NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR

Cachar, Assam

B.Tech. VIth Sem

Subject Code: CS-317

Subject Name: Graphics and Multimedia Lab

Submitted By:

Name : Subhojit Ghimire

Sch. Id. : 1912160

Branch : CSE – B

1. **Perform Translation and Scaling Transformations on a 3-D object.**

➔ **CODE:**

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

using namespace std;
typedef float Matrix4 [4][4];

Matrix4 theMatrix;
int choice;
float output [8][3];
float tx, ty, tz;
float sx, sy, sz;
//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}
//};
static GLfloat input [8][3] = {
    {80, 80, -100}, {180, 80, -100}, {180, 180, -100}, {80, 180, -100},
    {60, 60, 0}, {160, 60, 0}, {160, 160, 0}, {60, 160, 0}
};

void myinit (void) {
    glClearColor (1.0, 1.0, 1.0, 1.0);
    glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    glOrtho (-500.0, 500.0, -500.0, 500.0, -500.0, 500.0);
    glEnable (GL_DEPTH_TEST);
}

void drawQuadrants () {
    glPointSize (3.0);
    glColor3f (0.0f, 0.5f, 0.5f);

    glBegin (GL_LINE_LOOP);
        glVertex3f (-500.0, 0.0, 0.0);
        glVertex3f (500.0, 0.0, 0.0);
    glEnd ();

    glBegin (GL_LINE_LOOP);
        glVertex3f (0.0, -500.0, 0.0);
        glVertex3f (0.0, 500.0, 0.0);
    glEnd ();
}
```

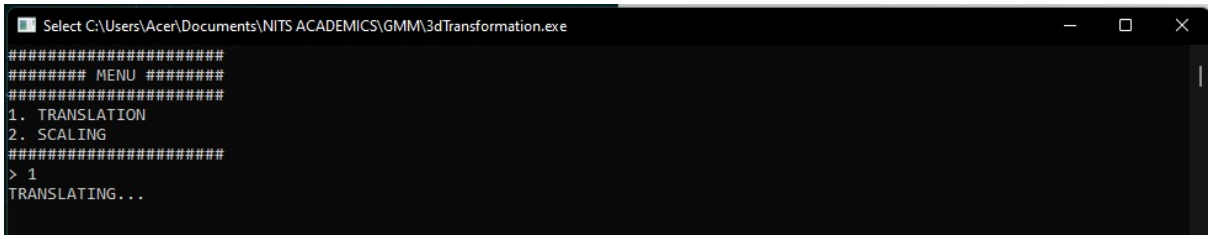
```
void transformPoints () {
    float tmp;
    for (int kk = 0; kk < 8; ++kk)
        for (int ii = 0; ii < 3; ++ii) {
            output [kk][ii]  = theMatrix [ii][0] * input [kk][0];
            output [kk][ii] += theMatrix [ii][1] * input [kk][1];
            output [kk][ii] += theMatrix [ii][2] * input [kk][2];
            output [kk][ii] += theMatrix [ii][3];
        }
}

void setIdentityM (Matrix4 mat) {
for (int ii = 0; ii < 4; ++ii)
    for (int jj = 0; jj < 4; ++jj)
        mat [ii][jj] = (ii == jj);
}

void multiplyM (Matrix4 matA, Matrix4 matB) {
    Matrix4 tmp;
    for (int ii = 0; ii < 4; ++ii)
        for (int jj = 0; jj < 4; ++jj) {
            tmp [ii][jj]  = matA [ii][0] * matB [0][jj];
            tmp [ii][jj] += matA [ii][1] * matB [1][jj];
            tmp [ii][jj] += matA [ii][2] * matB [2][jj];
            tmp [ii][jj] += matA [ii][3] * matB [3][jj];
        }
    for (int ii = 0; ii < 4; ++ii)
        for (int jj = 0; jj < 4; ++jj)
            theMatrix [ii][jj] = tmp [ii][jj];
}

void translate (int tx, int ty, int tz) {
    Matrix4 mat;
    setIdentityM (mat);
    mat [0][3] = tx;
    mat [1][3] = ty;
    mat [3][3] = tz;
    multiplyM (mat, theMatrix);
}

void scale (float sx, float sy, float sz) {
    Matrix4 mat;
    setIdentityM (mat);
    mat [0][0] = sx;
    mat [0][3] = (1 - sx);
    mat [1][1] = sy;
    mat [1][3] = (1 - sy);
```

```
    mat [2][2] = sz;
    mat [2][3] = (1 - sy);
    multiplyM (mat, theMatrix);
}


void draw (float outMat [8][3]) {
    int ii;
    glColor3f (0.7, 0.4, 0.7);
    glBegin (GL_LINE_LOOP);
        glVertex3f (outMat [0][0], outMat [0][1], outMat [0][2]);
        glVertex3f (outMat [1][0], outMat [1][1], outMat [1][2]);
        glVertex3f (outMat [2][0], outMat [2][1], outMat [2][2]);
        glVertex3f (outMat [3][0], outMat [3][1], outMat [3][2]);
    glEnd();

    ii = 0;
    glColor3f (0.8, 0.6, 0.5);
    glBegin (GL_LINE_LOOP);
        glVertex3s (outMat [0 + ii][0], outMat [0 + ii][1], outMat [0 +
ii][2]);
        glVertex3s (outMat [1 + ii][0], outMat [1 + ii][1], outMat [1 +
ii][2]);
        glVertex3s (outMat [5 + ii][0], outMat [5 + ii][1], outMat [5 +
ii][2]);
        glVertex3s (outMat [4 + ii][0], outMat [4 + ii][1], outMat [4 +
ii][2]);
    glEnd();

    glColor3f (0.2, 0.4, 0.7);
    glBegin (GL_LINE_LOOP);
        glVertex3f (outMat [0][0], outMat [0][1], outMat [0][2]);
        glVertex3f (outMat [3][0], outMat [3][1], outMat [3][2]);
        glVertex3f (outMat [7][0], outMat [7][1], outMat [7][2]);
        glVertex3f (outMat [4][0], outMat [4][1], outMat [4][2]);
    glEnd();

    ii = 1;
    glColor3f (0.5, 0.4, 0.3);
    glBegin (GL_LINE_LOOP);
        glVertex3s (outMat [0 + ii][0], outMat [0 + ii][1], outMat [0 +
ii][2]);
        glVertex3s (outMat [1 + ii][0], outMat [1 + ii][1], outMat [1 +
ii][2]);
        glVertex3s (outMat [5 + ii][0], outMat [5 + ii][1], outMat [5 +
ii][2]);
        glVertex3s (outMat [4 + ii][0], outMat [4 + ii][1], outMat [4 +
ii][2]);
```

```
    glEnd();

    ii = 2;
    glColor3f (0.5, 0.6, 0.2);
    glBegin(GL_LINE_LOOP);
        glVertex3s (outMat [0 + ii][0], outMat [0 + ii][1], outMat [0 +
ii][2]);
        glVertex3s (outMat [1 + ii][0], outMat [1 + ii][1], outMat [1 +
ii][2]);
        glVertex3s (outMat [5 + ii][0], outMat [5 + ii][1], outMat [5 +
ii][2]);
        glVertex3s (outMat [4 + ii][0], outMat [4 + ii][1], outMat [4 +
ii][2]);
    glEnd();

    ii = 4;
    glColor3f (0.7, 0.3, 0.4);
    glBegin(GL_LINE_LOOP);
        glVertex3f (outMat [0 + ii][0], outMat [0 + ii][1], outMat [0 +
ii][2]);
        glVertex3f (outMat [1 + ii][0], outMat [1 + ii][1], outMat [1 +
ii][2]);
        glVertex3f (outMat [2 + ii][0], outMat [2 + ii][1], outMat [2 +
ii][2]);
        glVertex3f (outMat [3 + ii][0], outMat [3 + ii][1], outMat [3 +
ii][2]);
    glEnd();
}

void display () {
    drawQuadrants ();
    glColor3f (1.0, 0.0, 0.0);
    draw (input);
    setIdentityM (theMatrix);

    cout << "######################\n"
            "######## MENU ########\n"
            "######################\n"
            "1. TRANSLATION\n"
            "2. SCALING\n"
            "######################\n"
            "> ";
    cin >> choice;

    switch (choice) {
        case 1: printf ("TRANSLATING...\n");
                tx = 105;
                ty = 110;
```

```
                tz = 0;
                translate (tx, ty, tz);
                break;
        case 2: printf ("SCALING...\n");
                sx = 0.5;
                sy = 0.7;
                sz = 0.2;
                scale (sx, sy, sz);
                break;
        default: printf ("ERROR: OPTION NOT AVAILABLE!");
    }
    transformPoints ();
    draw (output);
    glFlush ();
}

int main (int argc, char **argv) {
    glutInit (&argc, argv);
    glutInitWindowSize (1000, 1000);
    glutInitWindowPosition (500, 0);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
    glutCreateWindow ("3D Transformations");
    myinit ();
    glutDisplayFunc (display);
    glutMainLoop ();
    return 0;
}
```

OUTPUT:



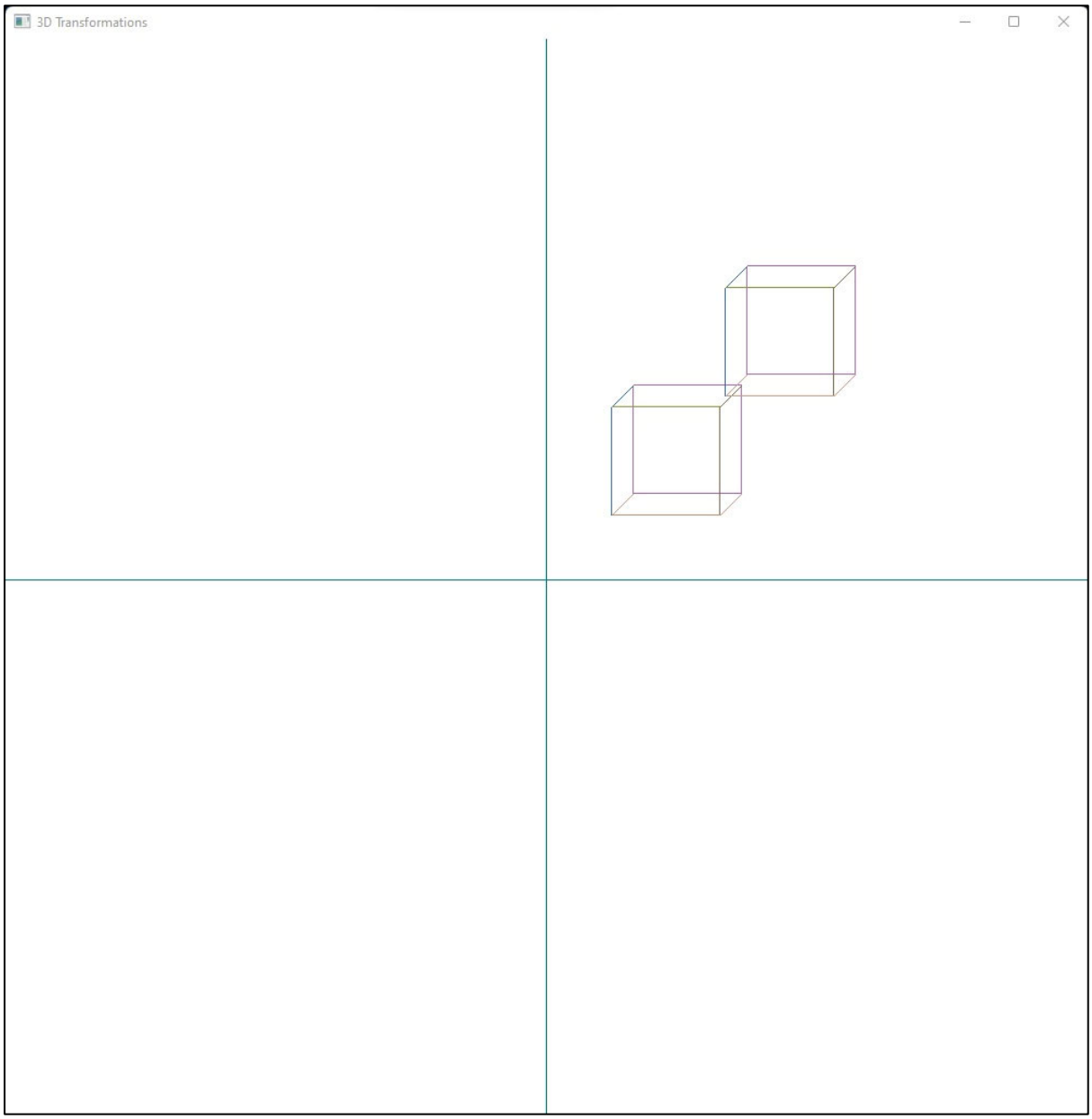
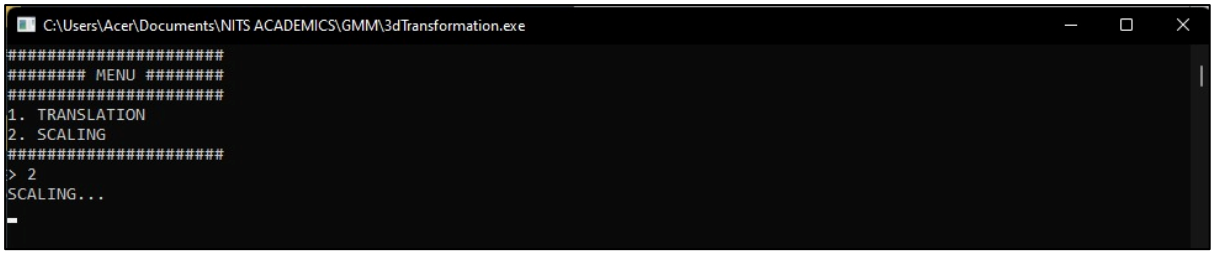


Fig.: 3-D Translation



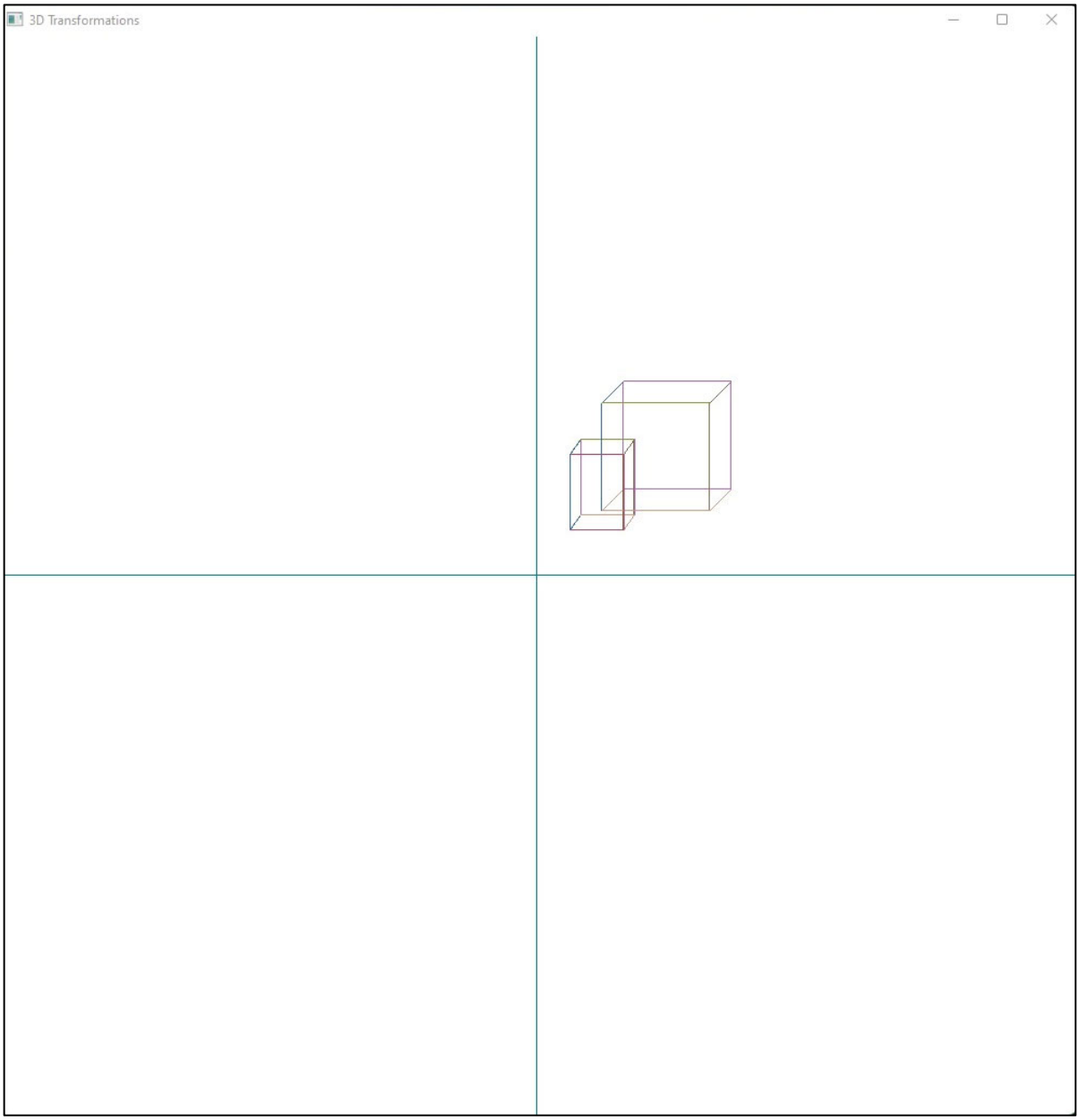


Fig.: 3-D Scaling