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

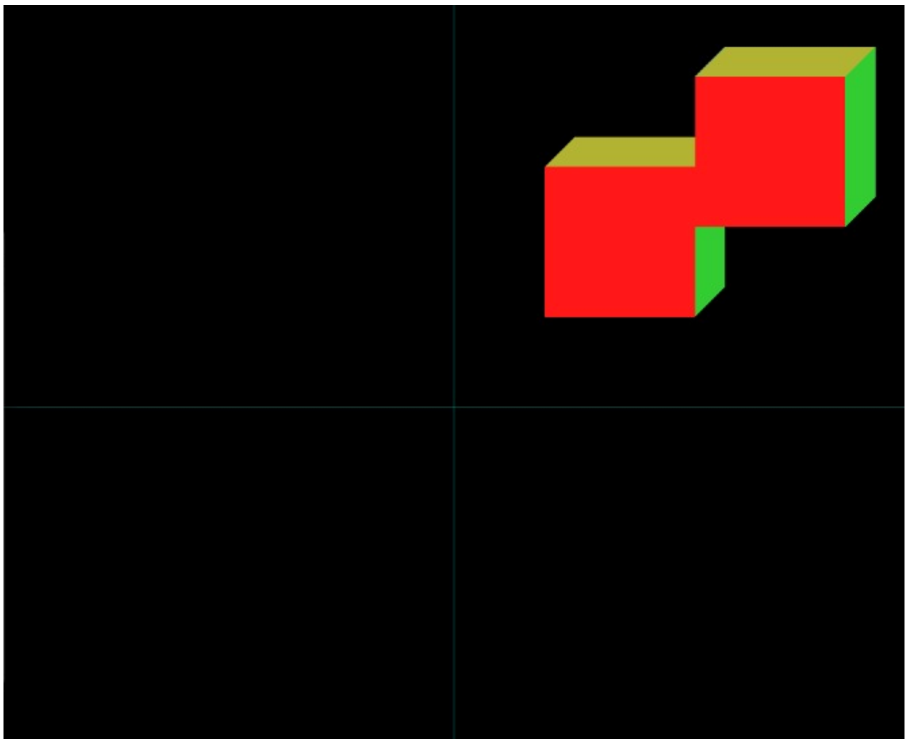
1. **Perform Translation and Scaling Transformations on a 3-D object.**

➔ **CODE:**

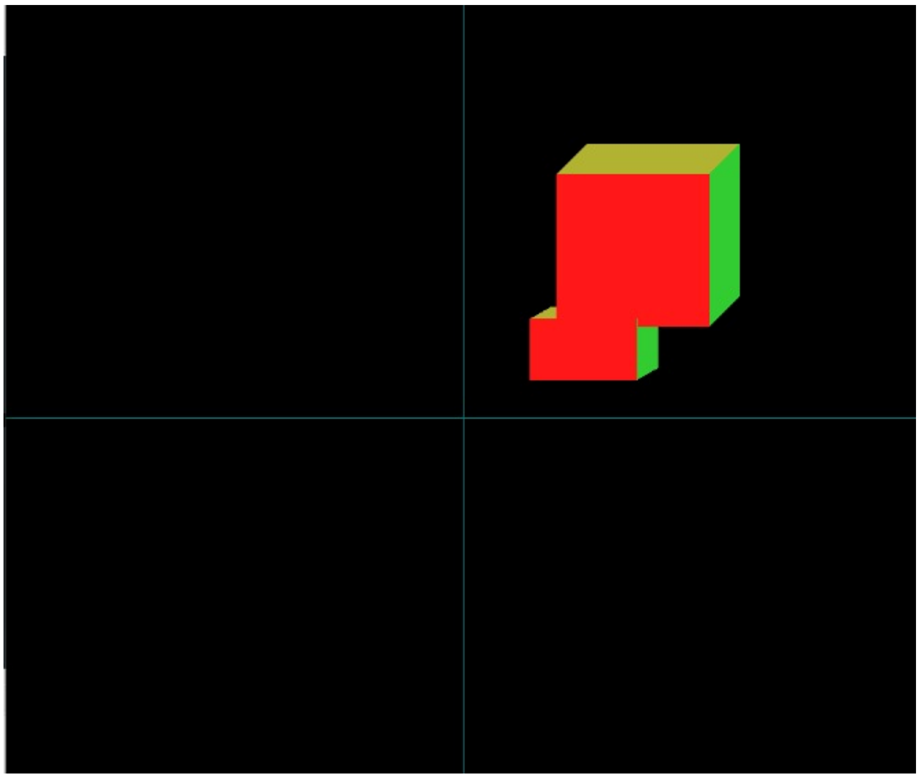
```
#include <iostream>
#include <math.h>
#include <GL/glut.h>

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}};
void myinit (void) {
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    glOrtho (-150.0, 150.0, -150.0, 150.0, -150.0, 150.0);
    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 k = 0; k < 8; ++k)
        for (int i = 0; i < 3; ++i) {
            output [k][i]  = theMatrix [i][0] * input [k][0];
            output [k][i] += theMatrix [i][1] * input [k][1];
            output [k][i] += theMatrix [i][2] * input [k][2];
            output [k][i] += theMatrix [i][3];
        }
}

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

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

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]) {
    glBegin (GL_QUADS);

        glColor3f (0.7, 0.4, 0.5);
        glVertex3fv (outMat [0]);
        glVertex3fv (outMat [1]);
        glVertex3fv (outMat [2]);
        glVertex3fv (outMat [3]);

        glColor3f (0.8, 0.2, 0.4);
        glVertex3fv (outMat [0]);
        glVertex3fv (outMat [1]);
        glVertex3fv (outMat [5]);
        glVertex3fv (outMat [4]);

        glColor3f (0.3, 0.6, 0.7);
        glVertex3fv (outMat [0]);
        glVertex3fv (outMat [4]);
        glVertex3fv (outMat [7]);
        glVertex3fv (outMat [3]);

        glColor3f (0.2, 0.8, 0.2);
        glVertex3fv (outMat [1]);
        glVertex3fv (outMat [2]);
        glVertex3fv (outMat [6]);
        glVertex3fv (outMat [5]);

        glColor3f (0.7, 0.7, 0.2);
        glVertex3fv (outMat [2]);
        glVertex3fv (outMat [3]);
        glVertex3fv (outMat [7]);
        glVertex3fv (outMat [6]);

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

    glEnd ();
}

void display () {
    drawQuadrants ();
    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); break;
        default: break;
    }
    transformPoints ();
    draw (output);
    glFlush ();
}

int main (int argc, char **argv) {
    glutInit (&argc, argv);
    cout << "Choose:\n1. Translation\n2. Scaling\n";
    cin >> choice;

    switch (choice) {
        case 1: tx = 50; ty = 30; tz = 0;
                break;
        case 2: sx = 0.7; sy = 0.4; sz = 0.2;
                break;
        default: break;
    }
    glutInitWindowSize (1000, 1000);
    glutInitWindowPosition (500, 0);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
    glutCreateWindow ("3-D Translation and Scaling");
    myinit ();
    glutDisplayFunc (display);
    glutMainLoop ();
    return 0;
}
```

OUTPUT:



3-D Translation



3-D Scaling