Lab - 7

Course Title	:	Computer Graphics
Course Code	:	CSE420
Semester	:	Summer 2025
Section	:	02

Submitted by:

Al Shahriyar Shrabon

(ID: 2019-3-60-098)

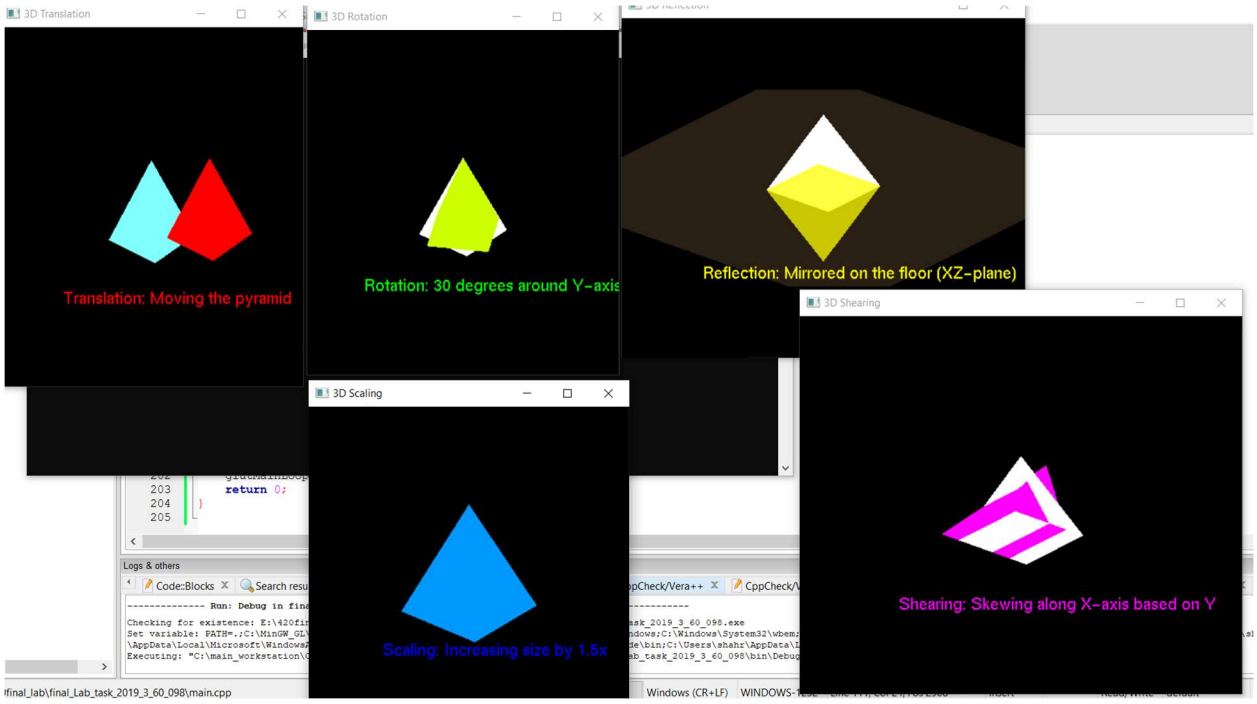
Submitted to:

Dr. Md. Tauhid Bin Iqbal

Assistant Professor

Department of Computer Science and Engineering

Output:



Source Code:

```
#include <iostream>
#include <GL/gl.h>
#include <GL/glut.h>
#include <string>

using namespace std;

void drawPyramid()
{
    glBegin(GL_QUADS);
    glVertex3f(-0.2f, -0.2f, -0.2f);
    glVertex3f( 0.2f, -0.2f, -0.2f);
    glVertex3f( 0.2f, -0.2f,  0.2f);
    glVertex3f(-0.2f, -0.2f,  0.2f);
    glEnd();
```

```
    glBegin(GL_TRIANGLES);
    glVertex3f( 0.0f,  0.3f,  0.0f);
    glVertex3f(-0.2f, -0.2f,  0.2f);
    glVertex3f( 0.2f, -0.2f,  0.2f);

    glVertex3f( 0.0f,  0.3f,  0.0f);
    glVertex3f( 0.2f, -0.2f,  0.2f);
    glVertex3f( 0.2f, -0.2f, -0.2f);

    glVertex3f( 0.0f,  0.3f,  0.0f);
    glVertex3f( 0.2f, -0.2f, -0.2f);
    glVertex3f(-0.2f, -0.2f, -0.2f);

    glVertex3f( 0.0f,  0.3f,  0.0f);
    glVertex3f(-0.2f, -0.2f, -0.2f);
    glVertex3f(-0.2f, -0.2f,  0.2f);
    glEnd();
}

void drawCenteredText(const char* text, float r, float g, float b, int windowWidth)
{
    glColor3f(r, g, b);

    float text_pixel_width = 0;
    for (const char *c = text; *c != '\0'; c++)
    {
        text_pixel_width += glutBitmapWidth(GLUT_BITMAP_HELVETICA_18, *c);
    }

    float normalized_text_width = text_pixel_width / windowWidth * 2.0f;

    glRasterPos2f(-normalized_text_width / 2.0f, -0.9f);

    for (const char* c = text; *c != '\0'; c++)
    {
        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, *c);
    }
}
void drawFloor()
{
    glColor4f(0.4f, 0.3f, 0.2f, 0.4f);
    glBegin(GL_QUADS);
    glVertex3f(-1.0f, 0.0f,  1.0f);
    glVertex3f( 1.0f, 0.0f,  1.0f);
    glVertex3f( 1.0f, 0.0f, -1.0f);
```

```
    glVertex3f(-1.0f, 0.0f, -1.0f);
    glEnd();
}
void setupView()
{
    glRotatef(30.0f, 1.0f, 0.0f, 0.0f);
    glRotatef(-40.0f, 0.0f, 1.0f, 0.0f);
}
void translate3D(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    setupView();
    glColor3f(0.5f, 1.0f, 1.0f);
    drawPyramid();
    glPushMatrix();
    glTranslatef(0.5f, 0.2f, 0.0f);
    glColor3f(1.0f, 0.0f, 0.0f);
    drawPyramid();
    glPopMatrix();
    drawCenteredText("Translation: Moving the pyramid", 1.0f, 0.0f, 0.0f,
                glutGet(GLUT_WINDOW_WIDTH));
    glFlush();
}
void rotate3D(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    setupView();
    glColor3f(1.0f, 1.0f, 1.0f);
    drawPyramid();
    glPushMatrix();
    glRotatef(30.0f, 0.0f, 1.0f, 0.0f);
    glColor3f(0.8f, 1.0f, 0.0f);
    drawPyramid();
    glPopMatrix();
    drawCenteredText("Rotation: 30 degrees around Y-axis", 0.0f, 1.0f, 0.0f,
                glutGet(GLUT_WINDOW_WIDTH));
    glFlush();
}
void scale3D(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    setupView();
    glColor3f(1.0f, 1.0f, 1.0f);
```

```
    drawPyramid();
    glPushMatrix();
    glScalef(1.5f, 1.5f, 1.5f);
    glColor3f(0.0f, 0.6f, 1.0f);
    drawPyramid();
    glPopMatrix();
    drawCenteredText("Scaling: Increasing size by 1.5x", 0.0f, 0.0f, 1.0f,
                     glutGet(GLUT_WINDOW_WIDTH));
    glFlush();
}
void reflect3D(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    setupView();
    glEnable(GL_BLEND);
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
    drawFloor();
    glPushMatrix();
    glTranslatef(0.0f, 0.2f, 0.0f);
    glColor3f(1.0f, 1.0f, 1.0f);
    drawPyramid();
    glPopMatrix();
    glPushMatrix();
    glScalef(1.0f, -1.0f, 1.0f);
    glTranslatef(0.0f, 0.2f, 0.0f);
    glColor4f(1.0f, 1.0f, 0.0f, 0.5f);
    drawPyramid();
    glPopMatrix();
    glDisable(GL_BLEND);
    drawCenteredText("Reflection: Mirrored on the floor (XZ-plane)", 1.0f, 1.0f, 0.0f,
                     glutGet(GLUT_WINDOW_WIDTH));
    glFlush();
}
void shear3D(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    setupView();
    glColor3f(1.0f, 1.0f, 1.0f);
    drawPyramid();
    glPushMatrix();
    GLfloat shearMatrix[16] =
    {
        1.0f, 0.0f, 0.0f, 0.0f,
        0.5f, 1.0f, 0.0f, 0.0f,
```

```
        0.0f, 0.0f, 1.0f, 0.0f,
        0.0f, 0.0f, 0.0f, 1.0f
    };
    glMultMatrixf(shearMatrix);
    glColor3f(1.0f, 0.0f, 1.0f);
    drawPyramid();
    glPopMatrix();
    drawCenteredText("Shearing: Skewing along X-axis based on Y", 1.0f, 0.0f, 1.0f,
                     glutGet(GLUT_WINDOW_WIDTH));
    glFlush();
}
void initWindows(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
    int windowWidth = 500;
    int windowHeight = 500;
    glutInitWindowSize(windowWidth, windowHeight);
    glutInitWindowPosition(50, 50);
    glutCreateWindow("3D Translation");
    glutDisplayFunc(translate3D);
    glutInitWindowSize(windowWidth, windowHeight);
    glutInitWindowPosition(560, 50);
    glutCreateWindow("3D Rotation");
    glutDisplayFunc(rotate3D);
    glutInitWindowSize(windowWidth, windowHeight);
    glutInitWindowPosition(50, 560);
    glutCreateWindow("3D Scaling");
    glutDisplayFunc(scale3D);
    glutInitWindowSize(windowWidth, windowHeight);
    glutInitWindowPosition(560, 560);
    glutCreateWindow("3D Reflection");
    glutDisplayFunc(reflect3D);
    glutInitWindowSize(windowWidth, windowHeight);
    glutInitWindowPosition(1070, 50);
    glutCreateWindow("3D Shearing");
    glutDisplayFunc(shear3D);
}

void initGL()
{
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    glEnable(GL_DEPTH_TEST);
}
int main(int argc, char** argv)
{
```

```
    initWindows(argc, argv);
    initGL();
    glutMainLoop();
    return 0;
}
```