**Practical No 9**

**Implementation of 3D Transformations (only coordinates calculation).**

**Aim: Write a program to implement a 3D Transformation.**

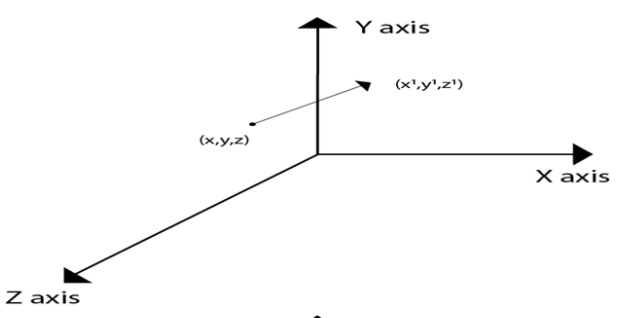
**Theory:**

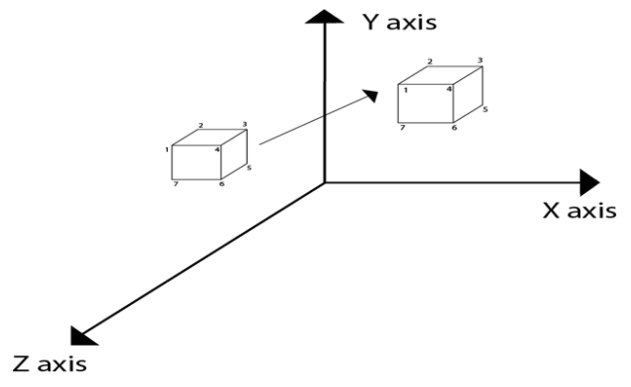
**Translation**

Three-dimensional transformation matrix for translation with homogeneous coordinates is as given below. It specifies three coordinates with their own translation factor. A translation in space is described by tx, ty and tz. It is easy to see that this matrix realizes the equations: x2=x1+tx

y2=y1+ty

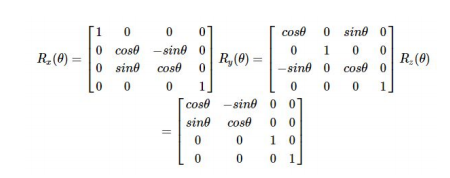
Like two dimensional transformations, an object is translated in three dimensions by transforming each vertex of the object.



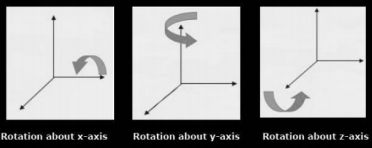
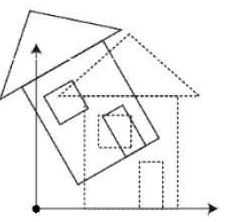


**Rotation**

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

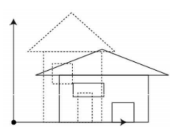


The following figure explains the rotation about various axes

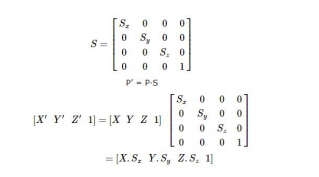
 

**Scaling**

You can change the size of an object using scaling transformation. In the scaling process, you either expand or compress the dimensions of the object. Scaling can be achieved by multiplying the original coordinates of the object with the scaling factor to get the desired result. The following figure shows the effect of 3D scaling

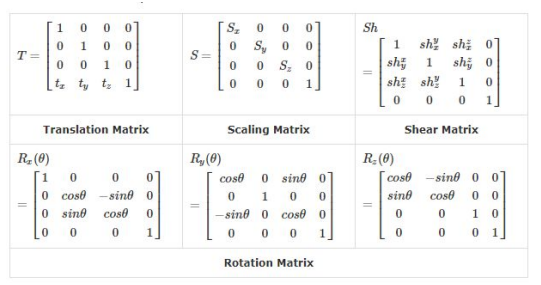


In 3D scaling operation, three coordinates are used. Let us assume that the original coordinates are (X, Y, Z), scaling factors are (SX,SY,Sz)(SX,SY,Sz) respectively, and the produced coordinates are (X’, Y’, Z’). This can be mathematically represented as shown below



**Transformation Matrices**

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



**Conclusion: We have implemented 3D Transformation.**

**Code:**

#include<iostream.h>

#include<stdlib.h>

#include<math.h>

#include<graphics.h>

#include<conio.h>

typedef struct

{

float x;

float y;

float z;

}

Point;

Point points;

float temp = 0;

void showPoint()

{

cout<<"("<<points.x<<","<<points.y<<","<<points.z<<")"<<endl;

}

void translate(float tx, float ty, float tz)

{

points.x += tx;

points.y += ty;

points.z += tz;

cout<<"After Translation, new point is :";

showPoint();

}

void rotatex(float angle)

{

angle = angle \* M\_PI / 180.0;

temp = points.y;

points.y = points.y \* cos(angle) - points.z \* sin(angle);

points.z = temp \* sin(angle) + points.z \* cos(angle);

cout<<"After rotation about x, new point is: ";

showPoint();

}

void rotatey(float angle)

{

angle = (angle \* M\_PI) / 180.0;

temp = points.z;

points.z = points.z \* cos(angle) - points.x \* sin(angle);

points.x = temp \* sin(angle) + points.x \* cos(angle);

cout<<"After rotation about y, new point is: ";

showPoint();

}

void rotatez(float angle)

{

angle = angle \* M\_PI / 180.0;

temp = points.x;

points.x = points.x \* cos(angle) - points.y \* sin(angle);

points.y = temp \* sin(angle) + points.y \*cos(angle);

cout<<"After rotation about z, new point is: ";

showPoint();

}

void scale(float sf, float xf, float yf, float zf)

{

points.x = points.x \* sf + (1 - sf) \* xf;

points.y = points.y \* sf + (1 - sf) \* yf;

points.z = points.z \* sf + (1 - sf) \* zf;

cout<<"After scaling, new point is: ";

showPoint();

}

void main()

{

clrscr();

float tx = 0, ty = 0, tz = 0;

float sf = 0, xf = 0, yf = 0, zf = 0;

int choose;

float angle;

cout<<"Enter the initial point you want to transform:";

cin>>points.x>>points.y>>points.z;

int ch=1;

cout<<"1. Translate"<<endl;

cout<<"2. Rotate about X axis"<<endl;

cout<<"3. Rotate about Y axis"<<endl;

cout<<"4. Rotate about Z axis"<<endl;

cout<<"5. Scale"<<endl;

cout<<"0. Exit"<<endl;

while(ch>0)

{

cout<<"Enter the choice:";

cin>>choose;

switch(choose)

{

case 1:

cout<<"Enter the value of tx, ty and tz: ";

cin>>tx>>ty>>tz;

translate(tx, ty, tz);

break;

case 2:

cout<<"Enter the angle: ";

cin>>angle;

rotatex(angle);

break;

case 3:

cout<<"Enter the angle: ";

cin>>angle;

rotatey(angle);

break;

case 4:

cout<<"Enter the angle: ";

cin>>angle;

rotatez(angle);

break;

case 5:

cout<<"Enter the value of sf, xf, yf and zf: ";

cin>>sf>>xf>>yf>>zf;

scale(sf, xf, yf, zf);

break;

case 0: exit(0);

ch=0;

break;

default:

break;

}

}

getch();

}

**Output:**

