Lab2

Name: Abdelaziz Mohamed Abdelaziz

I.D: 19015941

Screenshots of code flow:

```
/* write user interaction here for good practice */
#include <GL/glew.h>
#include <GL/freeglut.h>
#include <iostream>
enum choice {ortho=1, prespective=2};
enum userInput {zoomIn='i', zoomOut='o', stopSpinning=' ', CCW=GLUT_LEFT_BUTTON, CW= GLUT_RIGHT_BUTTON};
int userChoice = 0;
float orthoLeft = -50;
float orthoRight = 50;
float orthoBottom = -50;
float orthoTop = 50;
float orthoNear = -5;
float orthoFar = 5;
float fruLeft = -5;
float fruRight = 5;
float fruBottom = -5;
float fruTop = 5;
float fruNear = 5;
float fruFar = 100;
float offsetX = 100;
float offsetY = 100;
float zOffset = -15;//zoom out/in
float windowWidth = 500;
float windowHeight = 500;
float spinY = 0;
float spinZ = 0;
float spinSpeed = 5;
float currentSpin = 0;
float prevTime = 0;
// Drawing routine.
```

in case of ortho we draw a triangle with following point and handle rotation using glrotatef.

main.cpp

```
31  void drawScene(void)
32  {
33      glClear(GL_COLOR_BUFFER_BIT);
34      glLoadIdentity();
35      glColor3f(0.0, 0.0, 0.0);
36      glLineWidth(1.0); // Default line width.
37      glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
38      switch (userChoice) {
39      case ortho:
40          /*
41              write code below:
42              1- handle spinning hint: glRotatef
43              2- draw triangle within viewing box
44              Recommended points:
45              (0, 10, 0)
46              (-30, 0, 0)
47              (30, 0, 0)
48              you are encourged to change points location and observe its effects on rotation
49          */
50          // code here
51          //-------------------------------------------------------------------------------
52
53          glRotatef(currentSpin, 0, spinY, spinZ);
54          glBegin(GL_TRIANGLES);
55          glVertex3f(0, 10, 0);
56          glVertex3f(-30, 0, 0);
57          glVertex3f(30, 0, 0);
58          glEnd();
59          break;
60      case prespective:
```

While in case of prespective we draw a pyramid using following points and handle zoom in and zoom out using gltransalte and handle rotation using glrotate .

```
60      case prespective:
61          /*
62              write code below:
63              1- handle zoom in/out hint: glTranslatef
64              2- handle spinning hint: glRotatef
65              3- draw pyramid within frustum
66              Recommended points:
67              (0, 5, 0)
68              (5, 0, 5)
69              (5, 0, -5)
70              (-5, 0, -5)
71              (-5, 0, 5)
72              you are encourged to change points location and observe its effects on rotation
73          */
74          // code here
75          //-------------------------------------------------------
76
77          glTranslatef(0.0f, 0.0f,  zOffset);
78          glRotatef(currentSpin, 0, spinY, spinZ);
79          glBegin(GL_TRIANGLES);
80          glVertex3f(-5.0f, -5.0f, 5.0f);
81          glVertex3f(5.0f, -5.0f, 5.0f);
82          glVertex3f(0.0f, -5.0f, -5.0f);
83          glVertex3f(-5.0f, -5.0f, 5.0f);
84          glVertex3f(5.0f, -5.0f, 5.0f);
85          glVertex3f(0.0f, 5.0f, 0.0f);
86          glVertex3f(5.0f, -5.0f, 5.0f);
87          glVertex3f(0.0f, -5.0f, -5.0f);
```

```
        glVertex3f(0.0f, 5.0f, 0.0f);
        glVertex3f(5.0f, -5.0f, 5.0f);
        glVertex3f(0.0f, -5.0f, -5.0f);
        glVertex3f(0.0f, 5.0f, 0.0f);
        glVertex3f(-5.0f, -5.0f, 5.0f);
        glVertex3f(0.0f, -5.0f, -5.0f);
        glVertex3f(0.0f, 5.0f, 0.0f);
        glEnd();

        break;
    default:
        break;
    }
    glFlush();
}
// Initialization routine.
void setup(void)
{
    glClearColor(1.0, 1.0, 1.0, 0.0);
}
//spin logic
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
    //
```

In case of spin display in which we make it to make the object spin in counter clock wise direction and calculate current spin through delta angle

```
//spin logic
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
    //-------------------------------------------------------
    int currentTime = glutGet(GLUT_ELAPSED_TIME);
    int deltaTime = currentTime - prevTime;
    prevTime = currentTime;
    float deltaAngle = spinSpeed * (deltaTime / 1000.0);
    currentSpin += deltaAngle;
    glutPostRedisplay();

}
```

In case of spin display Reverse in which we make it to make the object spin in clock wise direction and calculate current spin through delta angle which delta angle be negative .

```
void spinDisplayReverse(void)
{
    /*
        write code below:
        1- change currentSpin according to spinSpeed (note: spinSpeed unit is dgree/second)
        2- mark window to be rerendered
    */
    // code here
    //-------------------------------------------------------------------
    int currentTime = glutGet(GLUT_ELAPSED_TIME);
    int deltaTime = currentTime - prevTime;
    prevTime = currentTime;
    float deltaAngle = -spinSpeed * (deltaTime / 1000.0);
    currentSpin += deltaAngle;
    glutPostRedisplay();

}
```

And here when user click on left click it call function glutIdle and the object start rotation in counter clock wise . And here when user click on right click it call function glutIdle and the object start rotation in clock wise.

```
142   //keyboard & mouse
143   void mouse(int button, int state, int x, int y)
144   {
145
146       switch (button)
147       {
148       case CCW:
149           /*
150           write code below:
151               1- assign spin logic to be invoked regularly (hint: glutIdleFunc)
152           */
153
154           //-----------------------------------------
155
156           spinSpeed = 5;
157           glutIdleFunc(spinDisplay);
158           break;
159       case CW:
160           /*
161           write code below:
162               1- assign reverse spin logic to be invoked regularly
163           */
164           // code here
165           //-----------------------------------------
166           spinSpeed = 5;
167           glutIdleFunc(spinDisplayReverse);
168           break;
169
170       default:
171           break;
172       }
```

And in case of zoom in we increase the value of zoffset to make the object zoomed in only the case of prepsective.

```
}
void keyInput(unsigned char key, int x, int y)
{
    switch (key)
    {
    case 27:
        exit(0);
        break;
    case zoomIn:{
        /*
        write code below:
            1- zoom in
            2- mark window for rerendering
        */
        // code here
        //---------------------------------------
         zOffset += 10;
         glLoadIdentity();
         glutPostRedisplay();
        break;
    }
```

And in case of zoom out we decrease the value of zoffset to make the object zoomed in only the case of prepsective.

And user when enter space the rotation stops.

```
193     }
194     case zoomOut:
195         /*
196         write code below:
197             1- zoom out
198             2- mark window for rerendering
199         */
200         // code here
201         //---------------------------------------
202
203         zOffset -= 10;
204         glutPostRedisplay();
205         break;
206     case stopSpinning:
207         /*
208         write code below:
209             1- stop spinning (hint: use NULL)
210         */
211         // code here
212         //---------------------------------------
213         spinSpeed = 0;
214         break;
215     default:
216         break;
217     }
```

and here initiate viewing for ortho by using glortho and put ortho variables in it

```
// OpenGL window reshape routine.
void resize(int w, int h)
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    switch (userChoice) {
    case ortho:
        /*
        write code below:
            1- initiate viewing box for parallel projection(use ortho variables (orthoLeft, orthoRight, ...))
        */
        // code here
        //-----------------------------------------------------------------
         glOrtho(orthoLeft , orthoRight, orthoBottom, orthoTop, orthoNear, orthoFar);
        break;
```

And here we set the values and initiate fru variables

And in function of printuserinteraction() : it is called at the first one time and as some instructions for the user

```
    case prespective:
        /*
        write code below:
            1- initiate frustum for perspective projection (use fru variables (fruLeft, fruRight, ...))
        */
        // code here
        //-----------------------------------------------------
        glFrustum(fruLeft, fruRight, fruBottom, fruTop, fruNear, fruFar);
        break;
    default:
        break;
    }
```

```
// user interaction
void printUserInteraction() {
    /*write code below:
     1- print user interaction(good practice)
    */
    // code here
    //---------------------------------------

    std::cout << "in order to rotate the scene in counter clock wise press left click \n";
    std::cout << "in order to rotate the scene in clock wise press right click \n";
    std::cout << "in order to stop rotation press space\n";
    std::cout << "in order to zoom in press i (in perspective only)\n";
    std::cout << "in order to zoom out press o (in perspective only)\n";
}
```

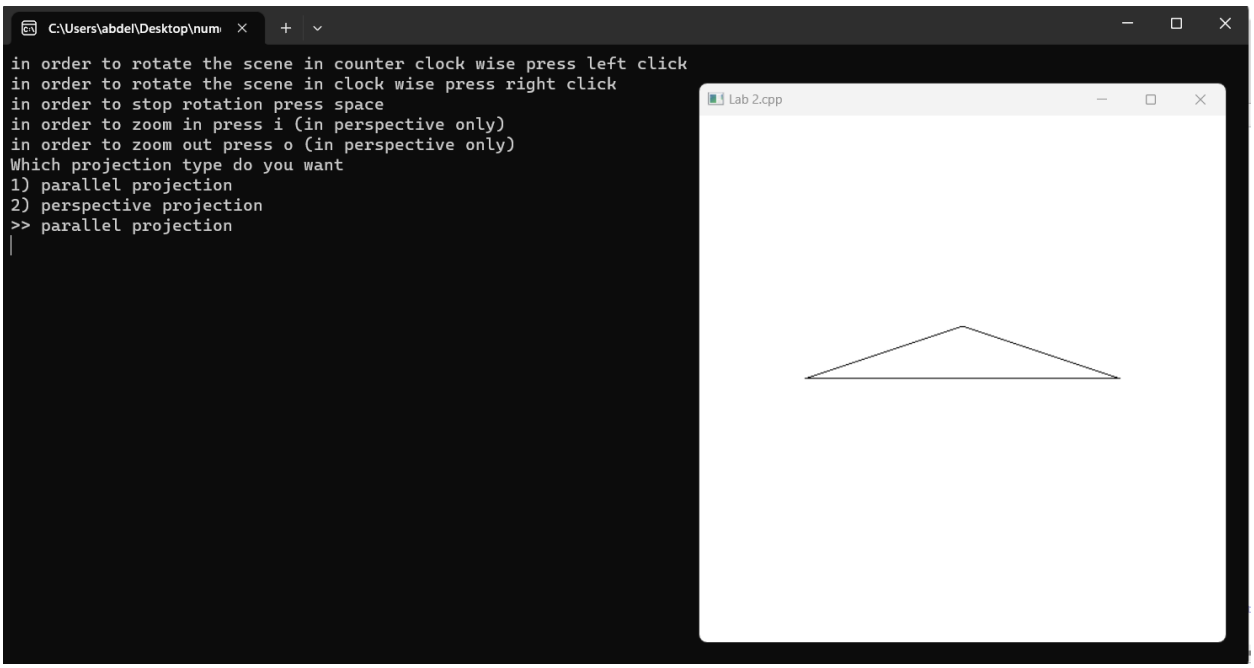
And here we take input from user and see if user choose parallel projection it set userchoice to 1 and spin z =1 (rotation around z) else if choose prepsective set user choice to 2 and spiny =1(rotation around y)

```
266   // Main routine.
267   int main(int argc, char** argv)
268   {
269       glutInit(&argc, argv);
270
271       glutInitContextVersion(4, 3);
272       glutInitContextProfile(GLUT_COMPATIBILITY_PROFILE);
273       printUserInteraction();
274       std::cout << "Which projection type do you want\n1) parallel projection\n2) perspective projection\n>> ";
275       /*
276       write code below:
277           1- accept input from user and assign the value to userChoice variable
278       */
279       // code here
280       //---------------------------------------------------------------------
281        std::string choice;
282        std::getline(std::cin, choice);
283        if (choice == "parallel projection")
284       {
285           spinZ =1;
286           userChoice  = 1;
287       }
288       else if (choice == "perspective projection")
289       {
290           spinY=1;
291           userChoice  = 2;
292       }
293
294       glutInitDisplayMode(GLUT SINGLE | GLUT RGBA);
```

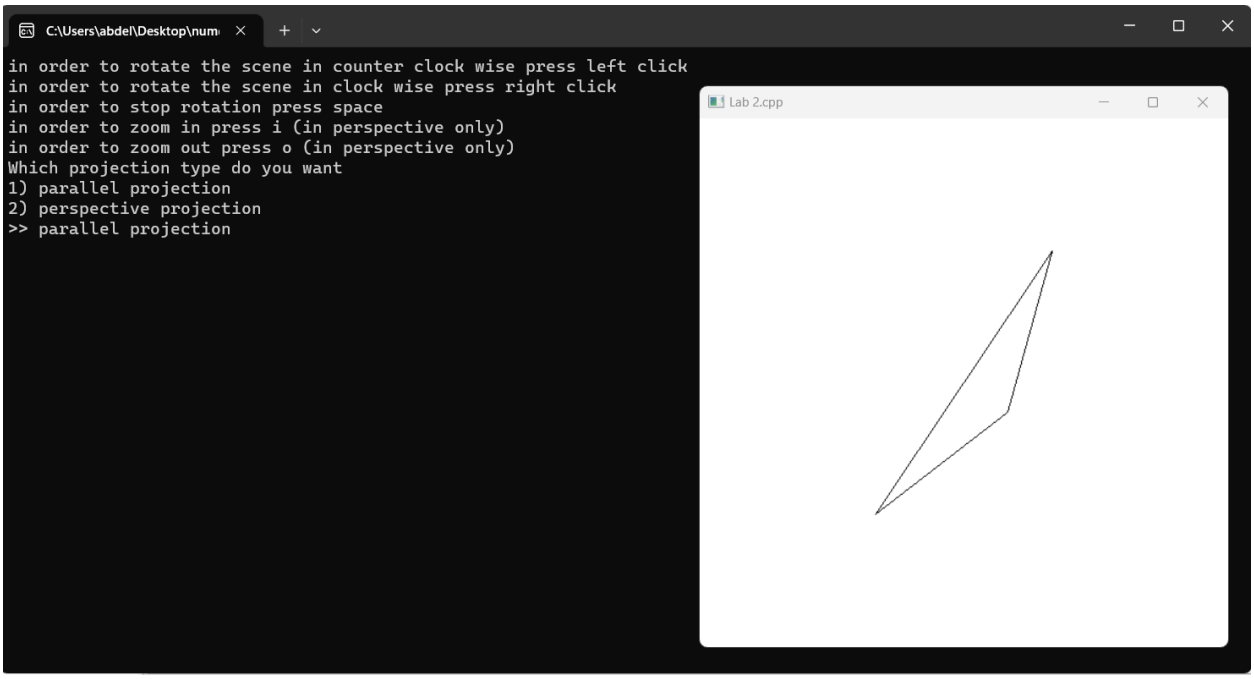
```
281        std::string choice;
282        std::getline(std::cin, choice);
283        if (choice == "parallel projection")
284       {
285           spinZ =1;
286           userChoice  = 1;
287       }
288       else if (choice == "perspective projection")
289       {
290           spinY=1;
291           userChoice  = 2;
292       }
293
294       glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
295       glutInitWindowSize(windowWidth, windowHeight);
296       glutInitWindowPosition(offsetX, offsetY);
297       glutCreateWindow("Lab 2.cpp");
298       glutDisplayFunc(drawScene);
299       glutReshapeFunc(resize);
300       glutKeyboardFunc(keyInput);
301       glutMouseFunc(mouse);
302       glewExperimental = GL_TRUE;
303       glewInit();
304
305       setup();
306
307       glutMainLoop();
308   }
309   //Note: At first user interaction, if scene spins unexpectadly you need to handle this behaviour (Hint: use prevTime variable)
310
```

Screenshots of sample run:

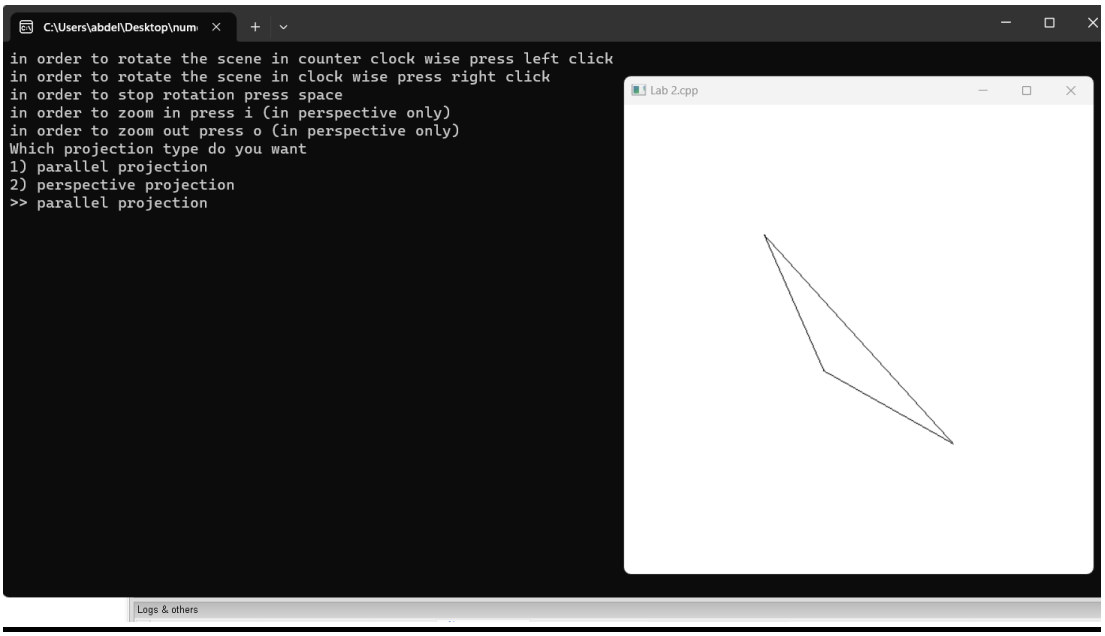
When user choose parallel projection it draws a triangle and the rotation will be on the z-axis.



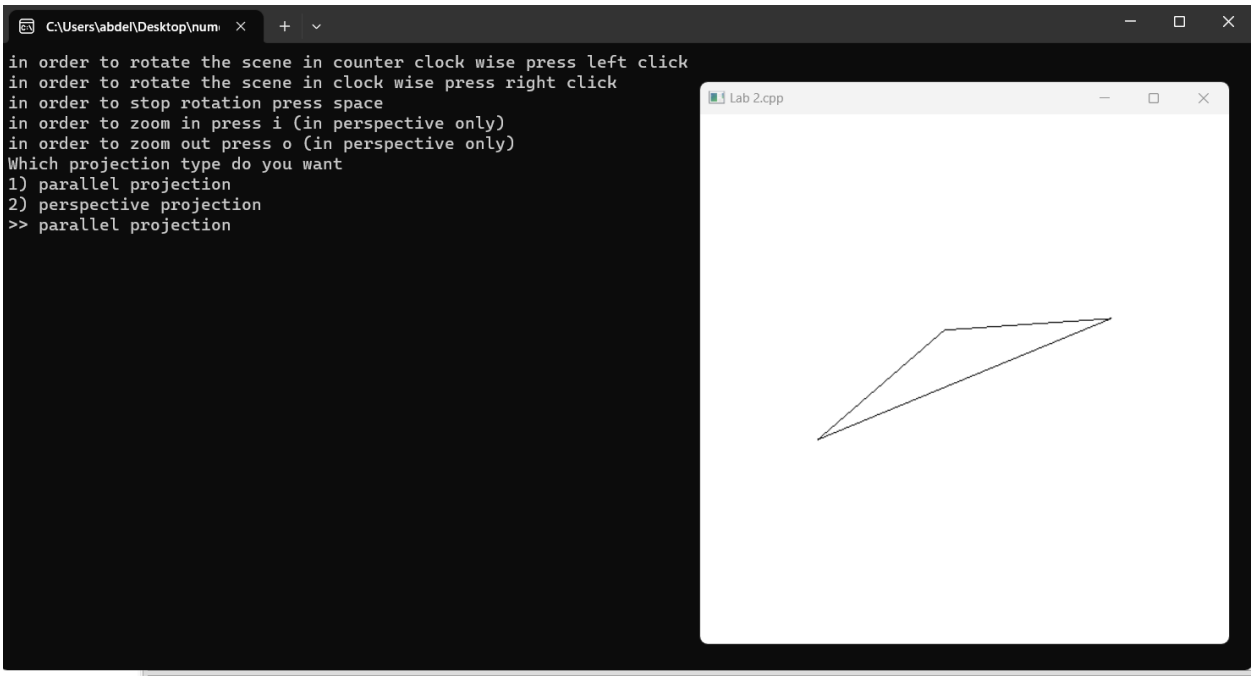
When user click left click it start rotate in counter clock wise in z-axis.



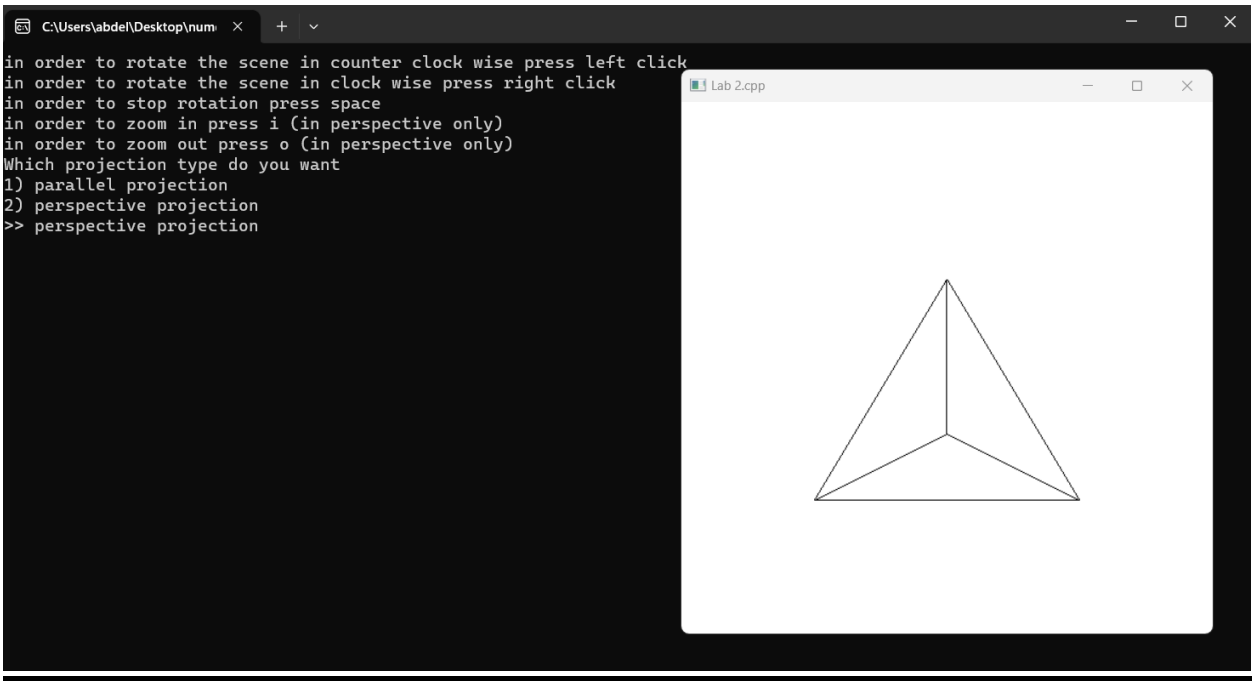
When user click on right click it start rotate in clock wise in z-axis.



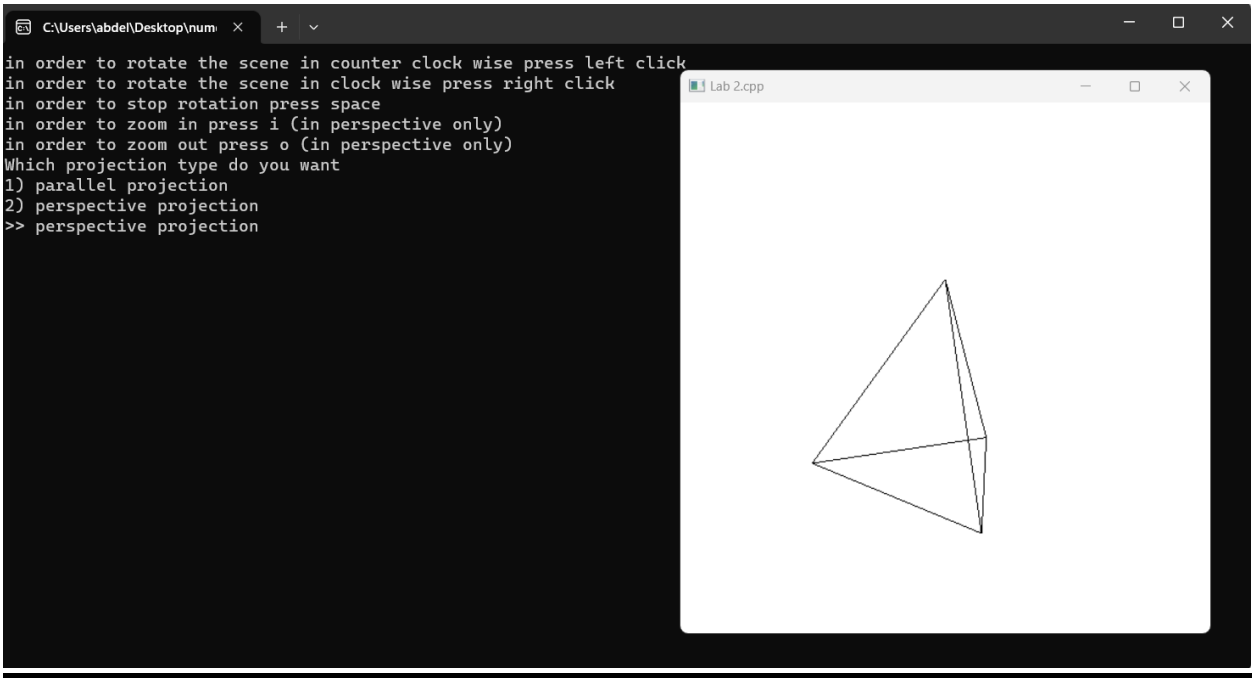
When user press on space it stop rotation .



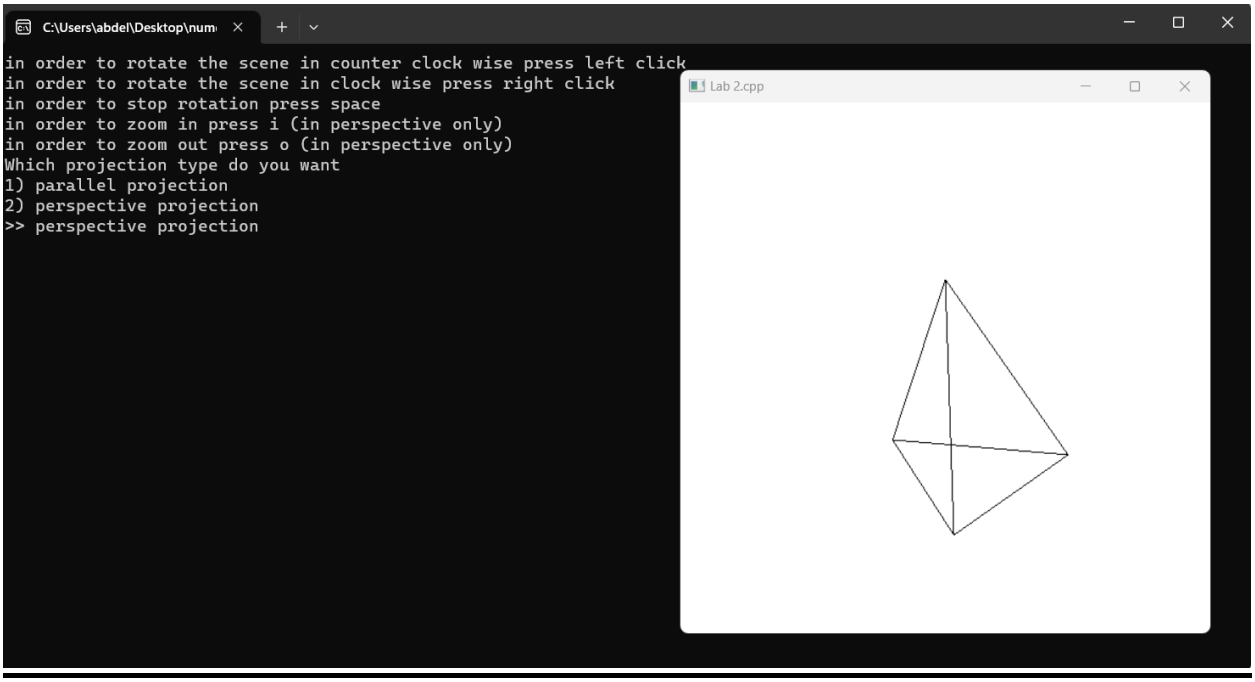
When user choose perspective projection it draws a pyramid and the rotation will be on the y-axis.



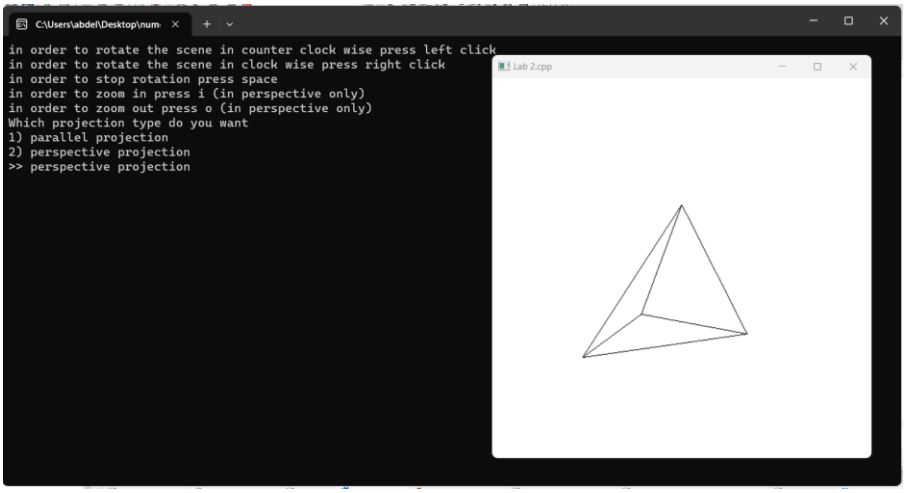
When user click right click it start rotate in clock wise in y-axis.



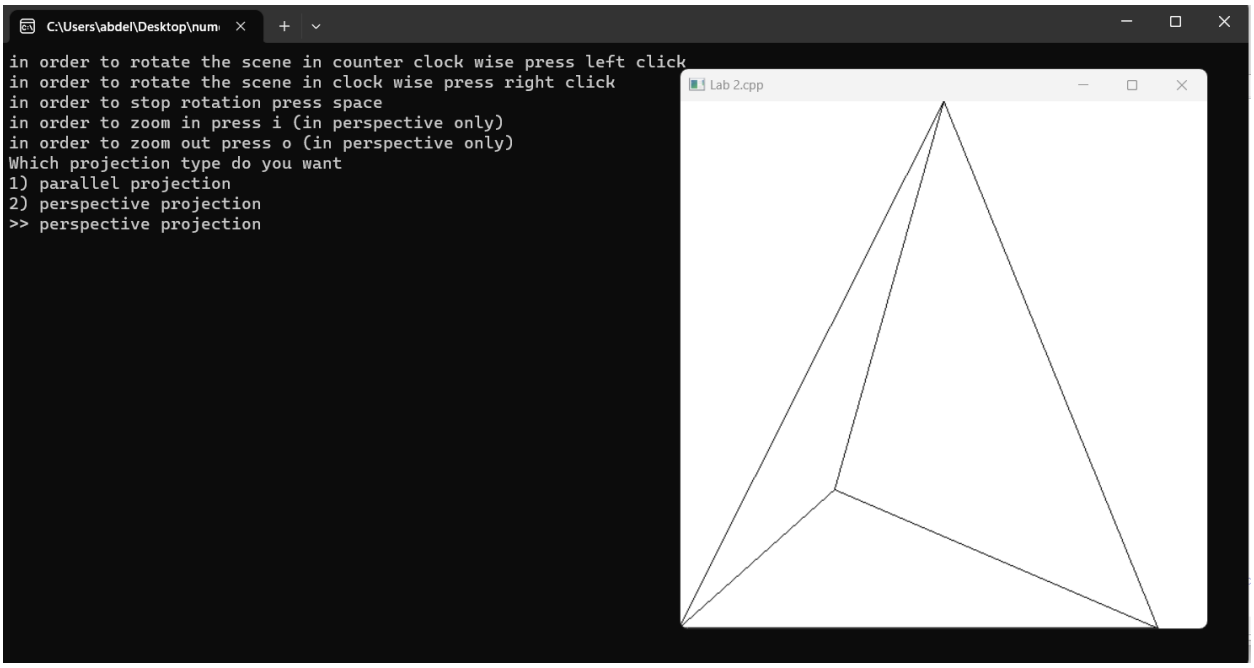
When user click left click it start rotate in counter clock wise in y-axis.



When user press on space it stop rotation .



when user press i it starts zoom in the pyramid.



when user press o it starts zoom out the pyramid.