**COMPUTER GRAPHICS ASSIGNMENT**

Q-1 Describe applications of Computer Graphics.

[Computer graphics](https://www.geeksforgeeks.org/computer-graphics-2/) deals with creation, manipulation and storage of different type of images and objects.

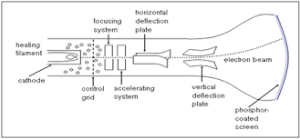
Some of the applications of computer graphics are:

1. **Computer Art:**  
   Using computer graphics we can create fine and commercial art which include animation packages, paint packages. These packages provide facilities for designing object shapes and specifying object motion.Cartoon drawing, paintings, logo design can also be done.
2. **Computer Aided Drawing:**  
   Designing of buildings, automobile, aircraft is done with the help of computer aided drawing, this helps in providing minute details to the drawing and producing more accurate and sharp drawings with better specifications.
3. **Presentation Graphics:**  
   For the preparation of reports or summarising the financial, statistical, mathematical, scientific, economic data for research reports, managerial reports, moreover creation of bar graphs, pie charts, time chart, can be done using the tools present in computer graphics.
4. **Entertainment:**  
   Computer graphics finds a major part of its utility in the movie industry and game industry. Used for creating motion pictures , music video, television shows, cartoon animation films. In the game industry where focus and interactivity are the key players, computer graphics helps in providing such features in the efficient way.
5. **Education:**  
   Computer generated models are extremely useful for teaching huge number of concepts and fundamentals in an easy to understand and learn manner. Using computer graphics many educational models can be created through which more interest can be generated among the students regarding the subject.
6. **Training:**  
   Specialised system for training like simulators can be used for training the candidates in a way that can be grasped in a short span of time with better understanding. Creation of training modules using computer graphics is simple and very useful.
7. **Visualisation:**  
   Today the need of visualise things have increased drastically, the need of visualisation can be seen in many advance technologies , data visualisation helps in finding insights of the data , to check and study the behaviour of processes around us we need appropriate visualisation which can be achieved through proper usage of computer graphics
8. **Image Processing:**  
   Various kinds of photographs or images require editing in order to be used in different places. Processing of existing images into refined ones for better interpretation is one of the many applications of computer graphics.
9. **Machine Drawing:**  
   Computer graphics is very frequently used for designing, modifying and creation of various parts of machine and the whole machine itself, the main reason behind using computer graphics for this purpose is the precision and clarity we get from such drawing is ultimate and extremely desired for the safe manufacturing of machine using these drawings.
10. **Graphical User Interface:**  
    The use of pictures, images, icons, pop-up menus, graphical objects helps in creating a user friendly environment where working is easy and pleasant, using computer graphics we can create such an atmosphere where everything can be automated and anyone can get the desired action performed in an easy fashion.

These are some of the applications of computer graphics due to which it’s popularity has increased to a huge extend and will keep on increasing with the progress in technology.

Q-2 What is refresh CRT? Explain function of CRT in detail with all characteristics.

* A beam of electrons (cathode rays), emitted by an electron gun, passes through focusing and deflection systems that direct the beam towards specified position on the phosphor-coated screen.
* The phosphor then emits a small spot of light at each position contacted by the electron beam.
* One way to keep the phosphor glowing is to redraw the picture repeatedly by quickly directing the electron beam back over the same points. This type of display is called a refresh CRT.

**[](https://graphicsnotes.files.wordpress.com/2014/11/crt.png)**

**Electron Gun**

* The primary components of an electron gun in a CRT are the heated metal cathode and a control grid.
* The cathode is heated by an electric current passed through a coil of wire called the filament.
* In the vacuum inside the CRT envelope, negatively charged electrons are then accelerated toward the phosphor coating by a high positive voltage.

**Control Grid :**

* Control grid is used to surround the cathode. Grid is cylindrical in shape. It is made up of metal.
* Grid has hole at one end, through which electrons get escaped.
* The control grid is kept at lower potential as compared to cathode, so that a electrostatic field can be created.
* It will direct that electrons through point source, so process of focusing will be simplified.

**Focusing System**

* The focusing system is to create a clear picture by focusing the electrons into a narrow beam. Otherwise, electrons would repel each other and beam would spread out as it reaches the screen.
* Focusing is accomplished with either electric or magnetic fields.

**Deflection System**

* Deflection of the electron beam can be controlled by either electric fields or magnetic fields.
* In case of magnetic field, two pairs of coils are used, one for horizontal deflection and other for vertical deflection.
* In case of electric field, two pairs of parallel plates are used, one for horizontal deflection and second for vertical deflection as shown in figure above.

**CRT Screen**

* The inside of the large end of a CRT is coated with a fluorescent material that gives off light when struck by electrons.
* When the electrons in the beam is collides with phosphor coating screen, they stopped and their kinetic energy is absorbed by the phosphor.
* Then a part of beam energy is converted into heat energy and the remainder part causes the electrons in the phosphor atom to move up to higher energy levels.

**Chracteristics**

**Persistence**

* It is defined as the time they continue to emit light after the CRT beam is removed.
* Persistence is defined as the time it takes the emitted light from the screen to decay to one-tenth of its original intensity.
* Lower-persistence phosphors require higher refresh rates to maintain a picture on the screen without flicker.
* A phosphor with low persistence is useful for animation; a high-persistence phosphor is useful for displaying highly complex, static pictures.

**Resolution**

* The number of points per centimeter that can be used be plotted horizontally and vertically. Or Total number of points in each direction.
* The resolution of a CRT is depend on
* type of phosphor
* intensity to be displayed

**Aspect Ratio**

* It is ratio of horizontal to vertical points.
* Example: An aspect ratio of 3/4 means that a vertical line plotted with three points has same length as horizontal line plotted with four points.

Q-3 Compare raster scan and vector scan display methods in detail.

Differentiate between Random and Raster Scan Display:

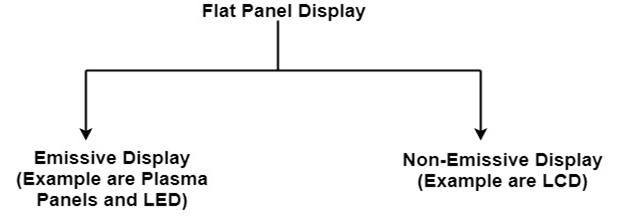
|  |  |
| --- | --- |
| **Random Scan** | **Raster Scan** |
| 1. It has high Resolution | 1. Its resolution is low. |
| 2. It is more expensive | 2. It is less expensive |
| 3. Any modification if needed is easy | 3.Modification is tough |
| 4. Solid pattern is tough to fill | 4.Solid pattern is easy 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. |

Q-4 Explain flat panel display in detail.

# 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.



**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:

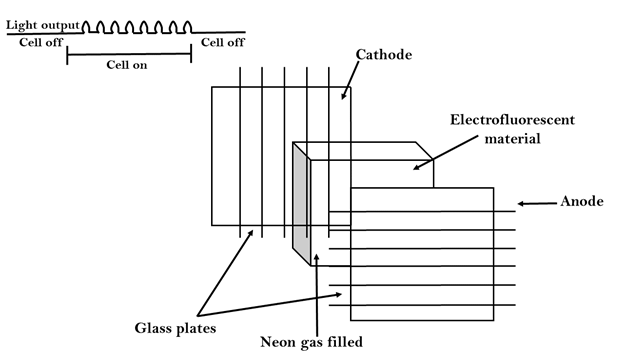
Plasma-Panels are also called as Gas-Discharge Display. It consists of an array of small lights. Lights are fluorescent in nature. The essential components of the plasma-panel display are:

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.

### Advantage:

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

### Disadvantage:

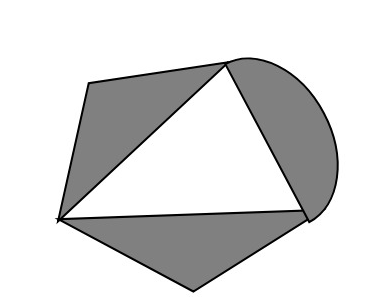
1. LCDs are temperature-dependent (0-70°C)
2. LCDs do not emit light; as a result, the image has very little contrast.
3. LCDs have no color capability.
4. The resolution is not as good as that of a CRT.

Q-7 Explain area filling algorithm flood fill.

# Flood Fill Algorithm:

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

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



In fill algorithm, we start from a specified interior point (x, y) and reassign all pixel values are currently set to a given interior color with the desired color. Using either a 4-connected or 8-connected approaches, we then step through pixel positions until all interior points have been repainted.

## Disadvantage:

1. Very slow algorithm
2. May be fail for large polygons
3. Initial pixel required more knowledge about surrounding pixels.

## Algorithm:

## Procedure floodfill (x, y,fill\_ color, old\_color: integer)

## If (getpixel (x, y)=old\_color)

## {

## setpixel (x, y, fill\_color);

## fill (x+1, y, fill\_color, old\_color);

## fill (x-1, y, fill\_color, old\_color);

## fill (x, y+1, fill\_color, old\_color);

## fill (x, y-1, fill\_color, old\_color);

## }

## }

## 8. Explain area filling algorithm boundary fill algorithm.

**Introduction :**Boundary Fill Algorithm starts at a pixel inside the polygon to be filled and paints the interior proceeding outwards towards the boundary. This algorithm works **only if** the color with which the region has to be filled and the color of the boundary of the region are different. If the boundary is of one single color, this approach proceeds outwards pixel by pixel until it hits the boundary of the region.  
***Boundary Fill Algorithm is recursive in nature.*** It takes an interior point(x, y), a fill color, and a boundary color as the input. The algorithm starts by checking the color of (x, y). If it’s color is not equal to the fill color and the boundary color, then it is painted with the fill color and the function is called for all the neighbours of (x, y). If a point is found to be of fill color or of boundary color, the function does not call its neighbours and returns. This process continues until all points up to the boundary color for the region have been tested.  
The boundary fill algorithm can be implemented by 4-connected pixels or 8-connected pixels.  
**4-connected pixels :**After painting a pixel, the function is called for four neighbouring points. These are the pixel positions that are right, left, above, and below the current pixel. Areas filled by this method are called 4-connected. Below given is the algorithm :   
**Algorithm :**

void boundaryFill4(int x, int y, int fill\_color,int boundary\_color)

{

if(getpixel(x, y) != boundary\_color &&

getpixel(x, y) != fill\_color)

{

putpixel(x, y, fill\_color);

boundaryFill4(x + 1, y, fill\_color, boundary\_color);

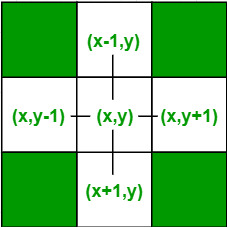
boundaryFill4(x, y + 1, fill\_color, boundary\_color);

boundaryFill4(x - 1, y, fill\_color, boundary\_color);

boundaryFill4(x, y - 1, fill\_color, boundary\_color);

}

}



**8-connected pixels :**More complex figures are filled using this approach. The pixels to be tested are the 8 neighbouring pixels, the pixel on the right, left, above, below and the 4 diagonal pixels. Areas filled by this method are called 8-connected. Below given is the algorithm :

**Algorithm :**

void boundaryFill8(int x, int y, int fill\_color,int boundary\_color)

{

if(getpixel(x, y) != boundary\_color &&

getpixel(x, y) != fill\_color)

{

putpixel(x, y, fill\_color);

boundaryFill8(x + 1, y, fill\_color, boundary\_color);

boundaryFill8(x, y + 1, fill\_color, boundary\_color);

boundaryFill8(x - 1, y, fill\_color, boundary\_color);

boundaryFill8(x, y - 1, fill\_color, boundary\_color);

boundaryFill8(x - 1, y - 1, fill\_color, boundary\_color);

boundaryFill8(x - 1, y + 1, fill\_color, boundary\_color);

boundaryFill8(x + 1, y - 1, fill\_color, boundary\_color);

boundaryFill8(x + 1, y + 1, fill\_color, boundary\_color);

}

}

Q-9 Explain scan line fill polygon filling algorithm..

**Scanline Polygon filling Algorithm**

Scanline filling is basically filling up of polygons using horizontal lines or scanlines. The purpose of the SLPF algorithm is to fill (color) the interior pixels of a polygon given only the vertices of the figure. To understand Scanline, think of the image being drawn by a single pen starting from bottom left, continuing to the right, plotting only points where there is a point present in the image, and when the line is complete, start from the next line and continue.   
This algorithm works by intersecting scanline with polygon edges and fills the polygon between pairs of intersections. 

https://media.geeksforgeeks.org/wp-content/uploads/scanlinefilling.png

**Special cases of polygon vertices:** 

1. If both lines intersecting at the vertex are on the same side of the scanline, consider it as two points.
2. If lines intersecting at the vertex are at opposite sides of the scanline, consider it as only one point.

**Components of Polygon fill:** 

1. **Edge Buckets:**It contains an edge’s information. The entries of edge bucket vary according to data structure you have used.In the example we are taking below, there are three edge buckets namely: ymax, xofymin,   
   slopeinverse.
2. **Edge Table:**It consistsof several edge lists -> holds all of the edges that compose the figure. When creating edges, the vertices of the edge need to be ordered from left to right and the edges are maintained in increasing yMin order. Filling is complete once all of the edges are removed from the ET
3. **Active List:** IT maintains the current edges being used to fill in the polygon.Edges aree pushed into the AL from the Edge Table when an edge’s yMin is equal to the current scan line being processed.   
   The Active List will be re-sorted after every pass.

**Data Structure:** 

Scan line polygon filling1

**Algorithm:**

1. We will process the polygon edge after edge, and store in the edge Table.

2. Storing is done by storing the edge in the same scanline edge tuple as

the lowermost point's y-coordinate value of the edge.

3. After addition of any edge in an edge tuple, the tuple is

sorted using insertion sort, according to its xofymin value.

4. After the whole polygon is added to the edge table,

the figure is now filled.

5. Filling is started from the first scanline at the bottom,

and continued till the top.

6. Now the active edge table is taken and the following things

are repeated for each scanline:

i. Copy all edge buckets of the designated scanline

to the active edge tuple

ii. Perform an insertion sort according

to the xofymin values

iii. Remove all edge buckets whose ymax is equal

or greater than the scanline

iv. Fillup pairs of edges in active tuple, if any vertex is got,

follow these instructions:

o If both lines intersecting at the vertex are on

the same side of the scanline, consider it as two points.

o If lines intersecting at the vertex are at

opposite sides of the scanline, consider it as only one point.

v. Update the xofymin by adding slopeinverse for each bucket.

## Q-10 Describe character generation methods.

**Q-11 Define transformation. Explain all basic transformations. (rotation, scaling, translation) (Note: any one will be asked in detail so prepare all)**

Transformation means changing some graphics into something else by applying rules. We can have various types of transformations such as translation, scaling up or down, rotation, shearing, etc. When a transformation takes place on a 2D plane, it is called 2D transformation.

Transformations play an important role in computer graphics to reposition the graphics on the screen and change their size or orientation.

## Homogenous Coordinates

To perform a sequence of transformation such as translation followed by rotation and scaling, we need to follow a sequential process −

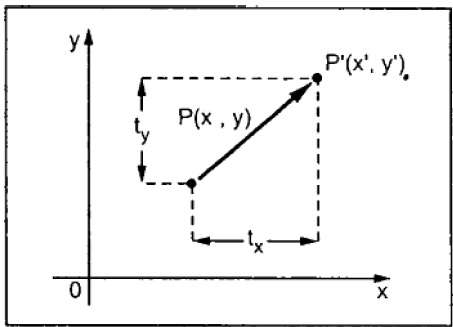
* Translate the coordinates,
* Rotate the translated coordinates, and then
* Scale the rotated coordinates to complete the composite transformation.

To shorten this process, we have to use 3×3 transformation matrix instead of 2×2 transformation matrix. To convert a 2×2 matrix to 3×3 matrix, we have to add an extra dummy coordinate W.

In this way, we can represent the point by 3 numbers instead of 2 numbers, which is called **Homogenous Coordinate** system. In this system, we can represent all the transformation equations in matrix multiplication. Any Cartesian point PX,YX,Y can be converted to homogenous coordinates by P’ (Xh, Yh, h).

## Translation

A translation moves an object to a different position on the screen. You can translate a point in 2D by adding translation coordinate (tx, ty) to the original coordinate X,YX,Y to get the new coordinate X′,Y′X′,Y′.



From the above figure, you can write that −

**X’ = X + tx**

**Y’ = Y + ty**

The pair (tx, ty) is called the translation vector or shift vector. The above equations can also be represented using the column vectors.

P=[X][Y]P=[X][Y] p' = [X′][Y′][X′][Y′]T = [tx][ty][tx][ty]

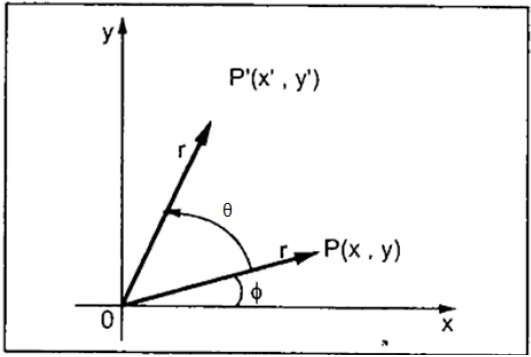
We can write it as −

**P’ = P + T**

## Rotation

In rotation, we rotate the object at particular angle θ thetatheta from its origin. From the following figure, we can see that the point PX,YX,Y is located at angle φ from the horizontal X coordinate with distance r from the origin.

Let us suppose you want to rotate it at the angle θ. After rotating it to a new location, you will get a new point P’ X′,Y′X′,Y′.



Using standard trigonometric the original coordinate of point PX,YX,Y can be represented as −

X=rcosϕ......(1)X=rcosϕ......(1)

Y=rsinϕ......(2)Y=rsinϕ......(2)

Same way we can represent the point P’ X′,Y′X′,Y′ as −

x′=rcos(ϕ+θ)=rcosϕcosθ−rsinϕsinθ.......(3)x′=rcos(ϕ+θ)=rcosϕcosθ−rsinϕsinθ.......(3)

y′=rsin(ϕ+θ)=rcosϕsinθ+rsinϕcosθ.......(4)y′=rsin(ϕ+θ)=rcosϕsinθ+rsinϕcosθ.......(4)

Substituting equation 11 & 22 in 33 & 44 respectively, we will get

x′=xcosθ−ysinθx′=xcosθ−ysinθ

y′=xsinθ+ycosθy′=xsinθ+ycosθ

Representing the above equation in matrix form,

[X′Y′]=[XY][cosθ−sinθsinθcosθ]OR[X′Y′]=[XY][cosθsinθ−sinθcosθ]OR

P’ = P . R

Where R is the rotation matrix

R=[cosθ−sinθsinθcosθ]R=[cosθsinθ−sinθcosθ]

The rotation angle can be positive and negative.

For positive rotation angle, we can use the above rotation matrix. However, for negative angle rotation, the matrix will change as shown below −

R=[cos(−θ)−sin(−θ)sin(−θ)cos(−θ)]R=[cos(−θ)sin(−θ)−sin(−θ)cos(−θ)]

=[cosθsinθ−sinθcosθ](∵cos(−θ)=cosθandsin(−θ)=−sinθ)=[cosθ−sinθsinθcosθ](∵cos(−θ)=cosθandsin(−θ)=−sinθ)

## Scaling

To change the size of an object, scaling transformation is used. In the scaling process, you either expand or compress the dimensions of the object. Scaling can be achieved by multiplying the original coordinates of the object with the scaling factor to get the desired result.

Let us assume that the original coordinates are X,YX,Y, the scaling factors are (SX, SY), and the produced coordinates are X′,Y′X′,Y′. This can be mathematically represented as shown below −

**X' = X . SX and Y' = Y . SY**

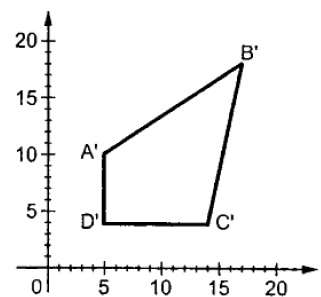
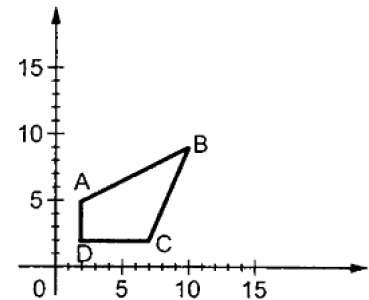
The scaling factor SX, SY scales the object in X and Y direction respectively. The above equations can also be represented in matrix form as below −

(X′Y′)=(XY)[Sx00Sy](X′Y′)=(XY)[Sx00Sy]

OR

**P’ = P . S**

Where S is the scaling matrix. The scaling process is shown in the following figure.

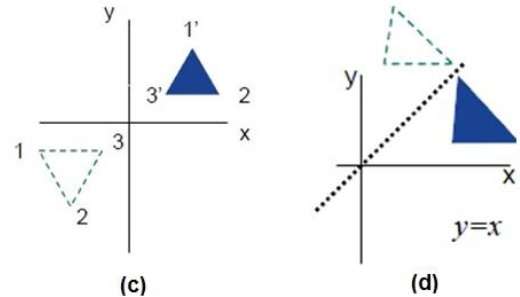
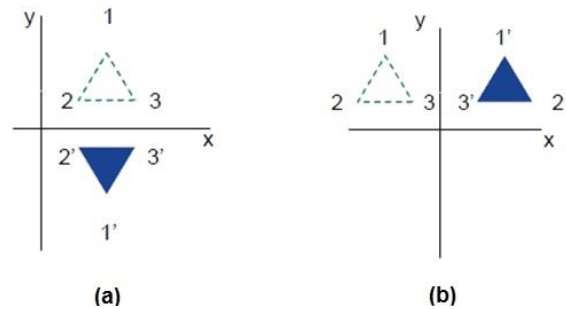


If we provide values less than 1 to the scaling factor S, then we can reduce the size of the object. If we provide values greater than 1, then we can increase the size of the object.

## Reflection

Reflection is the mirror image of original object. In other words, we can say that it is a rotation operation with 180°. In reflection transformation, the size of the object does not change.

The following figures show reflections with respect to X and Y axes, and about the origin respectively.

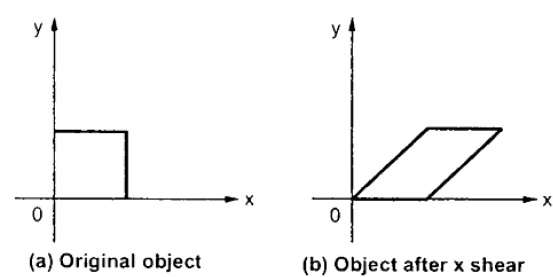


## Shear

A transformation that slants the shape of an object is called the shear transformation. There are two shear transformations **X-Shear** and **Y-Shear**. One shifts X coordinates values and other shifts Y coordinate values. However; in both the cases only one coordinate changes its coordinates and other preserves its values. Shearing is also termed as **Skewing**.

### X-Shear

The X-Shear preserves the Y coordinate and changes are made to X coordinates, which causes the vertical lines to tilt right or left as shown in below figure.



The transformation matrix for X-Shear can be represented as −

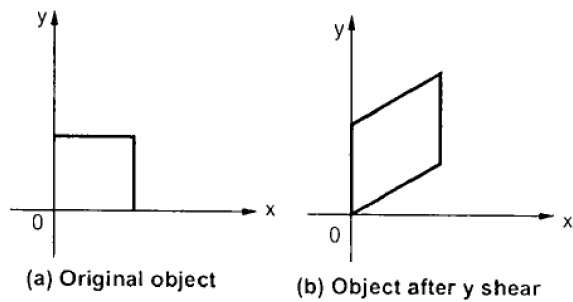
Xsh=⎡⎣⎢100shx10001⎤⎦⎥Xsh=[1shx0010001]

Y' = Y + Shy . X

X’ = X

### Y-Shear

The Y-Shear preserves the X coordinates and changes the Y coordinates which causes the horizontal lines to transform into lines which slopes up or down as shown in the following figure.



The Y-Shear can be represented in matrix from as −

Ysh⎡⎣⎢1shy0010001⎤⎦⎥Ysh[100shy10001]

X’ = X + Shx . Y

Y’ = Y

## Composite Transformation

If a transformation of the plane T1 is followed by a second plane transformation T2, then the result itself may be represented by a single transformation T which is the composition of T1 and T2 taken in that order. This is written as T = T1∙T2.

Composite transformation can be achieved by concatenation of transformation matrices to obtain a combined transformation matrix.

A combined matrix −

**[T][X] = [X] [T1] [T2] [T3] [T4] …. [Tn]**

Where [Ti] is any combination of

* Translation
* Scaling
* Shearing
* Rotation
* Reflection

The change in the order of transformation would lead to different results, as in general matrix multiplication is not cumulative, that is [A] . [B] ≠ [B] . [A] and the order of multiplication. The basic purpose of composing transformations is to gain efficiency by applying a single composed transformation to a point, rather than applying a series of transformation, one after another.

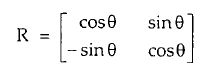
For example, to rotate an object about an arbitrary point (Xp, Yp), we have to carry out three steps −

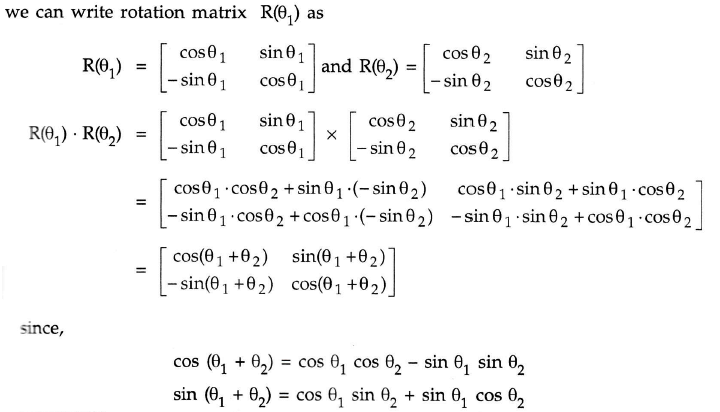
* Translate point (Xp, Yp) to the origin.
* Rotate it about the origin.
* Finally, translate the center of rotation back where it belonged.

**Q-12 Prove that two consecutive rotation are additive.**

To Show that the composition of two successive rotations are additive i.e. **R(Ɵ1). R(Ɵ2) = R(Ɵ1+ Ɵ2)**.

**Solution :-** The Rotation matrix R is given as,





**Q-13 Prove that transformation matrix multiplications are associative.**

**Q-14 Prove that two transformation matrix multiplications may be and may not be commutative**

**Q-15 Explain 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

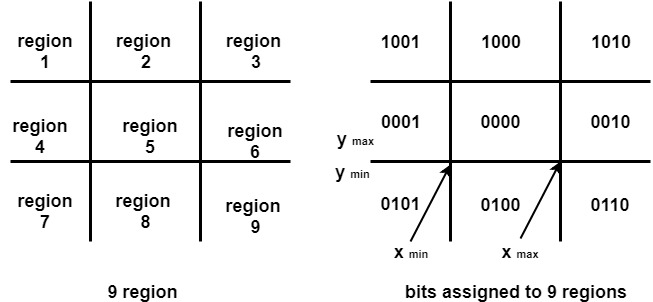
**1. Visible:** If a line lies within the window, i.e., both endpoints of the line lies within the window. A line is visible and will be displayed as it is.

**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 (x1,y2) and B (x2,y2) are endpoints of line.

xmin,xmax are coordinates of the window.

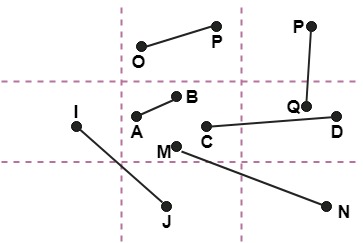
ymin,ymax are also coordinates of the window.  
          x1>xmax  
          x2>xmax  
          y1>ymax  
          y2>ymax  
          x1<xmin  
          x2<xmin  
          y1<ymin  
          y2<ymin

**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=(y2-y1 )(x2-x1)

**(a)** If bit 1 is "1" line intersects with left boundary of rectangle window  
                y3=y1+m(x-X1)  
                where X = Xwmin  
                where Xwminis the minimum value of X co-ordinate of window

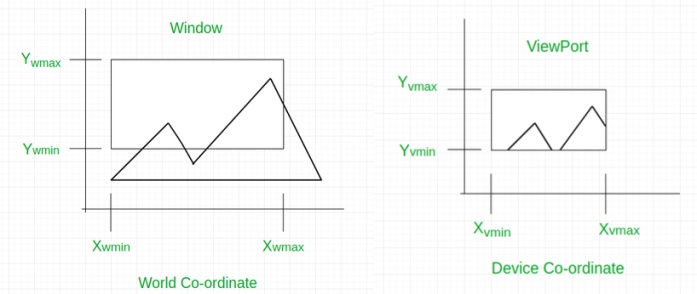
**(b)** If bit 2 is "1" line intersect with right boundary  
                y3=y1+m(X-X1)  
                where X = Xwmax  
                where X more is maximum value of X co-ordinate of the window

**(c)** If bit 3 is "1" line intersects with bottom boundary  
                X3=X1+(y-y1)/m  
                      where y = ywmin  
                ywmin is the minimum value of Y co-ordinate of the window

**(d)** If bit 4 is "1" line intersects with the top boundary  
                X3=X1+(y-y1)/m  
                      where y = ywmax  
                ywmax is the maximum value of Y co-ordinate of the window

Q-16 Explain window to viewport transformation.

**Window to Viewport Transformation**is the process of transforming 2D world-coordinate objects to device coordinates. Objects inside the world or clipping window are mapped to the viewport which is the area on the screen where world coordinates are mapped to be displayed.



***General Terms:***

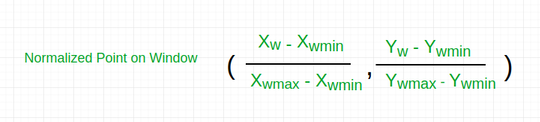
* **World coordinate –** It is the Cartesian coordinate w.r.t which we define the diagram, like Xwmin, Xwmax, Ywmin, Ywmax
* **Device Coordinate –**It is the screen coordinate where the objects are to be displayed, like Xvmin, Xvmax, Yvmin, Yvmax
* **Window –**It is the area on world coordinate selected for display.
* **ViewPort –**It is the area on the device coordinate where graphics is to be displayed.

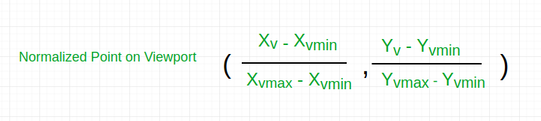
**Mathematical Calculation of Window to Viewport:**  
It may be possible that the size of the Viewport is much smaller or greater than the Window. In these cases, we have to increase or decrease the size of the Window according to the Viewport and for this, we need some mathematical calculations.

(xw, yw): A point on Window

(xv, yv): Corresponding point on Viewport

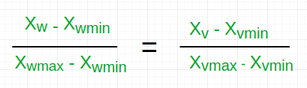
* we have to calculate the point **(xv, yv)**



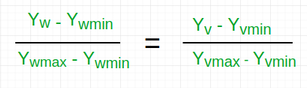


* Now the relative position of the object in Window and Viewport are same.

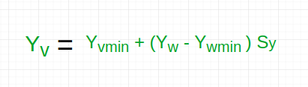
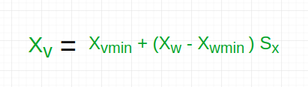
For x coordinate,



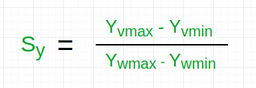
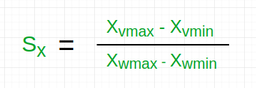
For y coordinate,



* so, after calculating for x and y coordinate, we get



* where sx is the scaling factor of x coordinate and sy is the scaling factor of y coordinate



**Q-17 Explain N-L-N line clipping algorithm**.

**Q-18.Explain liang barsky line clipping algorithm.**

The **Liang-Barsky algorithm** is a line clipping algorithm. This algorithm is more efficient than Cohen–Sutherland line clipping algorithm and can be extended to 3-Dimensional clipping. This algorithm is considered to be the faster parametric line-clipping algorithm. The following concepts are used in this clipping:

1. The parametric equation of the line.
2. The inequalities describing the range of the clipping window which is used to determine the intersections between the line and the clip window.

The parametric equation of a line can be given by,

X = x1 + t(x2-x1)

Y = y1 + t(y2-y1)

Where, t is between 0 and 1.

Then, writing the point-clipping conditions in the parametric form:

xwmin <= x1 + t(x2-x1) <= xwmax

ywmin <= y1 + t(y2-y1) <= ywmax

The above 4 inequalities can be expressed as,

tpk <= qk

Where k = 1, 2, 3, 4 (correspond to the left, right, bottom, and top boundaries, respectively).

The p and q are defined as,

p1 = -(x2-x1), q1 = x1 - xwmin (Left Boundary)

p2 = (x2-x1), q2 = xwmax - x1 (Right Boundary)

p3 = -(y2-y1), q3 = y1 - ywmin (Bottom Boundary)

p4 = (y2-y1), q4 = ywmax - y1 (Top Boundary)

When the line is parallel to a view window boundary, the p value for that boundary is zero.  
When pk < 0, as t increase line goes from the outside to inside (entering).  
When pk > 0, line goes from inside to outside (exiting).  
When pk = 0 and qk < 0 then line is trivially invisible because it is outside view window.  
When pk = 0 and qk > 0 then the line is inside the corresponding window boundary.

Using the following conditions, the position of line can be determined:

| Condition | Position of line |
| --- | --- |
| pk = 0 | parallel to the clipping boundaries |
| pk = 0 and qk < 0 | completely outside the boundary |
| pk = 0 and qk >= 0 | inside the parallel clipping boundary |
| pk < 0 | line proceeds from outside to inside |
| pk > 0 | line proceeds from inside to outside |

Parameters t1 and t2 can be calculated that define the part of line that lies within the clip rectangle.  
When,

1. pk < 0, maximum(0, qk/pk) is taken.
2. pk > 0, minimum(1, qk/pk) is taken.

If t1 > t2, the line is completely outside the clip window and it can be rejected. Otherwise, the endpoints of the clipped line are calculated from the two values of parameter t.

**Algorithm –**

1. Set tmin=0, tmax=1.
2. Calculate the values of t (t(left), t(right), t(top), t(bottom)),  
   (i) If t < tmin ignore that and move to the next edge.  
   (ii) else separate the t values as entering or exiting values using the inner product.  
   (iii) If t is entering value, set tmin = t; if t is existing value, set tmax = t.
3. If tmin < tmax, draw a line from (x1 + tmin(x2-x1), y1 + tmin(y2-y1)) to (x1 + tmax(x2-x1), y1 + tmax(y2-y1))
4. If the line crosses over the window, (x1 + tmin(x2-x1), y1 + tmin(y2-y1)) and (x1 + tmax(x2-x1), y1 + tmax(y2-y1)) are the intersection point of line and edge.

**19.Describe polygon clipping algorithm weiler athertan and Sutherland hodgeman.**

Weiler Atherton Polygon Clipping Algorithm is an algorithm made to allow clipping of even concave algorithms to be possible. Unlike Sutherland – Hodgman polygon clipping algorithm, this algorithm is able to clip concave polygons without leaving any residue behind.

### Algorithm:

**1. First make a list of all intersection points namely i1, i2, i3, ...**

**2. Classify those intersection points as entering or exiting.**

**3. Now, make two lists, one for the clipping polygon, and the other**

**for the clipped polygon.**

**4. Fill both the lists up in such a way that the intersection points**

**lie between the correct vertices of each of the polygon. That is**

**the clipping polygon list is filled up with all the vertices of**

**the clipping polygon along with the intersecting points lying**

**between the corresponding vertices.**

**5. Now, start at the 'to be clipped' polygon's list.**

**6. Choose the first intersection point which has been labelled as**

**an entering point. Follow the points in the list (looping back to**

**the top of the list, in case the list ends) and keep on pushing**

**them into a vector or something similar of the sorts. Keep on following**

**the list until an exiting intersection point is found.**

**7. Now switch the list to the 'polygon that is clipping' list, and find**

**the exiting the intersection that was previously encountered. Now keep**

**on following the points in this list (similar to how we followed the**

**previous list) until the entering intersection point is found (the**

**one that was found in the previous 'to be clipped' polygon's list).**

**8. This vector now formed by pushing all the encountered points in the**

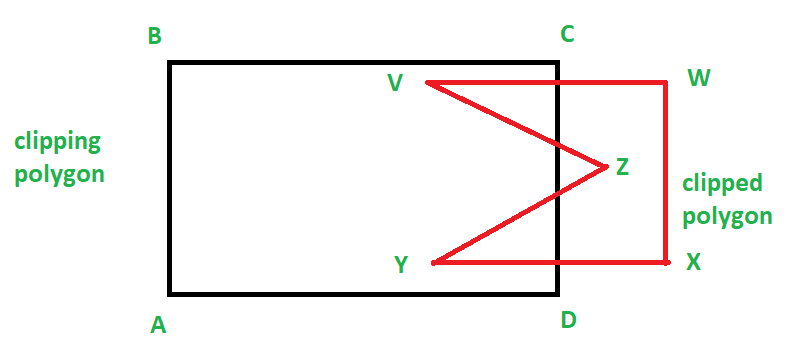
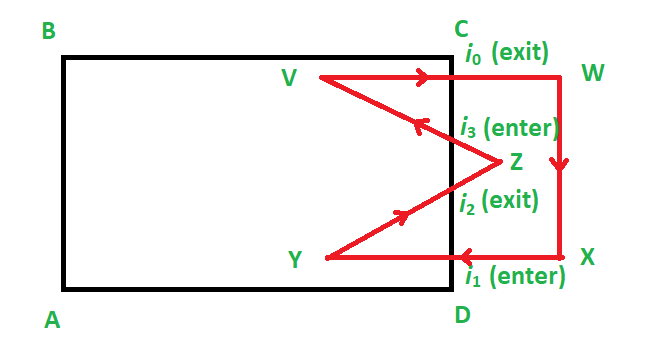
**two lists, is now the clipped polygon (one of the many clipped**

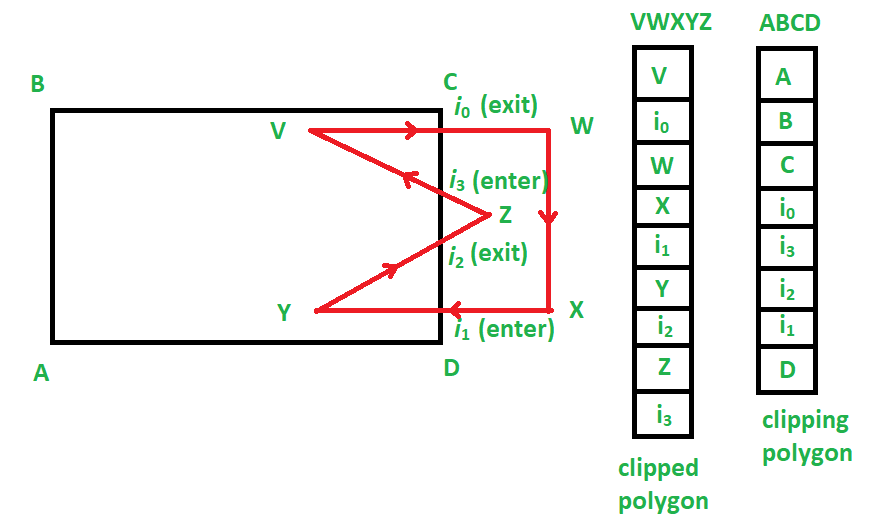
**polygons if any of the clipping polygons is concave).**

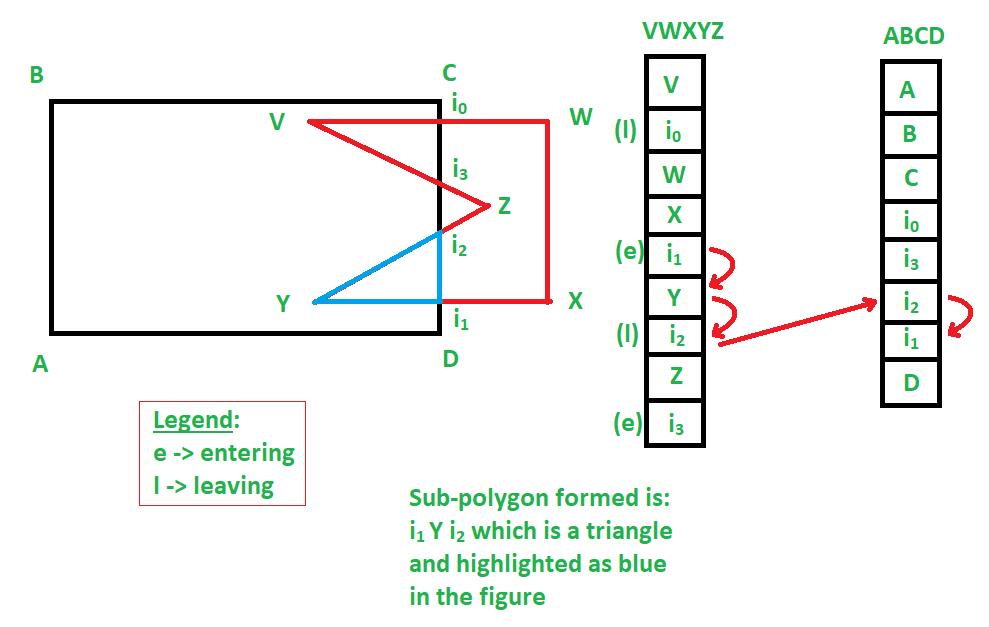
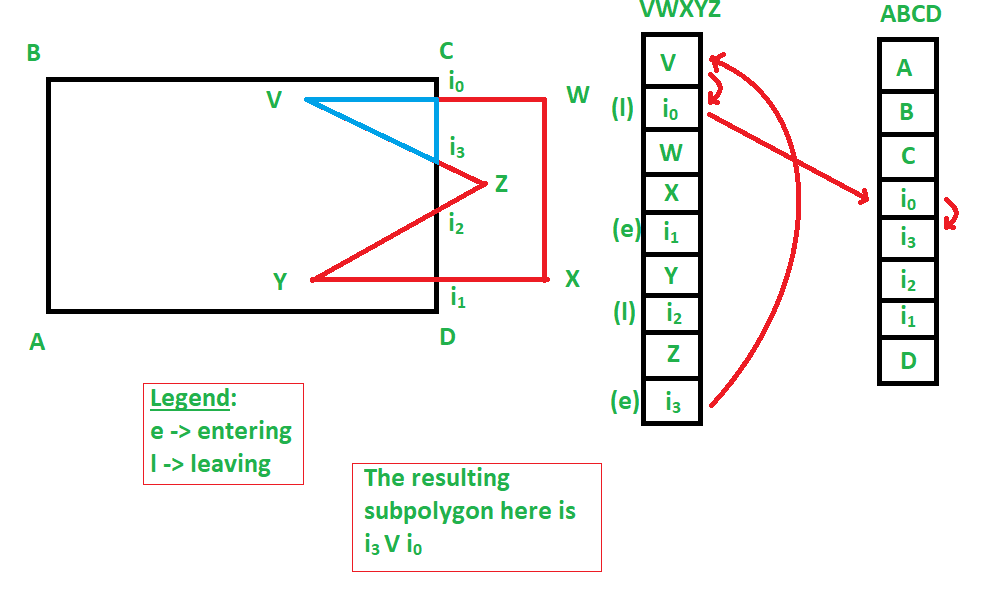
**9. Repeat this clipping procedure (i.e. from step 5) until all the**

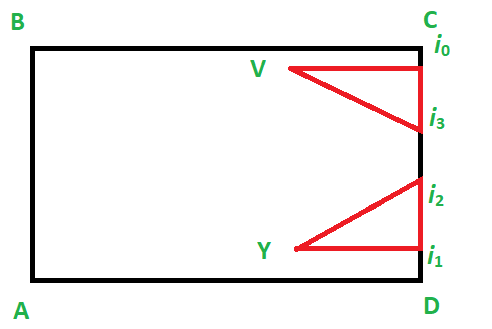
**entering intersection points have been visited once.**

### Explanation:

**1. Finding all the intersection points and grouping them**  
Here, let there be a polygon ABCD and another polygon VWXYZ. Let ABCD be the clipping polygon and let VWXYZ be the clipped polygon.  
  
So, we can find the intersection points using any method. For example, we can find the intersecting points separately and then find for each intersecting point find if it is entering or leaving, or, we can use Cyrus Beck and find all the intersecting points and also get if a point is entering or exiting. Refer [Cyrus Beck](https://www.geeksforgeeks.org/cyrus-beck/) for more information on this algorithm.  


**2. Making and filling of two lists**  
Now, we make two lists. One for the clipping polygon and one for the clipped polygon.  
Now this is how we fill it:  


**3. Running of the algorithm**  
We start at the clipped polygon’s list, i.e. VWXYZ.  
Now, we find the first intersecting point that is entering. Hence we choose i1.  
From here we begin the making of the list of vertices (or vector) to make a clipped sub-polygon.  
  
According to the given example, **i1 Y i2** is a clipped sub-polygon.  
Similarly, we get:  
  
**i0 V i3** as another sub-polygon also.

Hence, we were able to get two sub-polygons as a result of this polygon clipping, which involved a concave polygon, which resulted in:  
  
Similarly, this clipping works for convex polygons.

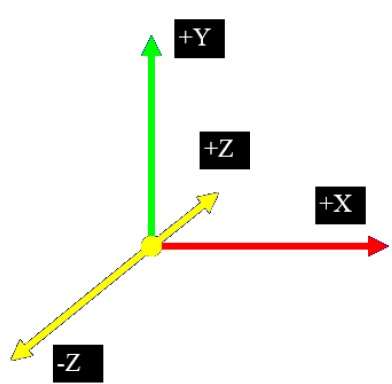
### Limitations:

This polygon clipping algorithm does not work for self – intersecting polygons, although some methods have been proposed to be able to solve this issue also, and have successfully worked.

**Q-20. Explain 3D display methods in detail.**

In the 2D system, we use only two coordinates X and Y but in 3D, an extra coordinate Z is added. 3D graphics techniques and their application are fundamental to the entertainment, games, and computer-aided design industries. It is a continuing area of research in scientific visualization.

Furthermore, 3D graphics components are now a part of almost every personal computer and, although traditionally intended for graphics-intensive software such as games, they are increasingly being used by other applications.

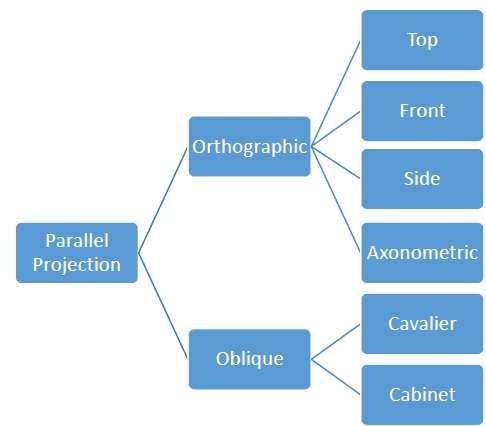


Parallel Projection

Parallel projection discards z-coordinate and parallel lines from each vertex on the object are extended until they intersect the view plane. In parallel projection, we specify a direction of projection instead of center of projection.

In parallel projection, the distance from the center of projection to project plane is infinite. In this type of projection, we connect the projected vertices by line segments which correspond to connections on the original object.

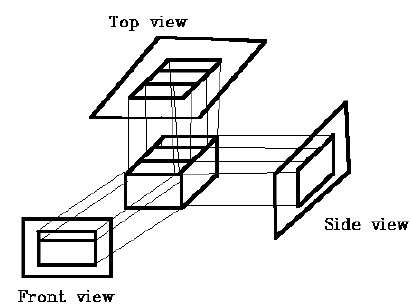
Parallel projections are less realistic, but they are good for exact measurements. In this type of projections, parallel lines remain parallel and angles are not preserved. Various types of parallel projections are shown in the following hierarchy.



Orthographic Projection

In orthographic projection the direction of projection is normal to the projection of the plane. There are three types of orthographic projections −

* Front Projection
* Top Projection
* Side Projection

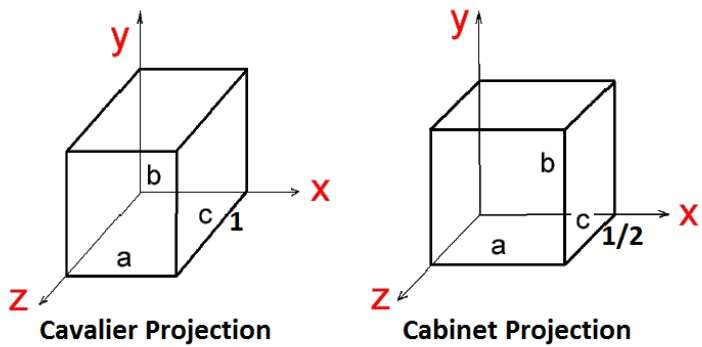


Oblique Projection

In oblique projection, the direction of projection is not normal to the projection of plane. In oblique projection, we can view the object better than orthographic projection.

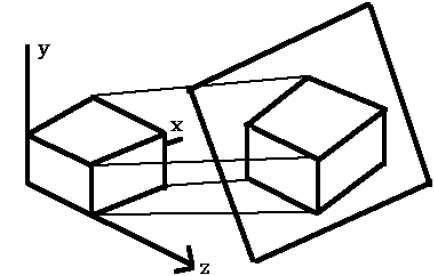
There are two types of oblique projections − **Cavalier** and **Cabinet**. The Cavalier projection makes 45° angle with the projection plane. The projection of a line perpendicular to the view plane has the same length as the line itself in Cavalier projection. In a cavalier projection, the foreshortening factors for all three principal directions are equal.

The Cabinet projection makes 63.4° angle with the projection plane. In Cabinet projection, lines perpendicular to the viewing surface are projected at ½ their actual length. Both the projections are shown in the following figure −



Isometric Projections

Orthographic projections that show more than one side of an object are called **axonometric orthographic projections**. The most common axonometric projection is an **isometric projection** where the projection plane intersects each coordinate axis in the model coordinate system at an equal distance. In this projection parallelism of lines are preserved but angles are not preserved. The following figure shows isometric projection −

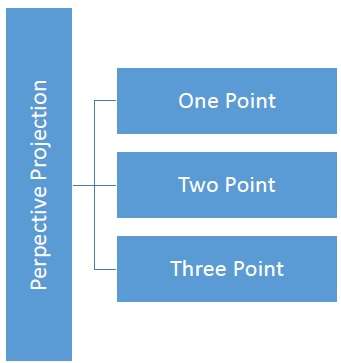


Perspective Projection

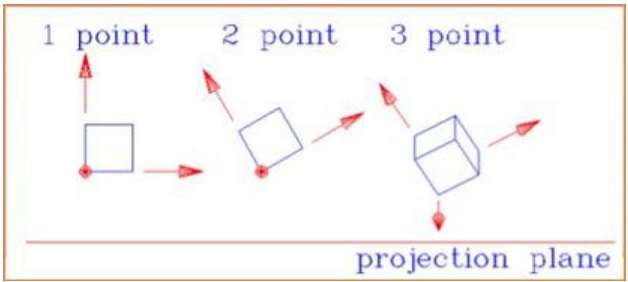
In perspective projection, the distance from the center of projection to project plane is finite and the size of the object varies inversely with distance which looks more realistic.

The distance and angles are not preserved and parallel lines do not remain parallel. Instead, they all converge at a single point called **center of projection** or **projection reference point**. There are 3 types of perspective projections which are shown in the following chart.

* **One point** perspective projection is simple to draw.
* **Two point** perspective projection gives better impression of depth.
* **Three point** perspective projection is most difficult to draw.



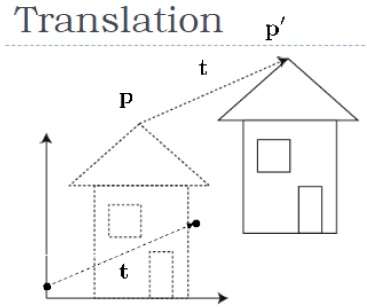
The following figure shows all the three types of perspective projection −



Translation

In 3D translation, we transfer the Z coordinate along with the X and Y coordinates. The process for translation in 3D is similar to 2D translation. A translation moves an object into a different position on the screen.

The following figure shows the effect of translation −



A point can be translated in 3D by adding translation coordinate (tx,ty,tz)(tx,ty,tz) to the original coordinate X,Y,ZX,Y,Z to get the new coordinate X′,Y′,Z′X′,Y′,Z′.

T=⎡⎣⎢⎢⎢⎢100tx010ty001tz0001⎤⎦⎥⎥⎥⎥T=[100001000010txtytz1]

P’ = P∙T

[X'Y'Z'1]=[XYZ1]⎡⎣⎢⎢⎢⎢100tx010ty001tz0001⎤⎦⎥⎥⎥⎥[X′Y′Z′1]=[XYZ1][100001000010txtytz1]

=[X+txY+tyZ+tz1]

**Q-21.What is anti-aliasing? Explain its methods**

**Antialiasing** is a technique used in computer graphics to remove the aliasing effect. The aliasing effect is the appearance of jagged edges or “jaggies” in a rasterized image (an image rendered using pixels). The problem of jagged edges technically occurs due to distortion of the image when scan conversion is done with sampling at a low frequency, which is also known as Undersampling. Aliasing occurs when real-world objects which comprise of smooth, continuous curves are rasterized using pixels.

Cause of anti-aliasing is **Undersampling**. Undersampling results in loss of information of the picture. Undersampling occurs when sampling is done at a frequency lower than Nyquist sampling frequency. To avoid this loss, we need to have our sampling frequency atleast twice that of highest frequency occurring in the object.

This minimum required frequency is referred to as **Nyquist sampling frequency (fs):**

fs =2\*fmax

This can also be stated as that our sampling interval should be no larger than half the cycle interval. This maximum required the sampling interval is called Nyquist sampling interval Δxs:

Δxs = Δxcycle/2

Where Δxcycle=1/fmax

**Methods of Antialiasing (AA) –**  
Aliasing is removed using four methods: Using high-resolution display, Post filtering (Supersampling), Pre-filtering (Area Sampling), Pixel phasing. These are explained as following below.

1. **Using high-resolution display:**  
   One way to reduce aliasing effect and increase sampling rate is to simply display objects at a higher resolution. Using high resolution, the jaggies become so small that they become indistinguishable by the human eye. Hence, jagged edges get blurred out and edges appear smooth.

**Practical applications:**  
For example retina displays in Apple devices, OLED displays have high pixel density due to which jaggies formed are so small that they blurred and indistinguishable by our eyes.

1. **Post filtering (Supersampling):**  
   In this method, we are increasing the sampling resolution by treating the screen as if it’s made of a much more fine grid, due to which the effective pixel size is reduced. But the screen resolution remains the same. Now, intensity from each subpixel is calculated and average intensity of the pixel is found from the average of intensities of subpixels. Thus we do sampling at higher resolution and display the image at lower resolution or resolution of the screen, hence this technique is called supersampling. This method is also known as post filtration as this procedure is done after generating the rasterized image.

**Practical applications:**  
In gaming, SSAA (Supersample Antialiasing) or FSAA (full-scene antialiasing) is used to create best image quality. It is often called the pure AA and hence is very slow and has a very high computational cost. This technique was widely used in early days when better AA techniques were not available. Different modes of SSAA available are: 2X, 4X, 8X, etc. denoting that sampling is done x times (more than) the current resolution.

A better style of AA is MSAA (multisampling Antialiasing) which is a faster and approximate style of supersampling AA.It has lesser computational cost. Better and sophisticated supersampling techniques are developed by graphics card companies like CSAA by NVIDIA and CFAA by AMD.

1. **Pre-filtering (Area Sampling):**  
   In area sampling, pixel intensities are calculated proportional to areas of overlap of each pixel with objects to be displayed. Here pixel color is computed based on the overlap of scene’s objects with a pixel area.  
     
   **For example:** Suppose, a line passes through two pixels. The pixel covering bigger portion(90%) of line displays 90% intensity while less area(10%) covering pixel displays 10-15% intensity. If pixel area overlaps with different color areas, then the final pixel color is taken as an average of colors of the overlap area. This method is also known as pre-filtering as this procedure is done BEFORE generating the rasterized image. It’s done using some graphics primitive algorithms.
2. **Pixel phasing:**  
   It’s a technique to remove aliasing. Here pixel positions are shifted to nearly approximate positions near object geometry. Some systems allow the size of individual pixels to be adjusted for distributing intensities which is helpful in pixel phasing.

**Other Applications of antialiasing techniques:**

1. **Compensating for line intensity differences:**  
   When a horizontal line and a diagonal line plotted on the raster display, the number of pixels required to display both lines is same, even though the diagonal line is 1.414 times larger than the horizontal line. This leads to a decrease in the intensity of the longer line. To compensate for this decrease in intensity, the intensity of pixels is assigned according to the length of line using anti-aliasing techniques.
2. **Anti-aliasing area boundaries:**  
   Anti-aliasing concepts can also be applied to remove jaggies along area boundaries. These procedures can be applied to scanline algorithms to smoothen out area boundaries .if repositioning of pixels is possible then pixel positions are adjusted to positions closer to area boundaries. Other methods adjust pixel intensity at a boundary position according to the percent of pixel area inside the boundary. These methods effectively smoothen out area boundaries.