UCS 1712 – GRAPHICS AND MULTIMEDIA LAB

ASSIGNMENT – 8

VISHAL N

185001198

CSEC

20.10.2021

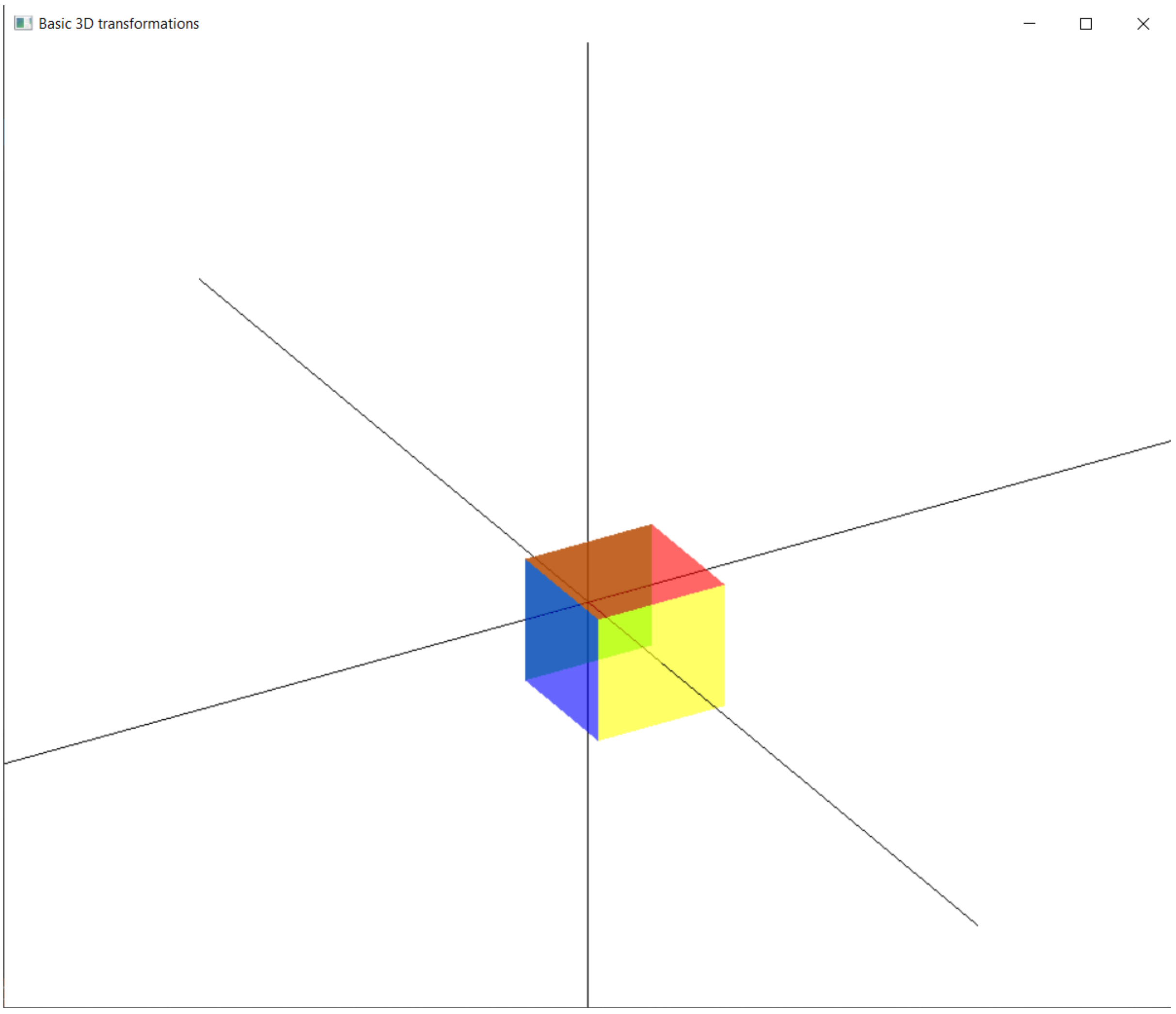
1. BASIC 3D TRANSFORMATIONS:

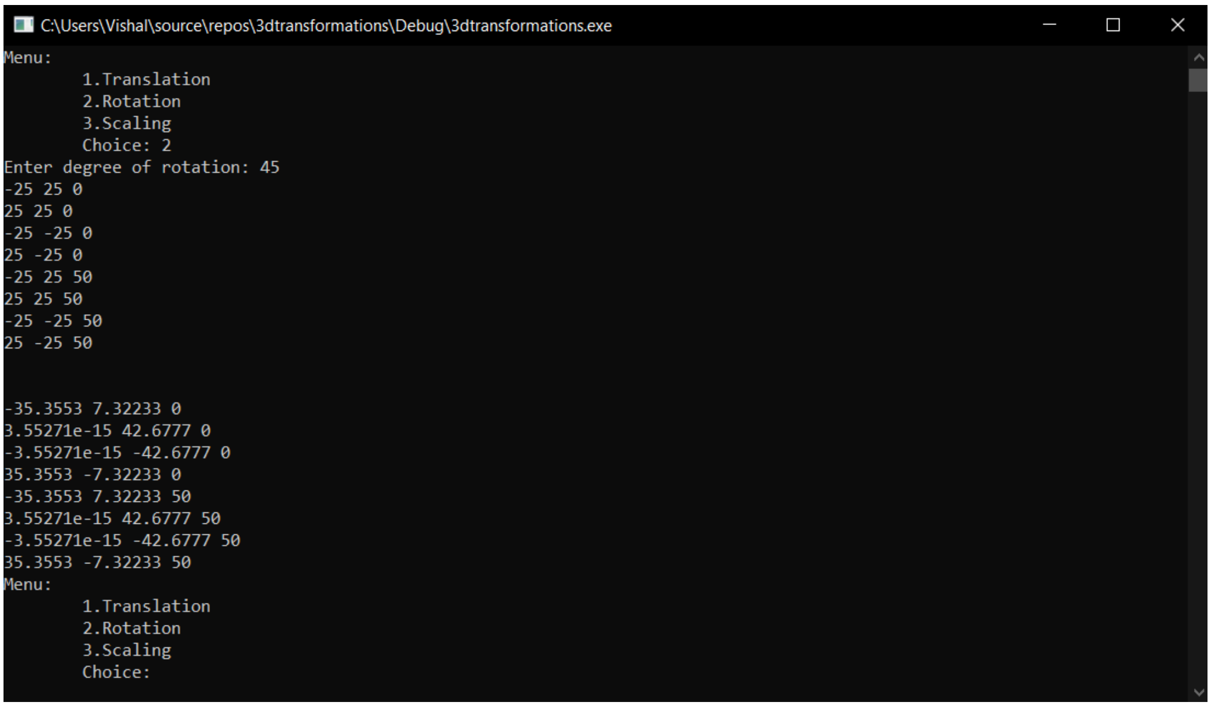
3dtransformations.cpp:

```
#include <windows.h>
#include <GL/glut.h>
#include <math.h>
#include <stdlib.h>
#include <iostream>
#include <vector>
#include <math.h>

#define M_PI 3.14159265358979323846
#define PI M_PI

using namespace std;

typedef struct Point {
    double x, y, z, h;
}Point;

typedef struct Face {
    Point v[4];
}Face;

typedef struct Cuboid {
    Point v[8];
    Face faces[6];
}Cuboid;

int assignList[6][4] = { {0, 1, 3, 2}, {0, 4, 5, 1}, {0, 4, 6, 2},
                         {4, 5, 7, 6}, {2, 6, 7, 3}, {1, 5, 7, 3}
};

float colors[6][3] = { {0, 1, 0}, {1, 0, 0}, {0, 0, 1},
                       {1, 1, 0}, {0, 1, 1}, {1, 0, 1}
};

double tMat[4][4];
bool tflag = false;
```

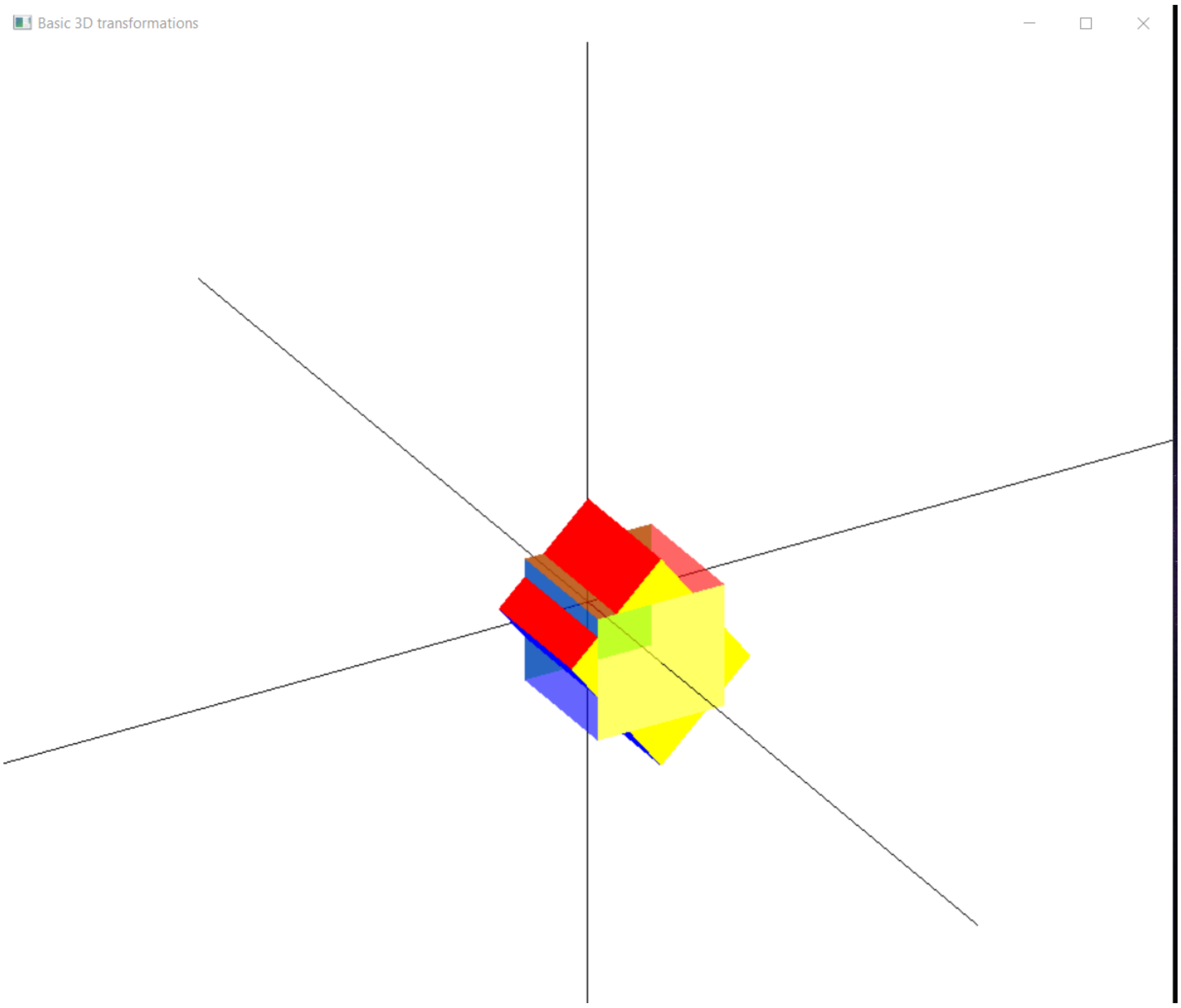
```
Cuboid cuboid, tcuboid;
Cuboid initCuboid() {
    Cuboid cuboid;
    double po[8][3] = { {-25, 25, 0}, {25, 25, 0}, {-25, -25, 0}, {25, -25,
0},
                        {-25, 25, 50}, {25, 25, 50}, {-25, -25, 50}, {25, -25, 50}
    };

    for (int i = 0; i < 8; i++) {
        cuboid.v[i].x = po[i][0];
        cuboid.v[i].y = po[i][1];
        cuboid.v[i].z = po[i][2];
        cuboid.v[i].h = 1;
    }
    for (int i = 0; i < 6; i++) {
        for (int j = 0; j < 4; j++) {
            cuboid.faces[i].v[j] = cuboid.v[assignList[i][j]];
        }
    }
    return cuboid;
}
void myInit() {
    glClearColor(1.0, 1.0, 1.0, 0.0);
    glColor3f(0, 0, 0);
    glEnable(GL_BLEND);
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
    glLoadIdentity();
    glOrtho(-200, 200, -200, 200, -200, 200);
    glEnable(GL_DEPTH_TEST);
}
void disp() {
    glRotatef(30, 1, 0, 0);
    glRotatef(30, 0, 1, 0);
}

void transformCuboid() {
    tflag = true;
    for (int i = 0; i < 8; i++) {
        cout << cuboid.v[i].x << " " << cuboid.v[i].y << " " << cuboid.v[i].z
<< "\n";
    }
    cout << "\n\n";
    for (int p = 0; p < 8; p++) {

        double pnt[4][1], pnt1[4][1];
        pnt[0][0] = cuboid.v[p].x;
        pnt[1][0] = cuboid.v[p].y;
        pnt[2][0] = cuboid.v[p].z;
        pnt[3][0] = cuboid.v[p].h;
        memset(pnt1, 0, sizeof(pnt1));
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 1; j++) {
                for (int k = 0; k < 4; k++) {
                    pnt1[i][j] += tMat[i][k] * pnt[k][j];
                }
            }
        }
        tcuboid.v[p].x = pnt1[0][0];
        tcuboid.v[p].y = pnt1[1][0];
        tcuboid.v[p].z = pnt1[2][0];
        tcuboid.v[p].h = pnt1[3][0];
    }
    for (int i = 0; i < 6; i++) {
        for (int j = 0; j < 4; j++) {
            tcuboid.faces[i].v[j] = tcuboid.v[assignList[i][j]];
        }
    }
    for (int i = 0; i < 8; i++) {
        cout << tcuboid.v[i].x << " " << tcuboid.v[i].y << " " <<
tcuboid.v[i].z << "\n";
    }
    glutPostRedisplay();
}

void getTransformMatrix() {
    memset(tMat, 0, sizeof(tMat));
    tMat[0][0] = tMat[1][1] = tMat[2][2] = tMat[3][3] = 1;
    int ch;
    cout << "Menu:\n\t1.Translation\n\t2.Rotation\n\t3.Scaling\n\tChoice: ";
    cin >> ch;
    switch (ch) {
    case 1:
        cout << "Enter translation parameters: ";
        cin >> tMat[0][3] >> tMat[1][3] >> tMat[2][3];
        break;
    case 2:
        cout << "Enter degree of rotation: ";
        double deg;
        cin >> deg;
        deg = deg * PI / 180;
        tMat[0][0] = cos(deg);
        tMat[0][1] = -sin(deg);
        tMat[1][0] = tMat[0][0];
        tMat[1][0] = -tMat[0][1];
        break;
    case 3:
        cout << "Enter scaling parameters: ";
        cin >> tMat[0][0] >> tMat[1][1] >> tMat[2][2];
        break;
    }
    transformCuboid();
}

void displayCuboid(Cuboid cuboid, double alpha = 0.6) {
    for (int i = 0; i < 6; i++) {
        glColor4f(colors[i][0], colors[i][1], colors[i][2], alpha);
        glBegin(GL_POLYGON);
        for (int j = 0; j < 4; j++) {
            glVertex3d(cuboid.faces[i].v[j].x, cuboid.faces[i].v[j].y,
cuboid.faces[i].v[j].z);
        }
        glEnd();
    }
}

void drawAxis() {
    glBegin(GL_LINES);
    glVertex3d(300, 0, 0);
    glVertex3d(-300, 0, 0);
    glEnd();
    glBegin(GL_LINES);
    glVertex3d(0, 300, 0);
    glVertex3d(0, -300, 0);
    glEnd();
    glBegin(GL_LINES);
    glVertex3d(0, 0, 300);
    glVertex3d(0, 0, -300);
    glEnd();
}

void myDisplay() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glColor4f(0, 0, 0, 1);
    drawAxis();
    displayCuboid(cuboid);
    if (tflag) displayCuboid(tcuboid, 1);
    glFlush();
    getTransformMatrix();
}

int main(int argc, char** argv) {

    cuboid = initCuboid();
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(960, 960);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("Basic 3D transformations");
    myInit();
    disp();
    glutDisplayFunc(myDisplay);
    glutMainLoop();
}
```

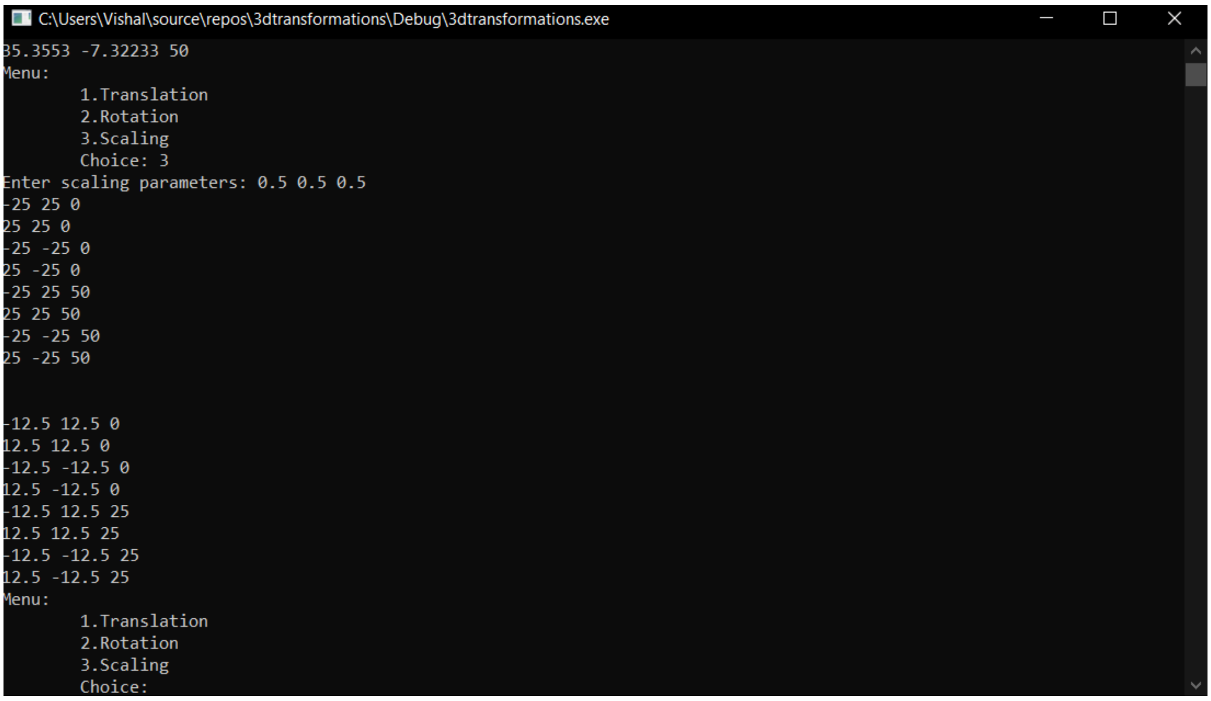
OUTPUTS:



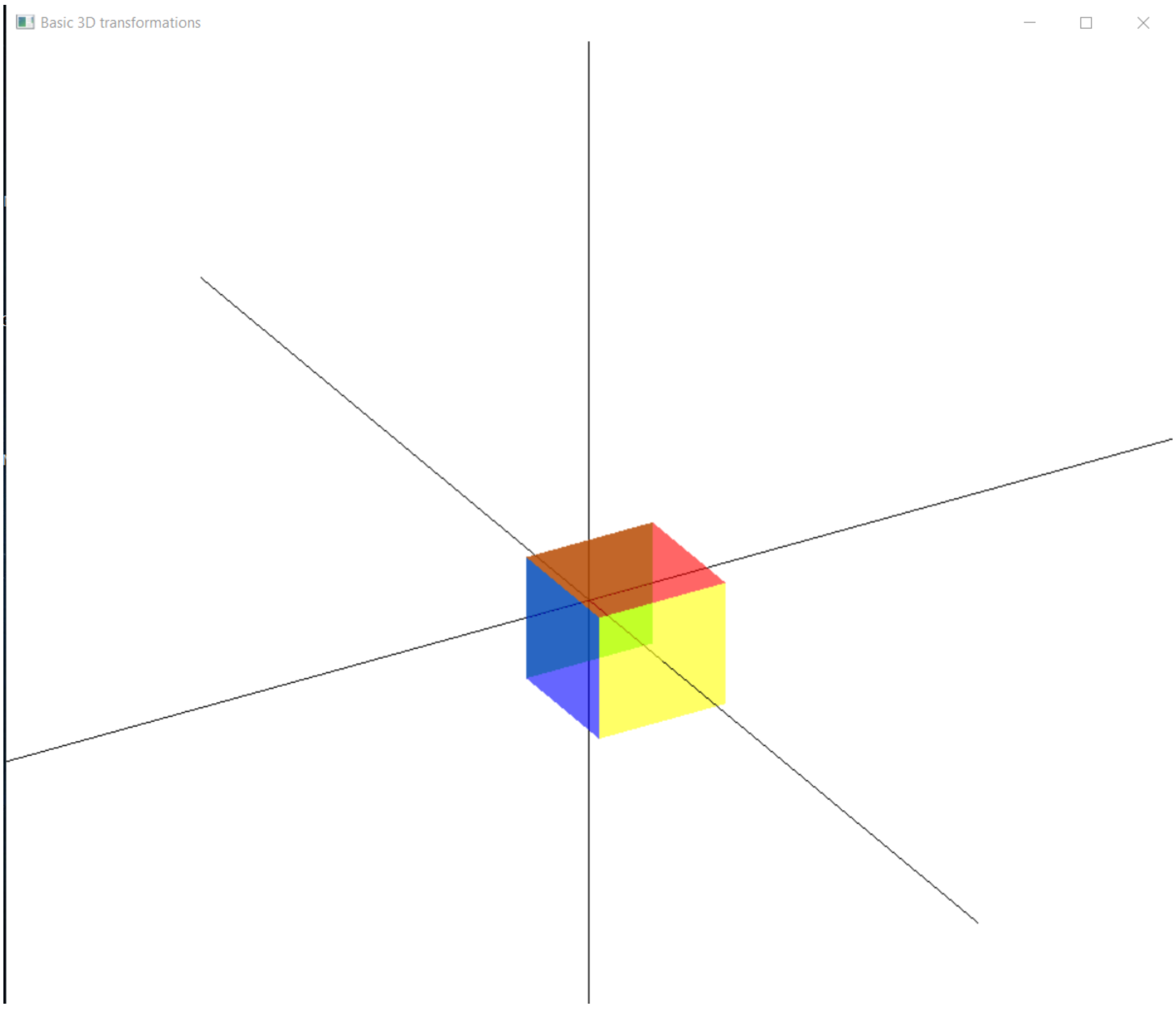


```
C:\Users\Vishal\source\repos\3dtransformations\Debug\3dtransformations.exe
Menu:
        1.Translation
        2.Rotation
        3.Scaling
        Choice: 2
Enter degree of rotation: 45
-25 25 0
25 25 0
-25 -25 0
25 -25 0
-25 25 50
25 25 50
-25 -25 50
25 -25 50


-35.3553 7.32233 0
3.55271e-15 42.6777 0
-3.55271e-15 -42.6777 0
35.3553 -7.32233 0
-35.3553 7.32233 50
3.55271e-15 42.6777 50
-3.55271e-15 -42.6777 50
35.3553 -7.32233 50
Menu:
        1.Translation
        2.Rotation
        3.Scaling
        Choice:
```





```
C:\Users\Vishal\source\repos\3dtransformations\Debug\3dtransformations.exe
35.3553 -7.32233 50
Menu:
        1.Translation
        2.Rotation
        3.Scaling
        Choice: 3
Enter scaling parameters: 0.5 0.5 0.5
-25 25 0
25 25 0
-25 -25 0
25 -25 0
-25 25 50
25 25 50
-25 -25 50
25 -25 50


-12.5 12.5 0
12.5 12.5 0
-12.5 -12.5 0
12.5 -12.5 0
-12.5 12.5 25
12.5 12.5 25
-12.5 -12.5 25
12.5 -12.5 25
Menu:
        1.Translation
        2.Rotation
        3.Scaling
        Choice:
```


Basic 3D transformations