COMPUTER GRAPHICS

LAB#2

RAMY AHMED EL SAYED

19015649

PROBLEM STATEMENT

You are required to create an OpenGL project using the project template. You should implement an application that asks user about the projection type (orthographic or perspective). If user chooses orthographic projection a triangle should be drawn; otherwise, a pyramid should be drawn.

At runtime, Input handling should be as follows:

- When user clicks left mouse button, scene should spin around specific axis in counterclockwise (CCW) manner.
- When user clicks right mouse button, scene should spin around specific axis in clockwise (CW) manner.
- When user presses space button, scene should stop spinning.
- When user presses ‘i’, you should zoom in the scene.
- When user presses ‘o’, you should zoom out the scene.

If user chooses parallel projection, the spinning should be around z-axis; otherwise spinning should be around y-axis. Use code in this link as starter code.

Code Description

Ortho Projection

Main Loop

```
glTranslatef(0, 0, 15 + zOffset);
glRotatef(currentSpin, 0, 0, 1);
glBegin(GL_TRIANGLES);
glVertex3f(0, 10, 0);
glVertex3f(-30, 0, 0);
glVertex3f(30, 0, 0);
glEnd();
```

glTranslatef is used for zooming in/out (which won't show any effect aside from trimming the shape itself. The zOffset value is changed in the keyinput function as follows:

```
case zoomIn:
    /*
    write code below:
        1- zoom in
        2- mark window for rerendering
    */
    // code here
    zOffset += 15;
    std::cout << "Zooming in" << "\n";
    glutPostRedisplay();
    //-------------------------------------
    break;
case zoomOut:
    /*
    write code below:
        1- zoom out
        2- mark window for rerendering
    */
    // code here
    zOffset -= 15;
    std::cout << "Zooming out" << "\n";
    glutPostRedisplay();
    //-------------------------------------
    break;
```

glRotatef is used to rotate the shape around the Z-Axis based on the spinning speed/direction, and the rotation changes as follows: eg CCW

```
void spinDisplay(void)
{
    /*
        write code below:
        1- change currentSpin according to spinSpeed (note: spinSpeed unit is dgree/second)
        2- mark window to be rerendered (hint: glutPostRedisplay, prveTime)
    */
    // code here
    currentSpin = currentSpin + (spinSpeed * (glutGet(GLUT_ELAPSED_TIME) - prevTime) / 1000);
    prevTime = glutGet(GLUT_ELAPSED_TIME);
    while ((glutGet(GLUT_ELAPSED_TIME) - prevTime) < 10)
    {

    }
    glutPostRedisplay();
    //-----------------------------------------------------
}
```

To make the increase non linear and dependent on our delta time which helps in synchronizing the rate of change with our frames, we multiply the spin speed with the difference in time between the current and last time the glutIdleFunc() was called.

Additionally, we add a small busy wait to control the frame rate generation (amount of spinDisplay calls) so that the processor doesn't get overwhelmed.

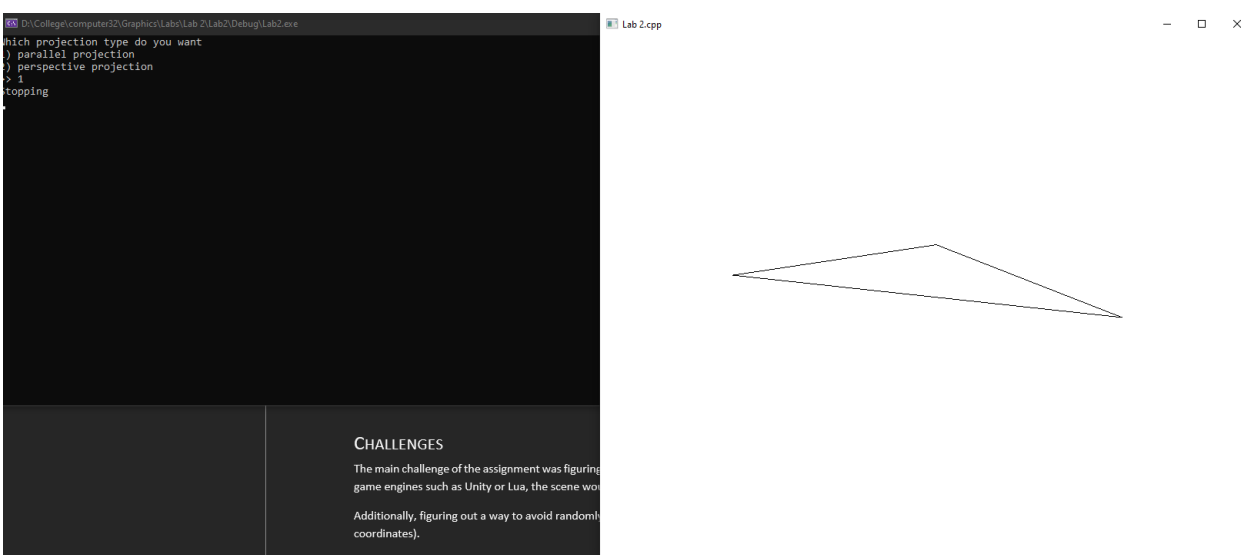
Perspective Projection (Frustum)

```
glTranslatef(0, 0,  zOffset);
glRotatef(currentSpin, 0, 1, 0);
glBegin(GL_TRIANGLES);
glVertex3f(0, 5, 0);
glVertex3f(5, 0, 5);
glVertex3f(5, 0, -5);

glVertex3f(0, 5, 0);
glVertex3f(5, 0, -5);
glVertex3f(-5, 0, -5);

glVertex3f(0, 5, 0);
glVertex3f(-5, 0, -5);
glVertex3f(-5, 0, 5);

glVertex3f(0, 5, 0);
glVertex3f(-5, 0, 5);
glVertex3f(5, 0, 5);

glEnd();
```

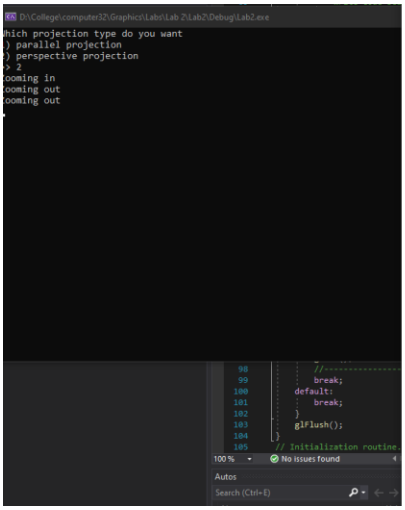
The same goes for Perspective Projection whilst the only difference being that the zoom effect caused by glTranslatef() is more apparent and doesn't simply trim the image.

Example Of Running Code

Ortho Projection



PERSPECTIVE PROJECTION



CHALLENGES

The main challenge was mostly trying to figure out how to deal with the input delay caused by the continuous glutPostRedisplay() function in the glutIdleFunc() call.