UNIT-1 : 2D AND 3D TRANSFORMATION & VIEWING

2D Transformation

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 –

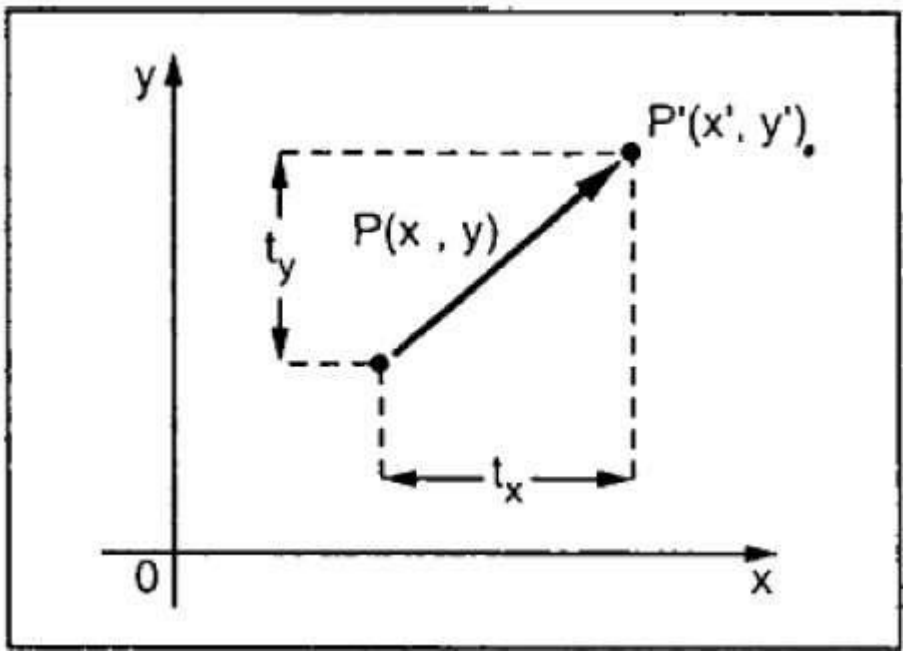
- Translate the coordinates,
- Rotate the translated coordinates, and then
- Scale the rotated coordinates to complete the composite transformation.
-

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 P(X, Y) can be converted to homogenous coordinates by P' (X_h, Y_h, 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 (t_x, t_y) to the original coordinate (X, Y) to get the new coordinate (X', Y').



From the above figure, you can write that –

$$X' = X + t_x$$

$$Y' = Y + t_y$$

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

$$P = \frac{[X]}{[Y]} \qquad p' = \frac{[X']}{[Y']} \qquad T = \frac{[t_x]}{[t_y]}$$

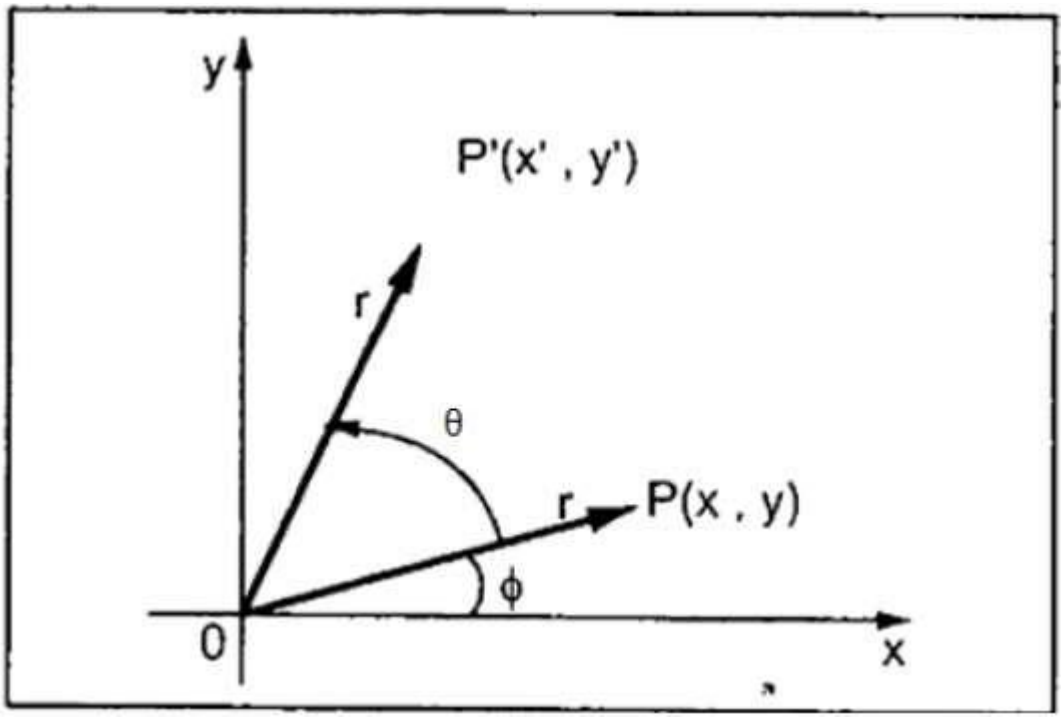
We can write it as –

$$P' = P + T$$

Rotation

In rotation, we rotate the object at particular angle θ (theta) from its origin. From the following figure, we can see that the point P(X, 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').



Using standard trigonometric the original coordinate of point P(X, Y) can be represented as –

$$X = r\,cos\,\phi.....(1)$$

$$Y = r\,sin\,\phi.....(2)$$

Same way we can represent the point P′ (X′, Y′) as –

$$x' = r\,cos\,(\phi + \theta) = r\,cos\,\phi\,cos\,\theta - r\,sin\,\phi\,sin\,\theta.....(3)$$

$$y' = r\,sin\,(\phi + \theta) = r\,cos\,\phi\,sin\,\theta + r\,sin\,\phi\,cos\,\theta.....(4)$$

Substituting equation (1) & (2) in (3) & (4) respectively, we will get

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

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

Representing the above equation in matrix form,

$$[X'Y'] = [XY]\begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} OR$$

P′ = P . R

Where R is the rotation matrix

Where R is the rotation matrix

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

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 –

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

$$= \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} (\because cos(-\theta) = cos\theta\ and\ sin(-\theta) = -sin\theta)$$

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, Y), the scaling factors are (S_X, S_Y), and the produced coordinates are (X′, Y′). This can be mathematically represented as shown below –

$$\mathbf{X' = X \cdot S_X \text{ and } Y' = Y \cdot S_Y}$$

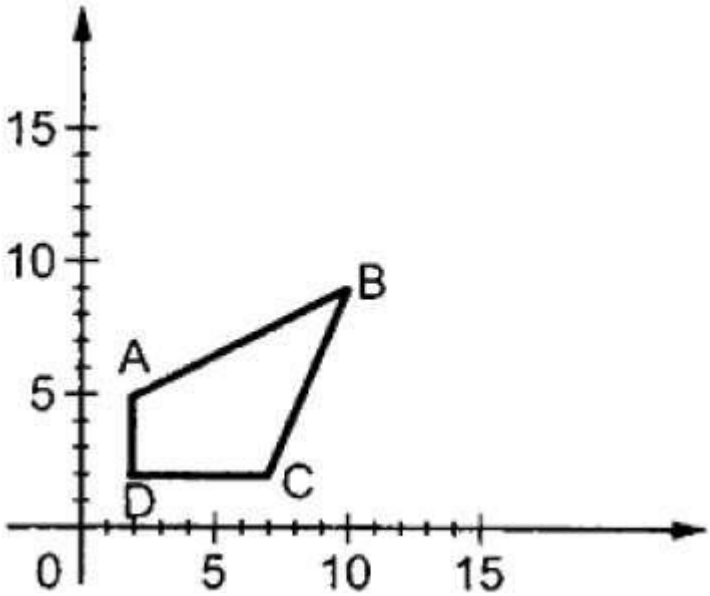
The scaling factor S_X, S_Y scales the object in X and Y direction respectively. The above equations can also be represented in matrix form as below –

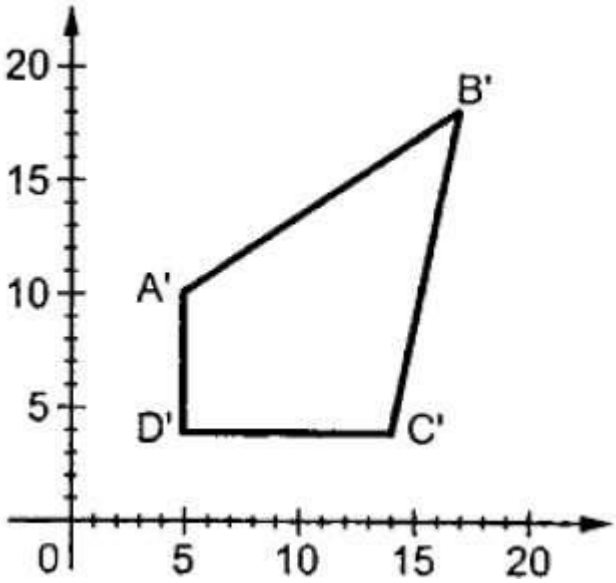
$$(X'Y')=(XY)[S_x00S_y](X'Y')=(XY)[Sx00Sy]$$

OR

$$\mathbf{P' = P \cdot S}$$

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



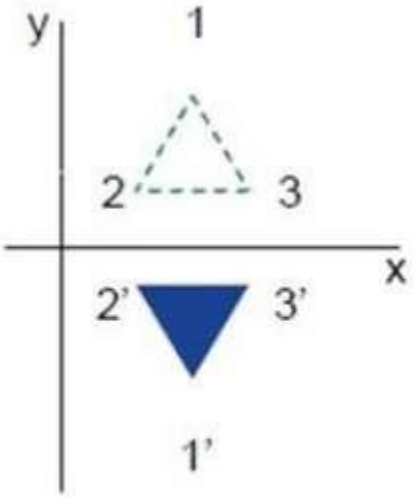


If we provide values less than 1 to the scaling factor S, then we can reduce the size of the object. If we provide values greater than 1, then we can increase the size of the object.

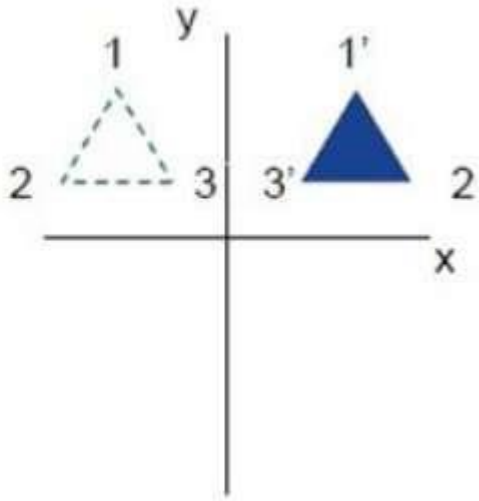
Reflection

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.

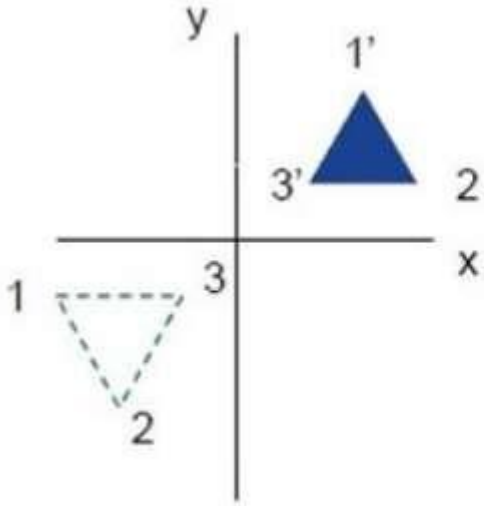
The following figures show reflections with respect to X and Y axes, and about the origin respectively.



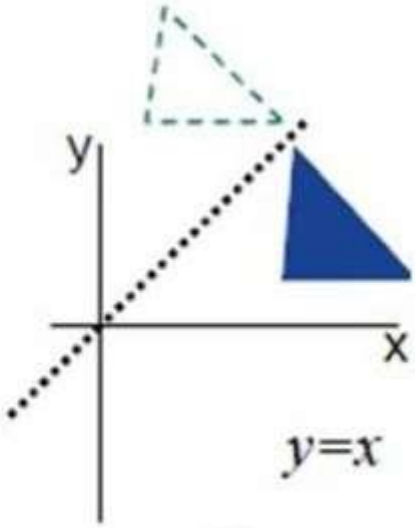
(a)



(b)



(c)



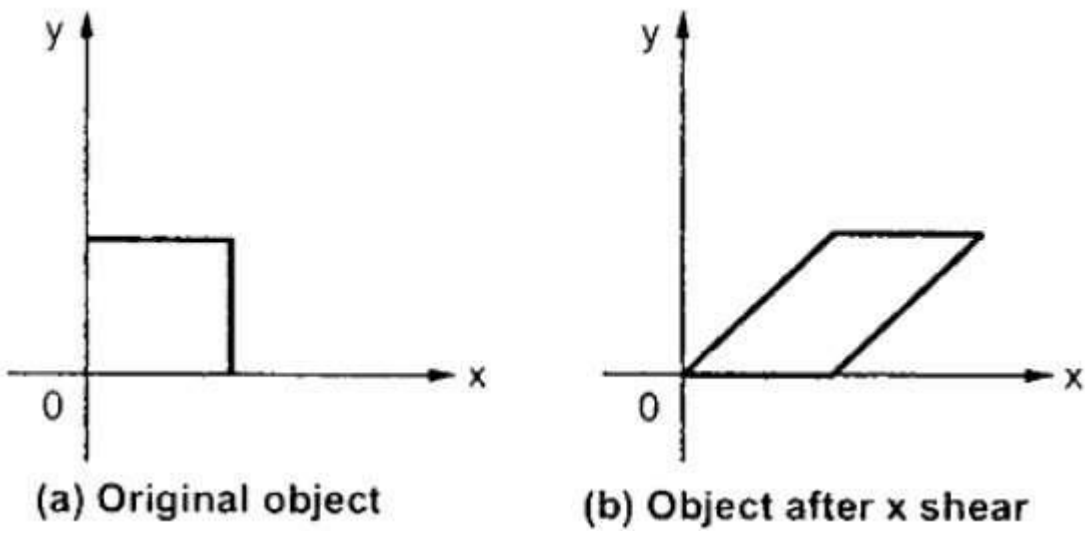
(d)

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 –

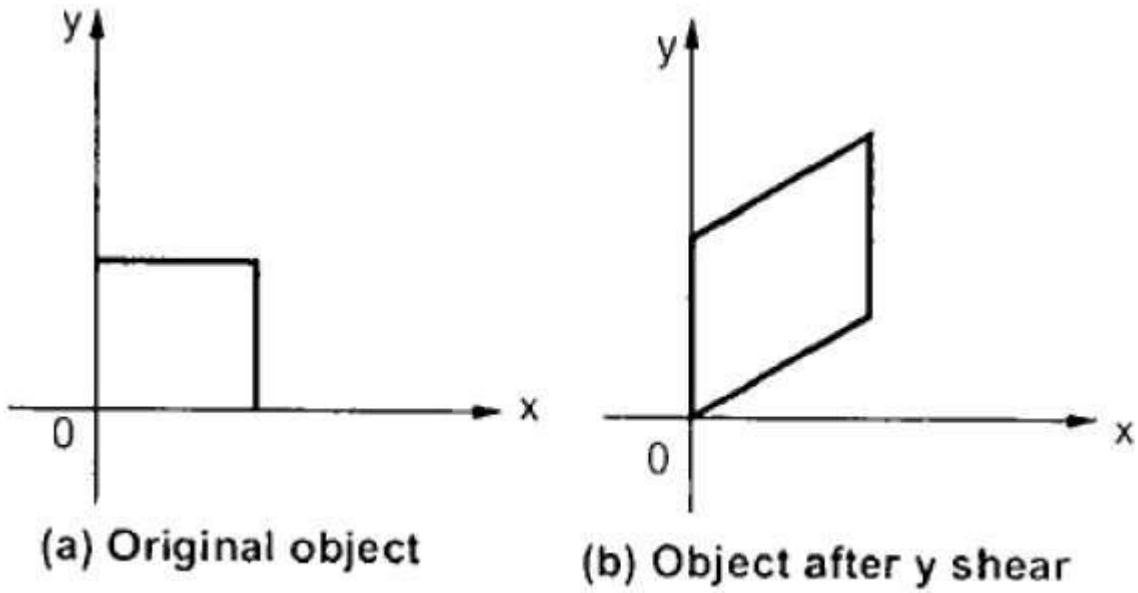
$$X_{sh} = \begin{bmatrix} 1 & shx & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$Y' = Y + Sh_y \cdot X$$

$$X' = X$$

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 –

$$Y_{sh} \begin{bmatrix} 1 & 0 & 0 \\ shy & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$X' = X + Sh_X \cdot Y$$

$$Y' = Y$$

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 (X_p, Y_p), we have to carry out three steps −

- Translate point (X_p, Y_p) to the origin.
- Rotate it about the origin.
- Finally, translate the center of rotation back where it belonged.

Explain Viewing transformation pipeline

The Viewing Transformation Pipeline:-

We know that the picture is stored in the computer memory using any convenient Cartesian co-ordinate system, referred to as World Co-Ordinate System (WCS). However, when picture is displayed on the display device it is measured in Physical Device Co-Ordinate System (PDCS) corresponding to the display device. Therefore, displaying an image of a picture involves mapping the co-ordinates of the Points and lines that form the picture into the appropriate physical device co-ordinate where the image is to be displayed. This mapping of co-ordinates is achieved with the use of co-ordinate transformation known as viewing transformation.

The viewing transformation which maps picture co-ordinates in the WCS to display co-ordinates in PDCS is performed by the following transformations.

- Converting world co-ordinates to viewing co-ordinates.
- Normalizing viewing co-ordinates.
- Converting normalized viewing co-ordinates to device co-ordinates.

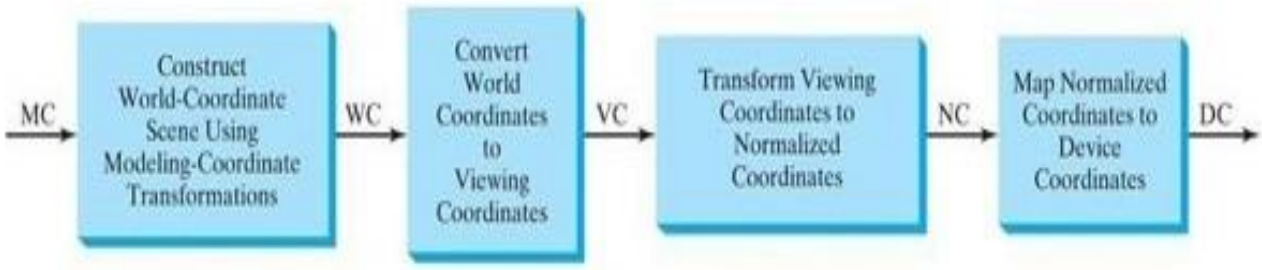


Fig. (c) Two-dimensional viewing transformation pipeline

The steps involved in viewing transformation:-

1. Construct the scene in world co-ordinate using the output primitives and attributes.
2. Obtain a particular orientation for the window by setting a two-dimensional viewing co-ordinate system in the world co-ordinate plane and define a window in the viewing co-ordinate system.
3. Use viewing co-ordinates reference frame to provide a method for setting up arbitrary orientations for rectangular windows.
4. Once the viewing reference frame is established, transform descriptions in world co-ordinates to viewing co-ordinates.
5. Define a view port in normalized co-ordinates and map the viewing co-ordinates description of the scene to normalized co-ordinates.
6. Clip all the parts of the picture which lie outside the viewport.

Define window and viewport. Derive window to viewport transformation.

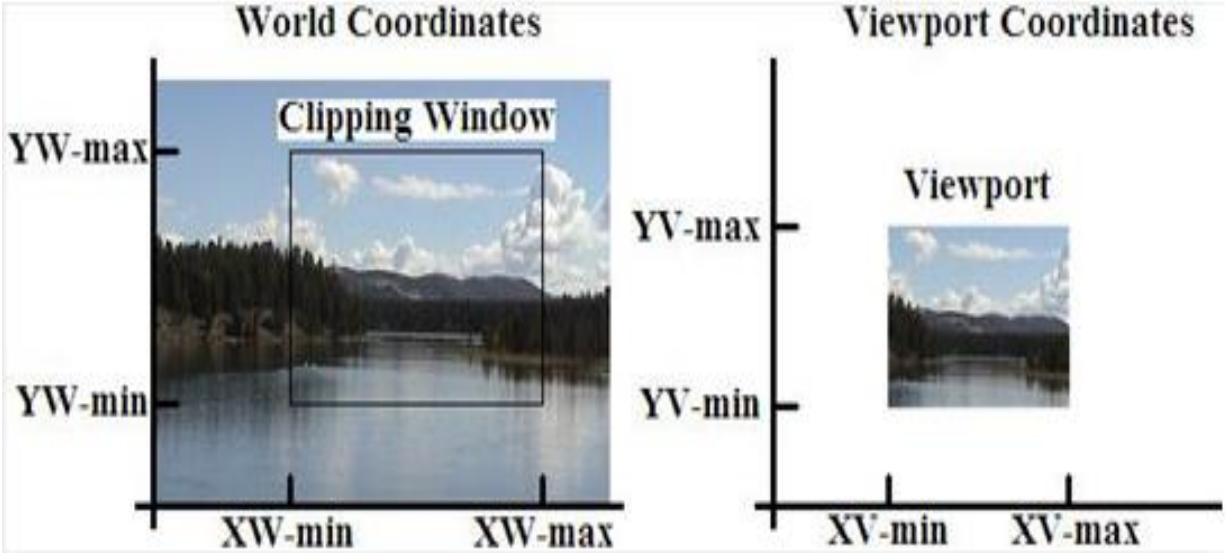
1. A world-coordinate area selected for display is called a window.
2. In computer graphics, a window is a graphical control element.
3. It consists of a visual area containing some of the graphical user interface of the program it belongs to and is framed by a window decoration.
4. A window defines a rectangular area in world coordinates. You define a window with a GWINDOW statement. You can define the window to be larger than, the same size as, or smaller than the actual range of data values, depending on whether you want to show all of the data or only part of the data.

Viewport:

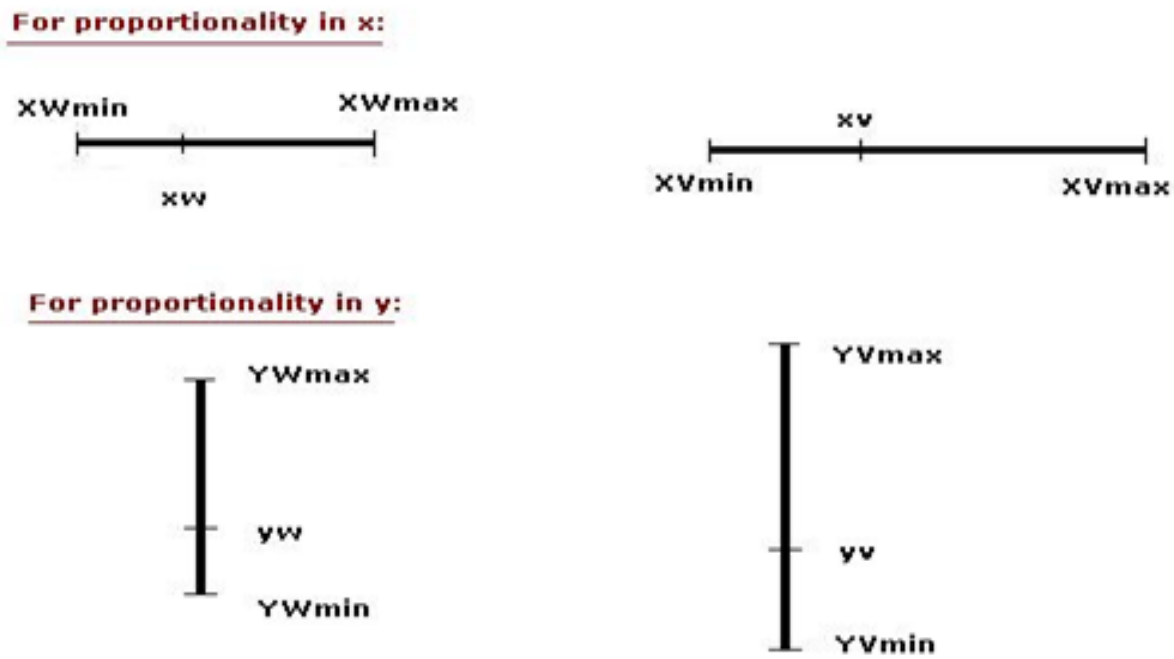
1. An area on a display device to which a window is mapped is called a viewport.
2. A viewport is a polygon viewing region in computer graphics. The viewport is an area expressed in rendering-device-specific coordinates, e.g. pixels for screen coordinates, in which the objects of interest are going to be rendered.
3. A viewport defines in normalized coordinates a rectangular area on the display device where the image of the data appears. You define a viewport with the GPORT command. You can have your graph take up the entire display device or show it in only a portion, say the upper-right part.

Window to viewport transformation:

1. Window-to-Viewport transformation is the process of transforming a two-dimensional, world-coordinate scene to device coordinates.
2. In particular, objects inside the world or clipping window are mapped to the viewport. The viewport is displayed in the interface window on the screen.
3. In other words, the clipping window is used to select the part of the scene that is to be displayed. The viewport then positions the scene on the output device.
4. **Example:**



1. This transformation involves developing formulas that start with a point in the world window, say (xw, yw).
2. The formula is used to produce a corresponding point in viewport coordinates, say (xv, yv). We would like for this mapping to be "proportional" in the sense that if xw is 30% of the way from the left edge of the world window, then xv is 30% of the way from the left edge of the viewport.
3. Similarly, if yw is 30% of the way from the bottom edge of the world window, then yv is 30% of the way from the bottom edge of the viewport. The picture below shows this proportionality.



1. Using this proportionality, the following ratios must be equal.

$$\frac{xv - xv_{min}}{xv_{max} - xv_{min}} = \frac{xw - xw_{min}}{xw_{max} - xw_{min}}$$

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

1. By solving these equations for the unknown viewport position (xv, yv), the following becomes true:

$$xv = S_x xw + t_x$$

$$yv = S_y yw + t_y$$

1. And the translation factors (Tx, Ty) would be:

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

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

1. The position of the viewport can be changed allowing objects to be viewed at different positions on the Interface Window.
2. Multiple viewports can also be used to display different sections of a scene at different screen positions. Also, by changing the dimensions of the viewport, the size and proportions of the objects being displayed can be manipulated.
3. Thus, a zooming affect can be achieved by successively mapping different dimensioned clipping windows on a fixed sized viewport.
4. If the aspect ratio of the world window and the viewport are different, then the image may look distorted.

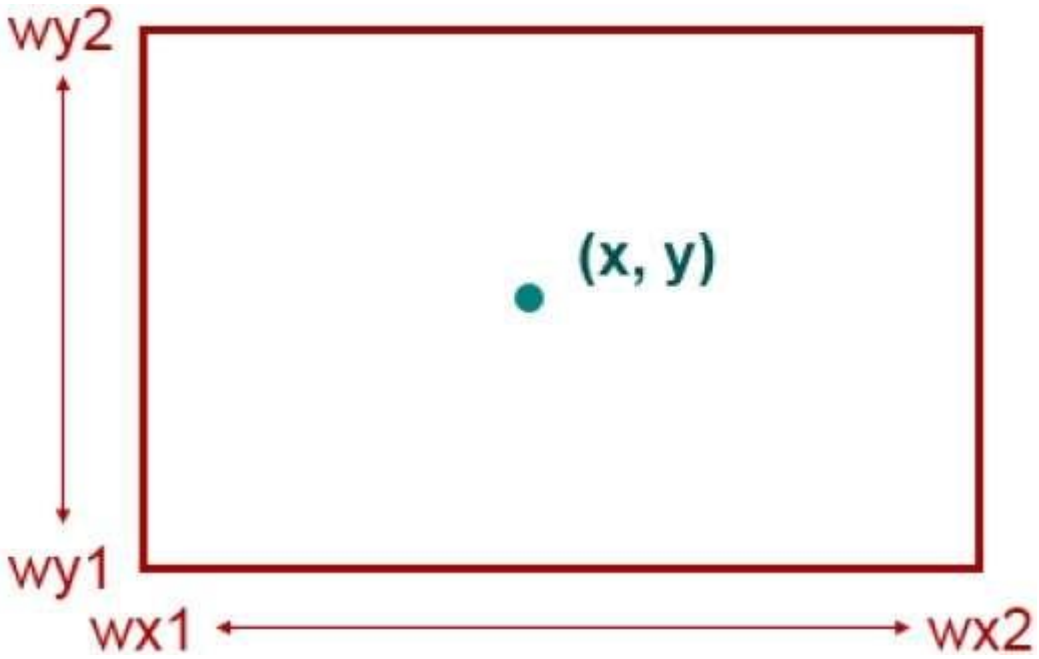
Viewing & Clipping

The primary use of clipping in computer graphics is to remove objects, lines, or line segments that are outside the viewing pane. The viewing transformation is insensitive to the position of points relative to the viewing volume – especially those points behind the viewer – and it is necessary to remove these points before generating the view.

Point Clipping

Clipping a point from a given window is very easy. Consider the following figure, where the rectangle indicates the window. Point clipping tells us whether the given point (X, Y) is within the given window or not; and decides whether we will use the minimum and maximum coordinates of the window.

The X-coordinate of the given point is inside the window, if X lies in between Wx1 ≤ X ≤ Wx2. Same way, Y coordinate of the given point is inside the window, if Y lies in between Wy1 ≤ Y ≤ Wy2.

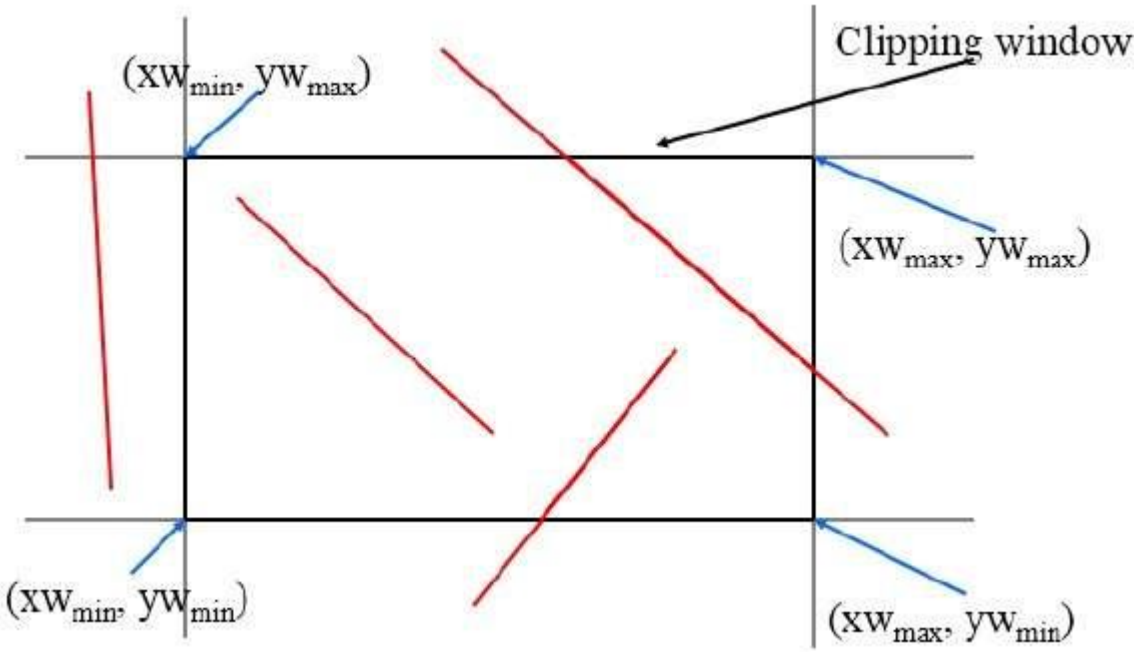


Line Clipping

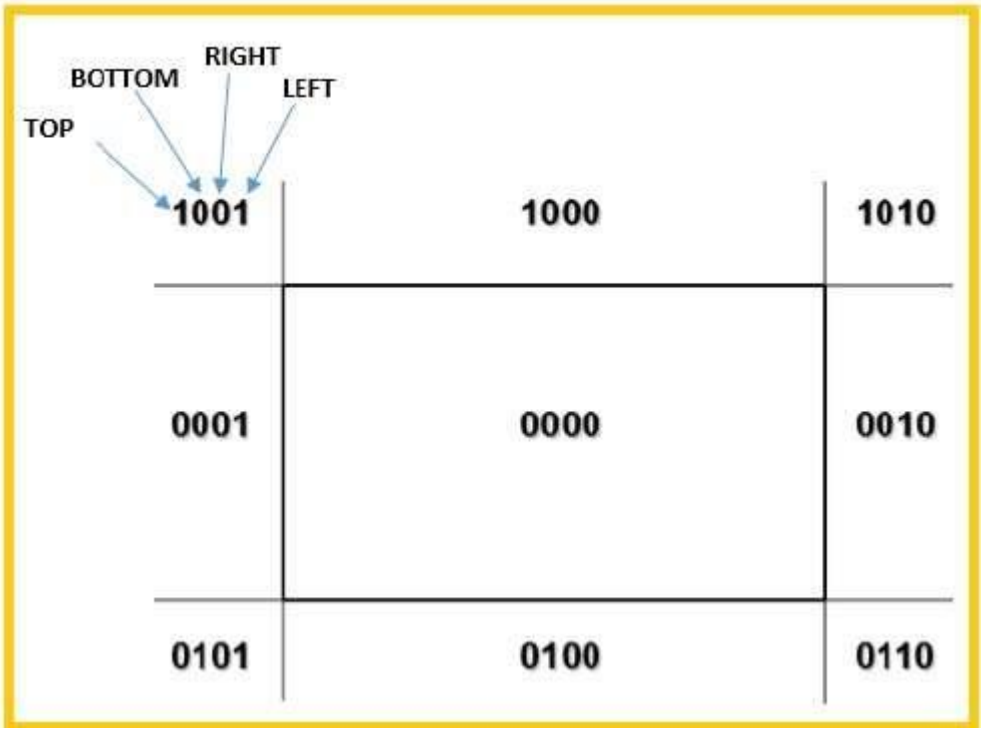
The concept of line clipping is same as point clipping. In line clipping, we will cut the portion of line which is outside of window and keep only the portion that is inside the window.

Cohen-Sutherland Line Clippings

This algorithm uses the clipping window as shown in the following figure. The minimum coordinate for the clipping region is (XW_{min}, YW_{min})(XWmin,YWmin) and the maximum coordinate for the clipping region is (XW_{max}, YW_{max})(XWmax,YWmax).



We will use 4-bits to divide the entire region. These 4 bits represent the Top, Bottom, Right, and Left of the region as shown in the following figure. Here, the **TOP** and **LEFT** bit is set to 1 because it is the **TOP-LEFT** corner.



There are 3 possibilities for the line –

- Line can be completely inside the window (This line should be accepted).
- Line can be completely outside of the window (This line will be completely removed from the region).
- Line can be partially inside the window (We will find intersection point and draw only that portion of line that is inside region).

Algorithm

Step 1 – Assign a region code for each endpoints.

Step 2 – If both endpoints have a region code **0000** then accept this line.

Step 3 – Else, perform the logical **AND**operation for both region codes.

Step 3.1 – If the result is not **0000,** then reject the line.

Step 3.2 – Else you need clipping.

Step 3.2.1 – Choose an endpoint of the line that is outside the window.

Step 3.2.2 – Find the intersection point at the window boundary (base on region code).

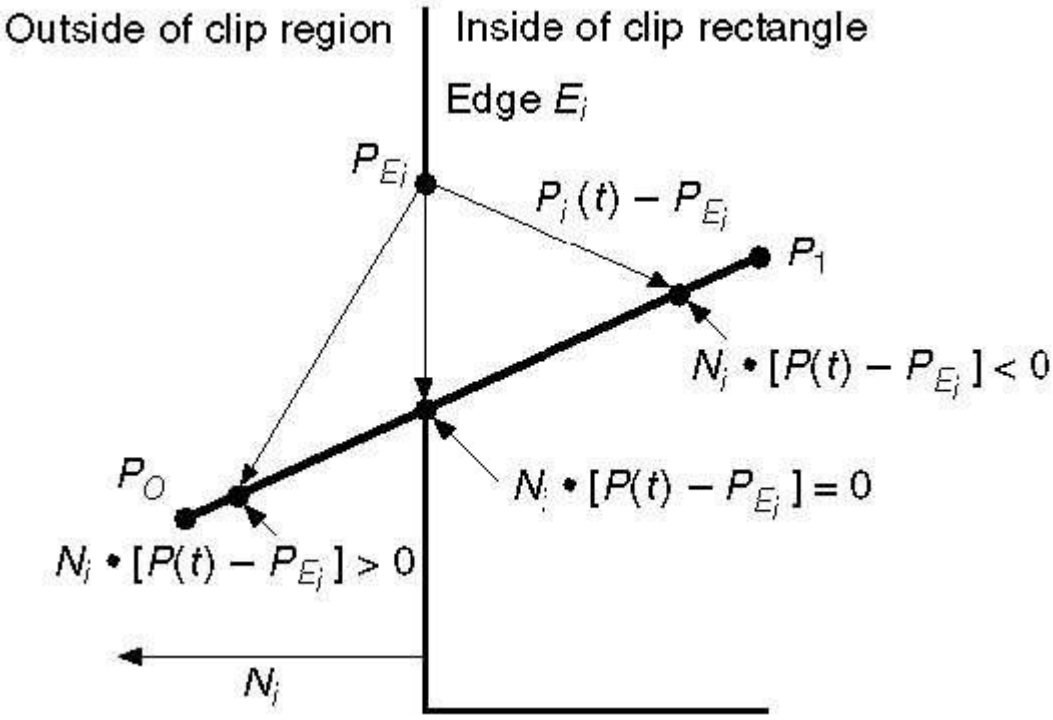
Step 3.2.3 – Replace endpoint with the intersection point and update the region code.

Step 3.2.4 – Repeat step 2 until we find a clipped line either trivially accepted or trivially rejected.

Step 4 – Repeat step 1 for other lines.

Cyrus-Beck Line Clipping Algorithm

This algorithm is more efficient than Cohen-Sutherland algorithm. It employs parametric line representation and simple dot products.



Parametric equation of line is –

```
P0P1:P(t) = P0 + t(P1 - P0)
```

Let N_i be the outward normal edge E_i. Now pick any arbitrary point P_{Ei} on edge E_i then the dot product $N_i.[P(t) - P_{Ei}]$ determines whether the point P(t) is "inside the clip edge" or "outside" the clip edge or "on" the clip edge.

The point P(t) is inside if $N_i.[P(t) - P_{Ei}] < 0$

The point P(t) is outside if $N_i.[P(t) - P_{Ei}] > 0$

The point P(t) is on the edge if $N_i.[P(t) - P_{Ei}] = 0$ (Intersection point)

$N_i.[P(t) - P_{Ei}] = 0$

$N_i.[P_0 + t(P_1 - P_0) - P_{Ei}] = 0$ (Replacing P(t) with $P_0 + t(P_1 - P_0)$)

$N_i.[P_0 - P_{Ei}] + N_i.t[P_1 - P_0] = 0$

$N_i.[P_0 - P_{Ei}] + N_i \cdot tD = 0$ (substituting D for $[P_1 - P_0]$)

$N_i.[P_0 - P_{Ei}] = - N_i \cdot tD$

The equation for t becomes,

The equation for t becomes,

$$t = \frac{N_i.[P_o - P_{Ei}]}{-N_i.D}$$

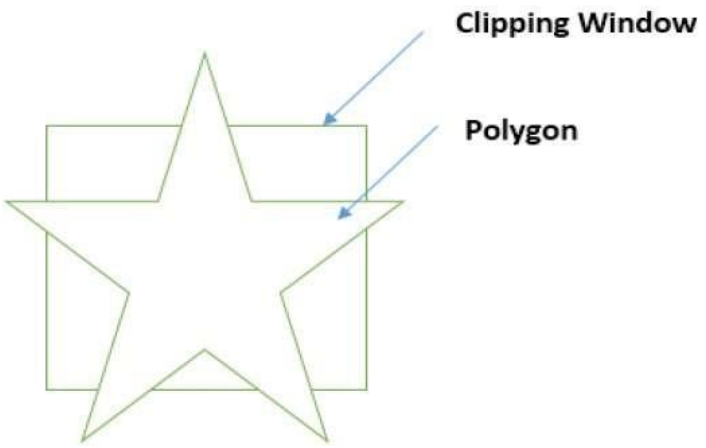
It is valid for the following conditions –

- $N_i \neq 0$ (error cannot happen)
- $D \neq 0$ ($P_1 \neq P_0$)
- $N_i \cdot D \neq 0$ (P_0P_1 not parallel to E_i)

Polygon Clipping (Sutherland Hodgman Algorithm)

A polygon can also be clipped by specifying the clipping window. Sutherland Hodgeman polygon clipping algorithm is used for polygon clipping. In this algorithm, all the vertices of the polygon are clipped against each edge of the clipping window.

First the polygon is clipped against the left edge of the polygon window to get new vertices of the polygon. These new vertices are used to clip the polygon against right edge, top edge, bottom edge, of the clipping window as shown in the following figure.



While processing an edge of a polygon with clipping window, an intersection point is found if edge is not completely inside clipping window and the a partial edge from the intersection point to the outside edge is clipped. The following figures show left, right, top and bottom edge clippings –

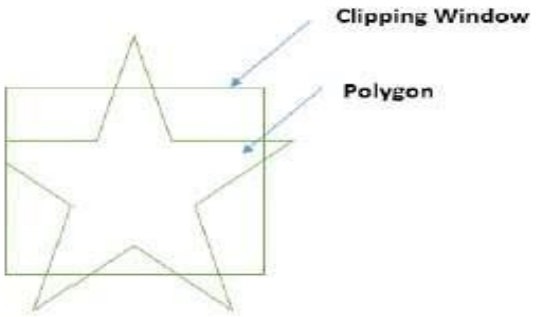


Figure: Clipping Left Edge

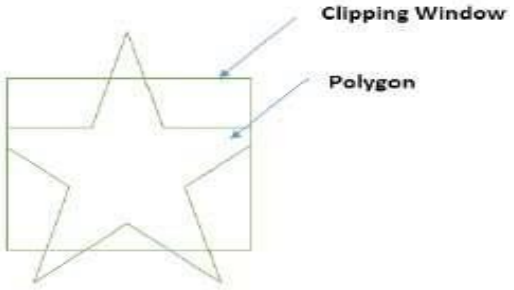


Figure: Clipping Right Edge

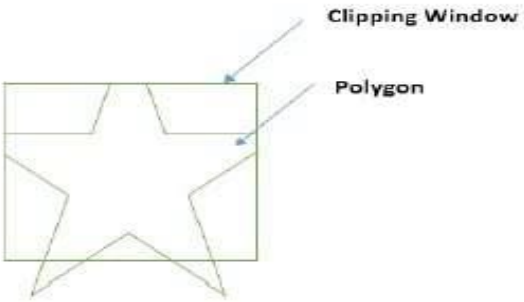


Figure: Clipping Top Edge

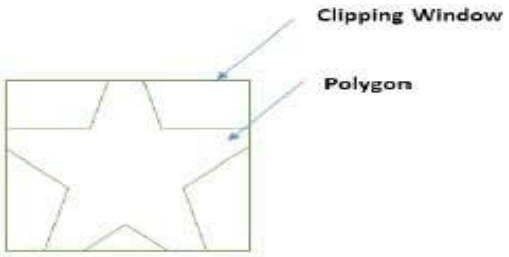


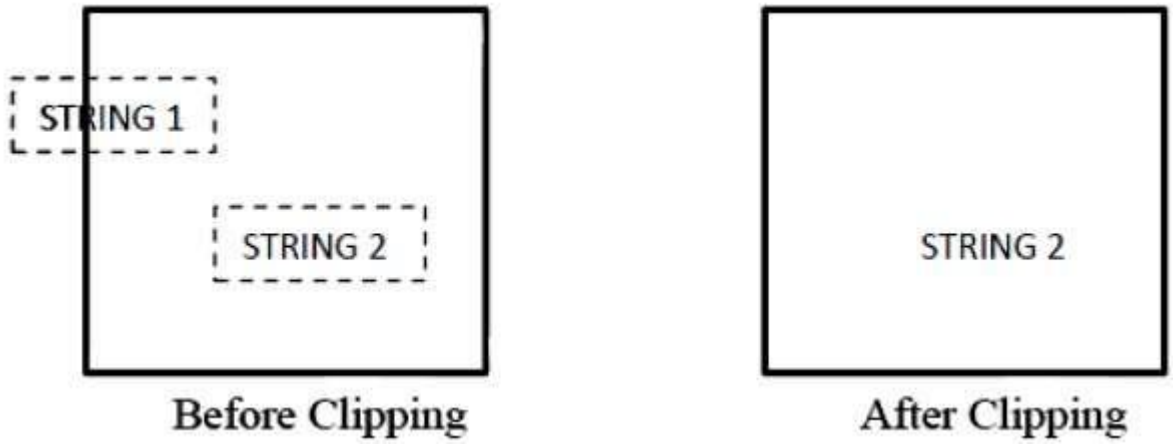
Figure: Clipping Bottom Edge

Text Clipping

Various techniques are used to provide text clipping in a computer graphics. It depends on the methods used to generate characters and the requirements of a particular application. There are three methods for text clipping which are listed below –

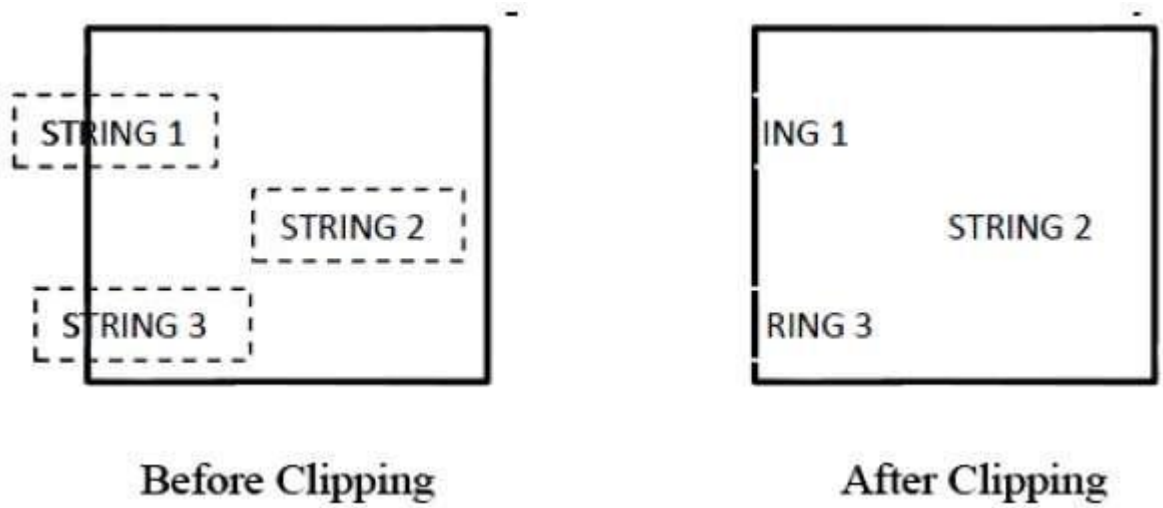
- All or none string clipping
- All or none character clipping
- Text clipping

The following figure shows all or none string clipping –



In all or none string clipping method, either we keep the entire string or we reject entire string based on the clipping window. As shown in the above figure, STRING2 is entirely inside the clipping window so we keep it and STRING1 being only partially inside the window, we reject.

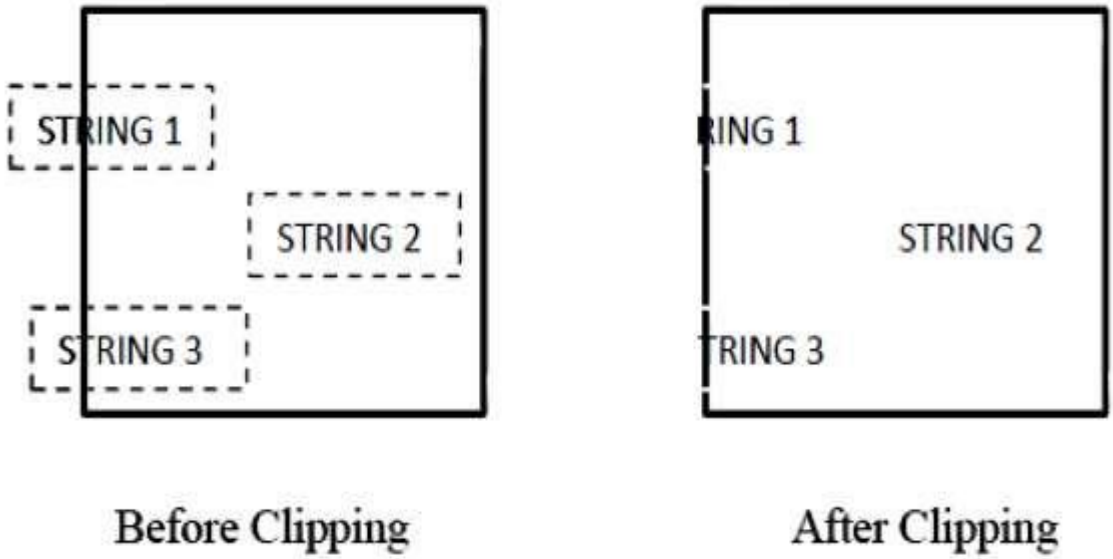
The following figure shows all or none character clipping –



This clipping method is based on characters rather than entire string. In this method if the string is entirely inside the clipping window, then we keep it. If it is partially outside the window, then –

- You reject only the portion of the string being outside
- If the character is on the boundary of the clipping window, then we discard that entire character and keep the rest string.

The following figure shows text clipping –



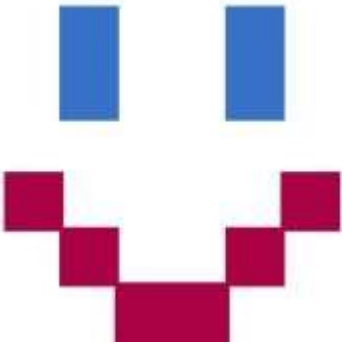
This clipping method is based on characters rather than the entire string. In this method if the string is entirely inside the clipping window, then we keep it. If it is partially outside the window, then

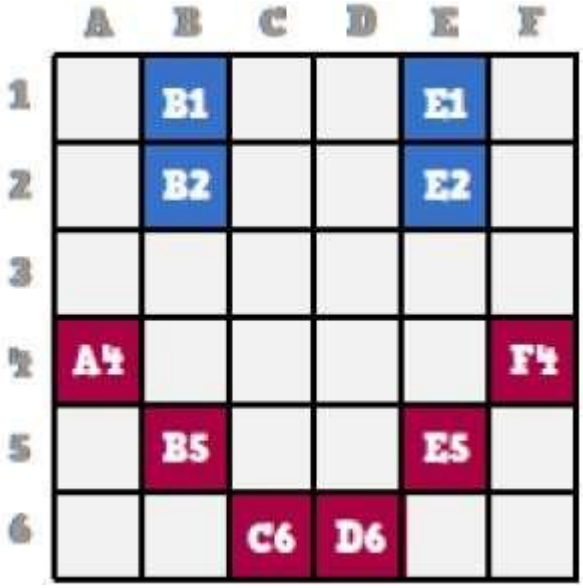
You reject only the portion of string being outside.

If the character is on the boundary of the clipping window, then we discard only that portion of character that is outside of the clipping window.

Bitmap Graphics

A bitmap is a collection of pixels that describes an image. It is a type of computer graphics that the computer uses to store and display pictures. In this type of graphics, images are stored bit by bit and hence it is named Bit-map graphics. For better understanding let us consider the following example where we draw a smiley face using bit-map graphics.

- 
- Now we will see how this smiley face is stored bit by bit in computer graphics.



By observing the original smiley face closely, we can see that there are two blue lines which are represented as B1, B2 and E1, E2 in the above figure.

In the same way, the smiley is represented using the combination bits of A4, B5, C6, D6, E5, and F4 respectively.

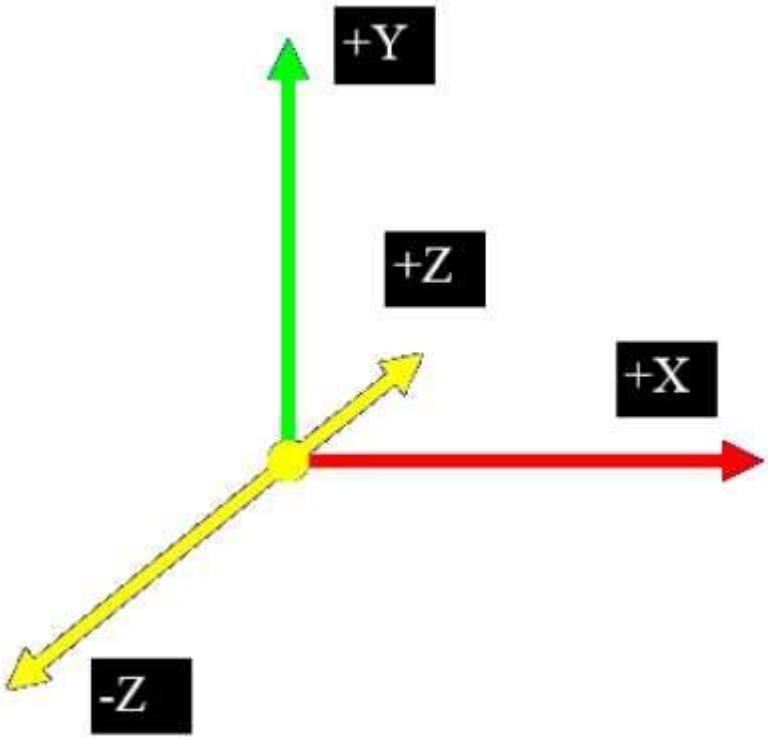
The main disadvantages of bitmap graphics are –

- We cannot resize the bitmap image. If you try to resize, the pixels get blurred.
- Colored bitmaps can be very large.

3D Computer Graphics

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.

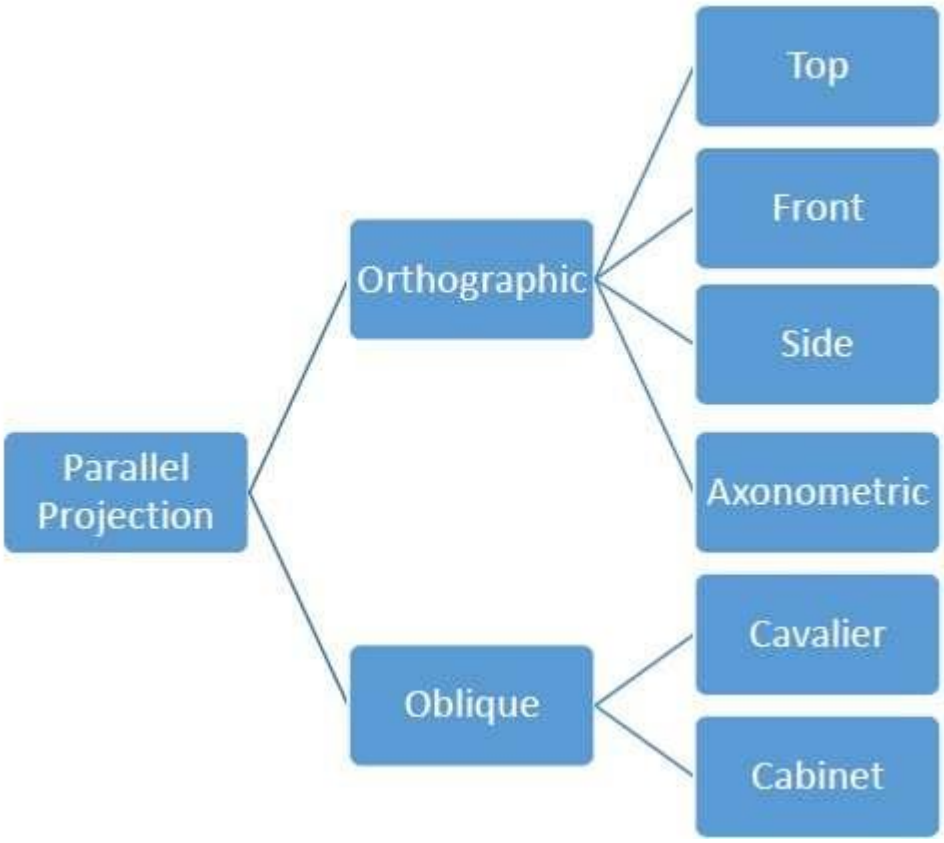


Parallel Projection

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.

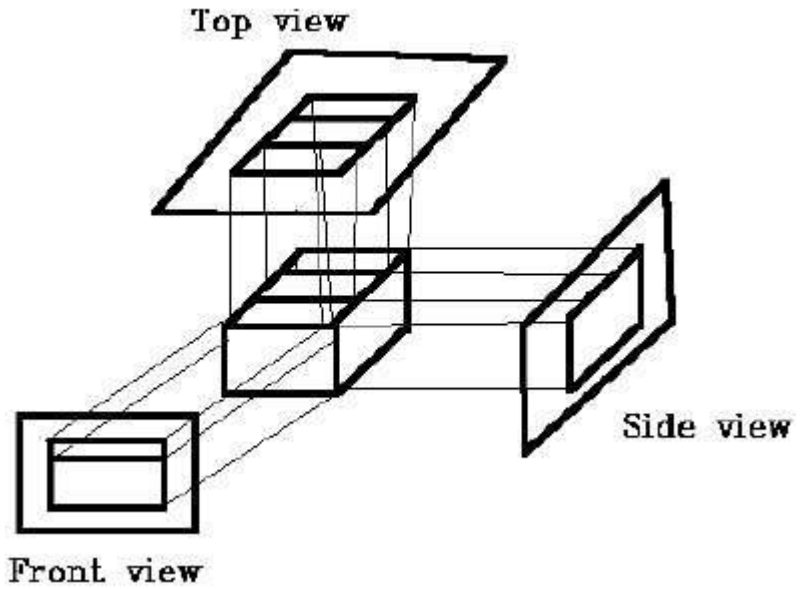
Parallel projections are less realistic, but they are good for exact measurements. In this type of projections, parallel lines remain parallel and angles are not preserved. Various types of parallel projections are shown in the following hierarchy.



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

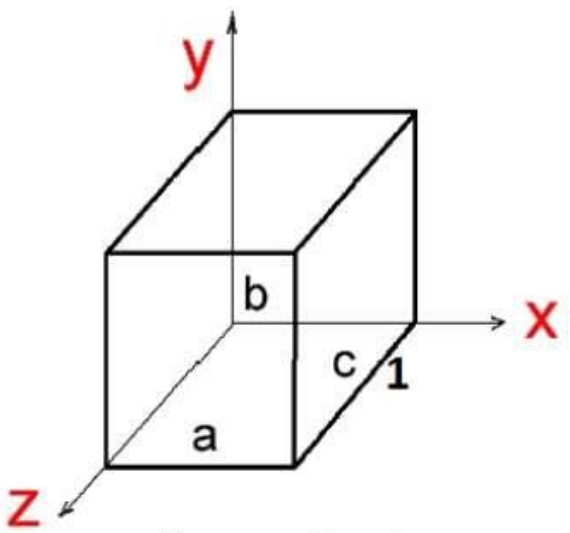


Oblique Projection

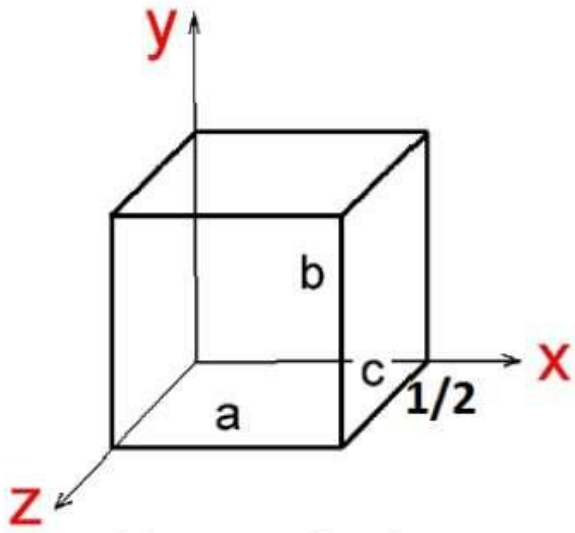
In oblique projection, the direction of projection is not normal to the projection of plane. In oblique projection, we can view the object better than orthographic projection.

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 –



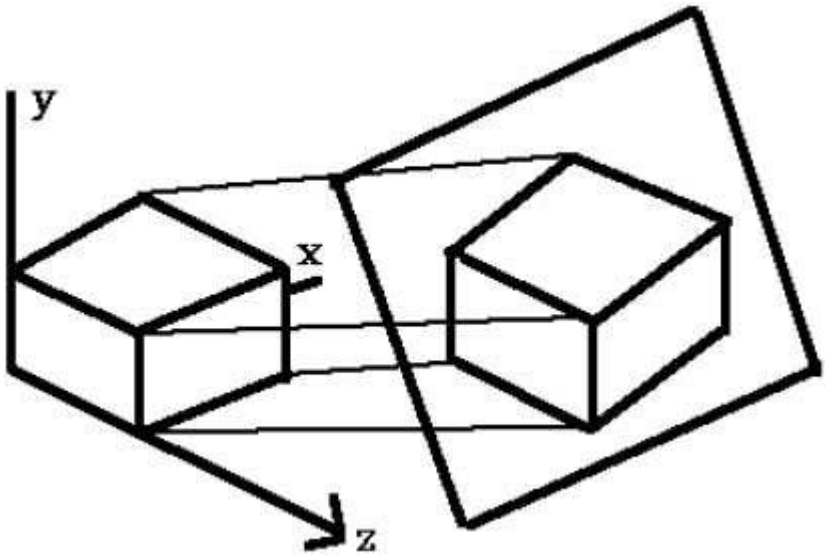
Cavalier Projection



Cabinet Projection

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 –

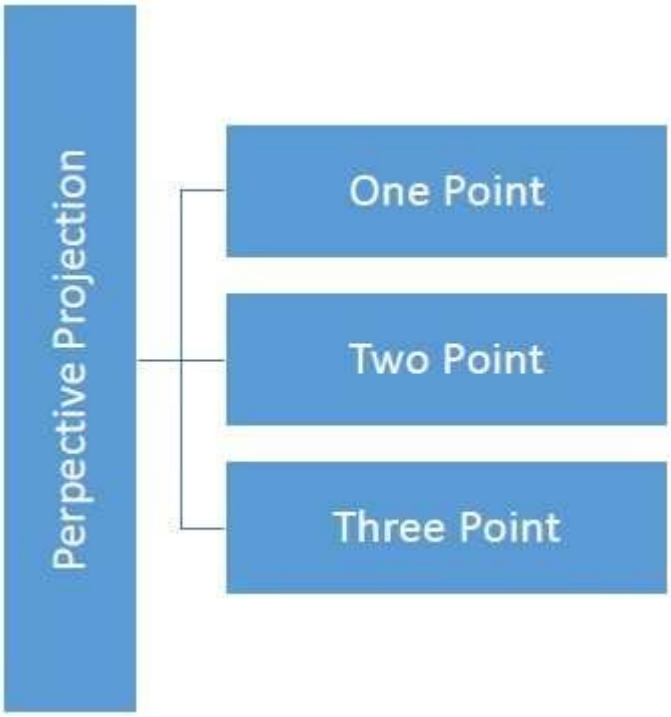


Perspective Projection

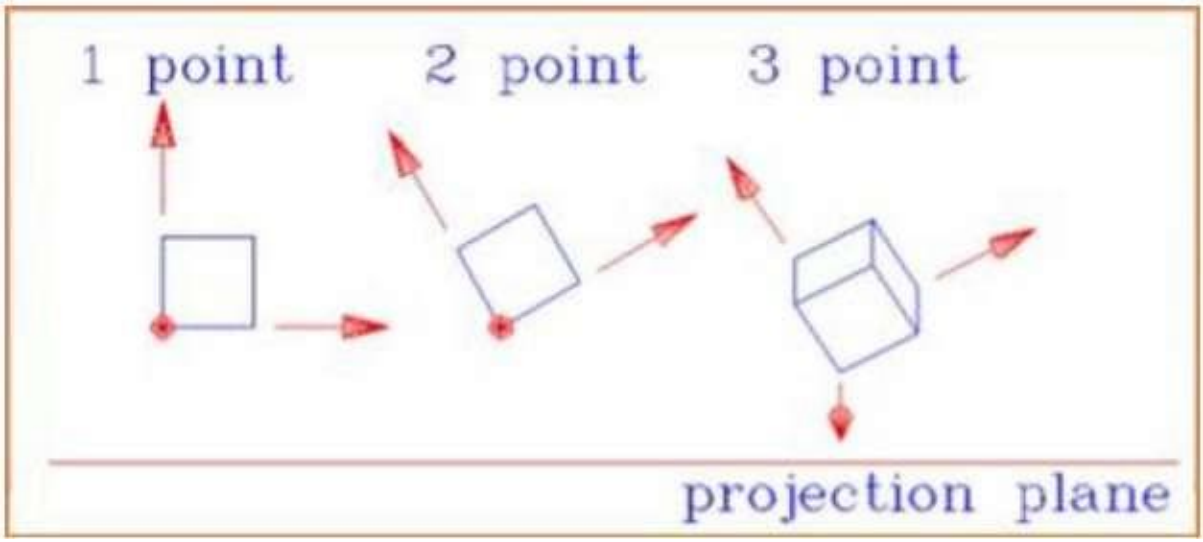
In perspective projection, the distance from the center of projection to project plane is finite and the size of the object varies inversely with distance which looks more realistic.

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.



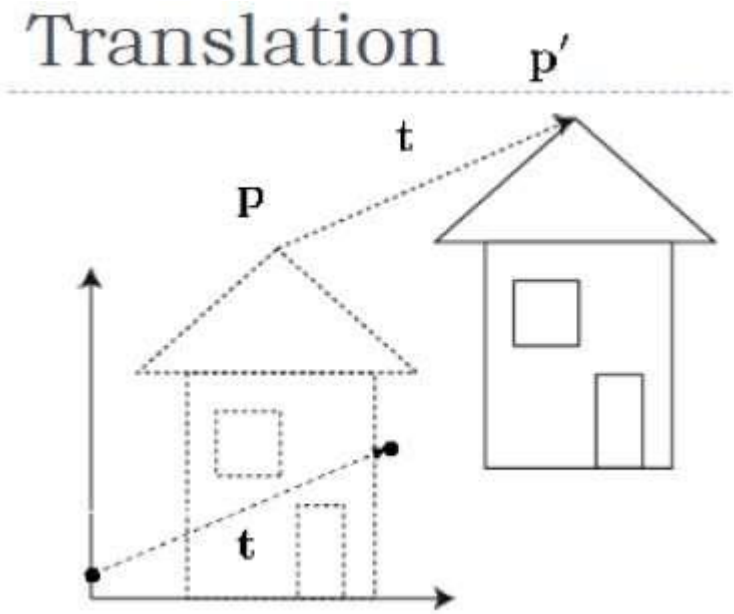
The following figure shows all the three types of perspective projection –



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 (t_x, t_y, t_z) to the original coordinate (X, Y, Z) to get the new coordinate (X′, Y′, Z′).

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

P′ = P·T

$$[X'\ Y'\ Z'\ 1] = [X\ Y\ Z\ 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

$$= [X + t_x \quad Y + t_y \quad Z + t_z \quad 1]$$

3D Transformation

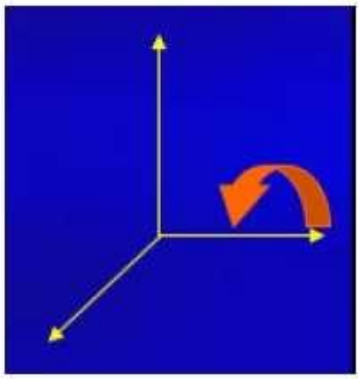
Rotation

3D rotation is not same as 2D rotation. In 3D rotation, we have to specify the angle of rotation along with the axis of rotation. We can perform 3D rotation about X, Y, and Z axes. They are represented in the matrix form as below –

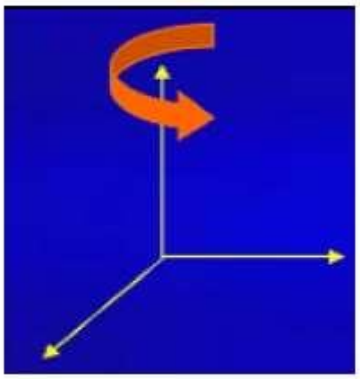
$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & -sin\theta & 0 \\ 0 & sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} R_y(\theta) = \begin{bmatrix} cos\theta & 0 & sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -sin\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} R_z(\theta)$$

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

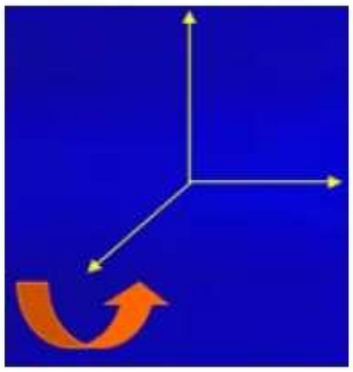
The following figure explains the rotation about various axes –



Rotation about x-axis



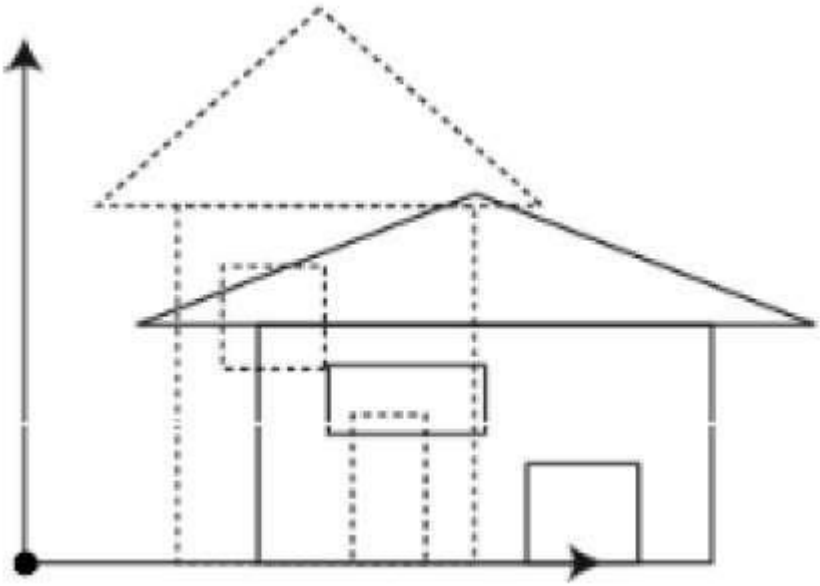
Rotation about y-axis



Rotation about z-axis

Scaling

You can change the size of an object using scaling transformation. 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. The following figure shows the effect of 3D scaling –



In 3D scaling operation, three coordinates are used. Let us assume that the original coordinates are (X, Y, Z), scaling factors are (S_X,S_Y,S_Z)(SX,SY,Sz) respectively, and the produced coordinates are (X', Y', Z'). This can be mathematically represented as shown below –

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

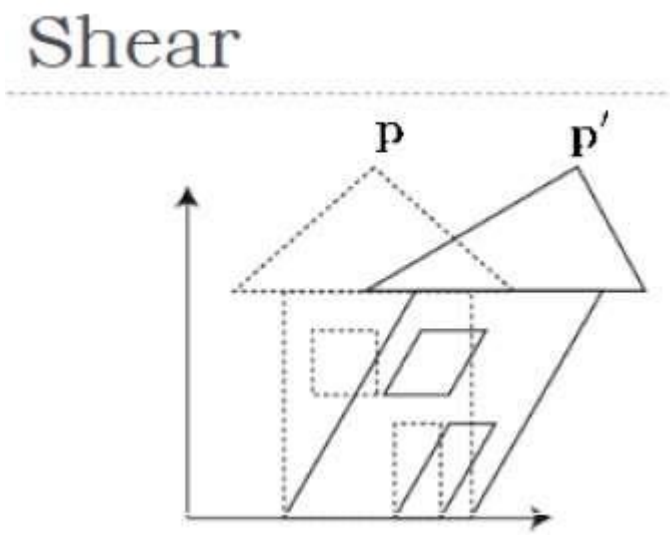
$$P' = P \cdot S$$

$$[X' \;\; Y' \;\; Z' \;\; 1] = [X \;\; Y \;\; Z \;\; 1] \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= [X.S_x \;\; Y.S_y \;\; Z.S_z \;\; 1]$$

Shear

A transformation that slants the shape of an object is called the **shear transformation**. Like in 2D shear, we can shear an object along the X-axis, Y-axis, or Z-axis in 3D.



As shown in the above figure, there is a coordinate P. You can shear it to get a new coordinate P', which can be represented in 3D matrix form as below –

$$Sh = \begin{bmatrix} 1 & sh_x^y & sh_x^z & 0 \\ sh_y^x & 1 & sh_y^z & 0 \\ sh_z^x & sh_z^y & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

P′ = P · Sh

$$X' = X + Sh_x^y Y + Sh_x^z Z$$

$$Y' = Sh_y^x X + Y + sh_y^z Z$$

$$Z' = Sh_z^x X + Sh_z^y Y + Z$$

Transformation Matrices

Transformation matrix is a basic tool for transformation. A matrix with n x m dimensions is multiplied with the coordinate of objects. Usually 3 x 3 or 4 x 4 matrices are used for transformation. For example, consider the following matrix for various operation.

$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$	$S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$Sh = \begin{bmatrix} 1 & sh_x^y & sh_x^z & 0 \\ sh_y^x & 1 & sh_y^z & 0 \\ sh_z^x & sh_z^y & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Translation Matrix	**Scaling Matrix**	**Shear Matrix**
$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & -sin\theta & 0 \\ 0 & sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$R_y(\theta) = \begin{bmatrix} cos\theta & 0 & sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -sin\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$R_z(\theta) = \begin{bmatrix} cos\theta & -sin\theta & 0 & 0 \\ sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Rotation Matrix		