UNIT 4:

Geometrical Transformations

2D Transformation

Two-dimensional transformations in computer graphics refer to the process of modifying the position, size, and orientation of 2D objects or shapes in a 2D coordinate system. These transformations are fundamental operations used to manipulate graphics elements and achieve various effects like translation, scaling, rotation, and shearing. Here are the common types of 2D transformations:

1. **Translation:** Translation involves moving an object by a specified distance along the x and y axes. It is represented by the translation vector (tx, ty), where tx is the horizontal translation, and ty is the vertical translation. Each point (x, y) of the object is translated to a new position (x', y') as follows: x' = x + tx y' = y + ty
2. **Scaling:** Scaling changes the size of an object by applying scaling factors sx and sy along the x and y axes, respectively. Scaling can make the object larger (sx, sy > 1) or smaller (0 < sx, sy < 1). The new coordinates (x', y') after scaling are calculated as: x' = x * sx y' = y * sy
3. **Rotation:** Rotation involves rotating an object around a specific point, often the origin (0, 0) or a user-defined center point (cx, cy). The angle of rotation is denoted by θ. To perform the rotation, the new coordinates (x', y') are determined as: x' = x * cos(θ) - y * sin(θ) y' = x * sin(θ) + y * cos(θ)
4. **Shearing:** Shearing distorts an object along one axis, leaving the other axis unchanged. There are two types of shearing: horizontal (shearing along the x-axis) and vertical (shearing along the y-axis). The amount of shearing is specified by the shearing factors (shx, shy). The new coordinates (x', y') after shearing are calculated as: x' = x + shx * y y' = y + shy * x
5. **Reflection:** Reflection flips or mirrors an object over a specific line, such as the x-axis, y-axis, or an arbitrary line defined by an angle. The reflection can be achieved by negating the corresponding coordinate values. For example, a reflection over the x-axis transforms (x, y) to (x, -y).
6. **Combining Transformations:** Multiple transformations can be combined to achieve more complex effects. The order in which transformations are applied matters. For

example, scaling followed by rotation yields different results than rotation followed by scaling.

Transformations play a crucial role in computer graphics, allowing objects to be positioned, oriented, and scaled as required for rendering and visualization. They are used in various applications, including 2D rendering, computer-aided design (CAD), video games, animations, and more. By applying appropriate transformations to objects, developers and designers can create dynamic and visually appealing graphics.

Homogeneous Coordinates

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

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

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

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

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

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

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

1. Translation	$\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\t_x & t_y & 1\end{bmatrix}$ or $\begin{bmatrix}1 & 0 & t_x\\0 & 1 & t_y\\0 & 0 & 1\end{bmatrix}$
2. Scaling	$\begin{bmatrix}S_x & 0 & 0\\0 & S_y & 0\\0 & 0 & 1\end{bmatrix}$
3. Rotation (clockwise)	$\begin{bmatrix}\cos\theta & -\sin\theta & 0\\\sin\theta & \cos\theta & 0\\0 & 0 & 1\end{bmatrix}$
4. Rotation (anti-clock)	$\begin{bmatrix}\cos\theta & \sin\theta & 0\\-\sin\theta & \cos\theta & 0\\0 & 0 & 1\end{bmatrix}$
5. Reflection against X axis	$\begin{bmatrix}1 & 0 & 0\\0 & -1 & 0\\0 & 0 & 1\end{bmatrix}$
6. Reflection against Y axis	$\begin{bmatrix}-1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}$
7. Reflection against origin	$\begin{bmatrix}-1 & 0 & 0\\0 & -1 & 0\\0 & 0 & 1\end{bmatrix}$
8. Reflection against line Y=X	$\begin{bmatrix}0 & 1 & 0\\1 & 0 & 0\\0 & 0 & 1\end{bmatrix}$
9. Reflection against Y= –X	$\begin{bmatrix}0 & -1 & 0\\-1 & 0 & 0\\0 & 0 & 1\end{bmatrix}$
10. Shearing in X direction	$\begin{bmatrix}1 & 0 & 0\\Sh_x & 1 & 0\\0 & 0 & 1\end{bmatrix}$
11. Shearing in Y direction	$\begin{bmatrix}1 & Sh_y & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}$
12. Shearing in both x and y direction	$\begin{bmatrix}1 & Sh_y & 0\\Sh_x & 1 & 0\\0 & 0 & 1\end{bmatrix}$

Matrix Representation of 2D Transformations

In computer graphics, 2D transformations can be represented using matrices. A 3x3 transformation matrix is used to encode translation, scaling, rotation, shearing, and reflection operations for 2D objects. Each type of transformation corresponds to a specific matrix, and multiple transformations can be combined by multiplying their respective matrices together.

Here are the matrix representations for some common 2D transformations:

1. **Translation:** Translation involves moving an object by a specified distance along the x and y axes. The translation matrix is given by:

Copy code

```
| 1 0 tx | | 0 1 ty | | 0 0 1 |
```

Where `tx` is the horizontal translation and `ty` is the vertical translation.

2. **Scaling:** Scaling changes the size of an object by applying scaling factors `sx` and `sy` along the x and y axes, respectively. The scaling matrix is given by:

Copy code

```
| sx 0 0 | | 0 sy 0 | | 0 0 1 |
```

Where `sx` is the scaling factor along the x-axis, and `sy` is the scaling factor along the y-axis.

3. **Rotation:** Rotation involves rotating an object around a specific point, often the origin (0, 0) or a user-defined center point (`cx`, `cy`). The rotation matrix is given by:

scssCopy code

```
| cos(θ) -sin(θ) 0 | | sin(θ) cos(θ) 0 | | 0 0 1 |
```

Where `θ` is the angle of rotation in radians.

4. **Shearing:** Shearing distorts an object along one axis while leaving the other axis unchanged. There are two types of shearing: horizontal (shearing along the x-axis) and vertical (shearing along the y-axis). The shearing matrix is given by:

Copy code

```
| 1 shx 0 | | shy 1 0 | | 0 0 1 |
```

Where `shx` is the horizontal shearing factor, and `shy` is the vertical shearing factor.

5. **Reflection:** Reflection flips or mirrors an object over a specific line, such as the x-axis, y-axis, or an arbitrary line defined by an angle. The reflection matrix depends on the axis of reflection:

- Reflection over the x-axis:

Copy code

```
| 1 0 0 | | 0 -1 0 | | 0 0 1 |
```

- Reflection over the y-axis:

Copy code

```
| -1 0 0 | | 0 1 0 | | 0 0 1 |
```

- Reflection over an arbitrary line with angle `α`:

scssCopy code

```
| cos(2α) sin(2α) 0 | | sin(2α) -cos(2α) 0 | | 0 0 1 |
```

To apply multiple transformations, you can multiply their respective matrices in the desired order. For example, to apply translation followed by rotation, you would multiply the translation matrix with the rotation matrix.

Using transformation matrices makes it easy to apply complex transformations to 2D objects efficiently. These matrices are essential in graphics pipelines for transforming objects before rendering and creating visually appealing graphics.

Reflection:

It is a transformation which produces a mirror image of an object. The mirror image can be either about x-axis or y-axis. The object is rotated by180°.

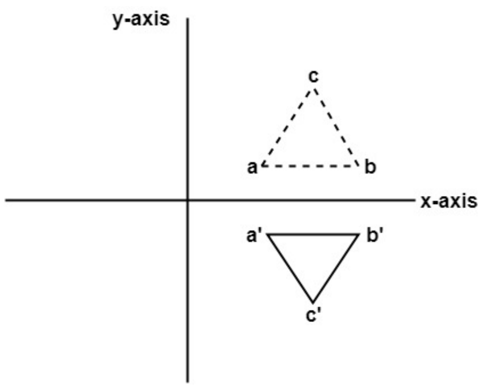
Types of Reflection:

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

1. Reflection about x-axis: The object can be reflected about x-axis with the help of the following matrix

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

In this transformation value of x will remain same whereas the value of y will become negative. Following figures shows the reflection of the object axis. The object will lie another side of the x-axis.

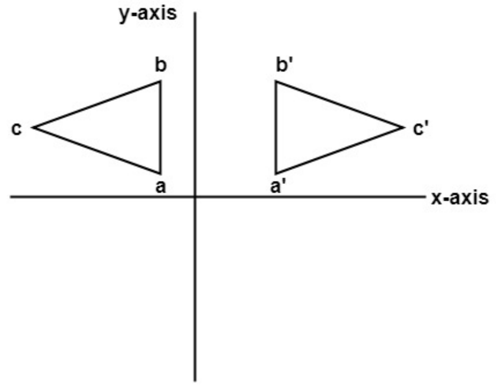


2. Reflection about y-axis: The object can be reflected about y-axis with the help of following transformation matrix

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

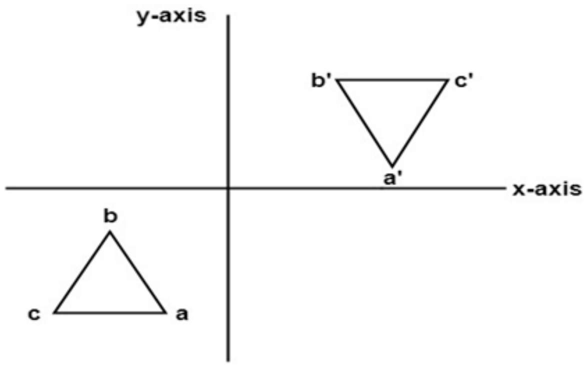
Here the values of x will be reversed, whereas the value of y will remain the same. The object will lie another side of the y-axis.

The following figure shows the reflection about the y-axis



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

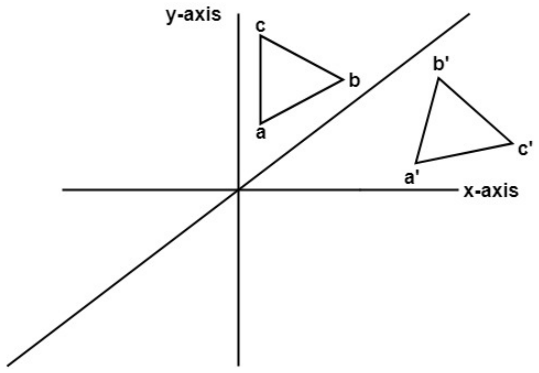
$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



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

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

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



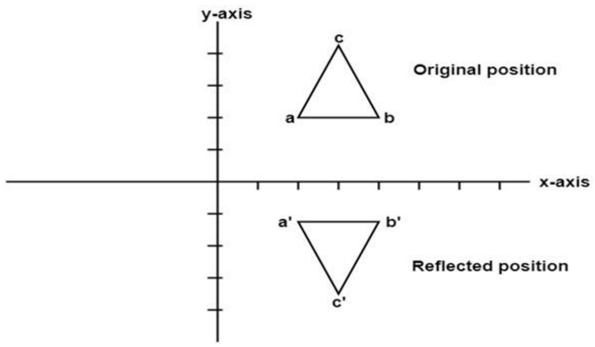
First of all, the object is rotated at 45°. The direction of rotation is clockwise. After it reflection is done concerning x-axis. The last step is the rotation of y=x back to its original position that is counterclockwise at 45°.

Example: A triangle ABC is given. The coordinates of A, B, C are given as

A (3 4)
B (6 4)
C (4 8)

Find reflected position of triangle i.e., to the x-axis.

Solution:



$$\text{The matrix for reflection about x axis} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The a point coordinates after reflection

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

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

The b point coordinates after reflection

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

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

The coordinate of point c after reflection

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

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

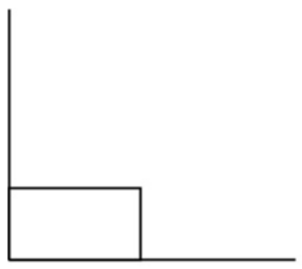
a (3, 4) becomes a^1 (3, -4)
b (6, 4) becomes b^1 (6, -4)
c (4, 8) becomes c^1 (4, -8)

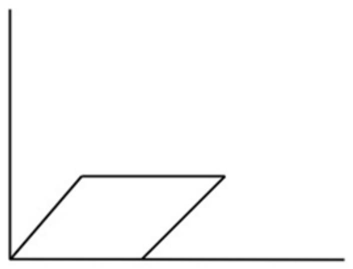
Shearing:

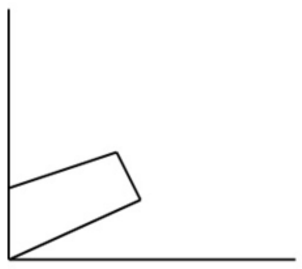
It is transformation which changes the shape of object. The sliding of layers of object occur. The shear can be in one direction or in two directions.

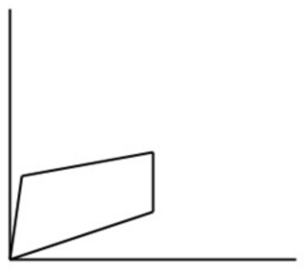
Shearing in the X-direction: In this horizontal shearing sliding of layers occur. The homogeneous matrix for shearing in the x-direction is shown below:

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


Original Object


Shear in X direction


Shear in Y direction


Shear in both directions

Shearing in the Y-direction: Here shearing is done by sliding along vertical or y-axis.

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

Shearing in X-Y directions: Here layers will be slided in both x as well as y direction. The sliding will be in horizontal as well as vertical direction. The shape of the object will be distorted. The matrix of shear in both directions is given by:

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

Inverse Transformations

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

Example1: Translation and its inverse matrix

Translation matrix

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

Inverse translation matrix

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

Example2: Rotation and its inverse matrix

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

Inverse Rotation Matrix

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

Viewing Transformation in computer graphics

Viewing transformation, also known as view transformation or view projection, is a crucial step in the graphics rendering pipeline that converts a 3D scene into a 2D image that can be displayed on a 2D screen. It involves transforming the 3D coordinates of objects in the scene into a new coordinate system, typically the 2D screen or view plane, so that they can be properly projected and rendered.

The viewing transformation is one of the key components of the graphics pipeline, which also includes modeling transformation, projection, and rasterization. The pipeline converts 3D objects and scenes into a 2D representation that can be displayed on the screen.

The viewing transformation typically includes the following steps:

1. **Camera Position and Orientation:** The first step is to define the position and orientation of the virtual camera in the 3D scene. The camera acts as the viewpoint through which the scene is observed. The camera position is represented by its coordinates (eye position), and its orientation is defined by its look-at point (a point towards which the camera is directed) and an up vector (specifying the camera's up direction).
2. **Viewing Frustum:** The viewing frustum is the 3D region that represents the portion of the scene visible from the camera. It is defined by six clipping planes: near, far, left, right, top, and bottom. The frustum helps determine which objects are visible and which are outside the camera's view.
3. **Viewing Transformation Matrix:** The viewing transformation is usually represented by a 4x4 transformation matrix, which combines translation, rotation, and scaling operations to position the camera and align it with the desired viewpoint. The matrix maps the 3D scene to a normalized view volume, which is a cube or parallelepiped centered at the origin.
4. **Perspective or Orthographic Projection:** After the viewing transformation, the scene is projected onto the 2D view plane using either a perspective or orthographic projection. In perspective projection, objects farther away from the camera appear smaller, mimicking how our eyes perceive depth. In orthographic projection, objects maintain their size regardless of their distance from the camera, resulting in a parallel projection.
5. **Viewport Transformation:** Finally, the 2D image is transformed from the normalized view volume to the actual screen coordinates (pixel coordinates) using a viewport transformation. The viewport maps the 2D view plane to the actual screen dimensions, allowing the scene to be rendered on the display.

The viewing transformation is essential for creating realistic 3D graphics and simulating 3D environments on a 2D screen. It enables the rendering of objects from different viewpoints, supporting interactive navigation and camera movement within the 3D scene

Normalize and Workstation transformation

Normalize transformation

- we have to define the picture co-ordinate in some units other than pixles and use the interprator to convert these co-ordinate to appropriate pixel value for a particular display device.
- the device independent unit is known as normalized device co-ordinator .
- The interprator use a simple linear formula to convert the normalize device co-ordinate to the actual device co-ordinate.

$$x = xn + Xw$$

$$y = yn + Yh$$

where x = actual device of x co-ordinate

y= actual device of y co-ordinate

xn = normalized x co-ordinate

yn = normalized y co- ordinate

Xw= width of the actual screen in pixel

Yh= actual height of the actual screen in pixel.

The transformation which maps the work co-ordinate to the normalized deviced co-ordinate is called Normalization transformation. It involve a scaling of x ,y co-ordinate then it also called scaling transformation.

Workstation transformation

- The transformation which maps the normalized device co-ordinate to physical device co-ordinate is known as workstation transfromation
- The window defined in world co-ordinates is first transformed into the normalized device co-ordinate then they are transformed into view- port co-ordinate.
- This window to view port transformation is called Workstation transformation.

This process is achieved by following steps :

1. The object together with its window is tanslated untill the lower left corner of the window is at the origin.
2. Object and the window are scaled until the window has the dimension of the view port.

Three Dimensional Transformations

The geometric transformations play a vital role in generating images of three Dimensional objects with the help of these transformations. The location of objects relative to others can be easily expressed. Sometimes viewpoint changes rapidly, or sometimes objects move in relation to each other. For this number of transformation can be carried out repeatedly.

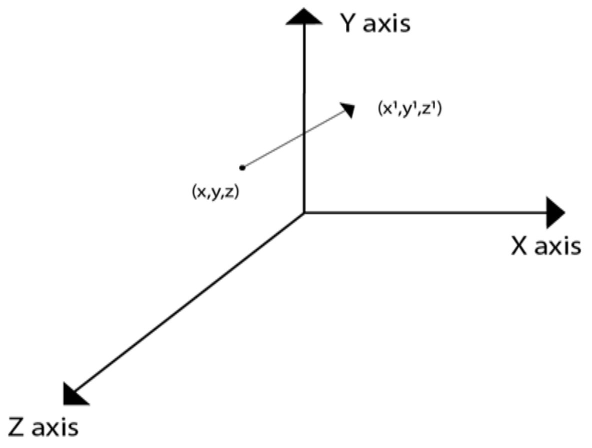
Translation

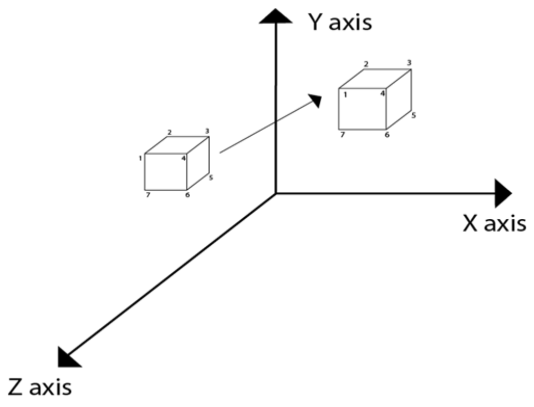
It is the movement of an object from one position to another position. Translation is done using translation vectors. There are three vectors in 3D instead of two. These vectors are in x, y, and z directions. Translation in the x-direction is represented using T_x. The translation is y-direction is represented using T_y. The translation in the z- direction is represented using T_z.

If P is a point having co-ordinates in three directions (x, y, z) is translated, then after translation its coordinates will be (x^1 y^1 z^1) after translation. T_x T_y T_z are translation vectors in x, y, and z directions respectively.

$x^1=x+ T_x$
$y^1=y+T_y$
$z^1=z+ T_z$

Three-dimensional transformations are performed by transforming each vertex of the object. If an object has five corners, then the translation will be accomplished by translating all five points to new locations. Following figure 1 shows the translation of point figure 2 shows the translation of the cube.





Matrix for translation

$$\begin{Bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{Bmatrix} \text{ or } \begin{Bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{Bmatrix}$$

Matrix representation of point translation

Point shown in fig is (x, y, z). It become (x^1,y^1,z^1) after translation. T_x T_y T_z are translation vector.

$$\begin{bmatrix} x^1 \\ y^1 \\ z^1 \\ 1 \end{bmatrix} = \begin{pmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Example: A point has coordinates in the x, y, z direction i.e., (5, 6, 7). The translation is done in the x-direction by 3 coordinate and y direction. Three coordinates and in the z-direction by two coordinates. Shift the object. Find coordinates of the new position.

Solution: Co-ordinate of the point are (5, 6, 7)
Translation vector in x direction = 3
Translation vector in y direction = 3
Translation vector in z direction = 2
Translation matrix is

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

Multiply co-ordinates of point with translation matrix

$$(x^1 y^1 z^1) = (5,6,7,1) \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 3 & 3 & 2 & 1 \end{pmatrix}$$

$$= [5+0+0+30+6+0+30+0+7+20+0+0+1] = [8991]$$

x becomes $x^1=8$
y becomes $y^1=9$
z becomes $z^1=9$