3 Transformation

3.1 TWO DIMENSIONAL TRANSFORMATION (2D)

We know that once the objects are created, the different applications may require the variations in these. For example, suppose that we have created a scene of a room. As we move along the room we find the object's position comes closer to us, it appears bigger and even its orientation will change. Thus we need to manipulate these objects. In some cases, the object size may be large, which needs to be stored in compressed form. All these requirements suggest us that we need to transform these objects. Essentially this process is carried out by means of transformations. Transformation is a process of changing the position of the object or may be any combination of these.

The objects are referenced by their co-ordinates. Geometric transformation allows us to calculate the new co-ordinates. In the following section we will study these transformations.

3.2 TRANSLATION

Translation is a process of changing the position of an object. Let P be an object point with the co-ordinates (x, y). We wish to move this point P to new position say P', having the co-ordinates (x', y'). refer Fig. 3.1.

Notice here that, P has been moved by some quantity in x direction as well as y-direction. Let t_x, t_y be the magnitudes by which we have moved in x and y directions respectively. Thus we can write,

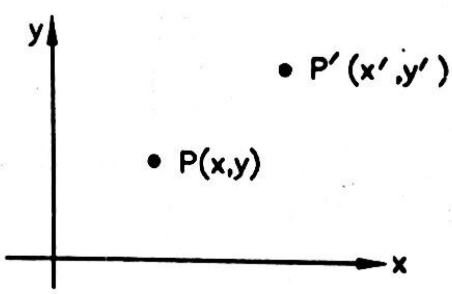


Fig. 3.1.

$$\left.\begin{aligned} x' &= x + t_x \\ y' &= y + t_y \end{aligned}\right\} \quad \ldots (i)$$

The quantities t_x, t_y are called as translation vectors. We can represent the equation in the matrix form. This infact simplifies the operations on complex objects. The required matrix form is :

$$\begin{bmatrix}\text{New object}\\ \text{Co-ordinate}\\ \text{Matrix}\end{bmatrix} = \begin{bmatrix}\text{Transformation}\\ \text{matrix}\end{bmatrix} \cdot \begin{bmatrix}\text{Previous object}\\ \text{co-ordinate}\\ \text{matrix}\end{bmatrix}$$

We wish to mention here that the transformation matrix should contain only the transformation factors and no co-ordinate values.

Fot the translation of point [Equation (*i*)], we can write it as :

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

Example 1 : Translate a square *ABCD* with the co-ordinates $A(0, 0)$, $B(5, 0)$, $C(5, 5)$, $D(0, 5)$ by 2 units in x direction and 3 units in *y* direction.

Solution : Translation

$$\begin{bmatrix}x\\ y\\ 1\end{bmatrix} = \begin{bmatrix}A & B & C & D\\ 0 & 5 & 5 & 0\\ 0 & 0 & 5 & 5\\ 1 & 1 & 1 & 1\end{bmatrix}$$

Given $t_x = 2 \quad t_y = 3$

The transformation matrix is = $\begin{bmatrix}1 & 0 & 2\\ 0 & 1 & 3\\ 0 & 0 & 1\end{bmatrix}$

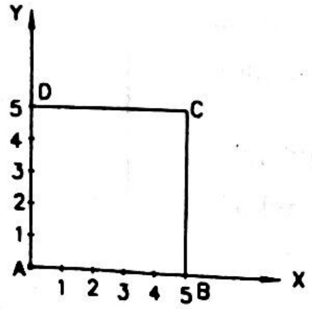
$$\begin{bmatrix}x'\\ y'\\ 1\end{bmatrix} = \begin{bmatrix}1 & 0 & 2\\ 0 & 1 & 3\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}0 & 5 & 5 & 0\\ 0 & 0 & 5 & 5\\ 1 & 1 & 1 & 1\end{bmatrix}$$

$$= \begin{bmatrix}2 & 7 & 7 & 2\\ 3 & 3 & 8 & 8\\ 1 & 1 & 1 & 1\end{bmatrix}$$

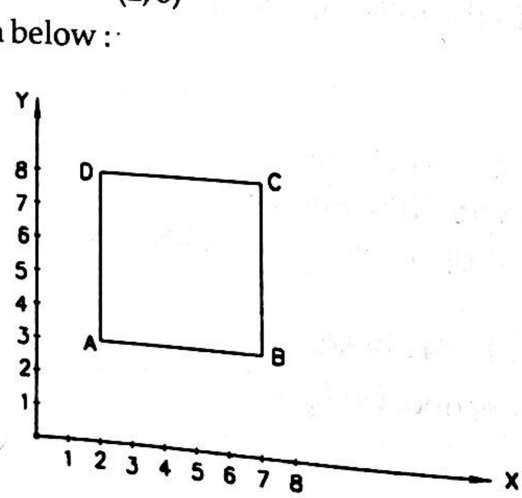
$A'(2,3)$, $B'(7,3)$

$C'(7,8)$; and $D'(2,8)$

The graphical representation is given below :



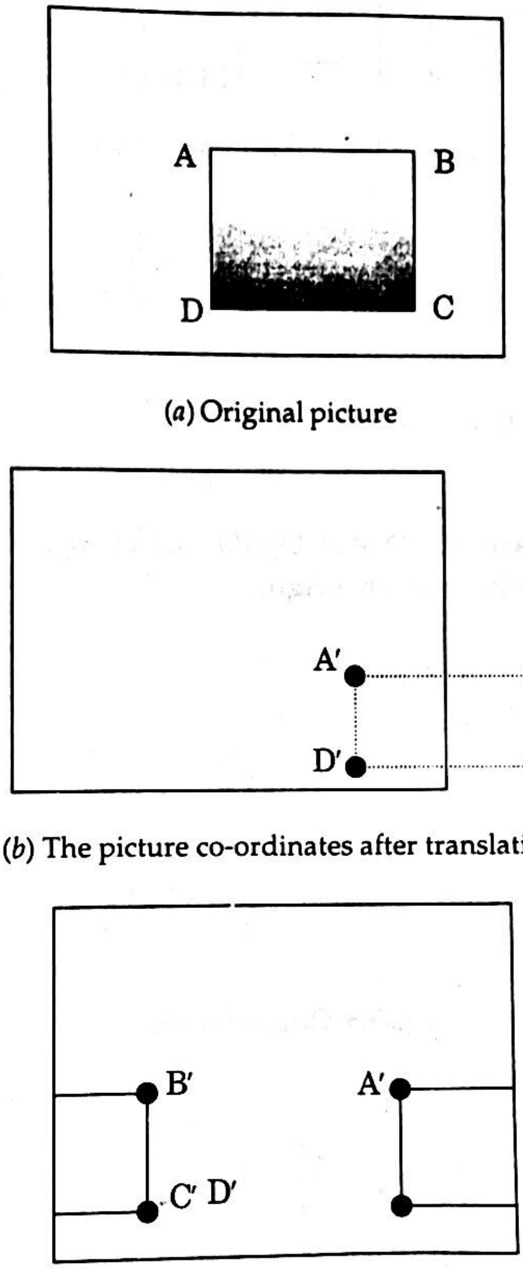
(A) Square before translation



(B) Square after translation

Fig. 3.2.

However, if the translation factors yield the new co-ordinates, which are greater than the device co-ordinates, then the system may result an error message. In this case we can clip the extended portion. In case, the system does not have any provision due to overflow, this effect is known as warp-around, where the extended points will be displayed on other side of the screen. Suppose, we have a square *ABCD* as shown in Fig 3.3 (*a*). Now after translation we are getting the picture like the one shown in Fig 3.3 (*b*). However due to wrap-around effect we actually see the square as distorted one; as shown in Fig 3.3 (*c*).



(*a*) Original picture

(*b*) The picture co-ordinates after translation

(*c*) Wrap-around effect

Fig. 3.3.

3.3 SCALING

Scaling is a transformation, which either magnifies or reduces the size of the object. If the scaling factor is less than zero, it reduces the size of the object, if scaling is greater than zero, then it magnifies the size of the object.

Let P be an object point with the coordinate (x, y) and that S_x, S_y be scaling factors in x and y directions respectively. So we can obtain the co ordinate of the scaled object as :

$$\left.\begin{aligned} x' &= x\,S\,x \\ y' &= y\,S\,y \end{aligned}\right\} \quad \ldots(i)$$

We can represent this in a matrix form, as :

$$\begin{bmatrix} \text{New object} \\ \text{Matrix} \end{bmatrix} = \begin{bmatrix} \text{Scaling} \\ \text{Matrix} \end{bmatrix} \begin{bmatrix} \text{Old} \\ \text{object} \\ \text{Matrix} \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

If we have the same scaling factors, then there will be uniform scaling.

$$\text{Scaling Matrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Example 2 : Scale the square ABCD, A(0, 0), B(3, 0), C(3, 3), D(0, 3) three units in x-direction and three units in y-direction with respect to origin.

Solution :

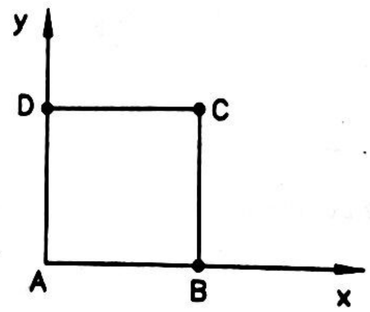


Fig. 3.5. Original object.

Given Here, $S_x = 3, S_y = 3$

The object matrix is :

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{array}{cccc} A & B & C & D \end{array} \begin{bmatrix} 0 & 3 & 3 & 0 \\ 0 & 0 & 3 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\text{Scaling matrix is} = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

New coordunals = Scaling matrix × object matrix

$$= \begin{bmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 3 & 3 & 0 \\ 0 & 0 & 3 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 9 & 9 & 0 \\ 0 & 0 & 9 & 9 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

A′ (0, 0)
B′ (9, 0)
C′ (9, 9)
and D′ (0, 9)

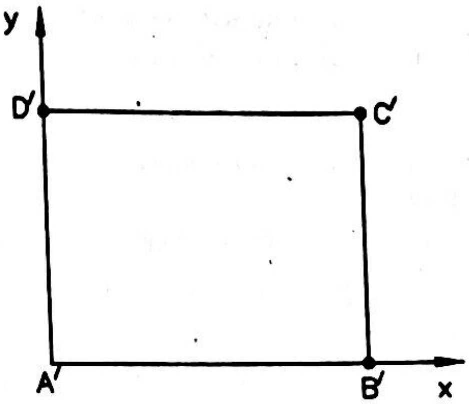


Fig. 3.6. Scaled object.

3.4 ROTATION

In this case we try to rotate the object by a given angle. Let us first take the simple case :

⇒ Object being rotated, be a point $P(x, y)$.

⇒ Let the angle of rotation be $\theta°$ in counter-clockwise direction.

⇒ Rotation is with respect to origin.

Refer Fig. 3.7

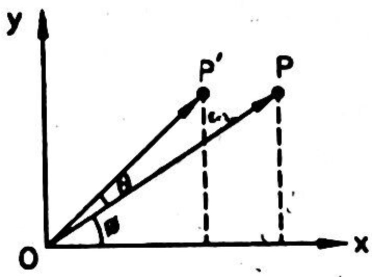


Fig. 3.7.

Note that, when the object is rotated about origin, it gets rotated with fixed radius. Hence from Fig. 3.7

$$\cos\theta = \frac{x}{r}$$

LAL
KKA

$$\sin\theta = \frac{y}{r}$$

$$\therefore \quad \boxed{\begin{aligned} x &= r\cos\theta \\ y &= r\sin\theta. \end{aligned}} \qquad ...(1)$$

Now let rotat me point $p(x, y)$ to $p^1(x^1, y^1)$ counter clockwise by angle ϕ.

$$\therefore \quad \cos(\theta + \phi) = \frac{x^1}{r}$$

$$\sin(\theta + \phi) = \frac{y^1}{r}$$

or

$$\boxed{\begin{aligned} x^1 &= r\cos(\theta + \phi) \\ y^1 &= r\sin\theta\ (\theta + \phi) \end{aligned}} \quad ...(2)$$

Expend Equations (2) we get

$$x^1 = r(\cos\theta \cos\phi - \sin\theta \sin\phi)$$
$$x^1 = r\cos\theta \cos\phi - r\sin\theta \sin\phi$$

and

$$y_1 = r(\sin\theta \cos\phi + \cos\theta \sin\phi)$$
$$= r\sin\theta \cos\phi + r\cos\theta \sin\phi$$

Therefore can we get

$$\begin{aligned} x^1 &= r\cos\theta \cos\phi - r\sin\theta \sin\phi \\ y^1 &= r\sin\theta \cos\phi + r\cos\theta \sin\phi \end{aligned} \quad ...(3)$$

Substitute equation (1) in (3) we get

$$x^1 = x\cos\phi - y\sin\phi$$
$$y^1 = x\sin\phi + y\cos\phi$$

We represent this in the matrix form as :

$$\begin{bmatrix} x^1 \\ y^1 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Example 3 : Obtain a matrix representation of an object by 45° about the origin.

Solution. $$R = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

(**Note :** *We are writing co-ordinate matrix of object as row wise.*)

$$R_{45^\circ} = \begin{bmatrix} \cos 45^\circ & -\sin 45^\circ \\ \sin 45^\circ & \cos 45^\circ \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$

$\left(\because \text{ After rationalizing by } \frac{1}{\sqrt{2}}\right)$

Example 4 : Rotate an object defined by $A(0, 0)$, $B(1, 0)$, $C(1, 1)$ and $D(0, 1)$ by 45° about origin.

Solution.

We want to find here,

$$R_{45^\circ} = \begin{bmatrix} \cos 45^\circ & -\sin 45^\circ & 0 \\ \sin 45^\circ & \cos 45^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{Object Matrix } ABCD = \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

Now coordinates after rotation $= R_{45^\circ} \times \square ABCD$

$$= \begin{pmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 0 & \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} \\ 0 & \frac{\sqrt{2}}{2} & \sqrt{2} & \frac{\sqrt{2}}{2} \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

$$A^1 = (0, 0)$$
$$B^1 = \left(\sqrt{2}/2, \sqrt{2}/2\right)$$
$$C^1 = (0, \sqrt{2})$$
$$D^1 = \left(\frac{-\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right) \quad \text{...co-ordinate rotation}$$

3.5 REFLECTION

Reflection is a transformation which generates the mirror image of an object. Let us consider the point object $P(x, y)$. For reflection we need to know the reference axis. Consider the case of reflection about x-axis.

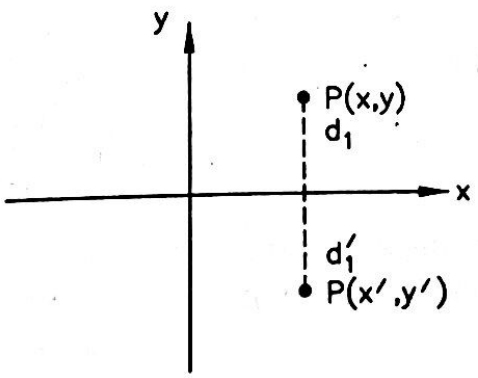


Fig. 3.8.

Here the d_1 and d'_1 are same.

$$\left.\begin{aligned} \Rightarrow \quad x' &= x \\ \Rightarrow \quad y' &= -y. \end{aligned}\right\} \quad \text{...}(i)$$

The eqn. (*i*), we represent in matrix form as follows :-

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

Similarly, reflection about *y*-axis is given by :

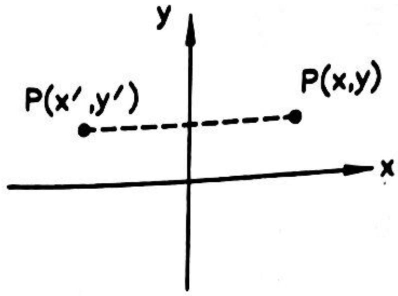


Fig. 3.9.

The eqn. (*ii*) we represent in matrix form as follows : -

$$\left.\begin{aligned} x' &= -x \\ y' &= y \end{aligned}\right\} \quad \text{...(ii)}$$

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

or

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

Program —Write a program Rotation of Triangle, Line and Rectangle.

Solution.

```
//rotation
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
#include <math.h>
float x1,y1,x2,y2,x,y,x3,y3,x4,y4,a;
int ch;
int main(void)
{
  int gdriver = DETECT, gmode, errorcode;
  initgraph(&gdriver, &gmode, "d:\\tc\\bgi");
  errorcode = graphresult();
  if (errorcode != grOk)
  {
printf("Graphics error: %s\n", grapherrormsg(errorcode));
printf("Press any key to halt:");
getch();
exit(1);
```

```
}
do
{
getch();
clrscr();
cout<<"  ############MAIN-MENU###############\n";
cout<<"          ROTATION\n";
cout<<"          1.LINE\n";
cout<<"          2.RECTANGLE\n";
cout<<"          3.TRIANGLE\n";
cout<<"enter your choice:0 for exit:\n";
cin>>ch;
switch(ch)

{
case 1 : cout<<"enter the values of line coordinates:";
          cin>>x1>>y1>>x2>>y2;
          cout<<"enter the value for angle of rotation:",
          cin>>a;
          cleardevice();
          line(x1,y1,x2,y2);
          a=a*(3.14/180);
          x1=(x1*cos(a))-(y1*sin(a));
          y1=(x1*sin(a))+(y1*cos(a));
          x2=(x2*cos(a))-(y2*sin(a));
          y2=(x2*sin(a))+(y2*cos(a));
          cout<<"now hit a key to see rotation:";
          getch();
          line(x1,y1,x2,y2);
          break;

case 2 : cout<<"enter the top left coordinates:";
          cin>>x1>>y1;
          cout<<"enter the bottom right coordinates:";
          cin>>x2>>y2;
          cout<<"enter the value for angle of rotation:";
          cin>>a;
          cleardevice();
          rectangle(x1,y1,x2,y2);
          a=a*(3.14/180);
          x1=(x1*cos(a))-(y1*sin(a));
          y1=(x1*sin(a))+(y1*cos(a));
          x2=(x2*cos(a))-(y2*sin(a));
          y2=(x2*sin(a))+(y2*cos(a));
          cout<<"now hit a key to see rotation:";
          getch();
```

```
                    recctangle(x1,y1,x2,y2);
                    break;
        case 3 :  cout<<"enter coordinates of line1:\n";
                    cin>>x1>>y1>>x2>>y2;
                    cout<<"enter coordinates for relative line:\n";
                    cin>>x3>>y3;
                    cout<<"enter the angle of rotation:\n";cin>>a;
                    cleardevice();
                    line(x1,y1,x2,y2);
                    moveto(x2,y2);
                    lineto(x3,y3);
                    moveto(x3,y3);
                    lineto(x1,y1);
                    a=a*(3.14/180);
                    x1=(x1*cos(a))-(y1*sin(a));
                    y1=(x1*sin(a))+(y1*cos(a));
                    x2=(x2*cos(a))-(y2*sin(a));
                    y2=(x2*sin(a))+(y2*cos(a));
                    x3=(x3*cos(a))-(y3*sin(a));
                    y3=(x3*sin(a))+(y3*cos(a));
                    cout<<"now hit a key to see rotation:";
                    getch();
                    moveto(x1,y1);
              }
        }
```

3.6 SHEARING TRANSFORMATION

When some enternal tangential force is applied to an object, it destorts the shape of the object such that the transformed shape appears as if the object was deformed due to its internal layer which slide over each other is called shear and a transformation is called shearing transformation.

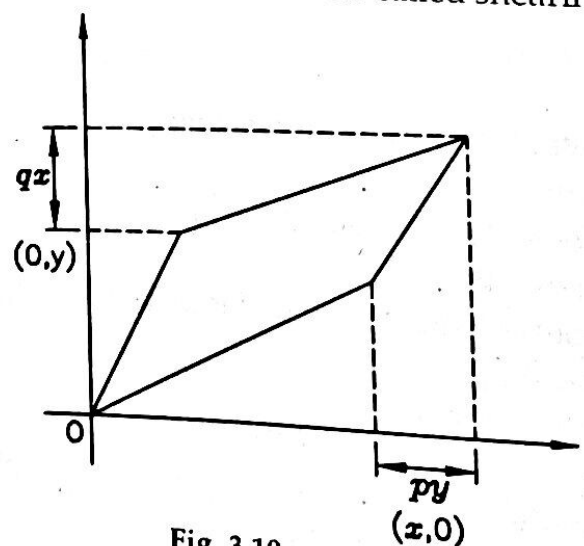


Fig. 3.10.

When shearing is applied in both x & y directions simultaneously, it is termed as simultaneous shearing, represented by the following matrix

$$= \begin{bmatrix} 1 & p \\ q & 1 \end{bmatrix}$$

Where p is shearing in x-direction and q is shearing in y direction.

When $q = 0$, an x-direction shearing relative to the x-axis is produced with the transformation matrix

$$\begin{bmatrix} x^1 \\ y^1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & p & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Where x' and y' can be represented in the form of following equations :

$$x_1 = x + p \cdot y$$
$$y_1 = y.$$

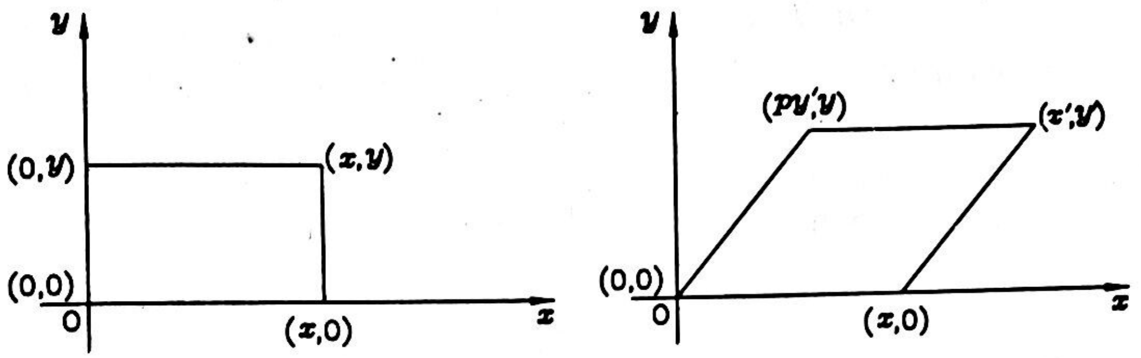


Fig. 3.11. Shearing in x-direction.

Similarly, when $p = 0$, an x-direction shearing relative to the x-axis is given by following matrix

$$\begin{bmatrix} x^1 \\ y^1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ q & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

where $x_1 = x$

and $y_1 = qx + y.$

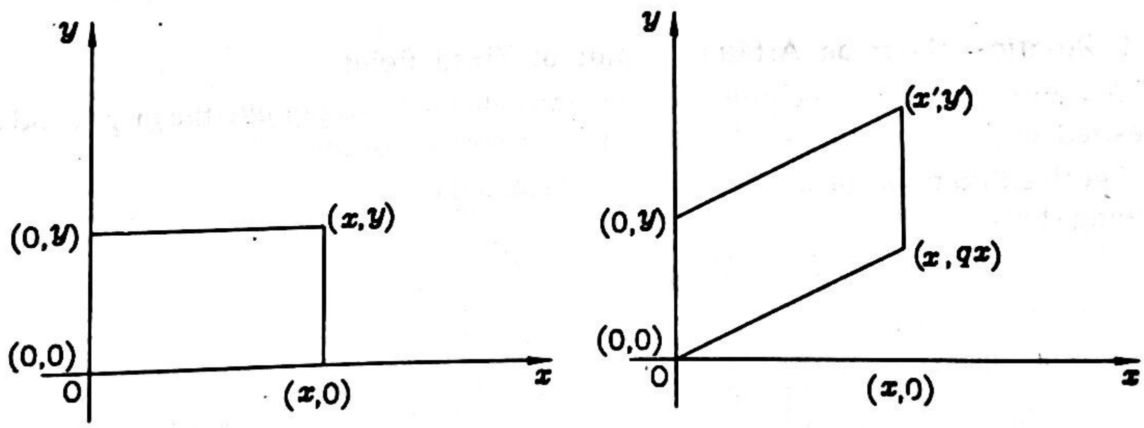


Fig. 3.12. Shearing in y-direction.

Example 5. Calculate the coordinates after shearing transformation on the square $A(0, 0)$, $B(2, 0)$, $C(2, 2)$ and $D(0, 2)$ when the shearing factors are 2 and 4.

Solution. Shearing Matrix = $\begin{bmatrix} 1 & 2 & 0 \\ 4 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$$\square ABCD = \begin{bmatrix} 0 & 2 & 2 & 0 \\ 0 & 0 & 2 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\square A^1B^1C^1D^1 = \text{Sh.} \square ABCD.$$

$$= \begin{pmatrix} 1 & 2 & 0 \\ 4 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 2 & 2 & 0 \\ 0 & 0 & 2 & 2 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

$$\square A^1B^1C^1D^1 = \begin{pmatrix} 0 & 2 & 6 & 4 \\ 0 & 8 & 10 & 2 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Thus, $A^1\,(0,0)$
$B^1\,(2,8)$
$C^1\,(6,10)$
$D^1\,(4,2)$

3.7 INSTANT TRANSFORMATION

The transformation in which the instance of the object is placed into a picture co-ordinate space, is called the instant transformation. Here the object is created using object co-ordinate space, and picture co-ordinate system is independent of the object space. The transformation techniques used here are corresponding to the three dimensional transformation.

3.8 COMPOSITE TRANSFORMATION

Transformation involves max than are basic transformation is called composite transformation. For example relation about an arbitrary point, mirror reflection about an arbitrary line rather than any axis etc.

3.8.1 Rotation about an Arbitrary Point or Fixed Point

In rotation about a fined point first we translate the fined point to the origin and then we rotate by desired angle and again translate back to its original position :

Let fixed point is $p(h, k)$ and point to be rotated counter clockwise is $p(x, y)$. This require the following steps :

$$R_{\theta, p} = T_v R_\theta T_{-v}$$

where $$T_v = \begin{bmatrix} 1 & 0 & h \\ 0 & 1 & k \\ 0 & 0 & 1 \end{bmatrix}, \quad R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$T_{-v} = \begin{bmatrix} 1 & 0 & -h \\ 0 & 1 & -k \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_{\theta,p} = \begin{bmatrix} 1 & 0 & h \\ 0 & 1 & k \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -h \\ 0 & 1 & -k \\ 0 & 0 & 1 \end{bmatrix}$$

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

3.8.2 Scaling about a Fixed Point

In scaling with respect to any fixed point involves the following steps :

1. Translate the arbitary point $p(h, k)$ to origin.
3. Scale to the point as desired.
4. Translate back to $P(h, k)$.

$$S(S_x, S_y), P(h, k) = T_v \, S_{sx} \, S_y \, T_{-v}$$

where

$$T_v = \begin{bmatrix} 1 & 0 & h \\ 0 & 1 & k \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_{-v} = \begin{bmatrix} 1 & 0 & -h \\ 0 & 1 & -k \\ 0 & 0 & 1 \end{bmatrix}$$

$$S(S_{x_1} S_y) = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S_{sn'} S_y p = \begin{bmatrix} S_x & 0 & -S_x h + h \\ 0 & 6 & -S_y k + k \\ 0 & 0 & 1 \end{bmatrix}$$

3.8.3 Reflection of an Object about any Line

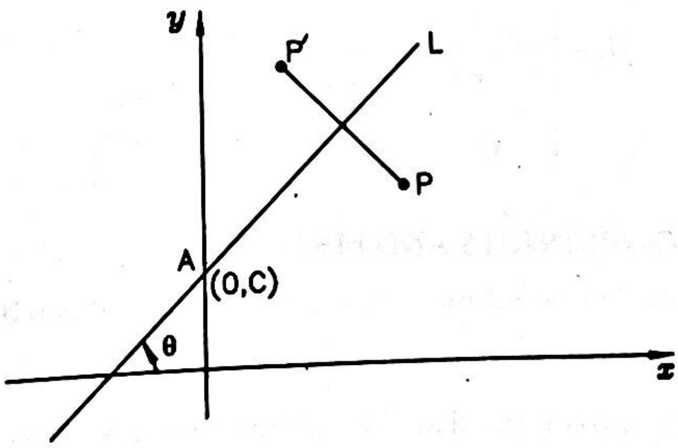


Fig. 3.13

Let line $y - m_x + c$ have a y-intercept $(0, c)$ and slope (c) with x-axis. The steps involved in this composite transformation are as follows :

1. Translate the intersection point A to the origin.
2. Rotate by $-\theta$, so that Line L aligns with x-axis.
3. Mirror reflect about the x-axis.
4. Rotate back by 0°.
5. Translate A back to the point $(0, c)$.

Therefore, we have,

$$M_L = T_v R_\theta M_x R_{-\theta} T_{-v}$$

where $v = C_J$

$$T_v = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & C \\ 0 & 0 & 1 \end{bmatrix}, \qquad T_{-v} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -C \\ 0 & 0 & 1 \end{bmatrix}$$

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

and

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

here stage is $m = \tan\theta$

therefore,

$$\sin\theta = \frac{m}{\sqrt{m^2+1}}$$

$$\cos = \frac{1}{\sqrt{m^2+1}}$$

After multiplication of all basic matrix we get

$$M_L = \begin{bmatrix} \frac{1-m^2}{m^2+1} & \frac{2m}{m^2+1} & \frac{-2Cm}{} \\ \frac{2m}{m^2+1} & \frac{m^2-1}{m^2+1} & \frac{2C}{m^2+1} \\ 0 & 0 & 1 \end{bmatrix}.$$

3.9 HOMOGENEOUS COORDINATES AND MATRICES

The formula for computing two-dimensional transformations can be represented in a uniform way with 3 × 3 matrices. Although they map two-dimensional space into two-dimensional space, 2 × 2 matrices cannot describe all three transformations; translation is impossible through a 2 × 2 matrix. We can get uniform matrix formalism if we define homogeneous coordinates.

We can represent a point with a Cartesian description as (x, y) or we can use homogeneous coordinates to represent it as $(x\ y\ 1)$ (1 at the end). We consider all homogeneous coordinates $(w^*x\ w^*y\ w)$ for every value of $w \neq 0$ as identical to $(x\ y\ 1)$.

Using normalized homogeneous coordinates and matrices, the transformation equations become :

Multiplying a normalized homogenous vector by any of the matrices shown will always yield a normalized homogeneous vector; that is, a vector with a 1 at the end. We can discard 1 from the result.

We concatenate transformations by multiplying the corresponding matrices in the order in which we want the transformations to occur. The resulting matrix expresses the overall transformation. This transformation has some advantage, nomatter how many 3×3 matrices we multiply with each other, the result is always a 3×3 matrix.

Suppose we have an object consisting of 200 vertices, which we will rotate around a point other than the origin, then translate, then scale about its center, then rotate about someother point, and so on. If we first express all transformations by homogeneous matrices and multiply them into the overall matrix, we can then simply subject vertex to one multiplication with this matrix and save computation time. Multiplying a vertex by the overall matrix requires roughly the same amount of computation as multiplying the vrtex by any one of the simple transformation matrices. If we do not use our matrix formalism, w must do many more computations.

Our matrix formalism is very efficient. No matter how many matrices of the form

$$\begin{pmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{pmatrix}$$

we multiply, the resulting matrix will always have the vector $(0\,0\,1)^T$ in its rightmost column. Multiplying a point $(x\ y\ 1)$ by this matrix will require a maximum of four multiplications and four additions.

The column (0 0 1) on the right is redundant and can be left out when specifying such a matrix.

SOLVED PROBLEMS

Problem 3.1. *Prove that two scaling transformations are commute i.e.* $S_1S_2 = S_2S_1$.

Solution. Let S_1 is scaling by factor p about origin and S_2 is scaling by factor q after S_1. We have,

$$S_1 = \begin{bmatrix} p & 0 & 0 \\ 0 & p & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S_2 = \begin{bmatrix} q & 0 & 0 \\ 0 & q & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

then $$S_1S_2 = \begin{bmatrix} p & 0 & 0 \\ 0 & p & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} q & 0 & 0 \\ 0 & q & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} pq & 0 & 0 \\ 0 & pq & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad ...(1)$$

and $$S_2 S_1 = \begin{bmatrix} q & 0 & 0 \\ 0 & q & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p & 0 & 0 \\ 0 & p & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S_2 S_1 = \begin{bmatrix} pq & 0 & 0 \\ 0 & pq & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

From equations (1) and (2) we suggests

$$\boxed{S_1 S_2 = S_2 S_1.}$$

Problem 3.2. *Prove that two 2D rotations about the origin are commute i.e.*

$$R_1 R_2 = R_2 R_1.$$

Solution. Let,
$$R_1 = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and}$$

Let
$$R_2 = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_1 R_2 = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

and
$$R_2 R_1 = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

$\therefore$
$$\boxed{R_1 R_2 = R_2 R_1.}$$

Problem 3.3. *Perform a 45° rotation of triangle $A(0, 0)$, $B(1, 1)$ and $C(5, 2)$.*

(a) about the origin and

(b) about the point $P(-1, -1)$.

Solution. Object Matrix =
$$\begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{bmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} \end{array}$$

Rotation Matrix $R_{45^\circ} = \begin{bmatrix} \cos 45^\circ & -\sin 45^\circ & 0 \\ \sin 45^\circ & \cos 45^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

New Coordinates $= R_{45^\circ} \times$ Object Matrix

$$= \begin{pmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 0 & 0 & \frac{5\sqrt{2}}{2} - 2 \\ 0 & \frac{\sqrt{2}}{2} + \frac{5\sqrt{2}}{2} & \frac{5\sqrt{2}}{2} + \sqrt{2} \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & \frac{3\sqrt{2}}{2} \\ 0 & \sqrt{2} & \frac{7\sqrt{2}}{2} \\ 0 & 0 & 1 \end{pmatrix}$$

$\therefore$

$A^1\ (0, 0)$

$B^1\ (0, \sqrt{2}\ )$

$C^1 \left(\frac{3\sqrt{2}}{2}, \frac{1/\sqrt{2}}{2}\right)$

(*b*) We have $R_{\theta, p} = T_v \cdot R_\theta \cdot T_{-v}$

$$R_{45^\circ, p} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & -1 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & -1 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & -1 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{-\sqrt{2}}{2}-1 & -1 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & -1 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}-2}{2} & -1 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & -1 \\ 0 & 1 & 1 \end{bmatrix}$$

$$\therefore \quad \Delta' ABC = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}-2}{2} & -1 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & -1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & .5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} -1 & \frac{\sqrt{2}}{2}+\frac{\sqrt{2}-2}{2}+-1 & \frac{5\sqrt{2}}{2}+\sqrt{2}-2-1 \\ -1 & \frac{\sqrt{2}}{2}+\frac{\sqrt{2}}{2}-1 & \frac{5\sqrt{2}}{2}+\frac{2\sqrt{2}}{2}-1 \\ 1 & 2 & 3 \end{bmatrix}.$$

Problem 3.4. *Define the shearing transformation.*

Solution. The shearing transformation is defined by the matrix

$$\begin{pmatrix} 1 & a \\ b & 1 \end{pmatrix}$$

Now if $a = 0$, it is called shearing x, and when $b = 0$ it is shearing in y-direction. If a and b ar not zero it defines the shearing transformation in both the directions.

Problem 3.5. *The triangle ABC is, A (0, 0) B(50) and C(0, 5). The shearing transformation with a = 3 and b = 4 is ...*

Solution. The shearing transformation is

$$S_T = \begin{pmatrix} 1 & a \\ b & 1 \end{pmatrix} = \begin{pmatrix} 1 & 3 \\ 4 & 1 \end{pmatrix}$$

The object matrix is :

$$Obj = \begin{pmatrix} A & B & C \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{pmatrix}$$

$$[A'B'C'] = S_T.Obj$$

$$= \begin{pmatrix} 1 & 3 \\ 4 & 1 \end{pmatrix} \begin{pmatrix} 0 & 5 & 0 \\ 0 & 0 & 5 \end{pmatrix}$$

$$= \begin{pmatrix} A' & B' & C' \\ 0 & 5 & 15 \\ 0 & 20 & 5 \end{pmatrix}$$

Problem 3.6. *Write a program shearing of Line, Triangle and Rectangle.*

Solution.

```
//shearing

#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
int x1,y1,x2,y2,x,y,x3,y3,x4,y4,ch;
int main(void)
{
        int gdriver = DETECT, gmode, errorcode;
        initgraph(&gdriver, &gmode, "d:\\tc\\bgi");
        errorcode = graphresult();
        if (errorcode != grOk)
        {
                printf("Graphics error: %s\n", grapherrormsg(errorcode));
                printf("Press any key to halt:");
                getch();
                exit(1);
}
 do
 {
 getch();
 clrscr();
 cout<<"  #############MAIN-ENU###############\n";
 cout<<"                    SHEARING\n";
```

```
cout<<"                1.LINE\n";
cout<<"                2.RECTANGLE\n";
cout<<"                3.TRIANGLE\n";
cout<<"enter your choice:0 for exit:\n";
cin>>ch;
switch(ch)
{
case 1 :        cout<<"enter the values of line coordinates:";
                cin>>x1>>y1>>x2>>y2;
                cout<<"enter the value of shearing for xaxis:";
                cin>>x;
                cout<<"enter the value of shearing for y-axis:";
                cin>>y;
                cleardevice();
                line(x1,y1,x2,y2);
                cout<<"now hit a key to see shear in x_axis:";
                getch();
                line(x1,y1,x2*x,y2);
                cout<<"\nnow hit a key to see shear in y_axis:";
                getch();
                line(x1,y1,x2,y2*y);
                break;

case 2 :        cout<<"enter the top left coordinates:";
                cin>>x1>>y1;
                cout<<"enter the bottom right coordinates:";
                cin>>x2>>y2;
                cout<<"enter the values of shearing coordinate for x_shear:\n";
                cin>>x;
                cout<<"enter the values of shearing coordinate for y_shear:\n";
                cin>>y;

                cleardevice();
                rectangle(x1,y1,x2,y2);
                cout<<"now hit a key to see shear in x_axis:";
                getch();
                rectangle(x1,y1,x2*x,y2);
                cout<<"\nnow hit a key to see shear in y_axis:";
                getch();
                rectangle(x1,y1,x2,y2*y);
                break;
case 3 :      cout<<"enter coordinates of line1:\n";
                cin>>x1>>y1>>x2>>y2;
                cout<<"enter coordinates for relative line:\n";
                cin>>x3>>y3;
```

```
        cout<<"enter shear coordinate for x_shear:\n";cin>>x;
        cout<<"enter shear coordinate for y_shear:\n";cin>>x;

        cleardevice();
        line(x1,y1,x2,y2);
        moveto(x2,y2);
        lineto(x3,y3);
        moveto(x3,y3);
        lineto(x1,y1);
        cout<<"\nnow hit a key to see shear in x_axis:";
        getch();
        moveto(x1,y1);
        lineto(x2*x,y2);
        moveto(x2*x,y2);
        lineto(x3*x,y3);
        moveto(x3*x,y3);
        lineto(x1,y1);
        cout<<"\nnow hit a key to see shear in y_axis:";
        getch();
        moveto(x1,y1);
        lineto(x2,y2*y);
        moveto(x2,y2*y);
        lineto(x3,y3*y);
        moveto(x3,y3*y);
        lineto(x1,y1);
        break;
case 0 :break;
        default:cout<<"invalid choice";break;
        }}while(ch!=0);
        getch();
        closegraph();
        }
```

Problem 3.7. *Write a program translation of Line, Triangle and Rectangle.*

Solution.

```
//Translation

#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
int x1,y1,x2,y2,x,y,x3,y3,x4,y4,ch;
int main(void)
{
```

```
        int gdriver = DETECT, gmode, errorcode;
        initgraph(&gdriver, &gmode, "d:\\tc\\bgi");
        errorcode = graphresult();
        if (errorcode != grOk)
        {
                printf("Graphics error: %s\n", grapherrormsg(errorcode));
                printf("Press any key to halt:");
                getch();
                exit(1);
}
do
{
getch();
cleardevice();
cout<<"  ############MAIN-MENU##############\n";
cout<<"           TRANSLATION\n";
cout<<"           1.LINE\n";
cout<<"           2.RECTANGLE\n";
cout<<"           3.TRIANGLE\n";
cout<<"enter your choice:0 for exit:\n";
cin>>ch;
switch(ch)
{
case 1 :        cout<<"enter the values of line coordinates:";
                cin>>x1>>y1>>x2>>y2;
                cout<<"enter the translation oordinates:";cin>>x>>y;
                cleardevice();
                line(x1,y1,x2,y2);
                cout<<"now hit a key to see translation:";
                getch();
                line(x1+x,y1+y,x2+x,y2+y);
                break;

case 2 :        cout<<"enter the top left coordinates:";
                cin>>x1>>y1;
                cout<<"enter the bottom right coordinates:";
                cin>>x2>>y2;
                cout<<"enter the values of translation oordinates:\n";
                cin>>x>>y;
                cleardevice();
                rectangle(x1,y1,x2,y2);
                cout<<"now hit a key to see translation:";
                getch();
                rectangle(x1+x,y1+y,x2+x,y2+y);
                break;
```

```
case 3 :        cout<<"enter coordinates of line1:\n";
                cin>>x1>>y1>>x2>>y2;
                cout<<"enter coordinates for relative line:\n";
                cin>>x3>>y3;
                cout<<"enter translation coordinates:\n";cin>>x>>y;
                cleardevice();
                line(x1,y1,x2,y2);
                moveto(x2,y2);
                lineto(x3,y3);
                moveto(x3,y3);
                lineto(x1,y1);
                cout<<"now hit a key to see translation:";
                getch();
                moveto(x1+x,y1+y);
                lineto(x2+x,y2+y);
                moveto(x2+x,y2+y);
                lineto(x3+x,y3+y);
                moveto(x3+x,y3+y);
                lineto(x1+x,y1+y);
                break;
case 0 : break;
default:cout<<"invalid choice";break;
}}while(ch!=0);
getch();
closegraph();
}
```

Problem 3.8. *The pyramid defined by the coordinate A(0, 0, 0), B(1, 0, 0), C(0, 1, 0) and D(0, 0, 1) is rotated 45° about the line L that has the direction,*

$$V = J + K$$

and passing through point C(0, 1, 0). Find the coordinates of the rotated figure.

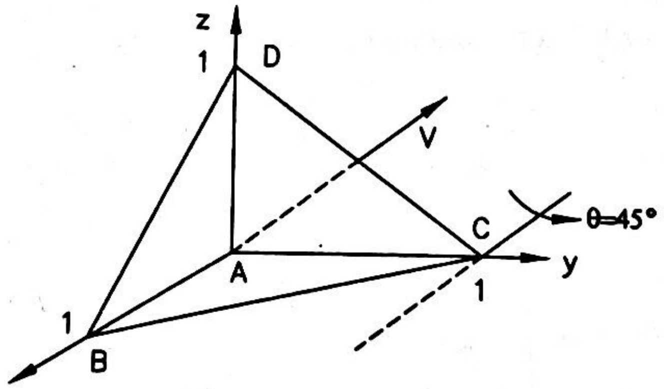


Fig. 3.14.

Solution. The rotation matrix $R_{\theta, L}$ can be found by concatenating the matrices,

i.e.,
$$R_{\theta, L} = T_{-P}^{-1} \cdot A_V^{-1} \cdot R_{\theta, K} \cdot A_V \cdot T_{-P}$$

when $P = (0, 1, 0)$, then

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

Now, $\boxed{V = J + K}$

with $a = 0, b = 1, c = 1$

we find, $|V| = \sqrt{a^2 + b^2 + c^2} = \sqrt{0+1+1} = \sqrt{2}$

$$\lambda = \sqrt{b^2 + c^2} = \sqrt{1+1} = \sqrt{2}$$

and
$$A_V = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 0 & 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad A_V^{-1} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Also,
$$R_{45^\circ, K} = \begin{pmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad T_{-P}^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Then,
$$R_{\theta, L} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{-1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{2+\sqrt{2}}{4} & \frac{2-\sqrt{2}}{4} & \frac{2-\sqrt{2}}{4} \\ \frac{-1}{2} & \frac{2-\sqrt{2}}{4} & \frac{2+\sqrt{2}}{4} & \frac{\sqrt{2}-2}{4} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

To find the coordinates of the rotated figure, we apply the rotation matrix $R_{\theta, L}$ to the matrix of homogeneous coordinates of the vertices A, B, C and D.

$$C = (ABCD) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

$$R_{\theta; L.C} = \begin{pmatrix} \frac{1}{2} & \frac{1+\sqrt{2}}{4} & 0 & 1 \\ \frac{2-\sqrt{2}}{4} & \frac{4-\sqrt{2}}{4} & 1 & \frac{2-\sqrt{2}}{2} \\ \frac{\sqrt{2}-2}{4} & \frac{\sqrt{2}-4}{4} & 0 & \frac{\sqrt{2}}{2} \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Then the rotated coordinates are,

$$A' = \left(\frac{1}{2}, \frac{2-\sqrt{2}}{4}, \frac{\sqrt{2}-2}{4}\right)$$

$$B' = \left(\frac{1+\sqrt{2}}{2}, \frac{4-\sqrt{2}}{4}, \frac{\sqrt{2}-4}{4}\right)$$

$$C' = (0, 1, 0)$$

$$D' = \left(1, \frac{2-\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)$$

Following is the rotated figure,

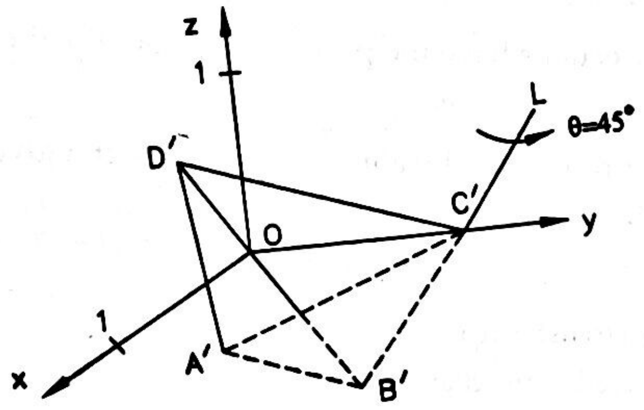


Fig. 3.15.

FLASH BACK

- ❖ **Transformation** is a process carried out by means of transformation these objects, or changing the size of the object or changing the orientation of the object or may be any combination of these.
- ❖ **Translation** is a process of changing the position of an object.
- ❖ **Scaling** is a transformation, which either magnifies or reduces the size of the object.
- ❖ **Rotation,** in this case we try to rotate the object by a given angle.
- ❖ **Instance Transformation** in which the instance of the object is placed into a picture co-ordinate space, is called the instance transformation.
- ❖ **Mirror Reflection,** we need to know the reference plane *i.e.* plane about which the reflection is to be taken.
- ❖ **Tilting Transformation** wherein the object is first rotated about x-axis and then about y-axis.
- ❖ **Projection** is a process of representing a three dimensional object or scene into two dimensional medium.
- ❖ **Prespective Projection,** this kind of projections are characterised by the kind of projectors used converging projectors.