Name : Harshada Gopal Rayate

Roll No : 19 [SEDA]

Subject : CGAVR

Assignment : 3D- Transformation

```
#define _CRT_SECURE_NO_WARNINGS
#include <GL/glut.h>
#include <iostream>
#include <cmath>

#define PI 3.14159265358979323846

using namespace std;

struct Point {
float x, y, z;
};

int num_points = 8;
Point points[8] = { {40,40,-50},{90,40,-50},{90,90,-50},{40,90,-
50},{30,30,0},{80,30,0},{80,80,0},{30,80,0} };
Point transformed_points[8];
int choice, c, ch, rot;
float tx, ty, tz, sx, sy, sz, angle;


void apply_transformation() {
float rad = angle * PI / 180.0;

switch (choice) {
case 1: // Translation
for (int i = 0; i < num_points; i++) {
```

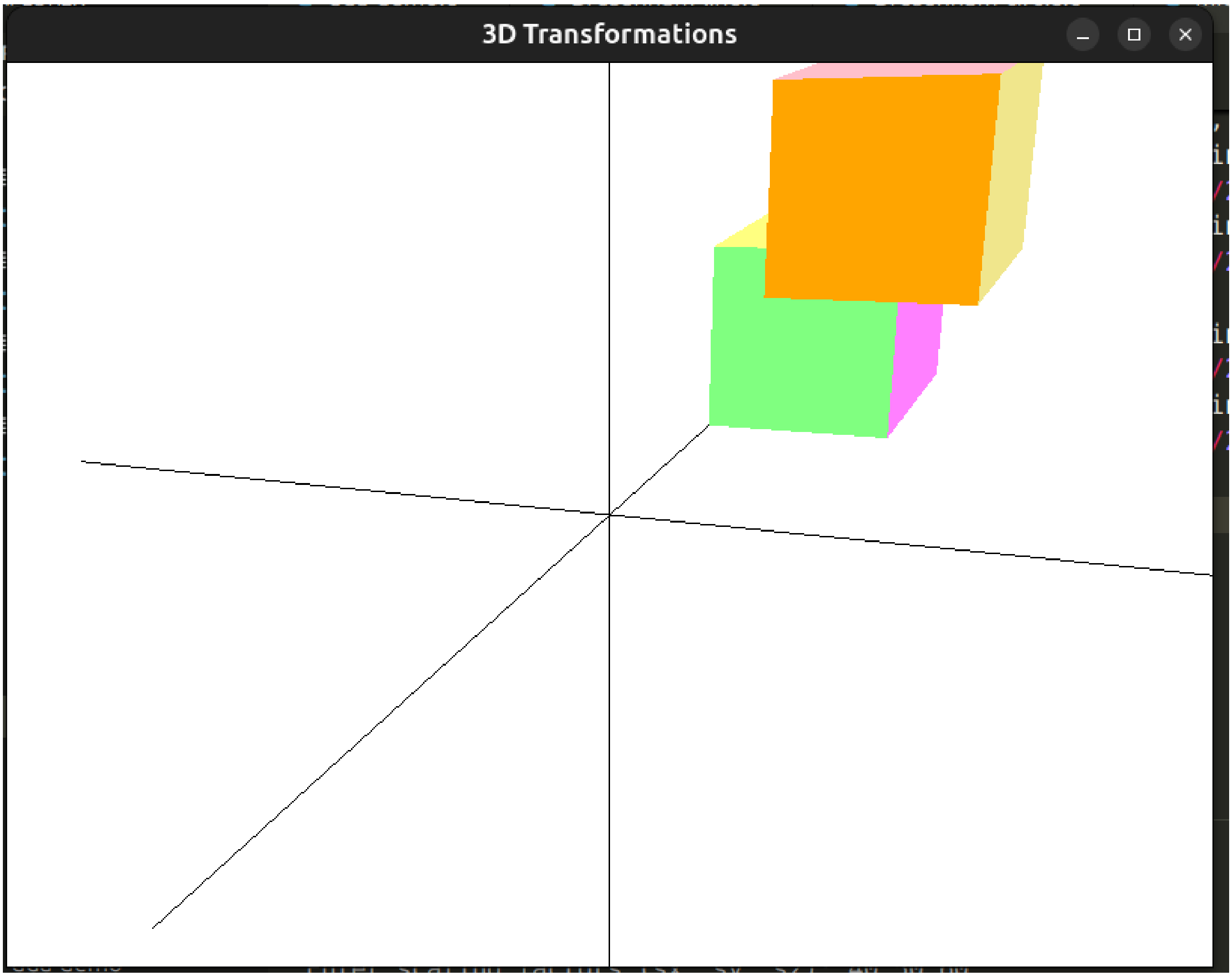
```
transformed_points[i].x = points[i].x + tx;
transformed_points[i].y = points[i].y + ty;
transformed_points[i].z = points[i].z + tz;
}
break;

case 2: // Scaling
for (int i = 0; i < num_points; i++) {
transformed_points[i].x = points[i].x * sx;
transformed_points[i].y = points[i].y * sy;
transformed_points[i].z = points[i].z * sz;
}
break;

case 3: // Rotation
if (rot == 1) { // Rotate around X axis
for (int i = 0; i < num_points; i++) {
transformed_points[i].x = points[i].x;
transformed_points[i].y = points[i].y * cos(rad) - points[i].z *

sin(rad);

transformed_points[i].z = points[i].y * sin(rad) + points[i].z *

cos(rad);
}
}
else if (rot == 2) { // Rotate around Y axis
for (int i = 0; i < num_points; i++) {
transformed_points[i].x = points[i].x * cos(rad) + points[i].z *

sin(rad);

transformed_points[i].y = points[i].y;
transformed_points[i].z = points[i].z * cos(rad) - points[i].x *

sin(rad);
}
}
else if (rot == 3) { // Rotate around Z axis
for (int i = 0; i < num_points; i++) {
transformed_points[i].x = points[i].x * cos(rad) - points[i].y *

sin(rad);

transformed_points[i].y = points[i].x * sin(rad) + points[i].y *

cos(rad);

transformed_points[i].z = points[i].z;
}
}
break;

default:
cout << "Invalid choice!" << endl;
break;
}
}

void draw_axes() {
glColor3f(0.0, 0.0, 0.0);
glBegin(GL_LINES);
glVertex3f(-200.0, 0.0, 0.0);
glVertex3f(200.0, 0.0, 0.0);
glVertex3f(0.0, -200.0, 0.0);
glVertex3f(0.0, 200.0, 0.0);
glVertex3f(0.0, 0.0, -200.0);
```

```
glVertex3f(0.0, 0.0, 200.0);
glEnd();
}


void draw_face(Point a, Point b, Point c, Point d, float r, float g, float bColor) {
glColor3f(r, g, bColor);
glBegin(GL_QUADS);
glVertex3f(a.x, a.y, a.z);
glVertex3f(b.x, b.y, b.z);
glVertex3f(c.x, c.y, c.z);
glVertex3f(d.x, d.y, d.z);
glEnd();
}

void display() {
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glLoadIdentity();
gluLookAt(100.0, 100.0, 300.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

draw_axes();

// Draw original points with new colors
draw_face(points[0], points[1], points[2], points[3], 1.0, 0.5, 0.5); // Light Red
draw_face(points[4], points[5], points[6], points[7], 0.5, 1.0, 0.5); // Light Green
draw_face(points[0], points[1], points[5], points[4], 0.5, 0.5, 1.0); // Light Blue
draw_face(points[2], points[3], points[7], points[6], 1.0, 1.0, 0.5); // Light Yellow
draw_face(points[1], points[2], points[6], points[5], 1.0, 0.5, 1.0); // Light Magenta
draw_face(points[0], points[3], points[7], points[4], 0.5, 1.0, 1.0); // Light Cyan

// Draw transformed points with new colors
draw_face(transformed_points[0], transformed_points[1], transformed_points[2],
transformed_points[3], 1.0, 1.0, 1.0); // White
draw_face(transformed_points[4], transformed_points[5], transformed_points[6],
transformed_points[7], 1.0, 165/255.f, 0); // Orange
draw_face(transformed_points[0], transformed_points[1], transformed_points[5],
transformed_points[4], 128/255.f, 128/255.f, 128/255.f); // Gray
draw_face(transformed_points[2], transformed_points[3], transformed_points[7],
transformed_points[6], 255/255.f, 192/255.f, 203/255.f); // Pink

draw_face(transformed_points[1], transformed_points[2], transformed_points[6],
transformed_points[5], 240/255.f, 230/255.f, 140/255.f); // Khaki
draw_face(transformed_points[0], transformed_points[3], transformed_points[7],
transformed_points[4], 135/255.f, 206/255.f, 235/255.f); // Sky Blue

glFlush();
}


void init() {
glClearColor(1.0, 1.0, 1.0, 1.0);
glEnable(GL_DEPTH_TEST);
}

void reshape(int w, int h) {
glViewport(0, 0, w, h);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(45.0, (float)w / h, 1.0, 1000.0);
glMatrixMode(GL_MODELVIEW);
}

int main(int argc, char** argv) {
cout << "3D Transformation\n";
cout << "1: Translation\n2: Scaling\n3: Rotation\n";
cout << "Enter your choice: ";
cin >> choice;
switch (choice) {
```

```
case 1:
cout << "Enter translation factors (tx, ty, tz): ";
cin >> tx >> ty >> tz;
break;
case 2:
cout << "Enter scaling factors (sx, sy, sz): ";


cin >> sx >> sy >> sz;
break;
case 3:
cout << "1. Rotation along X axis\n2. Rotation along Y axis\n3. Rotation along Z axis\n";
cout << "Enter your choice: ";
cin >> ch;
if (ch == 1) rot = 1;
else if (ch == 2) rot = 2;
else if (ch == 3) rot = 3;
cout << "Enter rotation angle (in degrees): ";
cin >> angle;
break;
default:
cout << "Invalid choice!" << endl;
return 0;
}

apply_transformation();

glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
glutInitWindowSize(800, 600);
glutCreateWindow("3D Transformations");

init();
glutDisplayFunc(display);
```

```
glutReshapeFunc(reshape);
glutMainLoop();
return 0;
}
```

OUTPUT :



3D Transformations