2

GEOMETRICAL TRANSFORMATION [10HR]

2.1 2D Transformations

In computer graphics, transformations of 2D objects are essential to many graphics applications. The transformations are used directly by application programs and within many graphics subroutines in application programs. Many applications use the geometric transformations to change the position, orientation, and size or shape of the object in drawing. Rotation, Translation and scaling are three major transformations that are extensively used by all most all graphical packages or graphical subroutines in applications. Other than these, reflection and shearing transformations are also used by some graphical packages.

Changing co-ordinate description of an object is called transformation.

Types of transformation:

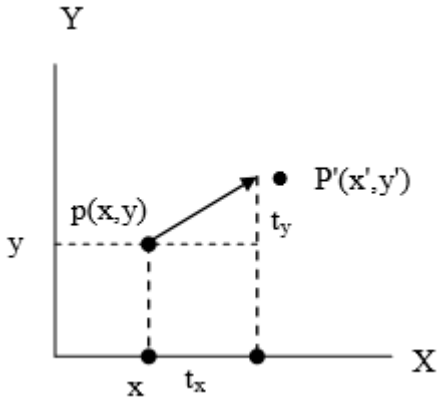
- Rigid body transformation (transformation without reformation in shape).
- Non rigid transformation (transformation with change in shape)

2D Translation:

A translation is applied to an object by re-positioning it along a straight line path from one co-ordinate location to another. We translate a two-dimensional point by adding translation distances, t_x, t_y to the respective co-ordinate values of original co-ordinate position (x,y) to move the point to a new position (x', y') as:

$x' = x + t_x$

$y' = y + t_y$



The translation distance pair (t_x, t_y) is known as translation vector or shift vector. We can express translation equations as matrix representations as :

$$P = \begin{bmatrix} x \\ y \end{bmatrix} \qquad P' = \begin{bmatrix} x' \\ y' \end{bmatrix} \qquad T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$\therefore P' = P + T$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Sometimes, matrix transformation are represented by co-ordinate rows vector instead of column vectors as:

$P = (x,y)$ $T = (t_x, t_y)$ $P' = P + T$

For translation of any object in Cartesian plane, we transform the distinct co-ordinates by the translation vector and re-draw image at the new transformed location.

For example: Translate the given points (2,5) by the translating value (3,3).

Solution:

Initial point (x,y) = (2,5)

$T_x = 3$

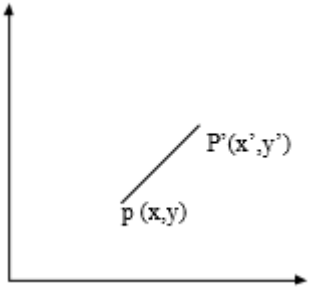
$T_y = 3$

$x' = x + t_x$

$y' = y + t_y$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \end{bmatrix} + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} 5 \\ 8 \end{bmatrix}$$



2D Rotation:

The 2D rotation is applied to re-position the object along a circular path in XY-plane. To generate rotation, we specify a rotation angle θ , through which the co-ordinates are to be rotated. Rotation can be made by angle θ either clockwise or anticlockwise direction. Besides the angle of rotation θ , there should be a pivot point through which the object is to be rotated. The positive θ rotates object in anti-clockwise direction and the negative value of θ rotates the object in clockwise direction.

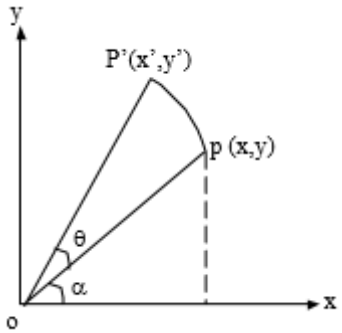
A line perpendicular to rotating plane and passing through pivot point is called axis of rotation.

Types of rotation:

a. About origin
b. About any point

About origin:

If P(x,y) is rotated to a new position P' (x', y') in anti-clockwise direction by an angle θ, then (x', y') can be calculated as follows:



Let the angle made by line OP with x-axis is α and the radius of circulation path is r then,

$x = r \cos\alpha$

$y = r \sin\alpha$

Also,

$x' = r\cos(\theta + \alpha)$

$y' = r \sin(\theta + \alpha)$

$x' = r(\cos\theta.\cos\alpha - \sin\theta.\sin\alpha)$

$= r\cos\theta.\cos\alpha - r\sin\theta.\sin\alpha$

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

$y' = r(\sin\theta.\cos\alpha + \cos\theta.\sin\alpha)$

$= r\sin\theta.\cos\alpha + r\cos\theta.\sin\alpha$

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

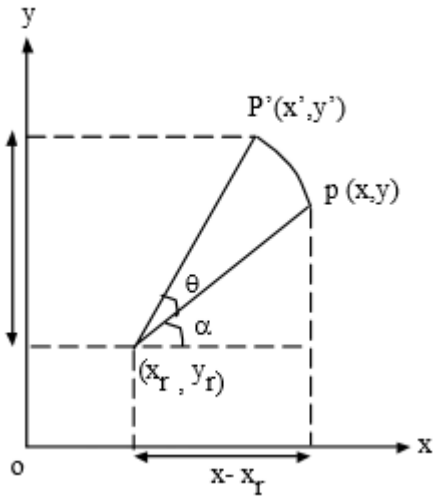
In matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Note: θ +ve (anticlockwise)

θ -ve (clockwise)

Rotation from any arbitrary pivot point (x_r, y_r):



- Let Q(x_r, y_r) is pivot point for rotation.
- P(x,y) is co-ordinate of point to be rotated by angle θ
- Let α is the angle made by QP with x-direction

Then angle made by QP' with x-direction is θ+α

Hence,

$\cos(\alpha+\theta) = (x' - x_r) / r$

Or, $r\cos(\alpha+\theta) = (x' - x_r)$

Or, $x' - x_r = r\cos\alpha\cos\theta - r\sin\alpha\sin\theta$

Since, $r\cos\alpha = x - x_r$

$r\sin\alpha = y - y_r$

$x' = x_r + (x-x_r)\cos\theta - (y-y_r)\sin\theta$ ---------- (1)

Similarly,

$\sin(\alpha+\theta) = (y' - y_r) / r$

Or, $r\sin(\alpha+\theta) = (y' - y_r)$

Or, $y' - y_r = r \sin\alpha \cos\theta + r \cos\alpha \sin\theta$
Since, $r\cos\alpha = x - x_r$
$r\sin\alpha = y - y_r$
$y' = y_r + (x-x_r)\sin\theta + (y-y_r)\cos\theta$ ---------- (2)

These equations (1) and (2) are the equations for rotation of a point (x,y) with angle θ taking pivot point (x_r, y_r).

The rotation about pivot point (x_r, y_r) can be achieve by sequence of translation, rotation about origin and reverse translation.

- Translate the point (x_r, y_r) and P(x,y) by translation vector $(-x_r, -y_r)$ with translates the pivot to origin and P(x,y) to $(x-x_r, y-y_r)$.
- Now apply the rotation equations when pivot is at origin to rotate the translated point $(x-x_r, y-y_r)$ as:
  $x_1 = (x-x_r)\cos\theta - (y-y_r)\sin\theta$
  $y_1 = (x-x_r)\sin\theta + (y-y_r)\cos\theta$
- Re-translate the rotated point (x_1, y_1) with translation vector (x_r, y_r) which is reverse translation to original translation. Finally we get the equation after successive transformation as:
  $x' = x_r + (x-x_r)\cos\theta - (y-y_r)\sin\theta$ ---------- (1)
  $y' = y_r + (x-x_r)\sin\theta + (y-y_r)\cos\theta$ ---------- (2)
  Which are actually the equations for rotation (x,y) from the pivot point (x_r, y_r).

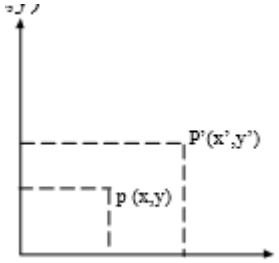
Question1: Rotate the triangle having co-ordinate (1, 2), (2, 3), (4, 5) by 60^0 about origin.
Question2: Rotate the rectangle having co-ordinate A(1,1), B(1,3), C(3,1), D(3,3) by 45^0 with pivot point (2,2).

2D Scaling:

A scaling transformation alters the size of the object. This operation can be carried out for polygon by multiplying the co-ordinate values (x,y) of each vertex by scaling factor s_x and s_y to produce transformed co-ordinates (x', y').

i.e. $x' = x.\ s_x$ and $y' = y.s_y$

Scaling factor s_x scales object in x-direction and s_y scales in y-direction. If the scaling factor is less than 1, the size of object is decreased and if it is greater than 1 the size of object is increased. The scaling factor = 1 for both direction does not change the size of the object. If both scaling factors have same value then the scaling is known as uniform scaling. If the value of s_x and s_y are different, then the scaling is known as differential scaling. The differential scaling is mostly used in the graphical package to change the shape of the object.



The matrix representation of the scaling is :

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

i.e. P' = S.P

Homogeneous co-ordinate representation of 2D Transformation:

- The homogeneous co-ordinate system provide a uniform frame-work for handling different geometric transformations, simply as multiplication of matrices.
- It extension to 3D is straight forward which also helps to produce perspective projections by use of matrix multiplication. We simply add a third co-ordinate to 2D point i.e.
  $(x,y) = (x_h, y_h, h)$ where, $x = x_h/h$
  $y = y_h/h$ where h is 1 usually for 2D case.
- By using this homogeneous co-ordinate system a 2D point would be (x,y,1). The matrix formulation for 2D translation for $T(t_x, t_y)$ is :

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$x' = x+t_x$

$y' = x+t_y$

For Rotation: R(θ) about origin the homogeneous matrix equation will be:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$ which gives the Y

$x' = x\cos\theta - y\sin\theta$ ---------- (1)

$y' = x\sin\theta + y\cos\theta$ ----------(2)

Which are equation for rotation of (x,y) with angle θ and taking pivot as origin.

Rotation about an arbitrary fixed pivot point (x_r, y_r):

For a fixed pivot point rotation, we can apply composite transformation as:

1. Translate fixed point (x_r, y_r) to the co-ordinate origin by $T(-x_r, -y_r)$.
2. Rotate with angle θ → R(θ).
3. Translate back to original position by $T(x_r, y_r)$.

This composite transformation is represented as:

$P' = T(x_r, y_r).\ R(\theta).\ T(-x_r, -y_r).\ P$

Which can be represented in matrix equation using homogeneous co-ordinate system as:

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

Expanding this equation, we get:

$x' = x_r + (x-x_r)\cos\theta - (y-y_r)\sin\theta$ ---------- (1)

$y' = y_r + (x-x_r)\sin\theta + (y-y_r)\cos\theta$ ---------- (2)

Which are actually the equations for rotation of (x,y) from the pivot point (x_r, y_r)

Scaling with scaling factors (s_x, s_y):

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Which exactly gives the equations:

$x' = x.s_x$

$y' = y.s_y$

Fixed point scaling:
Objects transformed with standard scaling equation are scaled as well as re-positioned. Scaling factor with value less then 1 moves object closer to the origin where as value of scaling factor greater than 1 moves object away from origin.

- To control the location of scaled object we can choose the position called fixed point. Let co-ordinates of fixed point = (x_f, y_f). It can be any point on the object.
- A polygon is then scaled relative to (x_f, y_f) by scaling the distance from each vertex to the fixed point.
- For a vertex with co-ordinate (x,y), the scaled co-ordinates (x', y') are calculated as:

$$x' = x_f (x-x_f)s_x$$
$$y' = y_f + (y-y_f)s_y$$
$$Or$$
$$x' = x.s_x + (1-s_x)x_f$$
$$y' = y.s_y + (1-s_y)y_f$$

  Where, $(1-s_x).x_f$ and $(1-s_y).y_f$ are constant for all points in object.

To represent fixed point scaling using matrix equations in homogeneous co-ordinate system, we can use composite transformation as in fixed point rotation.

- Translate object to the origin so that (x_f, y_f) lies at origin by $T(-x_f, -y_f)$.
- Scale the object with (s_x, s_y).
- Re-translate the object back to its original position so that fixed point (x_f, y_f) moves to its original position. In this case translation vector is $T(x_f, y_f)$.
  Therefore,

$$P' = [T(x_f, y_f).S(s_x, s_y).T(-x_f, -y_f)]P$$

  The homogeneous matrix equation for fixed point scaling is:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & x_f(1-s_x) \\ 0 & s_y & y_f(1-s_y) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Directive Scaling:
Standard and fixed point scaling scales object along x and y axis only. Directive scaling scales the object in any direction.

Let S_1 and S_2 are given directions for scaling at angle θ from co-ordinate axes as in figure below:

1. First perform the rotation so that directions S_1 and S_2 concide with x and y-axes.
2. Then the scaling transformation is applied to scale the object by given scaling factors (s_1, s_2).
3. Re-apply the rotation in opposite direction to return to their original orientation.

For any point P in object, the directive scaling position P' is given by following composite transformation.

$$P' = R^{-1}(\theta).S(s_1, s_2).R(\theta).P$$

For which, the homogeneous matrix equation is:

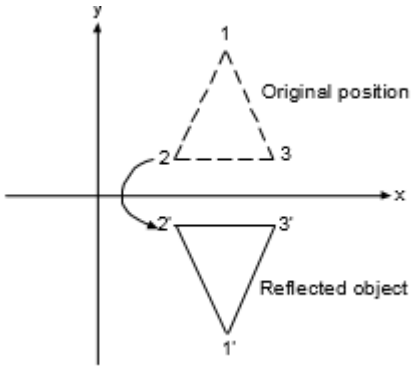
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Question: Scale the given triangle whose co-ordinate values are (2,3), (1,2), (3,4) where the scaling factor is (2,3) about (i) the origin (ii) about fixed point (1,2).

Reflection:

A reflection is a transformation that produces a mirror image of an object. In 2D-transformation, reflection is generated relative to an axis of reflection. The reflection of an object to an relative axis of reflection, is same as 180^0 rotation about the reflection axis.

a) Reflection about X-axis:



b)

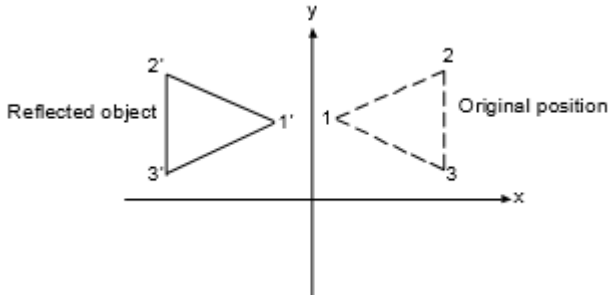
The line representing x-axis is y=0. The reflection of a point P(x,y) on x-axis, changes the y-coordinate sign i.e. reflection about x-axis, the reflected position of P(x,y) will be P' (x,-y). Hence, reflection in x-axis is accomplished with transformation equation:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

gives the reflection of a point.

To reflect a 2D object, reflect each distinct point of object by above equation, then joining the points with straight line, redraws the image for reflected image.

b) Reflection about y-axis:



The line representing y-axis is x=0. The reflection of a point P(x,y) on y-axis changes the sign of x-coordinate i.e. P(x,y) changes P'(-x,y). Hence reflection of a point on y-axis is obtained by following matrix equation:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

c) Reflection on an arbitrary axis:

The reflection on any arbitrary axis of reflection can be achieved by sequence of rotation and co-ordinate axes reflection matrices.

- First, rotate the arbitrary axis of reflection so that the axis of reflection and one of the co-ordinate axes coincide.
- Reflect the image on the co-ordinate axis to which the axis of reflection coincides.
- Rotate the axis of reflection back to its original position.

d) Reflection about origin:

The reflection on the line perpendicular to xy-plane and passing through flips x and y co-ordinates both. So, sign of x and y co-ordinate value changes. The equivalent matrix equation for the point is:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Shearing:

A transformation that distorts the shape of an object such that the transformed shape appears as if the object were composed of internal layers that had been caused to slide over each other is called shear.

x-direction shear: An x-direction shear relative to x-axis is produced with transformation matrix equation.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Which transforms $x'=x+y.sh_x$ and $y'=y$

y-direction shear: An y-direction shear relative to y-axis is produced by following transformation equation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Which transforms $x'=x$ and $y'=x.sh_y+y$

Composite Transformation:

With the matrix representation of transformation equations it is possible to setup a matrix for any sequence of transformations as a composite transformation matrix by calculating the matrix product of individual transformation.

For column matrix representation of coordinate positions we form composite transformation by multiplying matrices in order from right to left.

1. **Two successive translation are additive:**
   Let two successive translation vectors (t_{x1}, t_{y1}) and (t_{x2}, t_{y2}) are applied to a coordinate position P then,
   P' = $T(t_{x2}, t_{y2})$. {$T(t_{x1}, t_{y1})$.P} and P' = {$T(t_{x2}, t_{y2})$. $T(t_{x1}, t_{y1})$}.P
   Here the composite transformation matrix for this sequence of translation is:

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

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

2. **Two successive scaling operations are Multiplicative:**
   Let (s_{x1}, s_{y1}) and (s_{x2}, s_{y2}) be two successive vectors applied to a coordinate position P then the composite scaling matrix thus produced is:
   P' = $S(s_{x2}, s_{y2})$. {$S(s_{x1}, s_{y1})$.P} and P' = {$S(s_{x2}, s_{y2})$. $S(s_{x1}, s_{y1})$}
   Here, the composite transformation matrix for this sequence of translation is:

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

   Or, $S(s_{x2}, s_{y2})$. $S(s_{x1}, s_{y1}) = S(s_{x1}.s_{x2}, s_{y1}.s_{y2})$

3. **Two successive rotation operations are additive:**
   Let $R(\theta_1)$ and $R(\theta_2)$ be two successive rotations applied to a coordinate position P then the composite scaling matrix thus produced is,
   P' = $R(\theta_2)$. {$R(\theta_1)$.P} and P' = {$R(\theta_2)$. $R(\theta_1)$}.P
   Here, the composite transformation matrix for this sequence of translation is:

$$\begin{bmatrix}cos\theta_2 & -sin\theta_2\\ sin\theta_2 & cos\theta_2\end{bmatrix}.\begin{bmatrix}cos\theta_1 & -sin\theta_1\\ sin\theta_1 & cos\theta_1\end{bmatrix}$$

Or,

$$\begin{bmatrix}cos\theta_2.cos\theta_1 - sin\theta_2.sin\theta_1\\ sin\theta_2.cos\theta_1 + sin\theta_1.cos\theta_2\end{bmatrix}.\begin{bmatrix}-cos\theta_2.sin\theta_1 - sin\theta_2.cos\theta_1\\ cos\theta_2.cos\theta_1 - sin\theta_2.sin\theta_1\end{bmatrix}$$

Or,

$$\begin{bmatrix}\cos(\theta_2+\theta_2)\\ \sin(\theta_2+\theta_2)\end{bmatrix}.\begin{bmatrix}-\sin(\theta_2+\theta_2)\\ \cos(\theta_2+\theta_2)\end{bmatrix}$$

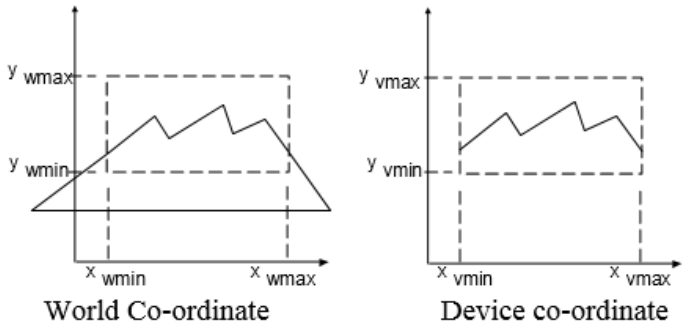
Or,

$R(\theta_2)$. $R(\theta_1) = R(\theta_1+\theta_2)$

2D Viewing:

Two-Dimensional viewing is the formal mechanism for displaying views of a picture on an output device. Typically, a graphics package allows a user to specify which part of a defined picture is to be displayed and where that part is to be placed on the display device. Any convenient Cartesian coordinate system, referred to as the world-coordinate reference frame, can be used to define the picture. For a two-dimensional picture, a view is selected by specifying a subarea of the total picture area. The picture parts within the selected areas are then mapped onto specified areas of the device coordinates. Transformations from world to device co-ordinates involve translation, rotation, and scaling operations, as well as procedures for deleting those parts of the picture that are outside the limits of a selected display area.

Window: A world-coordinate area selected for display
Viewport: An area on a display device to which a window is mapped.
"The window defines what is to be viewed; the viewport defines where it is to be displayed"



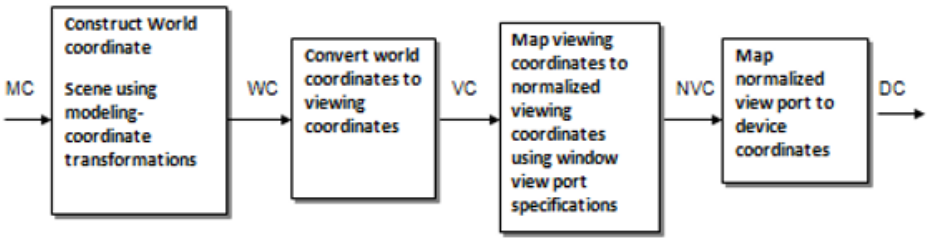
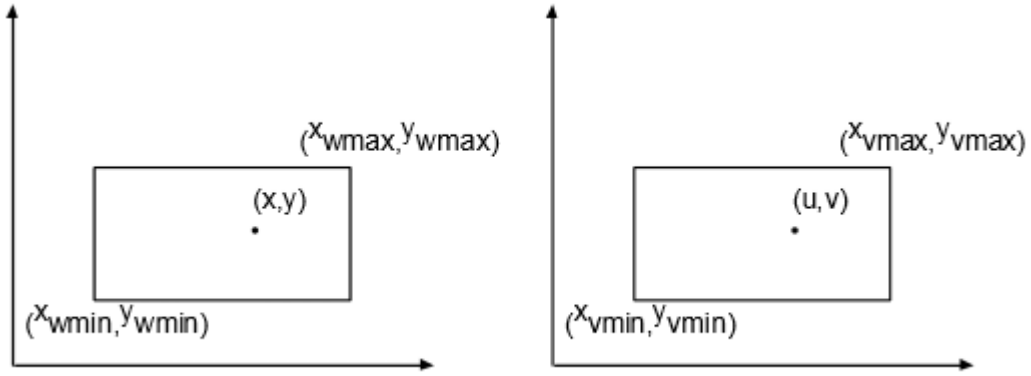


Fig: 2D viewing pipeline

Window to viewport coordinate transformation

Once object descriptions have been transferred to the viewing reference frame, we choose the window extents in viewing coordinates and select the viewport limits in normalized coordinates. Object descriptions are then transferred to normalized device coordinates. We do this using a transformation that maintains the same relative placement of objects in normalized space as they had in viewing coordinates.

A point at position (xw, yw) in the window is mapped into position (xv, yv) in the associated viewport.



To maintain the same relative placement in the viewports as in the window, we require that:

$$\frac{x - x_{w\min}}{x_{w\max} - x_{w\min}} = \frac{u - x_{v\min}}{x_{v\max} - x_{v\min}}$$

$$\text{Or, } \frac{y - y_{w\min}}{y_{w\max} - y_{w\min}} = \frac{u - y_{v\min}}{y_{v\max} - y_{v\min}}$$

$$S_x = \frac{x_{v\max} - x_{w\min}}{x_{w\max} - x_{w\min}}$$

$$S_y = \frac{y_{v\max} - y_{w\min}}{y_{w\max} - y_{w\min}}$$

Equations above can also be derived with a set of transformations that converts the window area into the viewport area. This conversion is performed with the following sequence of transformations:

1. Perform a scaling transformation using a fixed-point position of (xw_{min}, yw_{min}) that scales the window area to the size of the viewport.
2. Translate the scaled window area to the position of the viewport.

Relative proportions of objects are maintained if the scaling factors are the same ($s_x = s_y$). Otherwise, world objects will be stretched or contracted in either the x or y direction when displayed on the output device.

Clipping:
Any procedure that identifies those portions of a picture that are either inside or outside of a specified region of space is referred to as a clipping algorithm, or simply clipping. The region against which an object is clipped is called a clip window.

Applications of clipping:

- Extracting part of a defined scene for viewing
- Identifying visible surfaces in three-dimensional views
- Antialiasing line segments or object boundaries
- Creating objects using solid-modeling procedures
- Displaying a multi-window environment
- Drawing and painting operations that allow parts of a picture to be selected for copying, moving, erasing, or duplicating.

Depending on the application, the clip window can be a general polygon or it can even have curved boundaries.

Point Clipping: Assuming that the clip window is a rectangle in standard position, we save a point P = (x, y) for display if the following inequalities are satisfied:

$$Xw_{min} \le x \le xw_{max}$$

$$Yw_{min} \le y \le yw_{max}$$

Where the edges of the clip window (xw_{min}, xw_{max}, yw_{min}, yw_{max}) can be either the world-coordinate window boundaries or viewport boundaries. If any one of these four inequalities is not satisfied, the point is clipped (not saved for display).

Although point clipping is applied less often than line or polygon clipping, it can be applied to scenes involving explosions or sea foam that are modeled with particles (points) distributed in some region of the scene.

Line Clipping: A line clipping procedure involves several parts:

- First, we can test a given line segment to determine whether it lies completely inside the clipping window.
- If it does not, we try to determine whether it lies completely outside the window.
- Finally, if we cannot identify a line as completely inside or completely outside, we must perform intersection calculations with one or more clipping boundaries. We process lines through the "inside-outside" tests by checking the line endpoints.

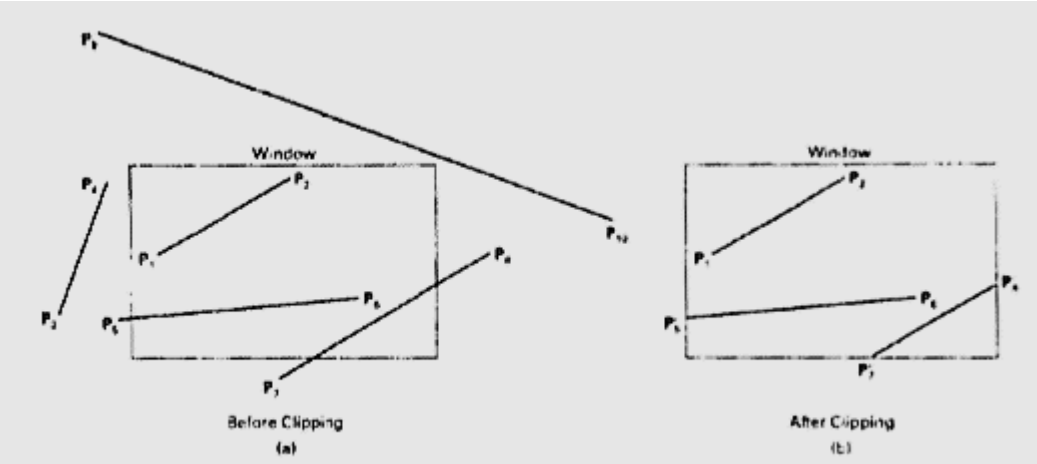


Fig: Line-clipping against a rectangular window

For a line segment with endpoints (x_1, y_1) and (x_2, y_2) and one or both endpoints outside the clipping rectangle, the parametric representation

$$x = x_1 + u(x_2-x_1)$$
$$y = y1 + u(y2-y1),\ 0 \leq u \leq 1$$

can be used to determine values of parameter u for intersections with the clipping boundary coordinates. If the value of u for an intersection with a rectangle boundary edge is outside the range 0 to 1, the line does not enter the interior of the window at the boundary. If the value of u is within the range from 0 to 1, the line segment does indeed cross into the clipping area. This method can be applied to each clipping boundary edge in turn to determine whether any part of the line segment is to be displayed. Line segments that are parallel to window edges can be handled as special cases.

Polygon Clipping: To clip polygons, we need to modify the line-clipping procedures discussed in the previous section. A polygon boundary processed with a line clipper may be displayed as a series of unconnected line segments depending on the orientation of the polygon to the clipping window.

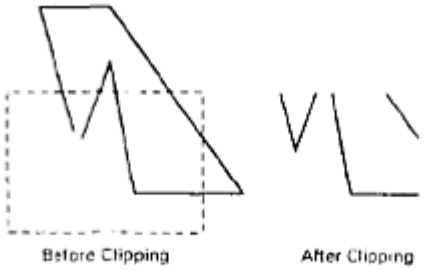


Fig: Display of a polygon processed by a line-dipping algorithm

What we really want to display is a bounded area after clipping, as in figure below:

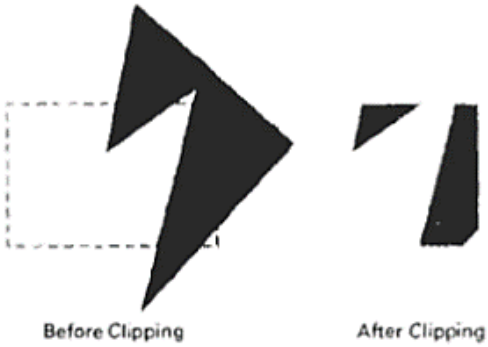


Fig: Display of a correctly clipped polygon

For polygon clipping, we require an algorithm that will generate one or more closed areas that are then scan-converted for the appropriate area fill. The output of a polygon clipper should be a sequence of vertices that defines the clipped polygon boundaries.

Curve Clipping:

Curve-clipping procedures will involve nonlinear equations and this requires more processing than for objects with linear boundaries. The bounding rectangle for a circle or other curved object can be used first to test for overlap with a rectangular clip window.

- If the bounding rectangle for the object is completely inside the window, we save the object.
- If the rectangle is determined to be completely outside window, we discard the object.

In either case, there is no further computation necessary. But if the bounding rectangle test fails, we can look for other computation-saving approaches.

- For a circle, we can use the coordinate extents of individual quadrants and then octants for preliminary testing before calculating curve-window intersections.
- For an ellipse, we can test the coordinate extents of individual quadrants

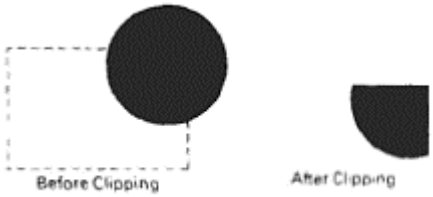


Fig: Clipping a filled Circle

2.2 Matrix representation of 3D transformations.

2D transformations can be represented by 3 x 3 matrices using homogenous coordinates, so 3D transformations can be represented by 4 x 4 matrices, providing we use homogeneous coordinate representations of points in 2 space as well.

Thus instead of representing a point as (x, y, z) , we represent if as (x , y , z, H), where two these quadruples represent the same point it one is a non-zero multiple of the other the quadruple (0,0,0,0) is not allowed.

A standard representation of a point (x , y ,z ,H) with H not zero is given by

$$(x/H , y/H, z/H, 1)$$

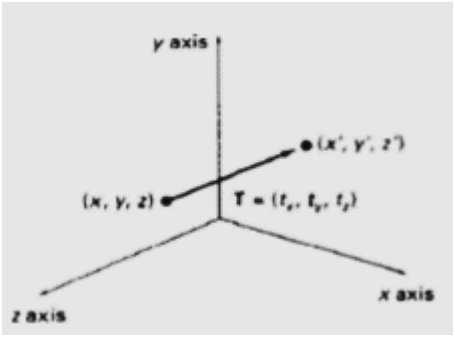
Transforming the point to this form is called homogenizing.

Translation:

A point is translated from position P = (x, y, z) to position P' = (x', y', z') with the matrix operation.

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

or $P' = T.P$



Parameters t_x, t_y, t_z specify translation distances for the coordinate directions x, y and z.
This matrix representation is equivalent to three equations:

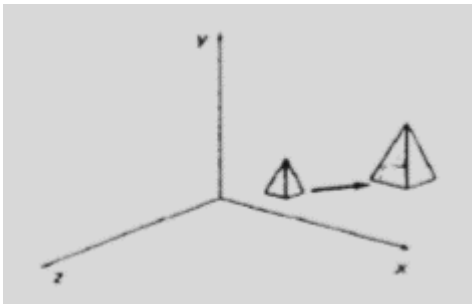
$$x' = x+t_x$$
$$y' = y+t_y$$
$$z' = z+t_z$$

Scaling:
Scaling changes size of an object and repositions the object relative to the coordinate origin.
If transformation parameters are not all equal then figure gets distorted. So we can preserve the original shape of an object with uniform scaling($s_x= s_y= s_z$).

Matrix expression for scaling transformation of a position P = (x,y,z) relative to the coordinate origin can be written as :

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

or $P' = S . P$

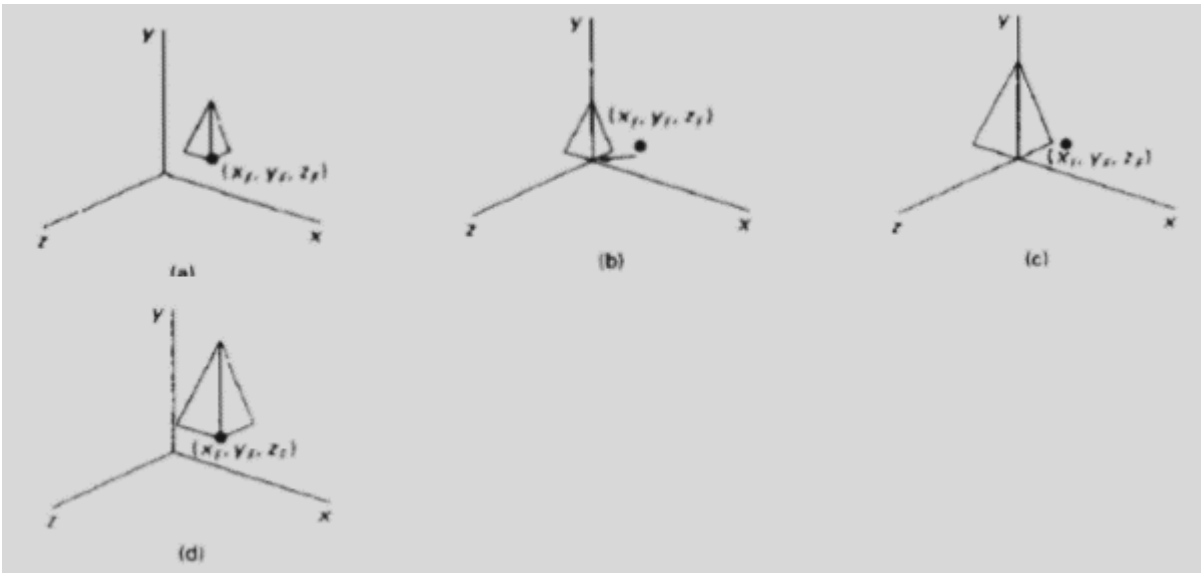


Scaling with respect to a selected fixed point (x_f , y_f , z_f) can be represented with:

i. Translate fixed point to the origin.
ii. Scale object relative to the coordinate origin.
iii. Translate fixed point back to its original position.

OR

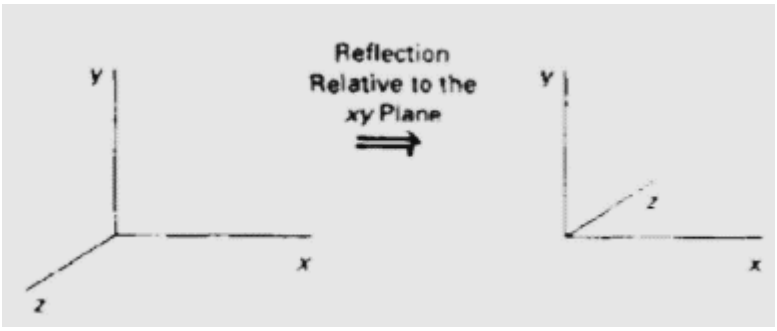
$$T(x_f, y_f, z_f) . S(s_x, s_y, s_z) . T(-x_f, -y_f, -z_f)$$



Reflection:

Reflections with respect to a plane are equivalent to 180^0 rotations in four dimensional space.

$$RFz = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



This transformation changes the sign of the z coordinates, leaving the x and y coordinate values unchanged.

Transformation matrices for inverting x and y values are defined similarly, as reflections relative to the yz plane and xz plane.

Shearing:

Shearing transformations are used to modify object shapes.

In 2D, shearing about x-axis means x-values changes by amount proportional to y-values and y-values remains same. However in 3D, z-axis means x and y value change by amount proportional to z-value and z-value remains same. i.e

$$x' = x + Sh_x.z \qquad y' = y + Sh_y.z \qquad z' = z$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & S_{hx} & 0 \\ 0 & 1 & S_{hy} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

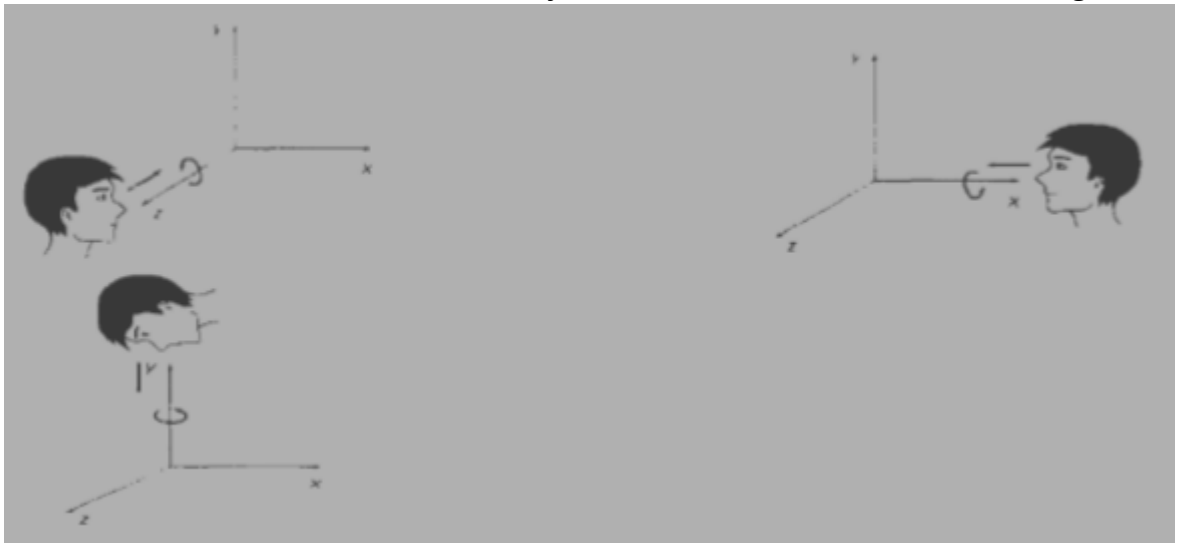
Similarly,

$$SH_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ S_{hx} & 1 & 0 & 0 \\ S_{hx} & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$SH_y = \begin{bmatrix} 1 & S_{hx} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & S_{hx} & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Rotation:

Designate the axis of rotation about which the object is to be rotated and the amount of angular rotation.



Axes that are parallel to the coordinate axes are easy to handle.

Coordinate axes Rotations:

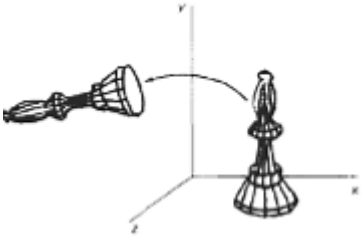
2D z-axis rotation equations are easily extended to 3D:

x' = xcosθ - y sinθ

y' = xsinθ + ycosθ

z' = z ---------- (i)

3D z-axis rotation equations are expressed in homogeneous coordinate form as:



$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} cos\theta & -sin\theta & 0 & 0 \\ sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

P' = Rz(θ) . P

or,

Cyclic permutation of the coordinate parameters x, y and z are used to get transformation equations for rotations about the other two coordinates:

$$x \rightarrow y \rightarrow z \rightarrow$$

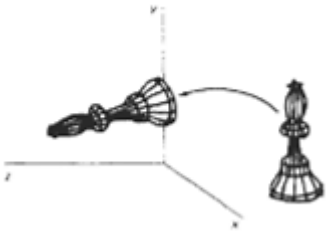
So, substituting permutations in (i) for an x axis rotation we get,

$y' = y\cos\theta - z\sin\theta$

$z' = y\sin\theta + z\cos\theta$

$x'=x$

3D x-axis rotation equations in homogenous coordinate form as:



$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Or,

$$P' = Rx(\theta).\ P$$

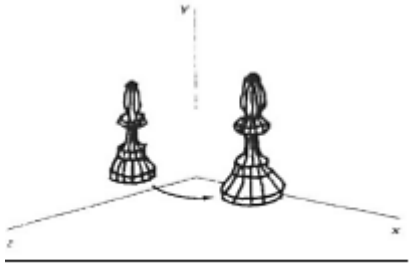
Substituting permutations in (i) for a y-axis rotation we get,

$z' = z\cos\theta - x\sin\theta$

$x' = z\sin\theta + x\cos\theta$

$y'=y$

3D y-axis rotation equations are expressed in homogenous coordinate form as:



$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & \cos\theta & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

$$P' = Ry(\theta)\ .\ P$$

Rotation about an axis parallel to one of the coordinate axes:

Steps:

i. Translate object so that rotation axis coincides with the parallel coordinate axis.
ii. Perform specified rotation about that axis.
iii. Translate object back to it's original position. i.e $P' = T^{-1}.Rx(\theta).T.P$

Rotation about an arbitrary axis:

An arbitrary axis in space passing through point (x_0, y_0, z_0) with direction cosines (c_x, c_y, c_z).

Steps:

i. Translate so that point (x_0, y_0, z_0) is at the origin of the coordinate system.
ii. Perform rotations to make axis of rotation coincident with the z axis.
iii. Rotate about the z axis by the angle θ.
iv. Perform the inverse of the combined rotation transformation.
v. Perform the inverse of the translation.

Reflection through an arbitrary plane:

General reflection matrices cause reflection through x=0, y=0, z=0 coordinate planes respectively.

Often it is necessary to reflect an object thru a plane other than one of these, which is obtained with the help of a series of transformations (composition).

i. Translate a known point P that lies in the reflection plane to the origin of the coordinate system.
ii. Rotate the normal vector to the reflection plane at the origin until it is coincident with the z axis. This makes the reflection plane the z=0 coordinate plane.
iii. After applying the above transformation to the object reflect the object thru the z=0 coordinate plane.

$$RFz = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

i.e.

iv. Perform the inverse transformation to those given above to achieve the desired result.

So, the general transformation matrix is:

$M(\theta) = T^{-1} . Rx^{-1}(\theta) . Ry^{-1}(\theta) . Rflct_z(\theta) . Ry(\theta). Rx\theta). T$

2.3 Viewing in 3D

In 2D we specify a window on 2D world and a view port on 2D view surface objects in world are clipped against window and are then transformed into view port for display.

Added complexity caused by:

i. Added dimension
ii. Display device are only 2D

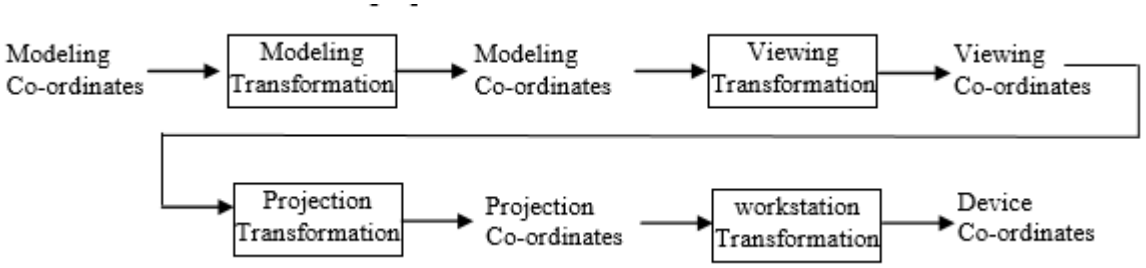
Solution is accomplished by introducing projections that transform 3D objects onto 2D plane.

In 3D we specify view volume (only those objects within view volume will appear in display on output device others are clipped from display) in world , projection onto projection plane and view port on view surface.

So objects in 3D world are clipped against 3D view volume and are then projected the contents of projection of view volume onto projection plane called window are then transformed (mapped onto) view port for display.

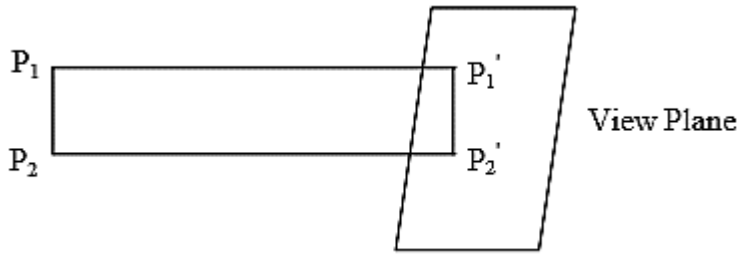
3D viewing pipeline:

The steps for computer generation of a view of 3D scene are analogous to the process of taking photograph by a camera. For a snapshot, we need to position the camera at a particular point in space and then need to decide camera orientation. Finally when we snap the shutter, the seen is cropped to the size of window of the camera and the light from the visible surfaces is projected into the camera film.



Projections: Once world co-ordinate description of the objects in a scene are converted to viewing co-ordinates, we can project the three dimensional objects onto the two dimensional view plane. There are two basic projection methods:

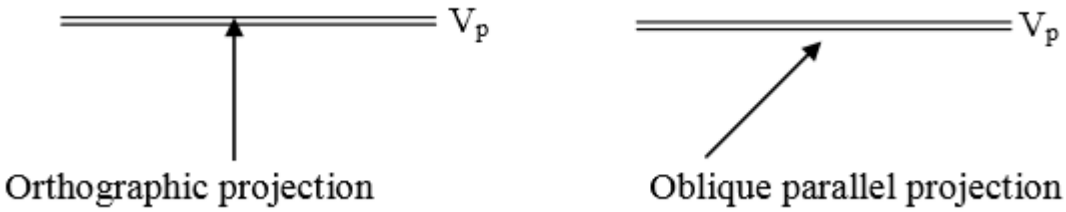
1. **Parallel Projection:** In parallel projection, co-ordinates positions are transformed to the view plane along parallel lines.



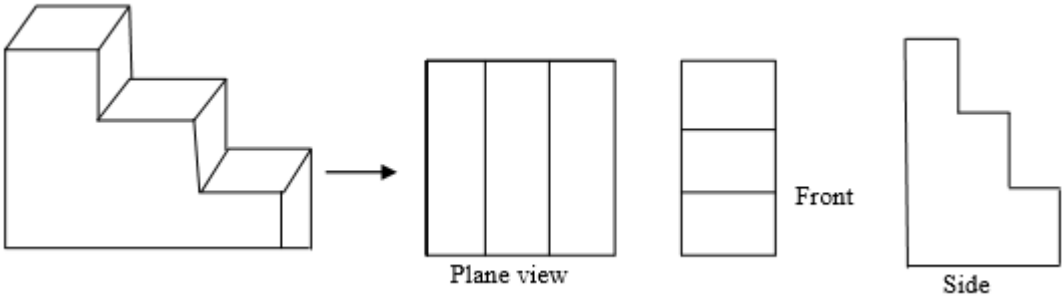
A parallel projection preserve relative proportions of objects and this is the method used in drafting to produce scale drawing of three-dimensional objects. Accurate view of various sides of 3D object is obtained with parallel projection. But it does not given a realistic appearance of a 3D-object.

A prospective projection, on the other hand, produces realistic views but does not preserve relative proportions. Projections of distance objects from view plane are smaller than the projections of objects of the same size that are closer to the projection place.

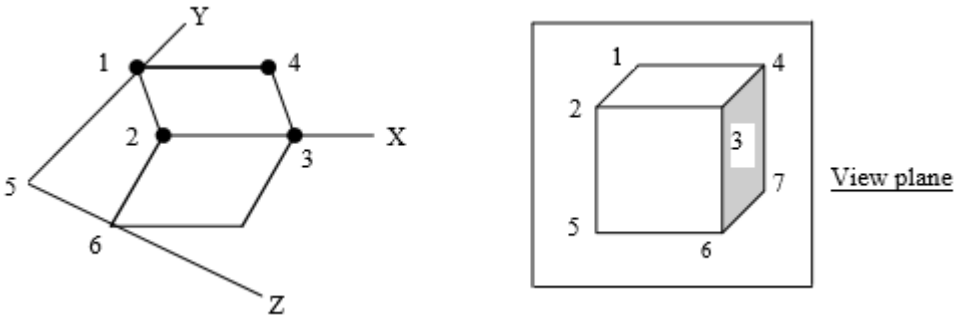
We can specify parallel projection withy projection vector that specifies the direction of projection line. When the projection lines are perpendicular to view plane, the projection is orthographic parallel projections. It projection line are not parallel to view plane then it is oblique parallel projection.



Orthographic projections are most often used to produce the front, side, and top views of an object. Front, side and rear orthographic are called elevations and the top orthographic view of object is known as plan view. Engineering and Architectural drawings commonly employ these orthographic projections.

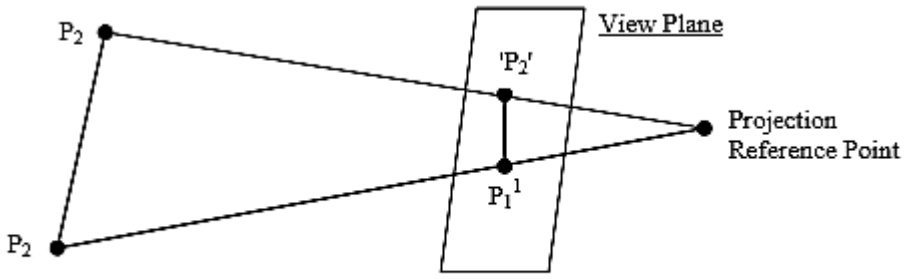


We also from orthographic projections that display more than one face of an object. Such views are called axonometric orthographic projections. the most commonly used axonometric projection is the isometric projection.



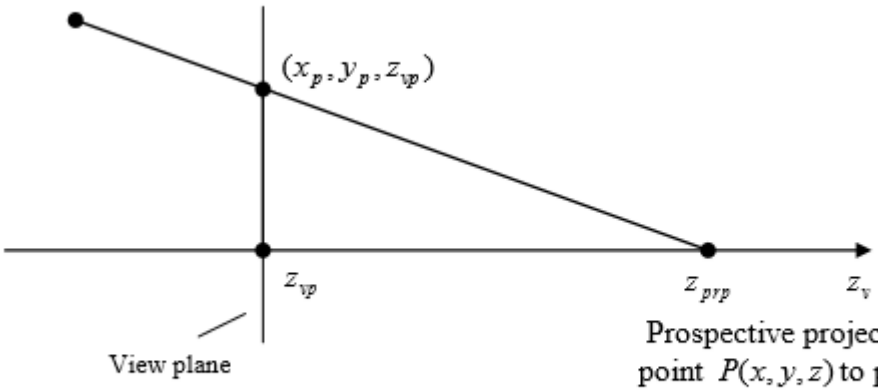
The transformation equation for orthographic projection is:
$x_p = x, y_p = y$, *z – coordinate value is preserved for the depth information.*

2. **Perspective Projection:** In prospective projection, objects positions are transformed to the view plane along lines that converge to a point called projection reference point (centre of projection). The projected view of an object is determined by calculating the intersection of the projection lines with the view plane.



To obtain a prospective projection of a three-dimensional object, we transform points along projection lines that meet at a point called projection reference point.

Suppose we set the projection reference point at position Z_{prp} along the Z_v axis, and we place the view plane at Z_{vp} as shown in fig:



Prospective projection of a point $P(x, y, z)$ to position (x_p, y_p, z_{vp}) on the view plane.

We can write equations describing co-ordinates positions along this prospective projection line in parametric form as:

$$x' = x - xu$$
$$y' = y - yu$$
$$z' = z - (z - z_{prp})u$$

Where u takes value from 0 to 1. If $u = 0$, we are at position $P = (x,y,z)$. If $u = 1$, we have projection reference point $(0, 0, z_{prp})$. On the view plane, $z' = z_{vp}$ then,

$$u = \frac{z_{vp} - z}{z_{prp} - z}$$

Substituting these value in equation for x', y',

$$x_p = x - x(\frac{z_{vp} - z}{z_{prp} - z}) = x(\frac{z_{prp} - z_{vp}}{z_{prp} - z}) = x(\frac{dp}{z_{prp} - z})$$

Similarly,

$$y_p = y(\frac{z_{prp} - z_{vp}}{z - z_{prp}}) = y(\frac{dp}{z_{prp} - z})$$

Where, $dp = z_{prp} - z_{vp}$ is the distance of the view plane from projection reference point.

Using 3D homogeneous co-ordinate representation, we can write prospective projection transformation matrix as:

$$\begin{bmatrix} x_h \\ y_h \\ z_h \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & z_{vp}/dp & \frac{-z_{vp}}{z_{prp}/dp} \\ 0 & 0 & 1/dp & z_{prp}/dp \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

In this representation the homogeneous factor:

$$h = \frac{z - z_{prp}}{dp}$$

Projection co-ordinates:

$$xp = {xh}/{h}, yp = {yh}/{h}$$

There are special case for prospective transformation.

When $z_{prp} = 0$,

$$x_p = x(\frac{z_{prp}}{z_{prp} - z}) = x(\frac{1}{1 - z/z_{prp}})$$

$$y_p = y(\frac{z_{prp}}{z_{prp} - z}) = y(\frac{1}{1 - z/z_{prp}})$$

Some graphics package, the projection point is always taken to be viewing co-ordinate origin. In this case, $z_{prp} = 0$.

$$\therefore x_p = x(\frac{zvp}{z}) = x(\frac{1}{z/zvp})$$

$$yp = y(\frac{zvp}{z}) = y\ (\frac{1}{z/zvp})$$

Exercise

1. Derive window to viewport transformation coefficient matrix. Explain the application of this matrix.
2. Explain the following term with practical applications. (i) 3D Rotation (ii) 2D Shear
3. Explain in detail about line clipping algorithm and its applications.
4. What do you mean by homogeneous coordinates? Explain it with equation and practical application.
5. Explain the following terms with practical applications. (a) 3D Mirror (ii) 2D Rotation
6. Explain in details about circle clipping algorithm. Where do you require circle clipping algorithm?
7. How can you represent 3D viewing? Explain it with equation and practical application.
8. Explain the following terms with practical applications. (i) 3D Translation (ii) 2D Mirror
9. Where do you require ellipse clipping algorithm? Explain in details about ellipse clipping algorithm.
10. Why homogeneous coordinates are used for transformation computations in computer graphics? Explain.
11. Differentiate between window port and view port. How are lines grouped into visible, invisible and partially visible categories in 2D clipping? Explain.
12. Write a procedure to fill the interior of a given ellipse with a specified pattern.
13. Show that two successive reflection about any line passing through the coordinate origins equivalent to a single rotation about the origin.
14. What do you mean by line clipping? Explain the procedures for line clipping.
15. Illustrate the windows to view point transformation with an example.
16. Explain the 2D and 3D transformations.

☺ ☺ ☺