**DEPARTMENT OF COMPUTER ENGINEERING**

**EXPERIMENT NO. 7**

**AIM:** To implement 2D transformation

**THEORY:**

1)What are homogeneous coordinates? Explain with an example.

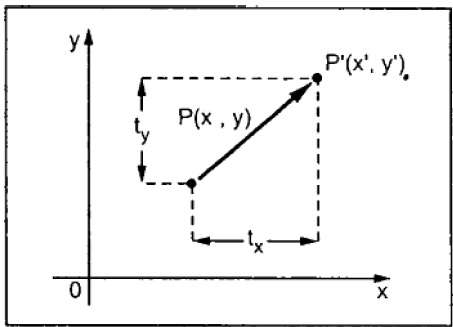
In mathematics, homogeneous coordinates or projective coordinates is a system of coordinates used in projective geometry, as Cartesian coordinates used in Euclidean geometry. It is a coordinate system that algebraically treats all points in the projective plane (both Euclidean and ideal) equally.

In particular, (x, y, 1) is such a system of homogeneous coordinates for the point (x, y). For example, the Cartesian point (1, 2) can be represented in homogeneous coordinates as (1, 2, 1) or (2, 4, 2). The original Cartesian coordinates are recovered by dividing the first two positions by the third.

2) Explain the five 2D transformations with 3x3 matrices and with appropriate examples

**Translation:**

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

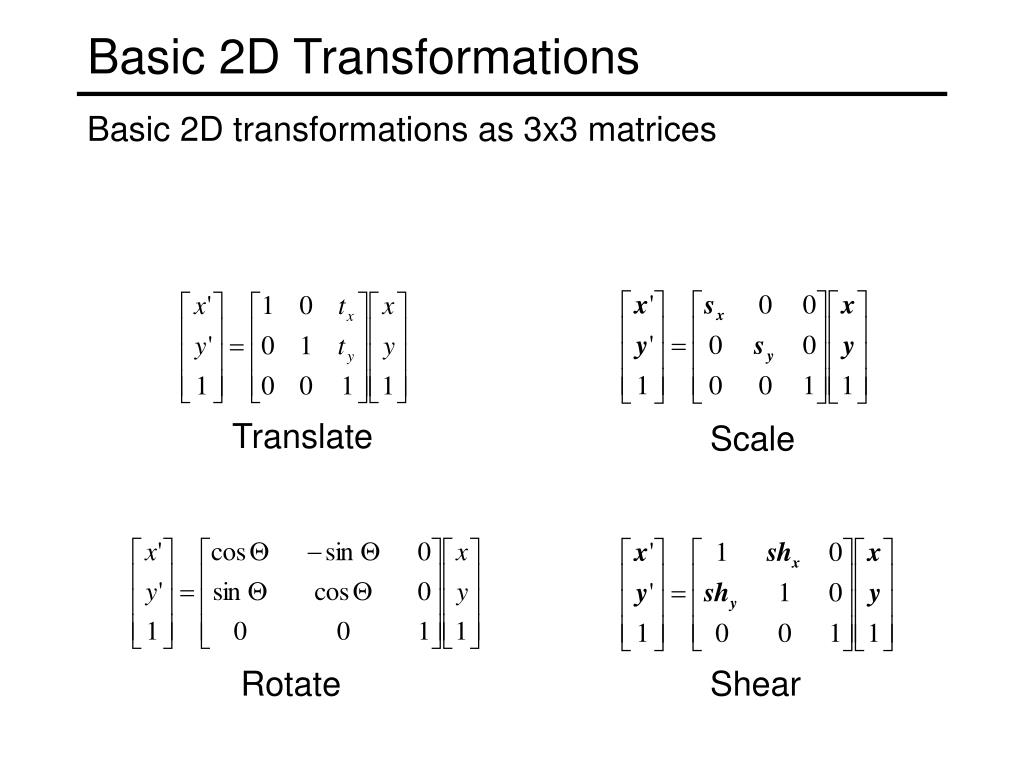


From the above figure, you can write that −

**X’ = X + tx**

**Y’ = Y + ty**

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

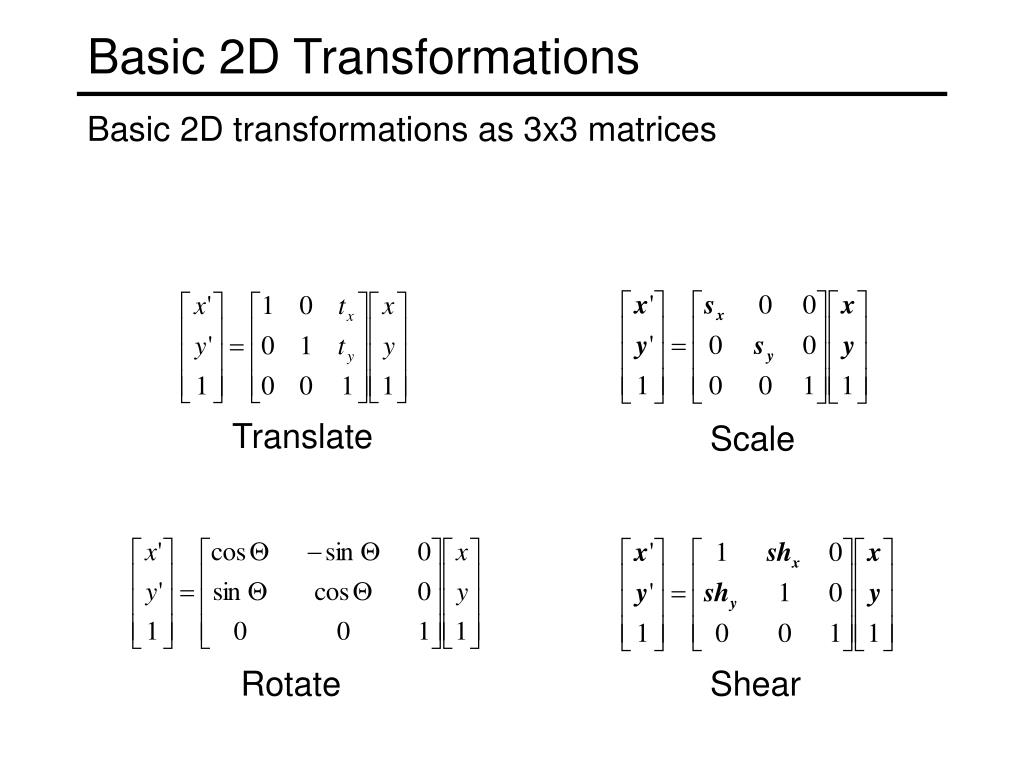


We can write it as − P’ = P + T

**Scaling matrix:**

We can use a 2 × 2 matrix to change or transform, a 2D vector. This kind of operation, which takes in a 2-vector and produces another 2-vector by a simple matrix multiplication, is a linear transformation.

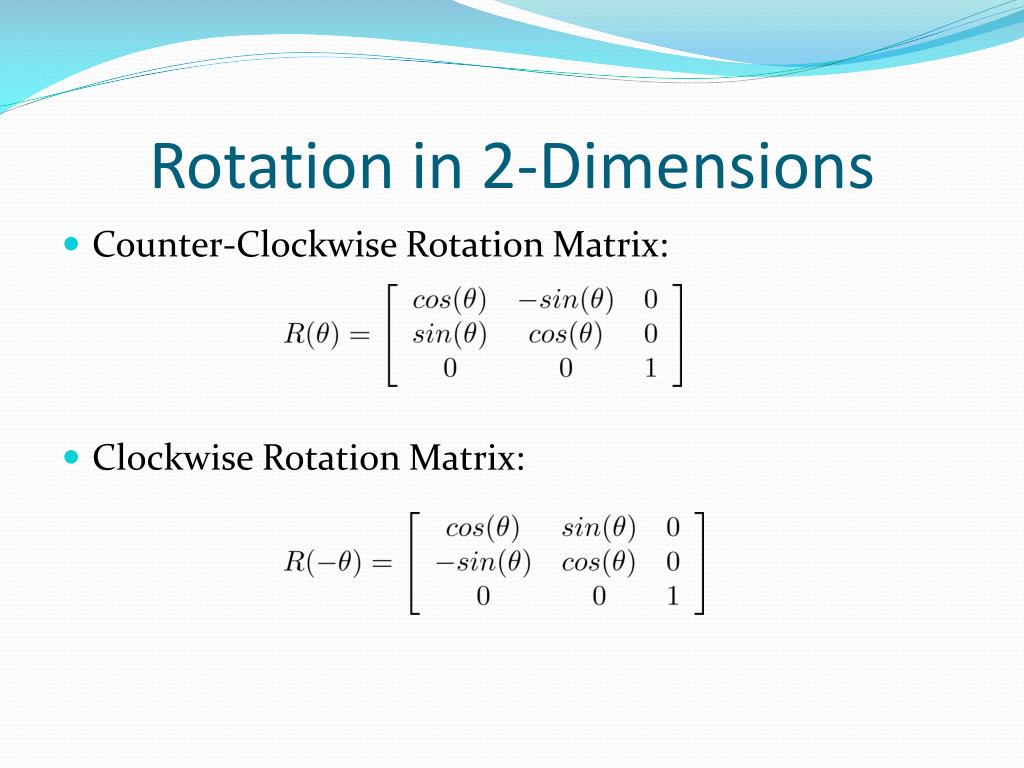
A scaling transformation alters size of an object. In the scaling process, we either compress or expand the dimension of the object. Scaling operation can be achieved by multiplying each vertex coordinate (x, y) of the polygon by scaling factor sx and sy to produce the transformed coordinates as (x’, y’). So, x’ = x \* sx and y’ = y \* sy. The scaling factor sx, sy scales the object in X and Y direction respectively. So, the above equation can be represented in matrix form:





**Rotation matrix:**

In 2D space, [rotation](https://www.cuemath.com/rotation-formula/) can occur about the x, y, or z-axis. Such a type of rotation that occurs about any one of the axis is known as a basic or elementary rotation. Given below are the rotation matrices that can rotate a vector through an angle about any particular axis.



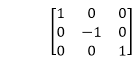
**Reflection matrix:**

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°.

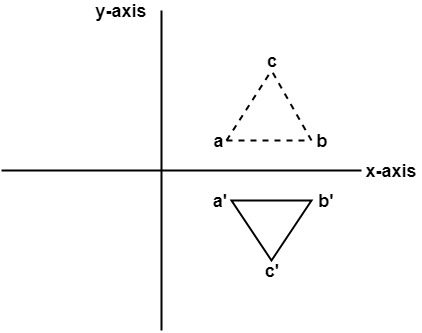
Types of reflection:

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



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.

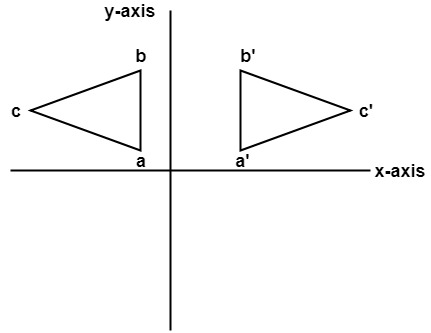


**2. Reflection about y-axis:** The object can be reflected about y-axis with the help of following transformation matrix

Reflection

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

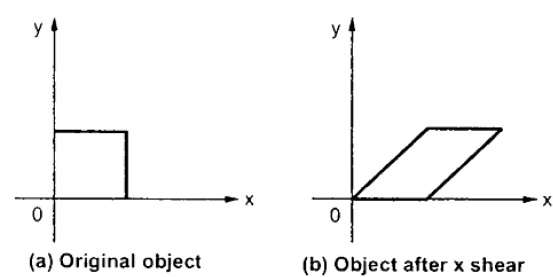


**Shear:**

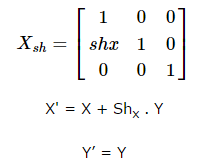
A transformation that slants the shape of an object is called the shear transformation. There are two shear transformations **X-Shear** and **Y-Shear**. One shifts X coordinates values and other shifts Y coordinate values. However; in both the cases only one coordinate changes its coordinates and other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

### X-Shear

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

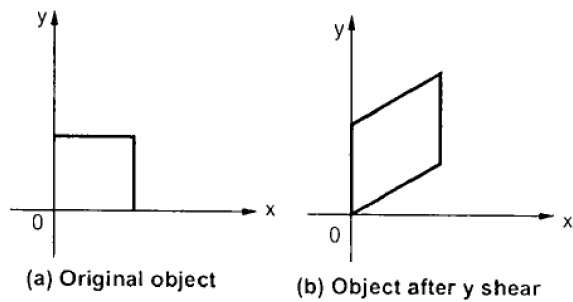


The transformation matrix for X-Shear can be represented as –

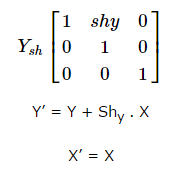


### Y-Shear

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



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



3)Explain the raster operations to perform simple transformation.

There are many different ways to perform raster transformations, ranging from simple translation and rotation to more complex operations such as scaling, shearing, and reflection. In addition, there are numerous software packages available that can be used to perform raster transformations.

1. **Scaling:**There are three main types of scaling: point scaling, line scaling, and area scaling. Point scaling is when the scale factor is applied to individual points. Line scaling is when the scale factor is applied to lines. Area scaling is when the scale factor is applied to areas.
2. **Shearing:**Shearing is a raster image transformation that changes the orientation of an image. Shearing can be used to create an illusion of depth or to make an image appear larger or smaller. Shearing can also be used to correct geometric distortions in an image.
3. **Reflection:**There are several ways to mathematically transform raster images. The most common (and computationally simplest) are translation, rotation, and scaling. The translation is the movement of an image along the x- and y-axis. Rotation is the transformation of an image around a certain point, usually the origin (0,0). Scaling is the resizing of an image, which can be done isotropically (maintaining the same aspect ratio) or anisotropically (changing the aspect ratio).

All of these transformations can be done using interpolation, which is when new pixel values are estimated based on known values. The quality of the transformation depends on the interpolation method used. The most common interpolation method is a bilinear interpolation, which gives each new pixel a value that is a weighted average of the known surrounding pixels.

The raster method of transformations is fast and easy to implement, but it can lead to some artifacts in the transformed image. These artifacts can be minimized by using higher-quality interpolation methods, such as bicubic interpolation.

**Advantages:**

There are many advantages to using the raster method of transformations as opposed to other methods. One advantage is that it is more accurate than other methods. This is because the raster method uses a grid system which makes it easy to determine the exact coordinates of each point on the image. This means that there is less chance of error when transforming an image.

Another advantage of the raster method is that it is less likely to cause distortion in an image. This is because the transformation is applied evenly across the entire image, rather than being applied selectively to certain areas. This makes it ideal for images that need to be transformed without changing their overall shape or appearance.

Finally, the raster method is also much faster than other methods. This is because all of the calculations are done by computer, rather than having to be done manually. This means that complex transformations can be carried out quickly and easily, without having to spend hours doing them by hand.

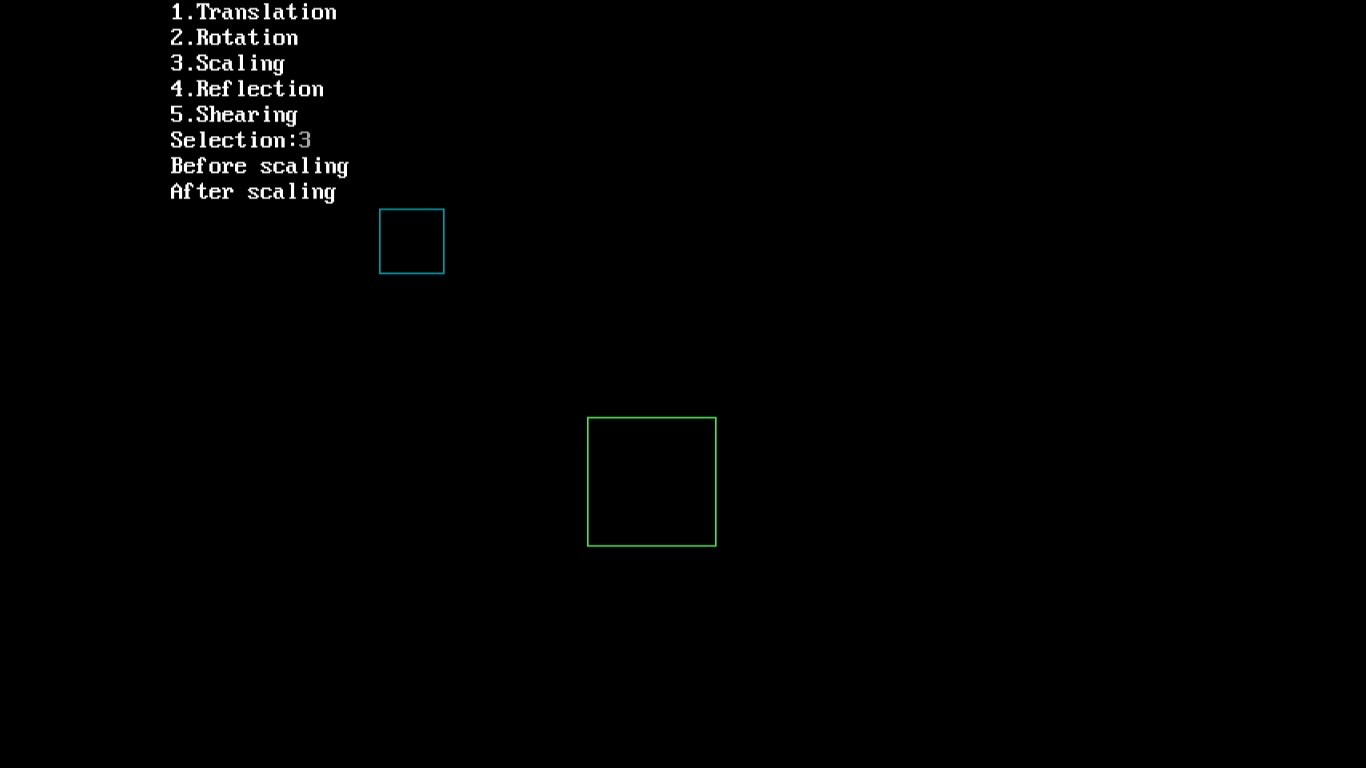
**Disadvantages:**

There are a few disadvantages to using the raster method of transformations. One is that it can be computationally expensive, especially if the raster data is large. Another is that it can be difficult to control the amount of distortion that occurs during the transformation process. Additionally, this method does not preserve all features of the original data, such as line work and text.

**Source Code:**

#include<stdio.h>  
#include<graphics.h>  
#include<stdlib.h>  
#include<math.h>  
#include<conio.h>  
int x1,y1,x2,y2;  
void translation()  
{  
int tx,ty,xn1,xn2,yn1,yn2;  
printf("\n Enter the translation\n");  
scanf("%d%d",&tx,&ty);  
cleardevice();  
outtextxy(400,100,"TRANSLATION");  
xn1=x1+tx;  
yn1=y1+ty;  
xn2=x2+tx;  
yn2=y2+ty;  
line(x1,y1,x2,y2);  
line(xn1,yn1,xn2,yn2);  
getch();  
}  
  
void scaling()  
{  
int xn1,xn2,yn1,yn2;  
float sx,sy;  
printf("Enter the scaling factor");  
scanf("%f%f",&sx,&sy);  
cleardevice();  
outtextxy(300,200,"SCALING");  
xn1=x1\*sx;  
yn1=y1\*sy;  
xn2=x2\*sx;  
yn2=y2\*sy;  
line(x1,y1,x2,y2);  
line(xn1,yn1,xn2,yn2);  
getch();  
}  
  
void rotation()  
{  
int r;  
float rx,xn1,xn2,yn1,yn2;  
printf("\n enter the angle for rotation");  
scanf("%d",&r);  
cleardevice();  
outtextxy(500,200,"ROTATION");  
rx=(r\*3.14)/180;  
xn1=x1\*cos(rx)-y1\*sin(rx);  
yn1=y1\*cos(rx)+x1\*sin(rx);  
xn2=x2\*cos(rx)-y2\*sin(rx);  
yn2=y2\*cos(rx)+x2\*sin(rx);  
line(x1,y1,x2,y2);  
line(xn1,yn1,xn2,yn2);  
getch();  
}  
  
void shearing()  
{  
float sh;  
float xn1,xn2,yn1,yn2;  
printf("\n Enter the value for shearing");  
scanf("%f",&sh);  
cleardevice();  
outtextxy(500,100,"SHEARING");  
xn1=x1+sh\*y1;  
yn1=y1;  
xn2=x2+sh\*y2;  
yn2=y2;  
line(x1,y1,x2,y2);  
line(xn1,yn1,xn2,yn2);  
getch();  
}  
  
void reflection()  
{  
int xn1,xn2,yn1,yn2;  
cleardevice();  
outtextxy(300,100,"REFLECTION");  
if((x1<y1)^(x2<y2))  
{  
xn1=x1+50;  
xn2=x2+50;  
yn1=y1;  
yn2=y2;  
}  
else  
{  
xn1=x1;  
xn2=x2;  
yn1=y1+50;  
yn2=y2+50;  
}  
line(x1,y1,x2,y2);  
line(xn1,yn1,xn2,yn2);  
getch();  
}  
  
void get()  
{  
printf("\n Enter the coordinates x1,y1,x2,y2");  
scanf("%d%d%d%d",&x1,&y1,&x2,&y2);  
outtextxy(200,100,"ORIGINAL OBJECT");  
line(x1,y1,x2,y2);  
getch();  
}  
  
void main()  
{  
int ch,gd=DETECT,gm;  
initgraph(&gd,&gm,"c:\\tc\\bgi");  
get();  
do  
{  
cleardevice();  
outtextxy(10,10,"1)TRANSLATION");  
outtextxy(10,20,"2)SCALING");  
outtextxy(10,30,"3)ROTATION");  
outtextxy(10,40,"4)SHEARING");  
outtextxy(10,50,"5)REFLECTION");  
outtextxy(10,60,"6)EXIT");  
outtextxy(10,70,"ENTER UR CHOICE:");  
scanf("%d",&ch);  
switch(ch)  
{  
case 1:  
translation();  
break;  
case 2:  
scaling();  
break;  
case 3:  
rotation();  
break;  
case 4:  
shearing();  
break;  
case 5:  
reflection();  
break;  
case 6:  
exit(0);  
}  
}while(ch<6);  
}

**Output:**

****

**Conclusion**:

I have understood how to implement 2D transformations in CG