood or provoking an emotional response from your viewer. Color theory and the color wheel provide a practical guide for graphic designers who want to select a single color or combine multiple colors in a harmonious- or intentionally discordant-way.
    - **Primary colors (red, yellow and blue)** are defined as the pure-pigment colors from which all others are made. There is no way to mix any other color to get red, yellow or blue. But mix them together, and you create all kinds of shades.
    - **Secondary colors (violet, green and orange)** are the immediate results of mixing two primary colors: Red and yellow make orange; blue and red make purple; and yellow and blue make green.
    - **Tertiary colors (red-orange, yellow-orange, yellow-green, blue-green, blue-violet and red-violet)** are the six colors that result from mixing a primary color and a secondary color.
4. **Texture** is the feel of a surface—furry, smooth, rough, soft, gooey or glossy. Most graphic designers must visually convey texture by using illusions to suggest how their work might feel if viewers could touch it.
5. **Type :** Whether you're choosing a font or creating your own typography for a graphic design project, it's important to make sure the type you use is legible and appropriate for your subject. Type affects the overall mood of a design, so consider whether your letters should be print or script, and whether they should have angles that are sharp or rounded.
6. **Space :** Spacing is a vital part of any designer's toolkit. It can give a design breathing room, increase its visual impact, balance out heavier visual elements, and emphasize images or messages that viewers should remember. Without enough space, a design can risk becoming too visually cluttered for your audience to understand.
7. **Image :** Whether graphic designers use photographs or illustrations, they rely on images to grab the audience's attention and express specific

messages. An image works on multiple levels simultaneously: It provides context for a designer’s communication, adds necessary drama or action, and creates an overall mood.

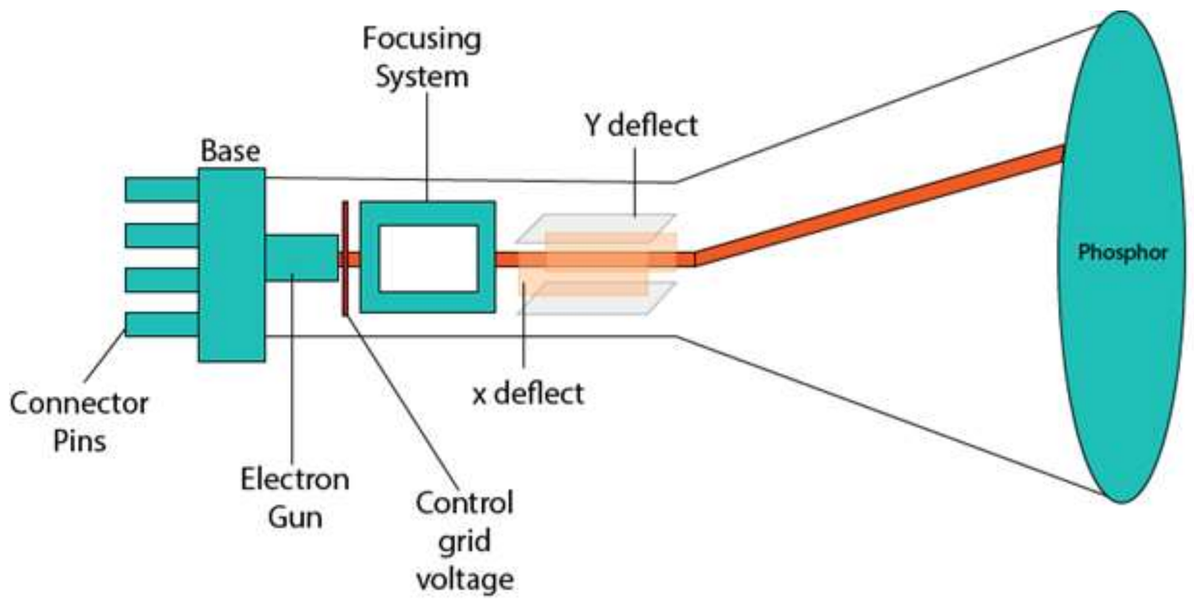
Applications

- **Computer Graphics are used for** an **aided design for engineering and architectural system-** These are used in electrical automobiles, electro-mechanical, mechanical, electronic devices. For example gears and bolts.
- **Computer Art –** MS Paint.
- **Presentation Graphics –** It is used to summarize financial statistical scientific or economic data. For example- Bar chart, Line chart.
- **Entertainment-** It is used in motion pictures, music videos, television gaming.
- **Education and training-** It is used to understand the operations of complex systems. It is also used for specialized system such for framing for captains, pilots and so on.
- **Visualization-** To study trends and patterns. For example- Analyzing satellite photo of earth.

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.

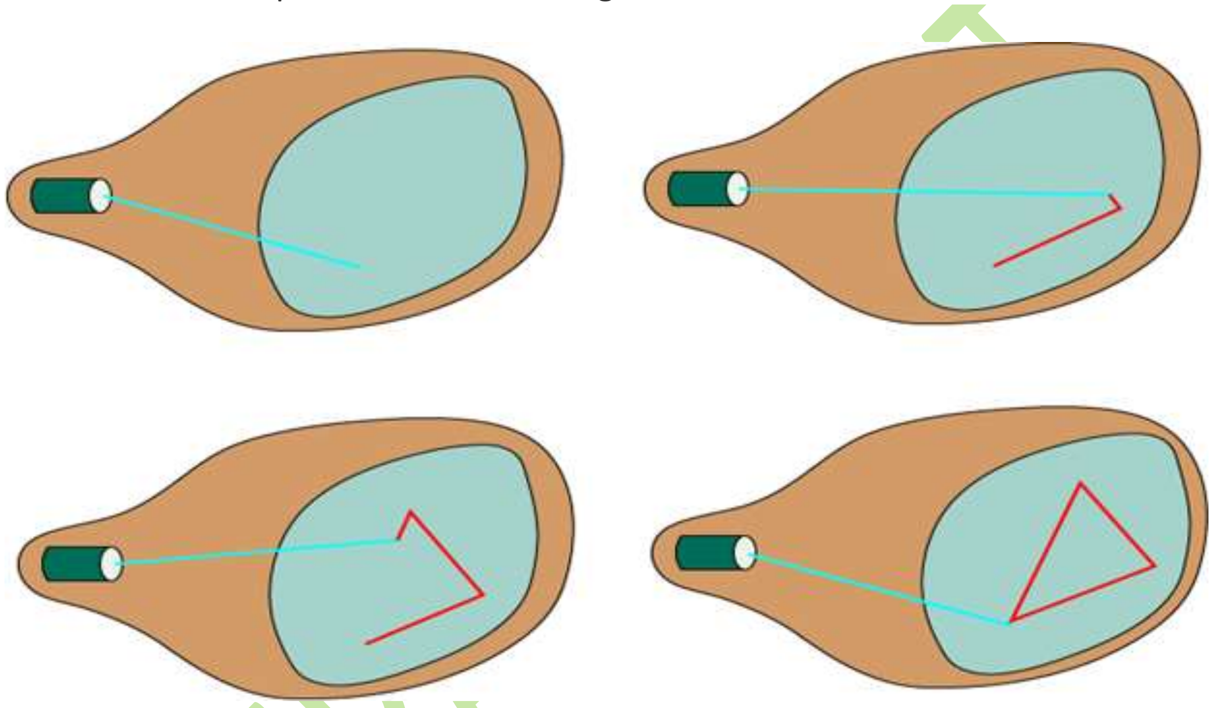
3. **Focusing system:** It is used to create a clear picture by focusing the electrons into a narrow beam.

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 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

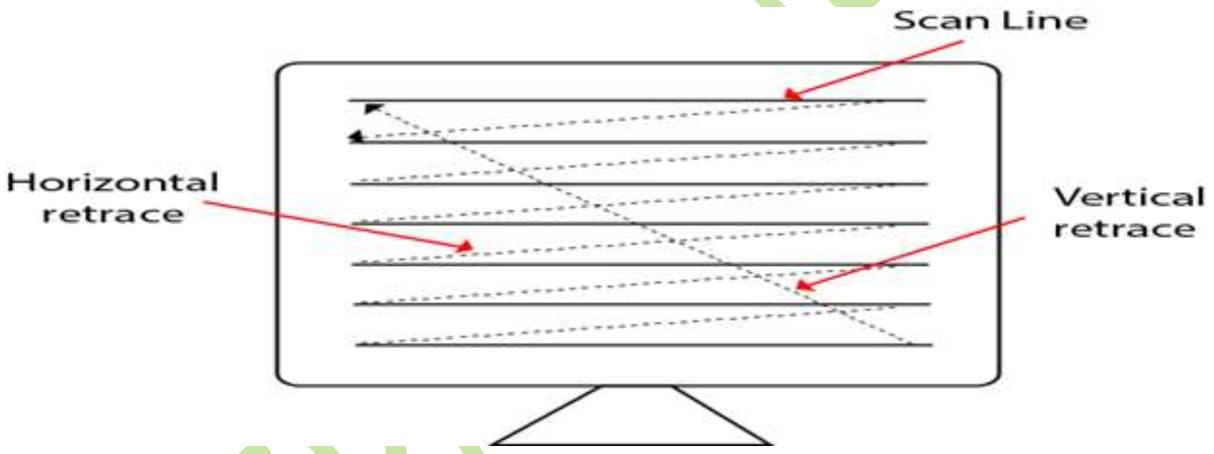
Disadvantages:

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:

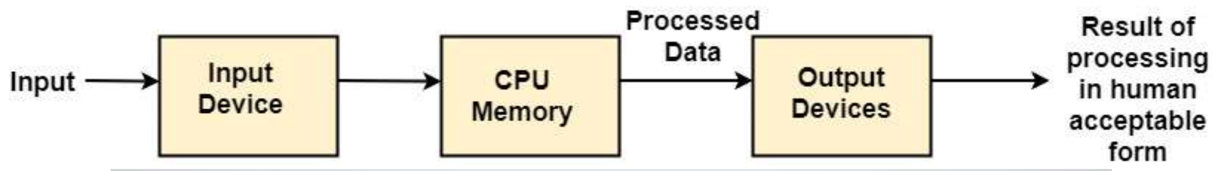
1. Low Resolution
2. Expensive

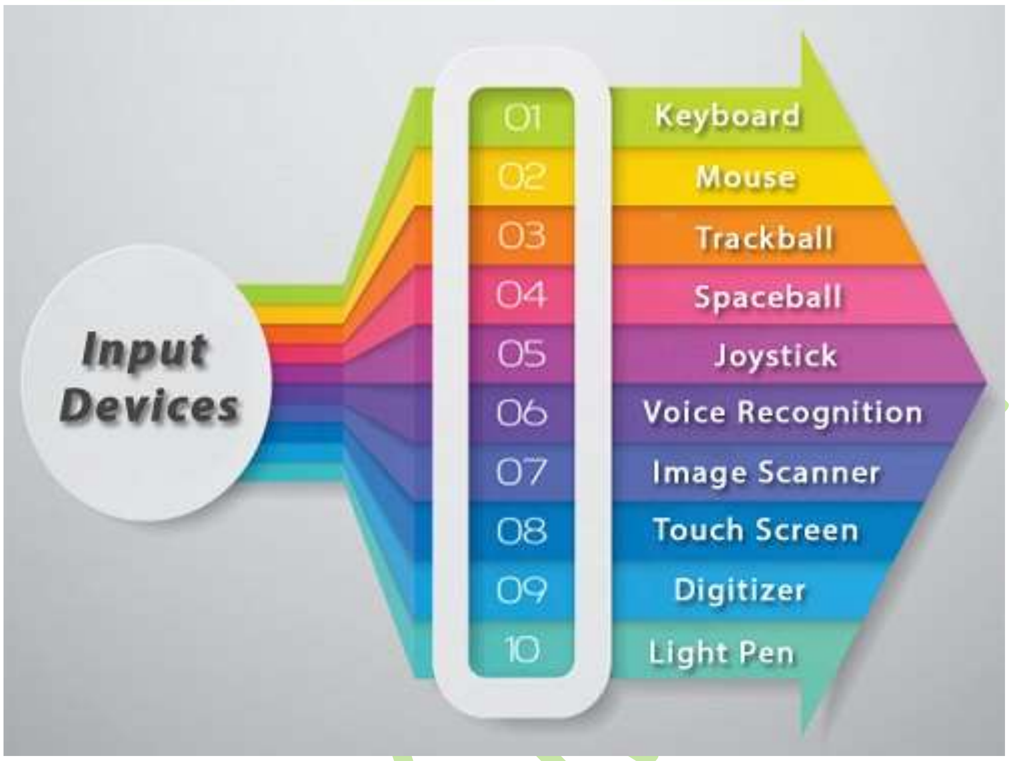
Differentiate between Random and Raster Scan Display:

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.

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.

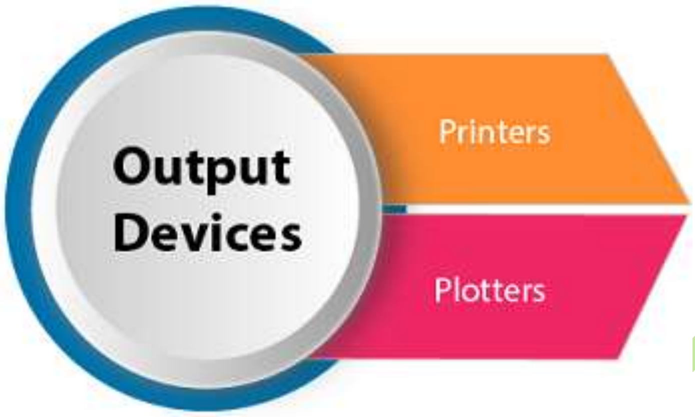




These Devices include:

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

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

Raster scan line

A scan line(also scanline) is one line, or row , in raster scanning pattern, such as a lone of video on a cathode ray tube(CRT) display of a television set or computer monitor.

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 $x_1<x_2$)

$x= x_1, y=y_1$ set $\Delta x=1$

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

Until $x = x_2$

Case2: When $|M|<1$ then (assume that $y_1<y_2$)

$x= x_1, y=y_1$ set $\Delta y=1$

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

Until $y \to y_2$

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:

Bresenham's(Mid-Point) 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'.

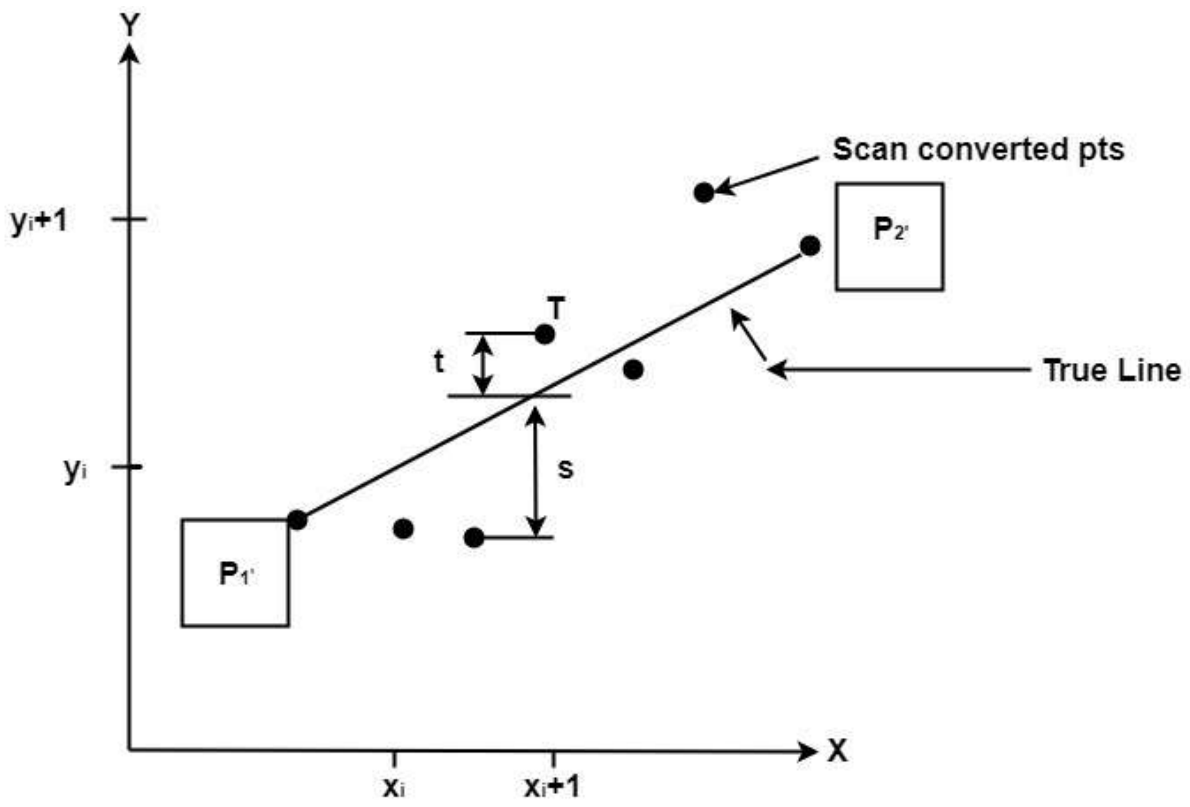


Fig: Scan Converting a line.

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) $<0 \Longrightarrow s < t$

The closest pixel is S

When (s-t) $\geq 0 \Longrightarrow s < t$

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 $\frac{\Delta y}{\Delta x}$ and introducing decision variable

$$d_i=\triangle x\ (s-t)$$
$$d_i=\triangle x\ (2\ \frac{\Delta y}{\Delta x}\ (x_i+1)+2b-2y_i-1)$$
$$=2\triangle xy_i-2\triangle y-1\triangle x.2b-2y_i\triangle x-\triangle x$$
$$d_i=2\triangle y.x_i-2\triangle x.y_i+c$$

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., $d_i \geq 0) \Longrightarrow y_{i+1}=y_i+1$

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

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

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

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 $m = \frac{\Delta y}{\Delta x}$, we have

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

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: 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.

2.3 SCAN – CONVERSION OF CIRCLE AND ELLIPSE

A circle with centre (x_c, y_c) and radius r can be represented in equation form in three ways

- Analytical representation: $r^2 = (x - x_c)^2 + (y - y_c)^2$
- Implicit representation : $(x - x_c)^2 + (y - y_c)^2 - r^2 = 0$
- Parametric representation: $x = x_c + r\cos\theta$
  $y = y_c + y\sin\theta$

A circle is symmetrical in nature. Eight – way symmetry can be used by reflecting each point about each 45° axis. The points obtained in this case are given below with illustration by figure.

$P_1 = (x, y)$	$P_5 = (-x, -y)$
$P_2 = (y, x)$	$P_6 = (-y, -x)$
$P_3 = (-y, x)$	$P_7 = (y, -x)$
$P_4 = (-x, y)$	$P_8 = (x, -y)$

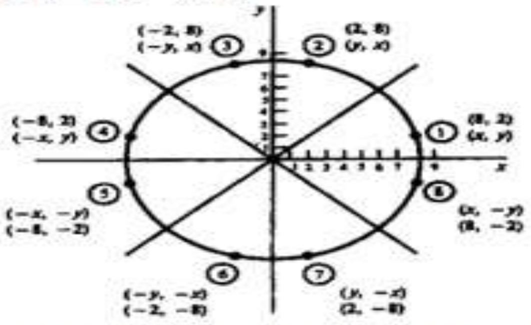


Figure 2.2 : Eight way symmetry of a circle

3.1 Bresenham's circle drawing algorithm

Let us define a procedure Bresenham_Circle (Xc,Yc, R) procedure for Bresenham's circle drawing algorithm for circle of radius R and centre (Xc, Yc)

```
Bresenham_Circle (Xc,Yc, R)
{
        Set X = 0;
        Set Y= R;
        Set D = 3 – 2R;
        While (X < Y)
          {
                Call Draw_Circle (Xc, Yc, X, Y);
                X=X+1;
                If (D < 0)
                { D = D + 4X + 6; }
                Else
                  {
                        Y = Y – 1;
                        D = D + 4(X – Y) + 10;
```

Follow for more

```
                }
            Call Draw_Circle (Xc, Yc, X, Y);
            }
    }
Draw_Circle (Xc, Yc, X, Y)
    {
        Call PutPixel (Xc + X, Yc, +Y);
        Call PutPixel (Xc – X, Yc, +Y);
        Call PutPixel (Xc + X, Yc, – Y);
        Call PutPixel (Xc – X, Yc, – Y);
        Call PutPixel (Xc + Y, Yc, + X);
        Call PutPixel (Xc – Y ,Yc, – X);
        Call PutPixel (Xc + Y, Yc, – X);
        Call PutPixel (Xc – Y, Yc, – X);
    }
```

2.3.2Midpoint circle drawing algorithm

This algorithm uses the implicit function of the circle in the following way

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

here $f(x, y) < 0$ means (x, y) is inside the circle

$f(x, y) = 0$ means (x, y) is on the circle

$f(x, y) > 0$ means (x, y) is outside the circle

The algorithm now follows as

```
        Midpoint_Circle( Xc, Yc, R)
            {
                Set X = 0;
                Set Y = R;
                Set P = 1 – R;
                While (X < Y)
                    {
                        Call Draw_Circle( Xc, Yc, X, Y);
                        X = X + 1;
                        If (P < 0)
                        {P = P + 2X + 6; }
                        Else
                            {
                                Y = Y – 1;
                                P = P + 2 (X – Y) + 1;
                            }
                        Call Draw_Circle( Xc, Yc, X, Y);
                    }
            }
```

2.3.3 Midpoint Ellipse Algorithm

This is a modified form of midpoint circle algorithm for drawing ellipse. The general equation of an ellipse in implicit form is

$f(x, y) = b^2x^2 + a^2y^2 - a^2b^2 = 0$

Now the algorithm for ellipse follows as

```
MidPoint_Ellipse( Xc, Yc, Rx, Ry)
    {
        /* Xc and Yc here denotes the x coordinate and y
        coordinate of the center of the ellipse and Rx and Ry
        are the x-radius and y-radius of the ellipse
        respectively */
        Set Sx = Rx * Rx;
        Set Sy = Ry * Ry;
        Set X = 0;
        Set Y = Ry;
        Set Px = 0;
        Set Py = 2 * Sx * Y;
        Call Draw_Ellipse (Xc, Yc, X, Y);
        Set P = Sy – (Sx * Ry) + (0.25 * Sx);/* First Region*/
        While ( Px<Py)
            {
                X = X + 1;
                Px = Px + 2 * Sy;
                If (P < 0)
                    {P = P + Sy + Px;}
                Else
                    {
                        Y = Y – 1;
                        Py = Py – 2 * Sx;
                        P = P + Sy + Px – Py;
                    }
                Call Draw_Ellipse (Xc, Yc, X, Y);
            }
        P = Sy * (X + 0.5)^2 + Sx * (Y – 1)^2 – Sx * Sy;
/*Second Region*/
        While (Y > 0)
            {
                Y = Y – 1;
                Py = Py – 2 * Sx;
                If (P > 0)
                    {P = P + Sx – Py;}
                Else
                    {
                        X = X + 1;
                        Px = Px + 2 * Sy;
                        P = P + Sx – Py + Px;
                    }
                Call Draw_Ellipse (Xc, Yc, X, Y);
            }
    }

Draw_Ellipse (Xc, Yc, X, Y)
    {
        Call PutPixel (Xc + X, Yc + Y);
        Call PutPixel (Xc – X, Yc + Y);
        Call PutPixel (Xc + X, Yc – Y);
        Call PutPixel (Xc – X, Yc – Y);
    }
```

2.4 DRAWING ELLIPSES AND OTHER CONICS

The equation of an ellipse with center at the origin is given as

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

Using standard parameterization, we can generate points on it as

$\theta \rightarrow (a\sin\theta, b\sin\theta)$

Differentiating the standard ellipse equation we get

$$\frac{dy}{dx}=-\frac{xb^2}{ya^2}$$

Now the DDA algorithm for circle can be applied to draw the ellipse.
Similarly a conic can be defined by the equation

$ax^2+bxy+cy^2+dx+ey+f=0$

If starting pixel on the conic is given, the adjacent pixel can be determined similar to the circle drawing algorithm.

8.3 POLYGON FILLING AND SEED FILL ALGORITHM

Polygon filling algorithm

There are two types of polygon filling algorithm.
1. Scan conversion polygon filling algorithm
2. Seed filling algorithms

Besides these algorithms we can use
a) Boundary fill algorithms and
b) Flood fill algorithm

8.4 SCAN-LINE ALGORITHM

Scan Line Algorithm.

In scanline filling algorithm, we take the intersection of each scanline with the edges of the polygon.

Steps :

1. Read n
2. Read (xi,yi) for all i=1,2,3......n
3. Read edges and store it in the array E which will be sorted accordingly to y axies.
4. Xmin=a; xmax=b; ymin=c; ymax=d
5. Take intersection
6. Take the scanline y=c and scan from x=a to x=b
7. Find the intersecting edges of E with y=c by comparing the y coordinate of the end points with y=c
8. Activate those edges
9. Scan through the line y=c and compute the next x position by appling the formulation
   $X_{k+1}= x_k +1/m$
   Check whether the point $(X_{k+1,}, Y_k)$ is inside or outside the polygon, by inside outside procedure. If the point $(X_{k+1,}, Y_k)$ is inside , paint it.
10. Repeat the procedure from Y_c to Y_d i.e. y=c to y=d.

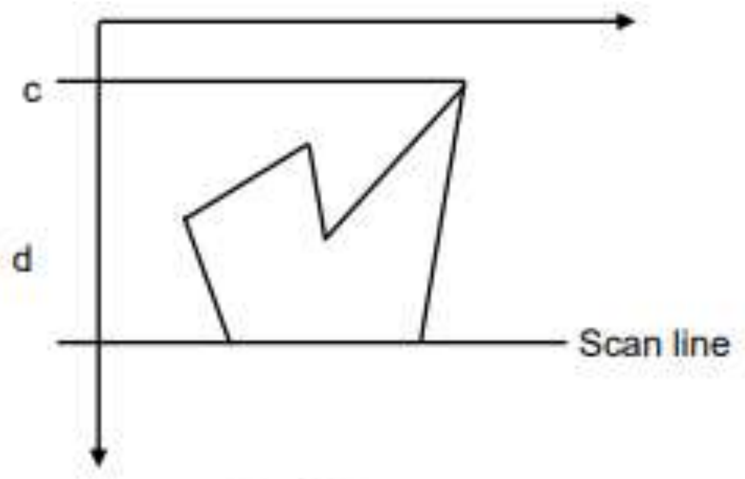


Fig. 8.4

Here, all the pixels inside the polygon can be painted without leaving any of the neighboring pixels. If the point of intersection of an edge and the scanline is a vertex, we shorten one of the edges. So that it will be contributed to the intersection is 1. If the endpoint of the two edges are on one side of the scan line and the contribution will be 2 if the other points are on the opposite side of the scanline.

The scan line filling algorithm can be applied for the curve closed boundary as follows:

1. Determine the pixels position along the curve boundary by using any of the incrementing methods
2. Filling the pixels by scanning through a scanline which spans between the boundary points. If the shape to be filled is regular geometrical figure like circle, ellipses etc. use symmetric property of geometrical figure to reduce boundary point calculations.

7.3 INTRODUCTION TO CLIPPING

The process which divides the given picture into two parts : visible and Invisible and allows to discard the invisible part is known as clipping. For clipping we need reference window called as clipping window.

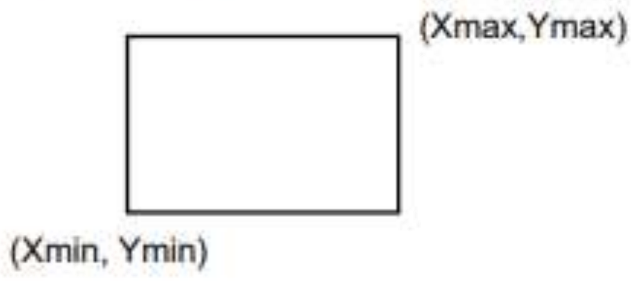


Fig. 7.3 Window

7.3.2 LINE CLIPPING

Discard the part of lines which lie outside the boundary of the window.

We require:

1. To identify the point of intersection of the line and window.
2. The portion in which it is to be clipped.
   The lines are divided into three categories.

a) Invisible
b) Visible
c) Partially Visible [Clipping Candidates]

To clip we give 4- bit code representation defined by

Bit 1	Bit 2	Bit 3	Bit 4
Ymax	Ymin	Xmax	Xmin

Fig. 7.5

Where, Bits take the volume either 0 or 1 and
Here, we divide the area containing the window as follows. Where, the coding is like this, Bit value = 1 if point lies outside the boundary OR

= 0 if point lies inside the boundary.
($Xmin \le X \le Xmax$ and $Ymin \le Y \le Ymax$)

Bit 1 tells you the position of the point related to Y=Ymax

Bit 2 tells you the position of the point related to Y=Ymin

Bit 3 tells you the position of the point related to X=Xmax

Bit 4 tells you the position of the point related to X=Xmin

1001	0001	0101
1000	0000 Clip Window	0100
1010	0010	0110

Fig. 7.6 Bit Code Representation

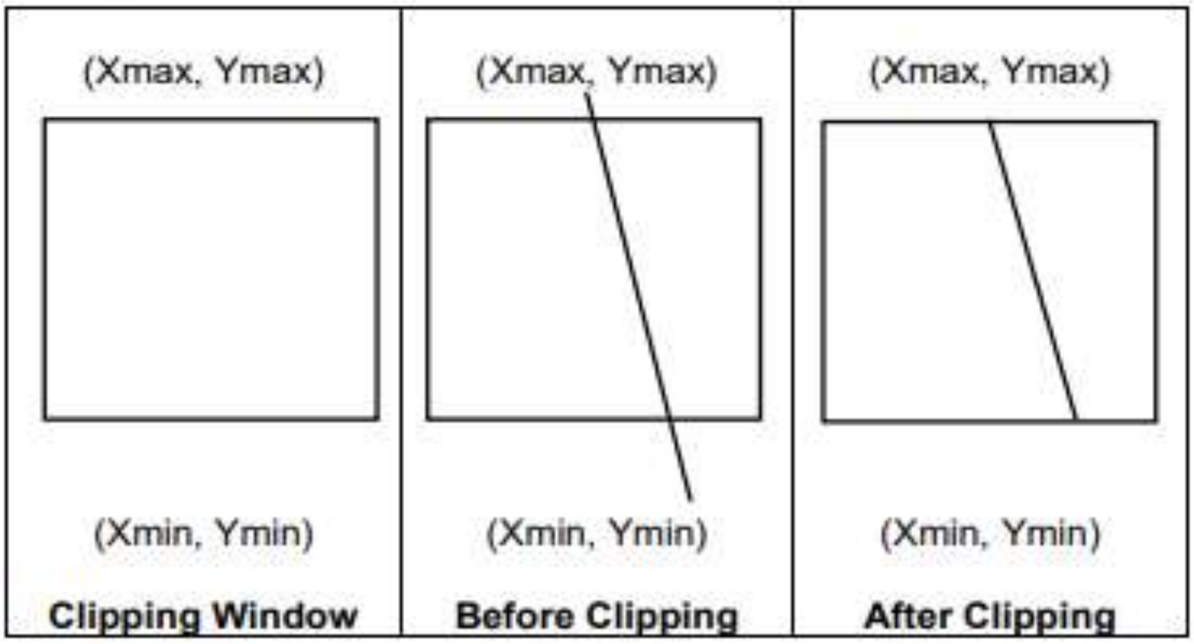
Rules for the visibility of the line:

1. If both the end points have bit code 0000 the line is visible.
2. If atleast one of the end point in non zero and
   a) The logical "AND"ing is 0000 then the line is Partially Visible
   b) If the logical "AND"ing isnon-zero then line is Not Visible.

Cohen-Sutherland Line Clipping Algorithm

For each line:

1. Assign codes to the endpoints
2. Accept if both codes are 0000, display line
3. Perform bitwise AND of codes
4. Reject if result is not 0000, return
5. Choose an endpoint outside the clipping rectangle
6. Test its code to determine which clip edge was crossed and find the intersection of the line and that clip edge (test the edges in a consistent order)
7. Replace endpoint (selected above) with intersection point
8. Repeat



7.4 INTRODUCTION TO A POLYGON CLIPPING

Polygon Clipping

Sutherland Hodgman Polygon Clipping algorithm

1. The polygon is stored by its vertices and edges, say v1, v2, v3 ,.......vn and e1, e2, e3,.... en.
2. Polygon is clipped by a window we need 4 clippers.

   Left clipper , Right Clipper, Bottom Clipper, Top Clipper
3. After clipping we get a different set of vertices say v1' , v2' , v3' ,....... vn'
4. Redraw the polygon by joining the vertices v1' , v2' , v3' ,....... vn' appropriately.

Algorithm:

1. Read v1, v2, v3 ,……vn coordinates of polygon.
2. Readcliping window. (Xmin, Ymin)(Xmax, Ymax)
3. For every edge do {
4. Compare the vertices of each edge of the polygon with the plane taken as the clipping plane.
5. Save the resulting intersections and vertices in the new list } // according to the possible relationships between the edge and the clipping boundary.
6. Draw the resulting polygon.

The output of the algorithm is a list of polygon vertices all of which are on the visible side of the clipping plane.

Here, the intersection of the polygon with the clipping plane is a line so every edge is individually compare with the clipping plane. This is achieved by considering two vertices of each edge which lies around the clipping boundary or plane. This results in 4 possible relationships between the edge and the clipping plane.

1st possibility:

If the 1st vertex of an edge lies outside the window boundary and the 2nd vertex lies inside the window boundary.

Here, point of intersection of the edge with the window boundaryand the second vertex are added to the putput vertex list (V1, v2)→(V1', v2)

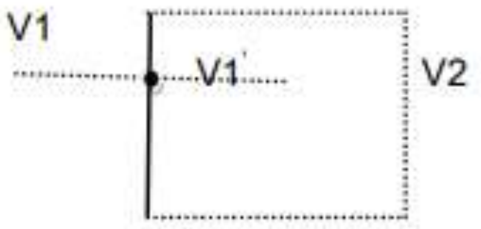


Fig. 7.8

2nd possibility:

If both the vertices of an edge are inside of the window boundary only the second vertex is added to the vertex list

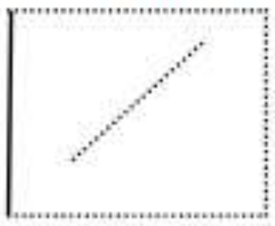


Fig. 7.9

3rd possibility:

If the 1st vertex is inside the window and 2nd vertex is outside only the intersection point is add to the output vertex list.

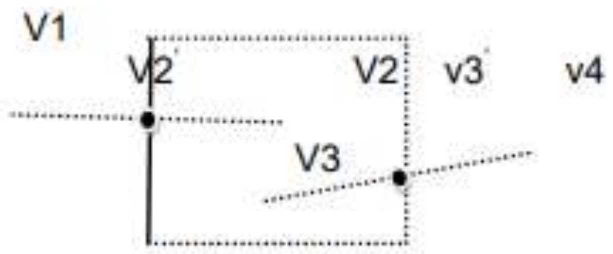


Fig. 7.10

4th possibility:

If both vertices are outside the window nothing is added to the vertex list.

Once all vertices are processed for one clipped boundary then the output list of vertices is clipped against the next window boundary going through above 4 possibilities. We have to consider the following points.

1) The visibility of the point. We apply inside-outside test.
2) Finding intersection of the edge with the clipping plane.

3.2 INTRODUCTION TO TRANSFORMATIONS

In computer graphics we often require to transform the coordinates of an object (position, orientation and size). One can view object transformation in two complementary ways:

(i) **Geometric transformation**: Object transformation takes place in relatively stationary coordinate system or background.

(ii) **Coordinate transformation**: In this view point, coordinate system is transformed instead of object.

On the basis of preservation, there are three classes of transformation

- **Rigid body**: Preserves distance and angle. Example – translation and rotation
- **Conformal**: Preserves angles. Example- translation, rotation and uniform scaling
- **Affine**: Preserves parallelism, means lines remains lines. Example- translation, rotation, scaling, shear and reflection

In general there are four attributes of an object that may be transformed

(i) Position(translation)
(ii) Size(scaling)
(iii) Orientation(rotation)
(iv) Shapes(shear)

3.4TYPES OF TRANSFORMATION IN TWO – DIMENSIONAL GRAPHICS

In 2D transformations, only planar coordinates are used. For this purpose a 2x2 transformation matrix is utilized. In general, 2D transformation includes following types of transformations:

I. Identity transformation
II. Scaling
III. Reflection
IV. Shear transformation
V. Rotation
VI. Translation

3.5 IDENTITY TRANSFORMATION

In identity transformation, each point is mapped onto itself. There is no change in the source image on applying identity transformation. Suppose T is the transformation matrix for identity transformation which operates on a point P (x, y) which produces point P' (x', y'), then

$$P'(x', y') = [x' \; y'] = [P]\,[T] = [x \; y]\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = [x \; y]$$

We can see that on applying identity transformation we obtain the same points. Here the identity transformation is

$$[T] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The identity transformation matrix is basically anxn matrix with ones on the main diagonal and zeros for other values.

3.6 SCALING

This transforms changes the size of the object. We perform this operation by multiplying scaling factors s_x and s_y to the original coordinate values (x, y) to obtain scaled new coordinates (x', y').

$x' = x.\, s_x$ $\qquad$ $y' = y.\, s_y$

In matrix form it can be represented as

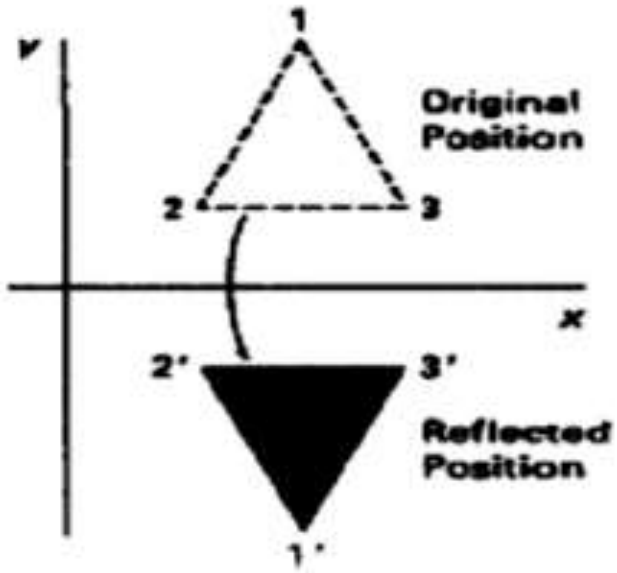
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

For same s_x and s_y, the scaling is called as uniform scaling. For different s_x and s_y , the scaling is called as differential scaling.

3.7REFLECTION

In reflection transformation, the mirror image of an object is formed. In two dimensions, it is achieved through rotating the object by 180 degrees about an axis known as axis of reflection lying in a plane. We can choose any plane of reflection in xy plane or perpendicular to xy plane.

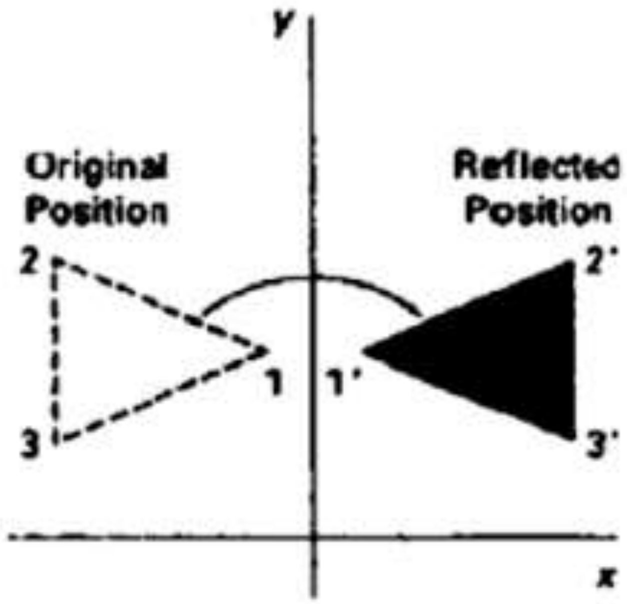
For example, reflection about x axis (y = 0) plane can be done with the transformation matrix



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

Figure 3.1 : Reflection transformation about x axis

A reflection about y axis (x = 0) is shown in the figure below which can be done by following transformation matrix.



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

Figure 3.2 : Reflection about y axis

3.8 SHEAR TRANSFORMATIONS

An object can be considered to be composed of different layers. In shear transformation, the shape of the object is distorted by producing the sliding effect of layers over each other. There are two general shearing transformations, one which shift coordinates of x axis and the other that shifts y coordinate values.

The transformation matrix for producing shear relative to x axis is

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

producing transformations

$$x' = x + sh_x.y, \quad y' = y$$

wheresh_x is a shear parameter which can take any real number value. The figure below demonstrates this transformation

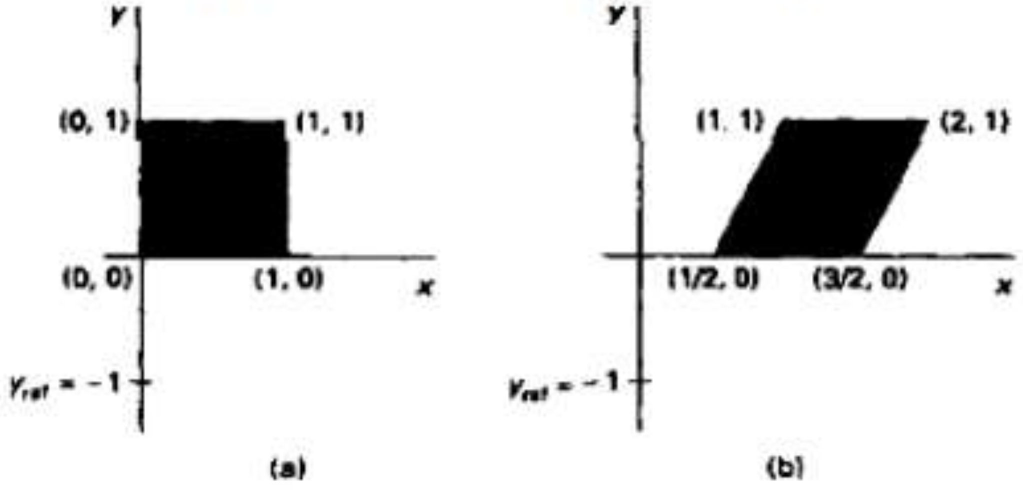


Figure 3.3 : 2D shear transformation

4.2 ROTATION

In rotation transformation, an object is repositioned along a circular path in the xy plane. The rotation is performed with certain angle θ, known as rotation angle. Rotation can be performed in two ways: about origin or about an arbitrary point called as rotation point or pivot point.

Rotation about origin: The pivot point here is the origin. We can obtain transformation equations for rotating a point (x, y) through an angleθ to obtain final point as (x', y') with the help of figure as

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

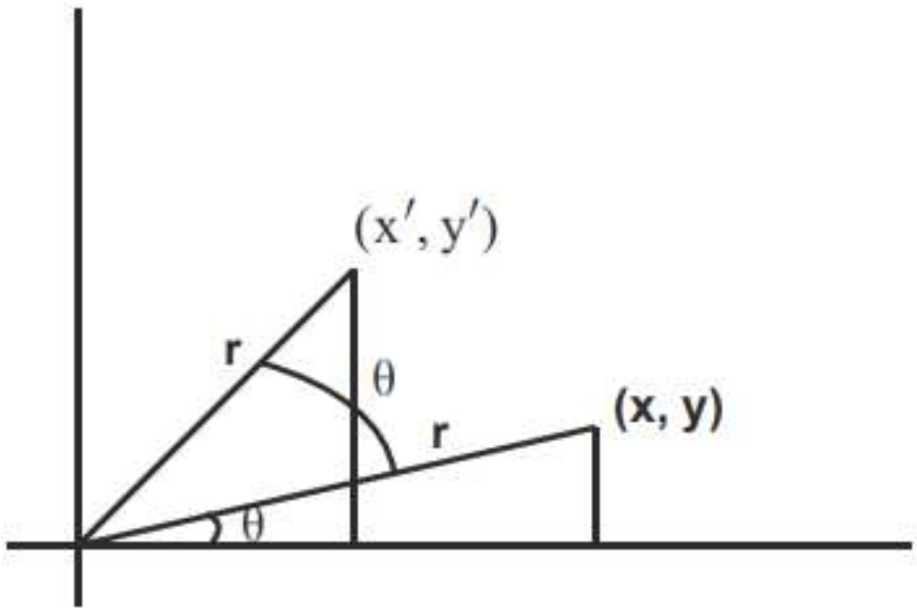


Figure 4.1 : Rotation about origin

The transformation matrix for rotation can be written as

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

Hence, the rotation transformation in matrix form can be represented as (x_1, y_1)

P' = R.P

4.3 TRANSLATION

The repositioning of the coordinates of an object along a straight line path is called as translation. The translation transformation is done by adding translation distance t_x and t_y to the original coordinate position (x, y) to obtain new position (x', y').

$$x' = x + t_x, \qquad y' = y + t_y$$

The pair (t_x, t_y) is called as translation vector or shift vector.
In the matrix form, it can be written as

$$P' = P + T \quad \text{, where}$$

$$P = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad P' = \begin{bmatrix} x'_1 \\ x'_2 \end{bmatrix}, \quad T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

The figure below shows the translation of an object. Here coordinate points defining the object are translated and then it is reconstructed.

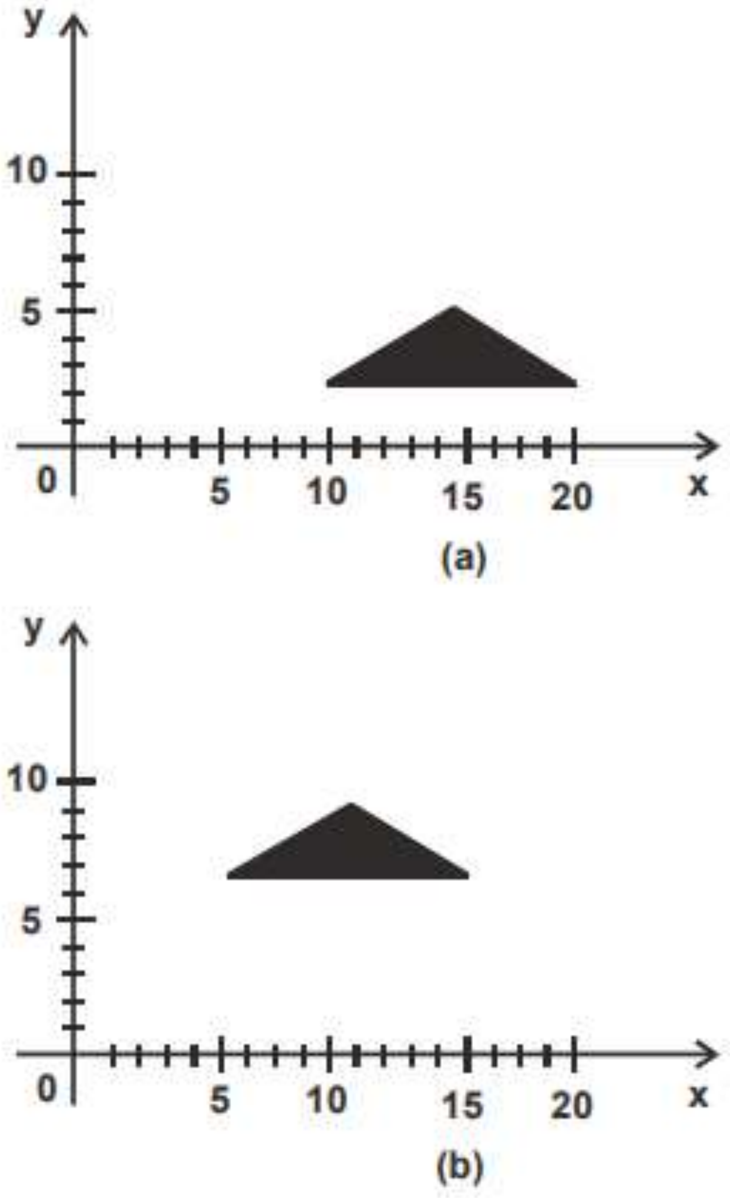


Figure 4.2 : 2D translation transformation

4.4ROTATION ABOUT AN ARBITRARY POINT

It is often required in many applications to rotate an object about an arbitrary point rather than the origin. Rotation about an arbitrary pivot point (x_r, y_r) is shown in the figure below

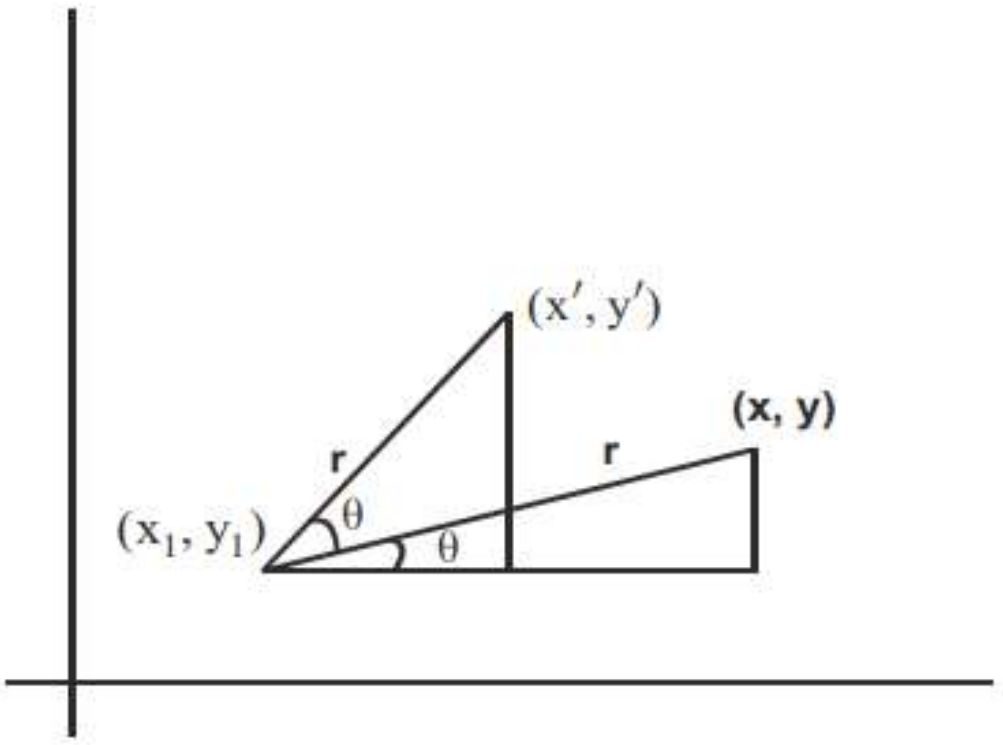


Figure 4.3 : Rotation about an arbitrary point

The corresponding equation obtained will be

$$x' = x_t + (x - x_t)\cos\theta - (y - y_t)\sin\theta$$
$$y' = y_t + (x - x_t)\sin\theta + (y - y_t)\cos\theta$$

We can see the difference between this rotation transformation from the previous one .This one contains the additive terms as well as the multiplicative factors on the coordinate values.

Let us understand the rotation about an arbitrary point through an example. Suppose we have to rotate a triangle ABC by 90 degree about a point (1, -1). The coordinates of the triangle are A (4, 0), B (8, 3) and C (6, 2).

The triangle ABC can be represented in matrix as

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} 4 & 0 \\ 8 & 3 \\ 6 & 2 \end{bmatrix}$$

The point about which the triangle has to be rotated be P = (-1, 1) and the rotation matrix for 90 degree rotation is

$$R_{90} = \begin{bmatrix} \cos 90 & \sin 90 \\ -\sin 90 & \cos 90 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Now we can perform the rotation operation in three steps. In first step we will have to translate the arbitrary point to the origin.

Let A'B'C' be the new coordinates obtained after translation operation.

$$\begin{bmatrix} A' \\ B' \\ C' \end{bmatrix} = ABC - P = \begin{bmatrix} 4 & 0 \\ 8 & 3 \\ 6 & 2 \end{bmatrix} - \begin{bmatrix} -1 & 1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 5 & -1 \\ 9 & 2 \\ 7 & 1 \end{bmatrix}$$

Now the second step is to rotate the object. Let A''B''C'' be new coordinates after applying rotation operation to A'B'C', then

$$\begin{bmatrix} A'' \\ B'' \\ C'' \end{bmatrix} = A'B'C'.[R_{90}] = \begin{bmatrix} 5 & -1 \\ 9 & 2 \\ 7 & 1 \end{bmatrix} . \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 5 \\ -2 & 9 \\ -1 & 7 \end{bmatrix}$$

In third step we translate back the coordinates

$$\begin{bmatrix} A''' \\ B''' \\ C''' \end{bmatrix} = \begin{bmatrix} A'' \\ B'' \\ C'' \end{bmatrix} + [P] = \begin{bmatrix} 1 & 5 \\ -2 & 9 \\ -1 & 7 \end{bmatrix} + \begin{bmatrix} -1 & 1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 6 \\ -3 & 10 \\ -2 & 8 \end{bmatrix}$$

The coordinates A'''B'''C''' is the required result.

4.5 COMBINED TRANSFORMATION

A sequence of transformation is said to be as composite or combined transformations can be represented by product of matrices. The product is obtained by multiplying the transformation matrices in order from right to left.

For example, two successive translations applied to position P to obtain P' is calculated as

$$P' = \{T(t_{x2}, t_{y2}).\ T(t_{x1}, t_{y1})\}.\ P$$

The expanded form of the multiplication of translation vectors of above equation can be written as

$$\begin{bmatrix} 1 & 0 & t_{x2} \\ 0 & 1 & t_{y2} \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & t_{x1} \\ 0 & 1 & t_{y1} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{x1} + t_{x2} \\ 0 & 1 & t_{y1} + t_{y2} \\ 0 & 0 & 1 \end{bmatrix}$$

$$T(t_{x2}, t_{y2}).\ T(t_{x1}, t_{y1}) = T(t_{x1} + t_{x2}, t_{y1} + t_{y2})$$