SSN COLLEGE OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UCS1712 – GRAPHICS AND MULTIMEDIA LAB
EX NO:7-3D-Transforamtions

Name : Mohamed Hashim G
RegNo : 185001094

AIM

To Write a c++ program using OPENGL to perform 3D transformations – translation, scaling and rotation

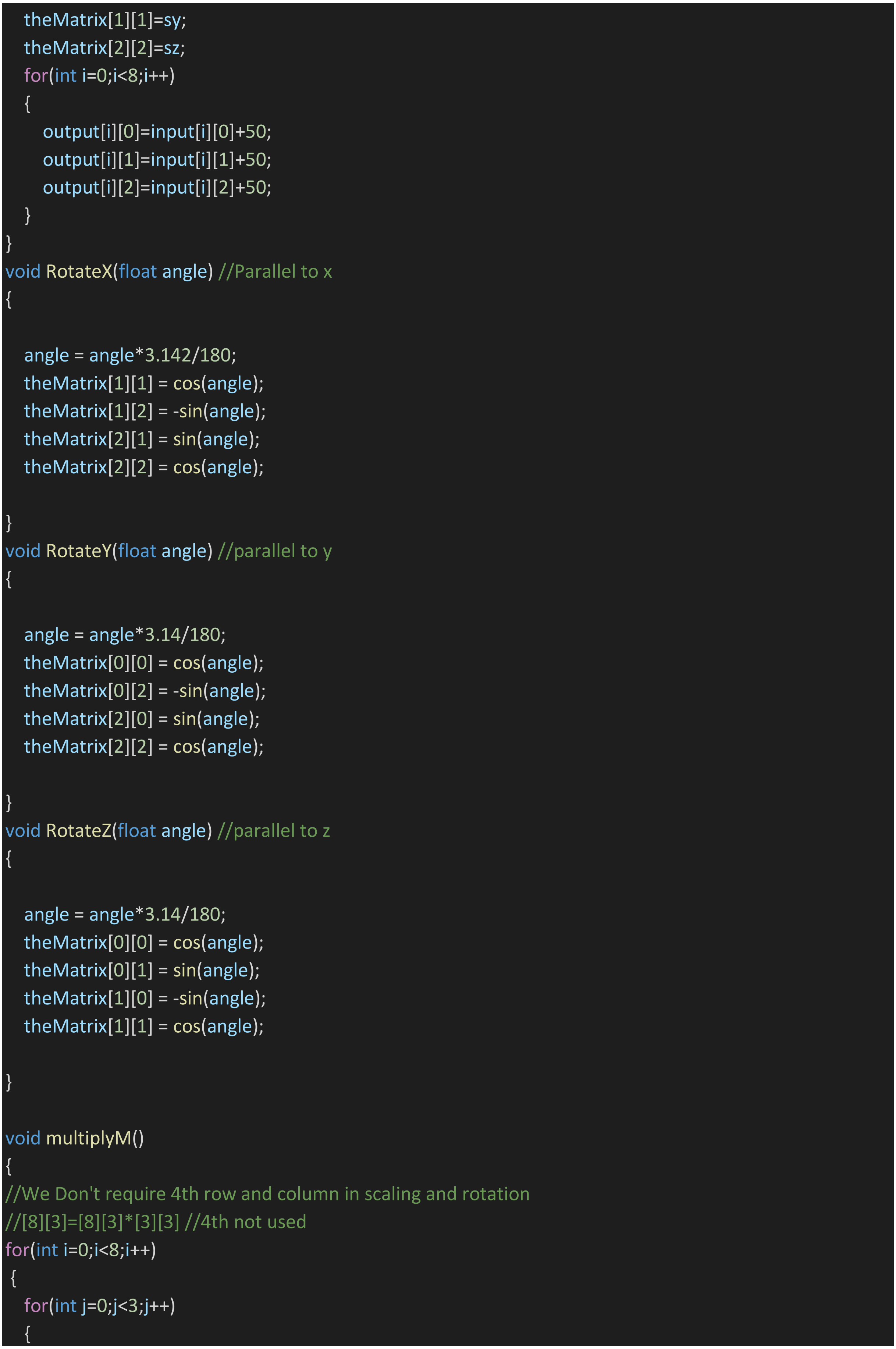
ALGORITHM:

```
1. Read the choice of operation to be performed.
2. For Translation:
    a. Read translation factor (tx,ty,tz).
    b. For each vertex (x,y,z) apply translation as follows:
        i. $x = x+tx$
        ii. $y = y+ty$
        iii. $z = z+tz$
3. For Rotation:
    a. Read the degree of rotation Ɵ for polygon and line.
    b. Read the axis of rotation.
    c. If axis of rotation is z-axis:
        i. For each vertex (x,y,z) the new vertex is computed as follows:
            1. $x = x\cos\theta - y\sin\theta$
            2. $y = x\sin\theta + y\cos\theta$
            3. $z = z$
    d. If axis of rotation is x-axis:
        i. For each vertex (x,y,z) the new vertex is computed as follows:
            1. $y = y\cos\theta - z\sin\theta$
            2. $z = y\sin\theta + z\cos\theta$
            3. $x = x$
    e. If axis of rotation is y-axis:
        i. For each vertex (x,y,z) the new vertex is computed as follows:
            1. $z = z\cos\theta - x\sin\theta$
            2. $x = z\sin\theta + x\cos\theta$
            3. $y = y$
4. For Scaling:
    a. Read scaling factor (sx,sy,sz).
    b. Set the fixed point (fx,fy,fz) as the first vertex.
    c. For each vertex (x,y,z) apply scaling as follows:
        i. $x = (x*sx)$
```

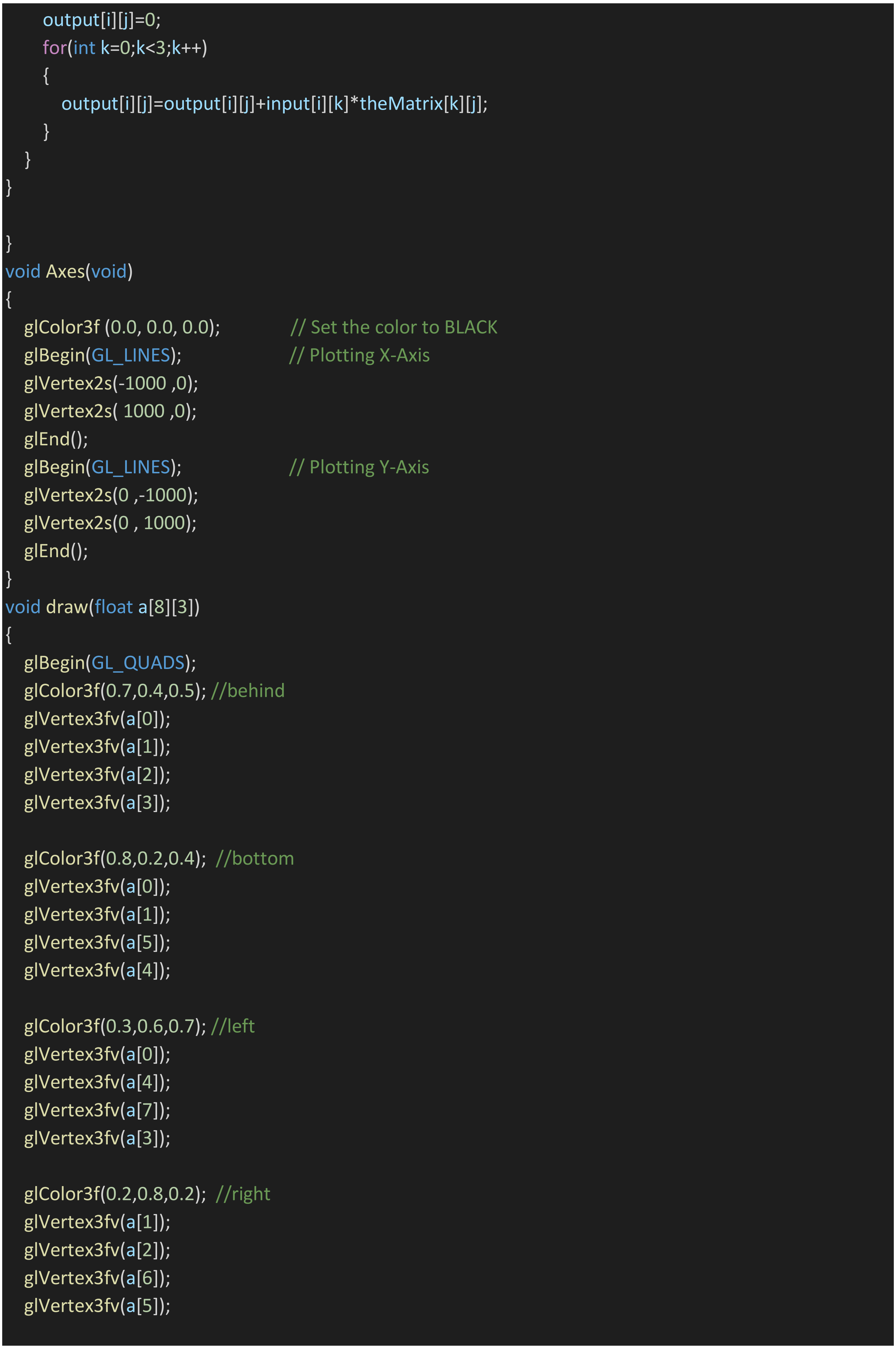
ii. y = (y*sy)
iii. z = (z*sz)

CODE:

```
#include<iostream>
#include<math.h>
#include<GL/glut.h>
using namespace std;

typedef float Matrix4 [4][4];

Matrix4 theMatrix;
static GLfloat input[8][3]=
{
    {40,40,-50},{90,40,-50},{90,90,-50},{40,90,-50},
    {30,30,0},{80,30,0},{80,80,0},{30,80,0}

};

float output[8][3];
float tx,ty,tz;
float sx,sy,sz;
float angle;

int choice,choiceRot;

void setIdentityM(Matrix4 m)
{
for(int i=0;i<4;i++)
    for(int j=0;j<4;j++)
        m[i][j]=(i==j);
}

void translate(int tx,int ty,int tz)
{

for(int i=0;i<8;i++)
{
output[i][0]=input[i][0]+tx;
output[i][1]=input[i][1]+ty;
output[i][2]=input[i][2]+tz;
}
}
void scale(int sx,int sy,int sz)
{
    theMatrix[0][0]=sx;
```



```
   theMatrix[1][1]=sy;
   theMatrix[2][2]=sz;
   for(int i=0;i<8;i++)
   {
      output[i][0]=input[i][0]+50;
      output[i][1]=input[i][1]+50;
      output[i][2]=input[i][2]+50;
   }
}
void RotateX(float angle) //Parallel to x
{

   angle = angle*3.142/180;
   theMatrix[1][1] = cos(angle);
   theMatrix[1][2] = -sin(angle);
   theMatrix[2][1] = sin(angle);
   theMatrix[2][2] = cos(angle);

}
void RotateY(float angle) //parallel to y
{

   angle = angle*3.14/180;
   theMatrix[0][0] = cos(angle);
   theMatrix[0][2] = -sin(angle);
   theMatrix[2][0] = sin(angle);
   theMatrix[2][2] = cos(angle);

}
void RotateZ(float angle) //parallel to z
{

   angle = angle*3.14/180;
   theMatrix[0][0] = cos(angle);
   theMatrix[0][1] = sin(angle);
   theMatrix[1][0] = -sin(angle);
   theMatrix[1][1] = cos(angle);

}

void multiplyM()
{
//We Don't require 4th row and column in scaling and rotation
//[8][3]=[8][3]*[3][3] //4th not used
for(int i=0;i<8;i++)
 {
   for(int j=0;j<3;j++)
   {
```



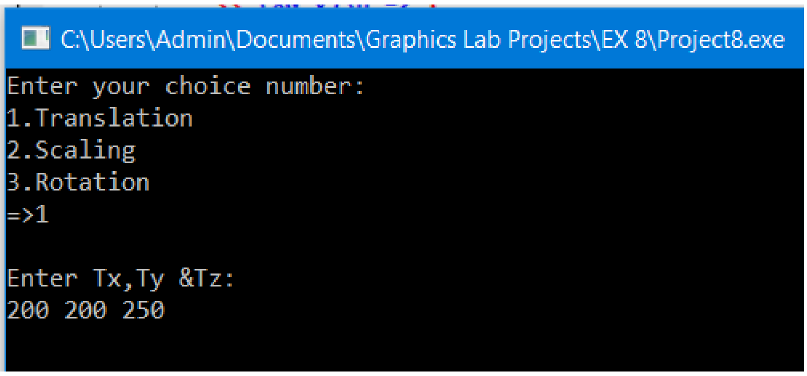
```
    output[i][j]=0;
    for(int k=0;k<3;k++)
    {
      output[i][j]=output[i][j]+input[i][k]*theMatrix[k][j];
    }
  }
}

}
void Axes(void)
{
  glColor3f (0.0, 0.0, 0.0);            // Set the color to BLACK
  glBegin(GL_LINES);                    // Plotting X-Axis
  glVertex2s(-1000 ,0);
  glVertex2s( 1000 ,0);
  glEnd();
  glBegin(GL_LINES);                    // Plotting Y-Axis
  glVertex2s(0 ,-1000);
  glVertex2s(0 , 1000);
  glEnd();
}
void draw(float a[8][3])
{
  glBegin(GL_QUADS);
  glColor3f(0.7,0.4,0.5); //behind
  glVertex3fv(a[0]);
  glVertex3fv(a[1]);
  glVertex3fv(a[2]);
  glVertex3fv(a[3]);

  glColor3f(0.8,0.2,0.4);  //bottom
  glVertex3fv(a[0]);
  glVertex3fv(a[1]);
  glVertex3fv(a[5]);
  glVertex3fv(a[4]);

  glColor3f(0.3,0.6,0.7); //left
  glVertex3fv(a[0]);
  glVertex3fv(a[4]);
  glVertex3fv(a[7]);
  glVertex3fv(a[3]);

  glColor3f(0.2,0.8,0.2);  //right
  glVertex3fv(a[1]);
  glVertex3fv(a[2]);
  glVertex3fv(a[6]);
  glVertex3fv(a[5]);
```

```
    glColor3f(0.7,0.7,0.2); //up
    glVertex3fv(a[2]);
    glVertex3fv(a[3]);
    glVertex3fv(a[7]);
    glVertex3fv(a[6]);

    glColor3f(1.0,0.1,0.1);
    glVertex3fv(a[4]);
    glVertex3fv(a[5]);
    glVertex3fv(a[6]);
    glVertex3fv(a[7]);

    glEnd();
}

void init()
{
    glClearColor(1.0,1.0,1.0,1.0); //set backgrond color to white
    glOrtho(-454.0,454.0,-250.0,250.0,-250.0,250.0);
    // Set the no. of Co-ordinates along X & Y axes and their gappings
    glEnable(GL_DEPTH_TEST);
     // To Render the surfaces Properly according to their depths
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    Axes();
    glColor3f(1.0,0.0,0.0);
    draw(input);
    setIdentityM(theMatrix);
    switch(choice)
    {
    case 1:
        translate(tx,ty,tz);
        break;
    case 2:
        scale(sx,sy,sz);
        multiplyM();
        break;
    case 3:
        switch (choiceRot)
        {
            case 1:
                RotateX(angle);
                break;
            case 2: RotateY(angle);
                break;
            case 3:
                RotateZ(angle);
                break;
            default:
                break;
        }
        multiplyM();
        break;
    }

    draw(output);
    glFlush();
}

int main(int argc, char** argv)
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB|GLUT_DEPTH);
    glutInitWindowSize(1362,750);
    glutInitWindowPosition(0,0);
    glutCreateWindow("3D TRANSFORMATIONS");
    init();
    cout<<"Enter your choice number:\n1.Translation\n2.Scaling\n3.Rotation\n=>";
    cin>>choice;
    switch (choice)
    {
        case 1:
            cout<<"\nEnter Tx,Ty &Tz: \n";
            cin>>tx>>ty>>tz;
            break;
        case 2:
            cout<<"\nEnter Sx,Sy & Sz: \n";
            cin>>sx>>sy>>sz;
            break;
        case 3:
            cout<<"Enter your choice for Rotation about axis:\n1.parallel to X-axis."
                <<"(y& z)\n2.parallel to Y-axis.(x& z)\n3.parallel to Z-axis."
                <<"(x& y)\n =>";
            cin>>choiceRot;
            switch (choiceRot) {
            case 1:
                cout<<"\nENter Rotation angle: ";
                cin>>angle;
                break;
            case 2:
                cout<<"\nENter Rotation angle: ";
                cin>>angle;
                break;
```

```
      case 3:
        cout<<"\nENter Rotation angle: ";
        cin>>angle;
        break;
      default:
        break;
      }
      break;
    default:
      break;
  }
  glutDisplayFunc(display);
  glutMainLoop();
  return 0;
}
```

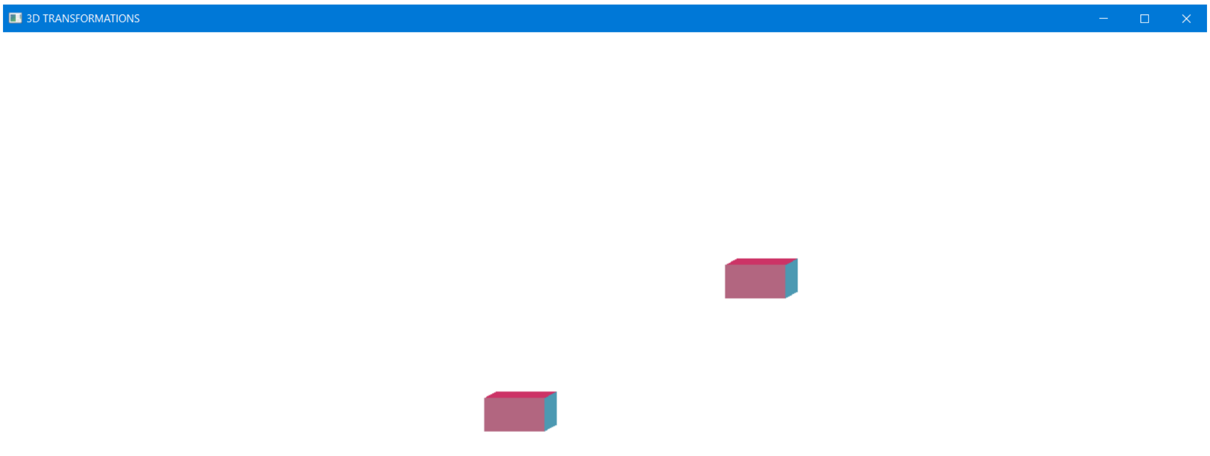
OUTPUT:

TRANSLATION:

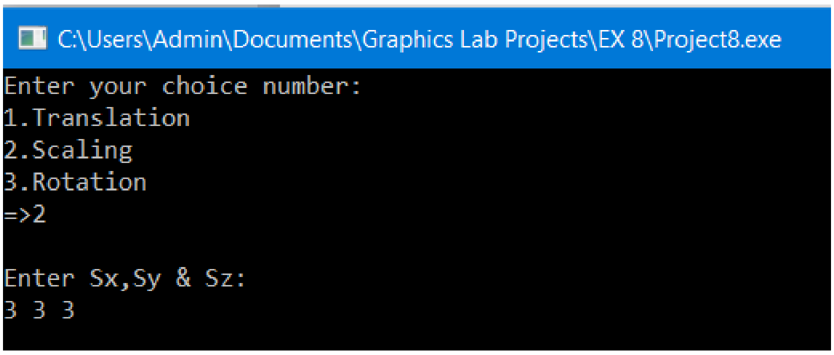


C:\Users\Admin\Documents\Graphics Lab Projects\EX 8\Project8.exe

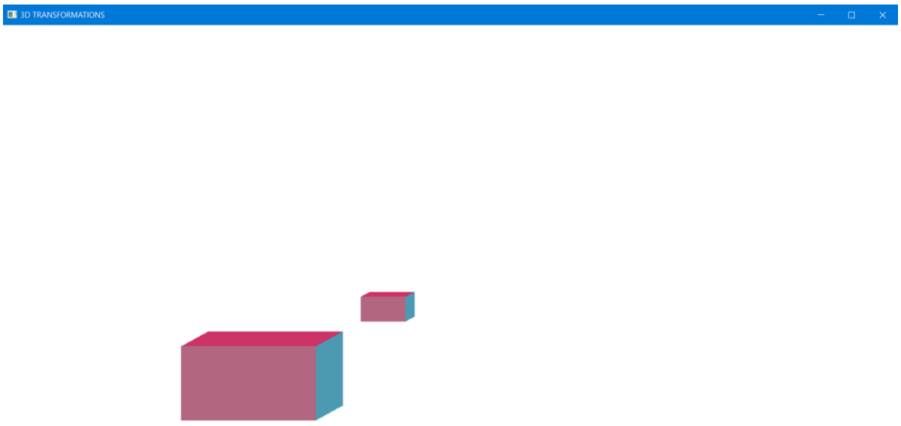
```
Enter your choice number:
1.Translation
2.Scaling
3.Rotation
=>1

Enter Tx,Ty &Tz:
200 200 250
```



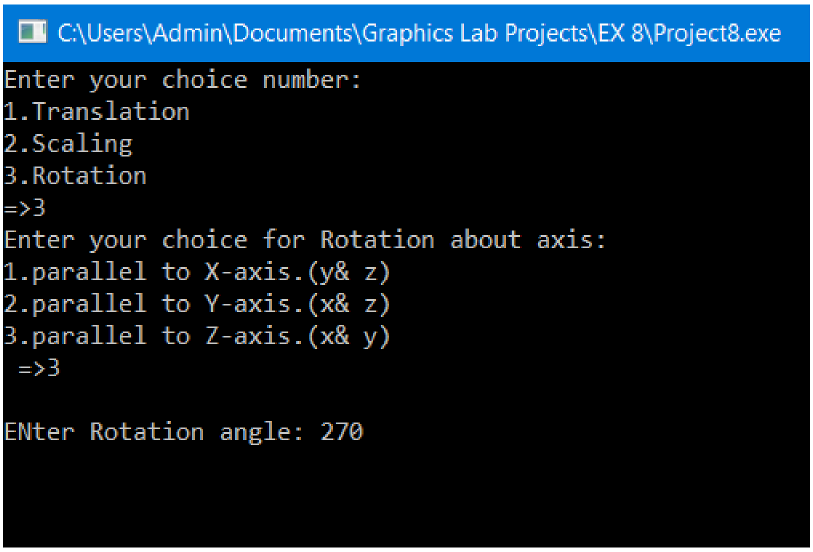
SCALING:



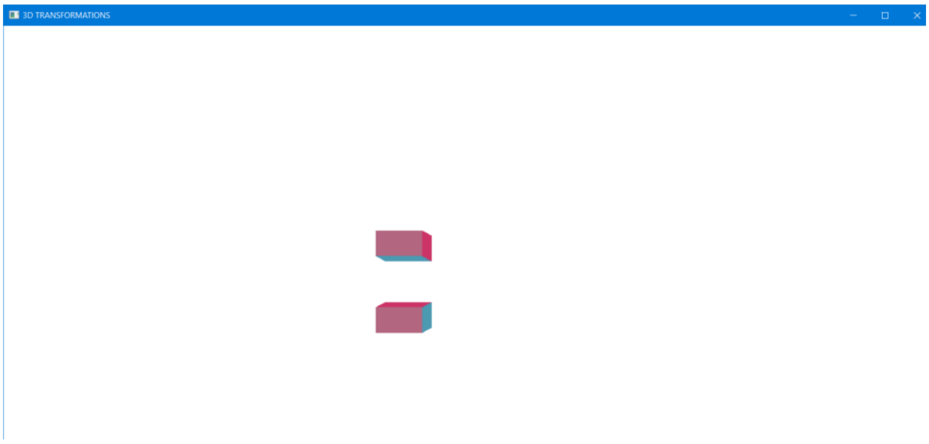
```
C:\Users\Admin\Documents\Graphics Lab Projects\EX 8\Project8.exe
Enter your choice number:
1.Translation
2.Scaling
3.Rotation
=>2

Enter Sx,Sy & Sz:
3 3 3
```



ROTATION:



```
C:\Users\Admin\Documents\Graphics Lab Projects\EX 8\Project8.exe
Enter your choice number:
1.Translation
2.Scaling
3.Rotation
=>3
Enter your choice for Rotation about axis:
1.parallel to X-axis.(y& z)
2.parallel to Y-axis.(x& z)
3.parallel to Z-axis.(x& y)
 =>3

ENter Rotation angle: 270
```



RESULT:

OPENGL programs to perform 3D transformations was designed and implemented successfully.