

Alexandria University

Faculty of Engineering

Computer Graphics

Lab #3 OpenGL

Name: Toka Ashraf Abo Elwafa Ahmed

ID: 19015539

Problem Statement:

You are required to create an OpenGL project using the project template. You should implement an application that asks user to choose between two types of shapes (Helix and Sphere).

Circular helix parametric equation: $X(t) = X_c + R * \cos(t)$

$Y(t) = Y_c + R * \sin(t)$

$Z(t) = P * t$

Where X_c , Y_c center of the helix, R is radius of the helix and P is the pitch of helix (the height of one complete helix turn).

You should choose X_c , Y_c such that helix appears fully within application window. At runtime, Input handling should be as follows:

- In case of sphere:
 - o When user presses Q/q, increase/decrease number of latitudinal slices.
 - o When user presses P/p, increase/decrease number of longitudinal slices.
 - o When user presses W/w, draw sphere in wireframe / draw filled sphere.
- In case of helix:
 - o When user presses R/r, increase/decrease radius of the helix
 - o When user presses H/h, increase/decrease pitch of helix.
 - o When user presses N/n, increase/decrease number of vertices used to draw the helix.

Number of turns of helix that should be drawn is 5 turns.

Code Flow:

1- Function of drawScene():

- After defining the enum for choosing between helix and sphere and userChoice variable input that taken from user:
 - If user choice is helix, then draw helix using the three parametric equations by looping around the number of turns multiplied by Pi when initialization is minus the number of turns multiplied by Pi and draw the vertex.
 - In each vertex, it randomizes a color using function of rand().
 - If user choice is sphere, then call function of drawSphere() that draw first half of the sphere then negate sign of the Y dimension to draw the second half of the sphere.

```
void drawScene(void)
{
    glClear( mask: GL_COLOR_BUFFER_BIT);
    glLoadIdentity();
    glColor3f( red: 0.0, green: 0.0, blue: 0.0);
    // Command to push the sphere, which is drawn centered at the origin,
    // into the viewing frustum.
    glTranslatef( x: 0.0, y: 0.0, z: offset);
    // Commands to turn the sphere.
    glRotatef( angle: Zangle, x: 0.0, y: 0.0, z: 1.0);
    glRotatef( angle: Yangle, x: 0.0, y: 1.0, z: 0.0);
    glRotatef( angle: Xangle, x: 1.0, y: 0.0, z: 0.0);
    switch(userChoice){
        case Helix:
            glPolygonMode( face: GL_FRONT_AND_BACK, mode: GL_LINE);
            glBegin( mode: GL_LINE_STRIP);
            for(float t = -no_of_turns * M_PI; t <= no_of_turns * M_PI; t += dt){
                glColor3ub( red: rand()%255, green: rand()%255, blue: rand()%255);
                glVertex3f( x: R_helix * cos( X: t), y: R_helix * sin( X: t), z: t * pitch);
            }
            glEnd();
            glFlush();
            break;
        case Sphere:
            drawSphere();
            break;
    }
```

```

void drawSphere(){
    glPolygonMode( face: GL_FRONT_AND_BACK, mode: model);
    int i, j;
    ////first half of the sphere

    for (j = 0; j < q; j++)
    {
        // One latitudinal triangle strip.
        glBegin(mode);
        for (i = 0; i <= p; i++)
        {
            glColor3f( red: 1.0, green: 0.0, blue: 0.0);
            glVertex3f( x: R_sphere * cos( X: (float)(j + 1) / q * M_PI / 2.0) * cos( X: 2.0 * (float)i / p * M_PI),
                       y: R_sphere * sin( X: (float)(j + 1) / q * M_PI / 2.0),
                       z: -R_sphere * cos( X: (float)(j + 1) / q * M_PI / 2.0) * sin( X: 2.0 * (float)i / p * M_PI));
            glColor3f( red: 0.0, green: 50.0, blue: 0.0);
            glVertex3f( x: R_sphere * cos( X: (float)j / q * M_PI / 2.0) * cos( X: 2.0 * (float)i / p * M_PI),
                       y: R_sphere * sin( X: (float)j / q * M_PI / 2.0),
                       z: -R_sphere * cos( X: (float)j / q * M_PI / 2.0) * sin( X: 2.0 * (float)i / p * M_PI));
        }
        glEnd();
    }

    ////second half of the sphere
    for (j = 0; j < q; j++)
    {
        // One latitudinal triangle strip.
        glBegin(mode);
        for (i = 0; i <= p; i++)
        {
            glColor3f( red: 0.0, green: 0.0, blue: 1.0);
            glVertex3f( x: R_sphere * cos( X: (float)(j + 1) / q * M_PI / 2.0) * cos( X: 2.0 * (float)i / p * M_PI),
                       y: -R_sphere * sin( X: (float)(j + 1) / q * M_PI / 2.0),
                       z: -R_sphere * cos( X: (float)(j + 1) / q * M_PI / 2.0) * sin( X: 2.0 * (float)i / p * M_PI));
            glColor3f( red: 0.0, green: 1.0, blue: 1.0);
            glVertex3f( x: R_sphere * cos( X: (float)j / q * M_PI / 2.0) * cos( X: 2.0 * (float)i / p * M_PI),
                       y: -R_sphere * sin( X: (float)j / q * M_PI / 2.0),
                       z: -R_sphere * cos( X: (float)j / q * M_PI / 2.0) * sin( X: 2.0 * (float)i / p * M_PI));
        }
        glEnd();
    }
    glFlush();
}

```

2- Function of keyInput():

In case of Sphere:

- ✓ If user presses button of P, then it increases the number of longitudinal slices by one and in case of pressing p, then decrease this number by one.
- ✓ If user presses button of Q, then it increases the number of latitudinal slices by one and in case of pressing q, then decrease this number by one.

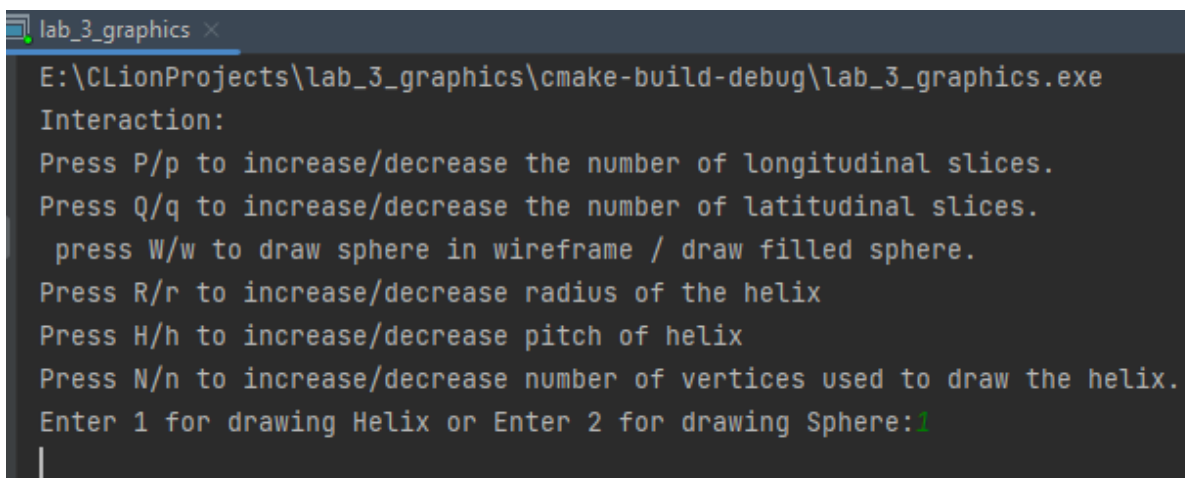
- ✓ If user presses button of W to draw sphere in wireframe, then mode of drawing shape is GL_LINE_LOOP and glPolygonMode remains GL_LINE to draw each wire separately.
And in case of pressing w, then mode of drawing shape remains GL_TRIANGLE_STRIP and glPolygonMode becomes GL_FILL to fill the sphere with certain color.

In case of Helix:

- ✓ If user presses button of R, then it increases the radius of helix by one and in case of r, decrease the radius by one.
- ✓ If user presses button of H, then increases the pitch by 0.1 and in case of h, it decreases by 0.1.
Variable of pitch is initialized with 0.3.
- ✓ If user presses button of N, then it increases the number of vertices by decreasing the variable of dt (it is the increasing rate in the loop) by 0.1 to it will draw more vertices. and in case of n, increasing the value to draw less vertices.

Test Cases:

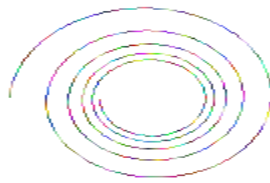
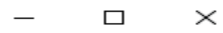
- After running the program, it will be required to enter number 1 in case of drawing the helix and enter 2 in case of sphere and store this value in variable of userChoice.




```
lab_3_graphics x
E:\CLionProjects\lab_3_graphics\cmake-build-debug\lab_3_graphics.exe
Interaction:
Press P/p to increase/decrease the number of longitudinal slices.
Press Q/q to increase/decrease the number of latitudinal slices.
press W/w to draw sphere in wireframe / draw filled sphere.
Press R/r to increase/decrease radius of the helix
Press H/h to increase/decrease pitch of helix
Press N/n to increase/decrease number of vertices used to draw the helix.
Enter 1 for drawing Helix or Enter 2 for drawing Sphere: 1
|
```

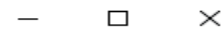
- After pressing 1, this will be shown:

 helix and sphere.cpp

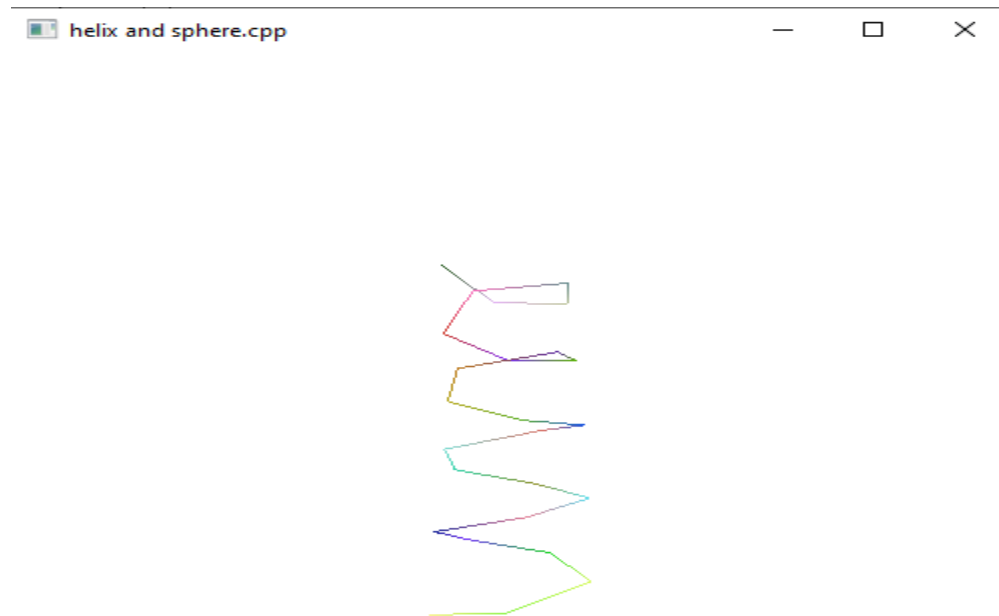


After making rotation in x direction:

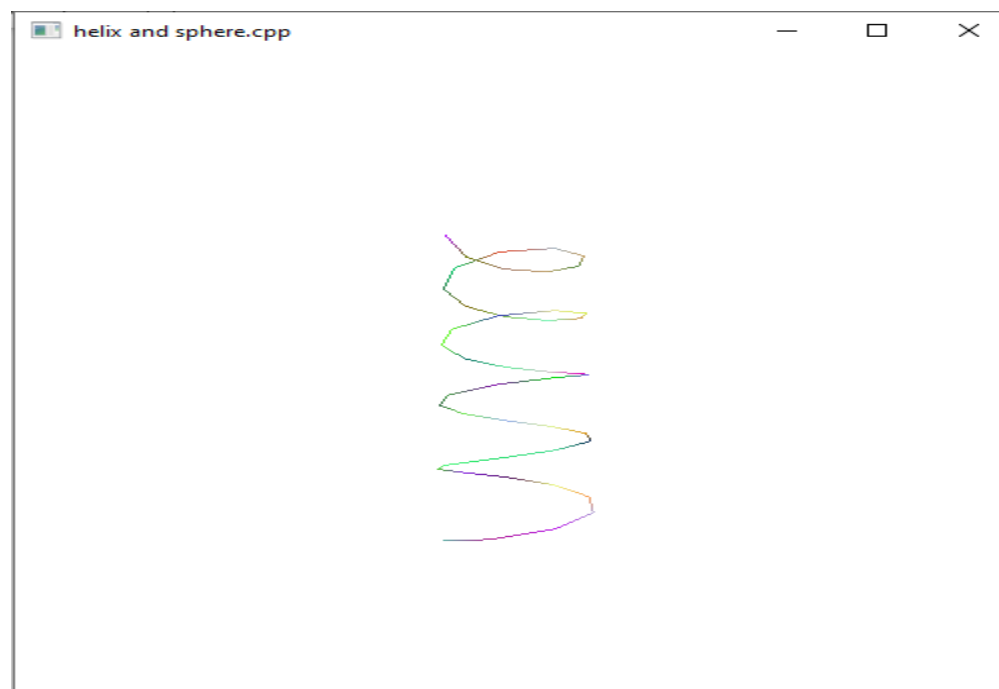
 helix and sphere.cpp



- After decreasing number of vertices:



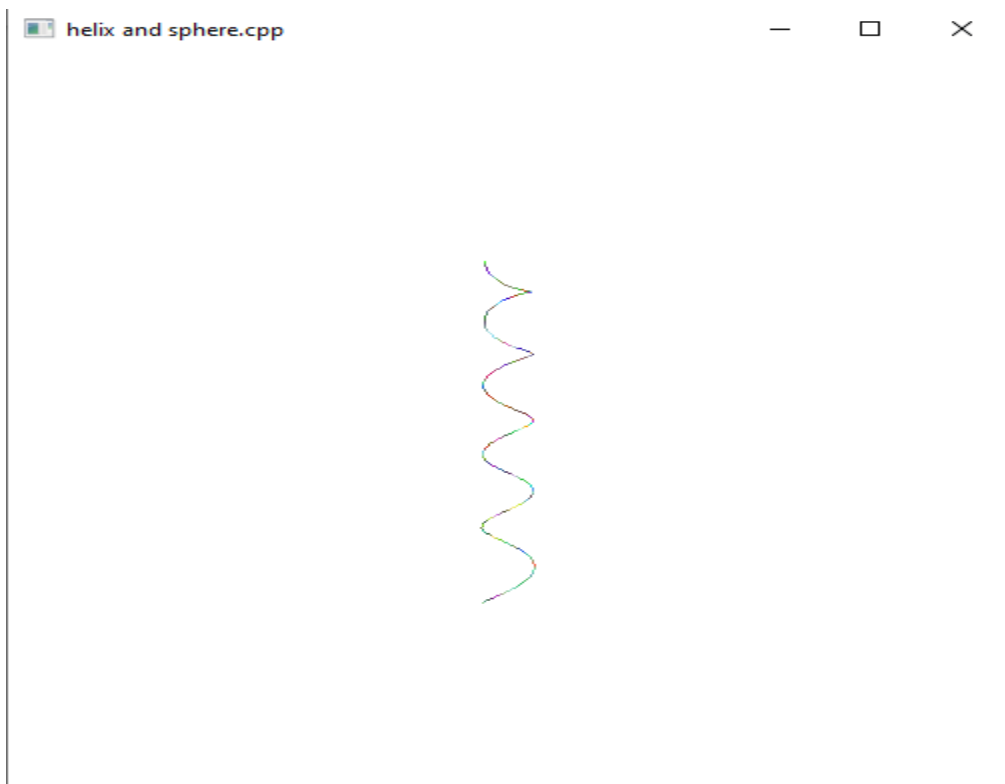
After increasing it again:



- After increasing the radius:



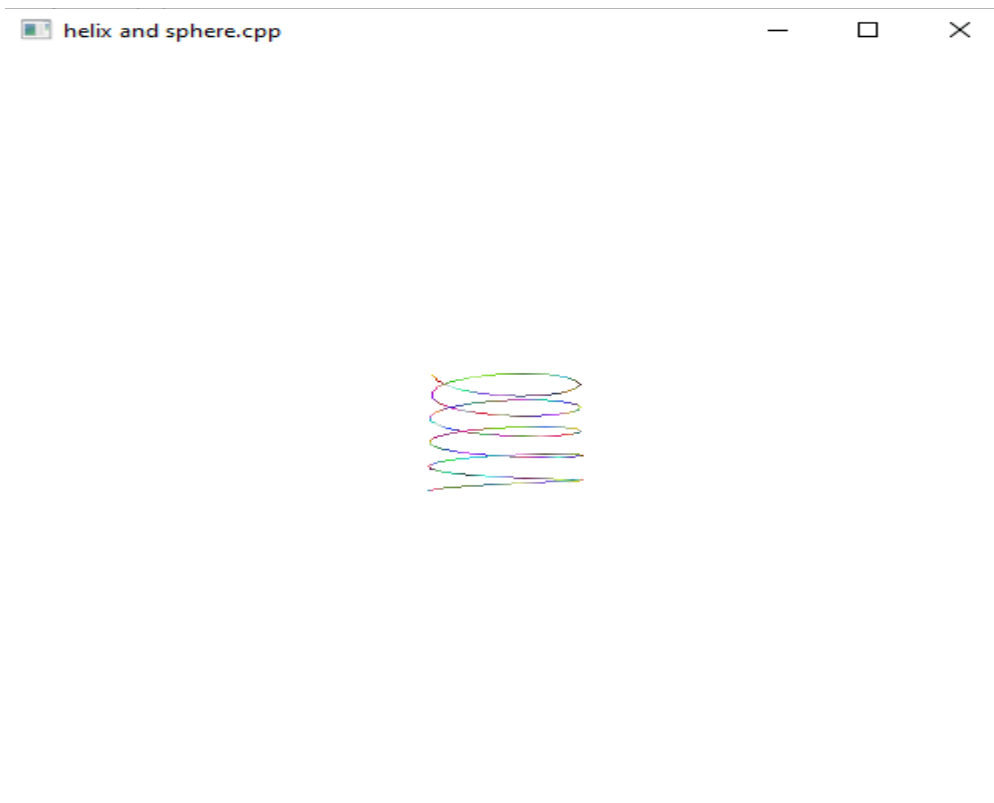
After decreasing it:



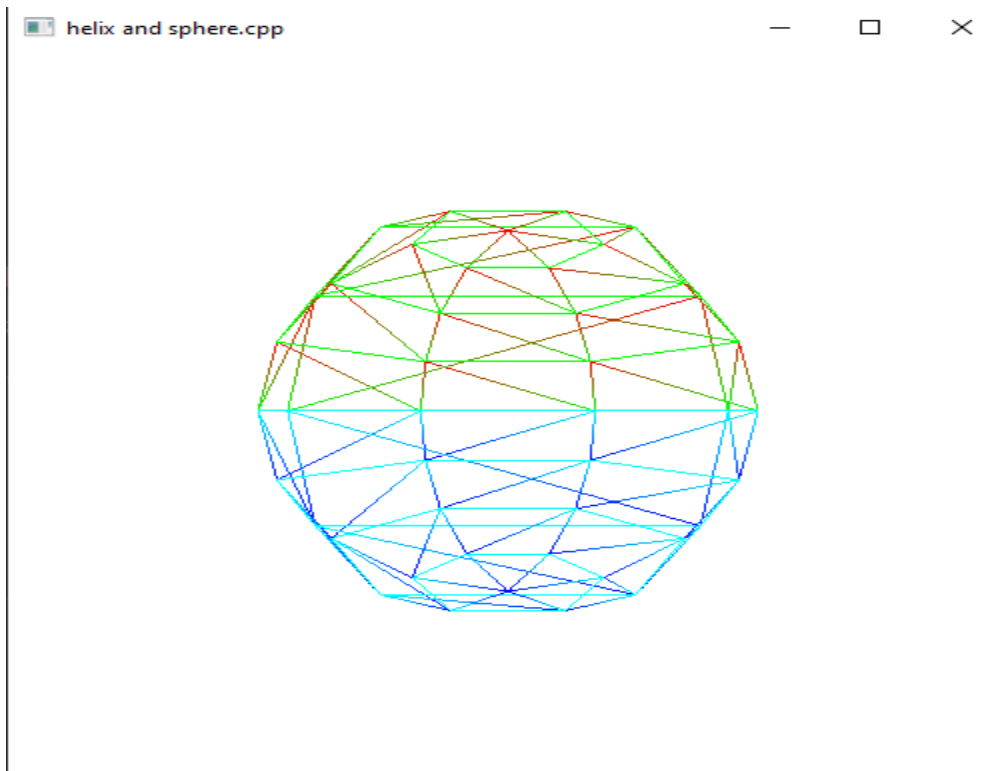
- After increasing the pitch:



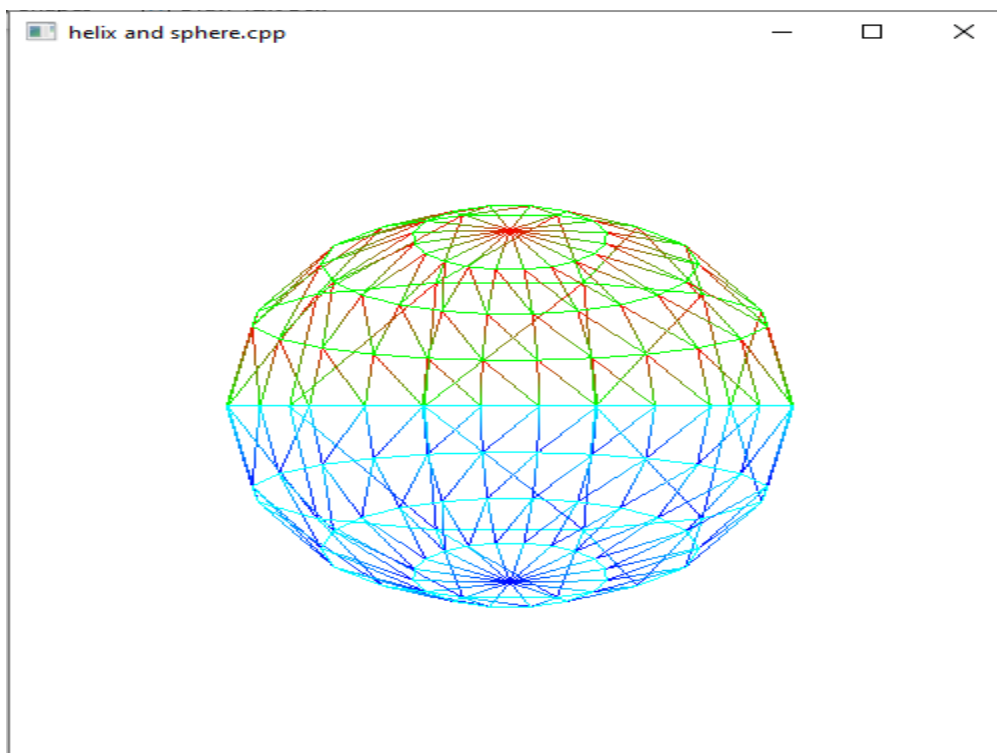
After decreasing it:



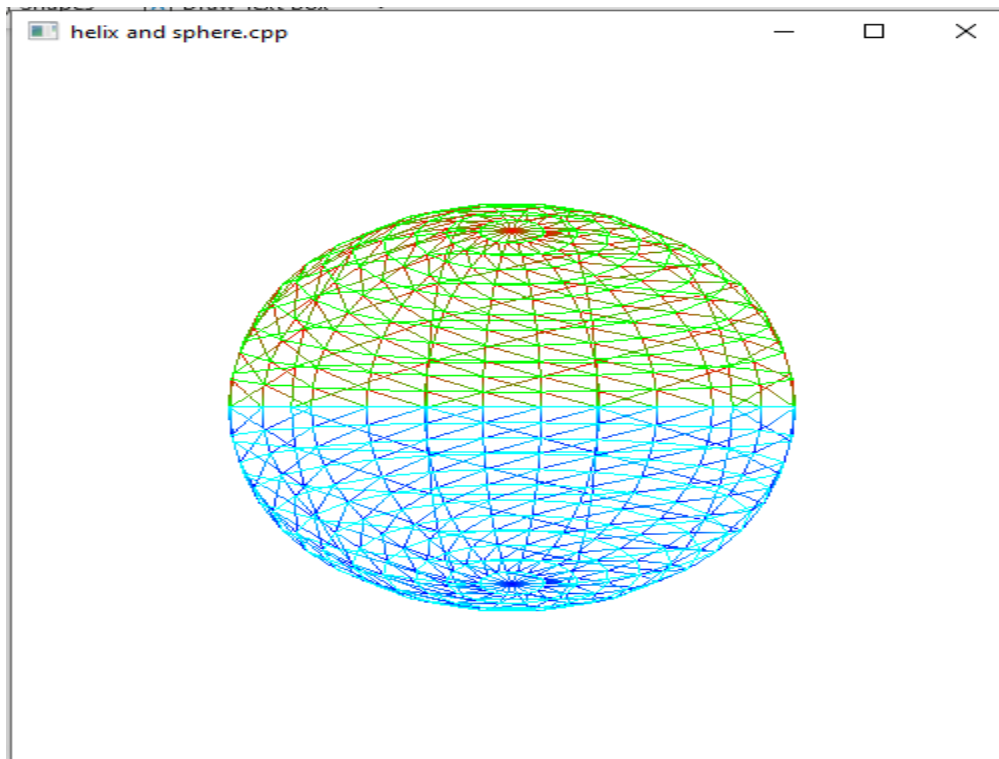
- After pressing 2, this will be shown:



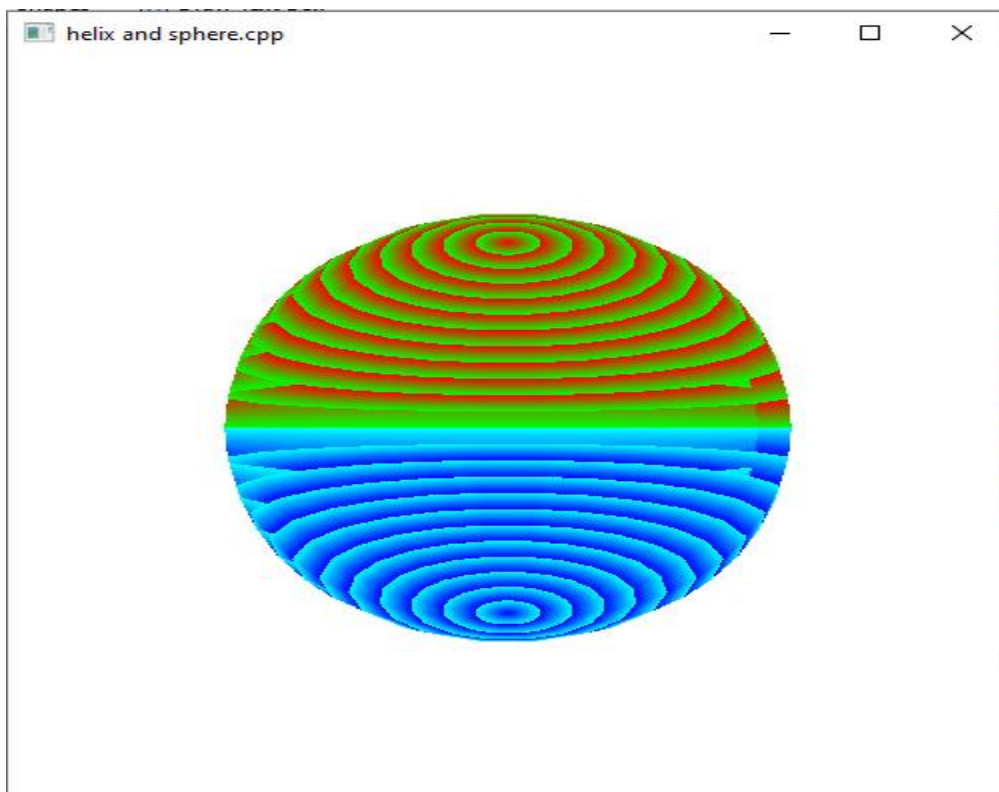
- After increasing the number of longitudinal slices:



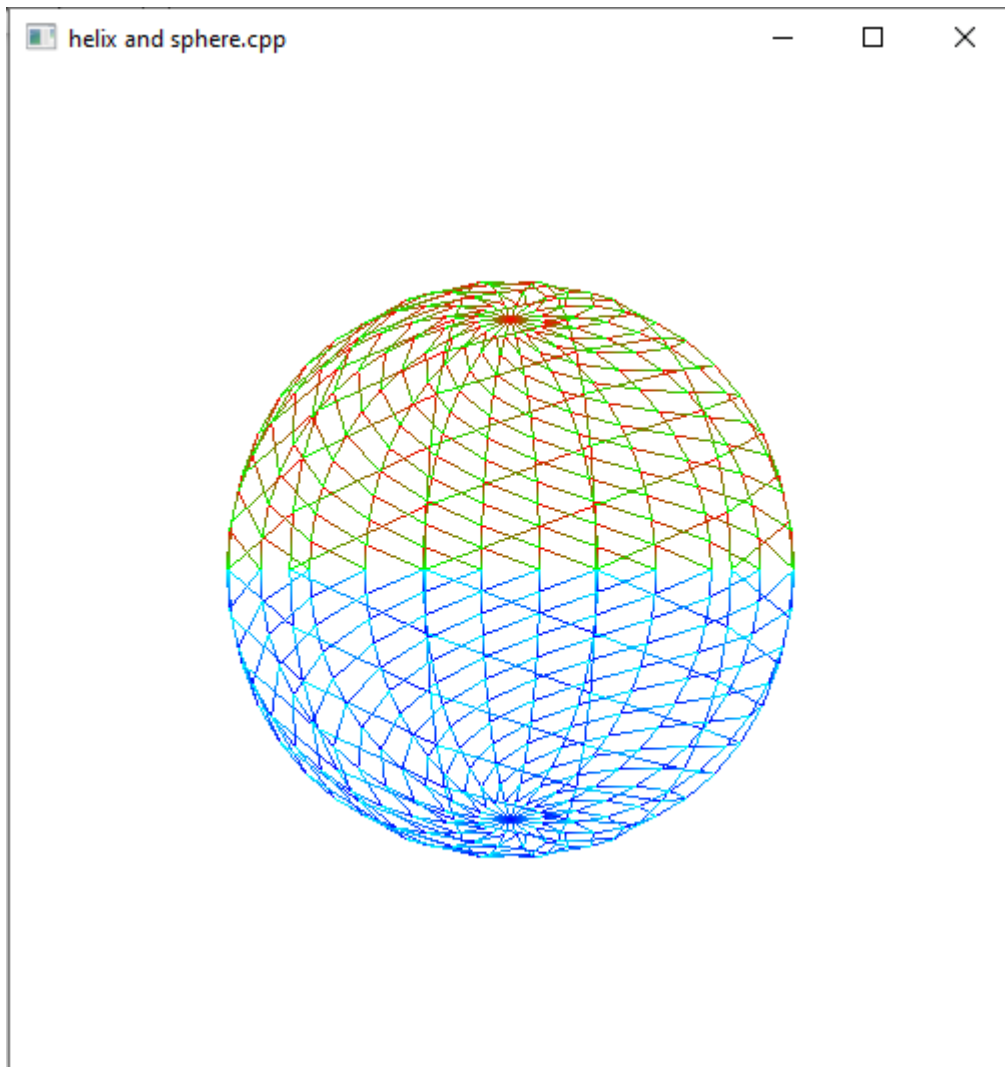
- After increasing the number of latitudinal slices:



- After pressing w to draw filled sphere:



- After pressing W to draw wireframe:



-