COMPUTER GRAPHICS

LAB#3

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

Number of turns of the helix that should be drawn is 5 turns. Use GL_FRONT_AND_BACK option for drawing mode. Also, for each vertex drawn you, should pick random color to draw vertex with. You should modify hemisphere code showed in the lab to full fill requirements.

Code Description

Sphere

The only modification added to the sphere main loop was simply repeating the inner loop but with an inverted Y value.

```
glBegin(GL_TRIANGLE_STRIP);
for (i = 0; i <= p; i++)
{
    glColor3f((float)rand() / RAND_MAX, (float)rand() / RAND_MAX, (float)rand() / RAND_MAX);
    glVertex3f(R * cos((float)(j + 1) / q * M_PI / 2.0) * cos(2.0 * (float)i / p * M_PI),
        R * sin((float)(j + 1) / q * M_PI / 2.0),
        -R * cos((float)(j + 1) / q * M_PI / 2.0) * sin(2.0 * (float)i / p * M_PI));
    glColor3f((float)rand() / RAND_MAX, (float)rand() / RAND_MAX, (float)rand() / RAND_MAX);
    glVertex3f(R * cos((float)j / q * M_PI / 2.0) * cos(2.0 * (float)i / p * M_PI),
        R * sin((float)j / q * M_PI / 2.0),
        -R * cos((float)j / q * M_PI / 2.0) * sin(2.0 * (float)i / p * M_PI));

}
glEnd();

glBegin(GL_TRIANGLE_STRIP);
for (i = 0; i <= p; i++)
{
    glColor3f((float)rand() / RAND_MAX, (float)rand() / RAND_MAX, (float)rand() / RAND_MAX);
    glVertex3f(R * cos((float)(j + 1) / q * M_PI / 2.0) * cos(2.0 * (float)i / p * M_PI),
        -R * sin((float)(j + 1) / q * M_PI / 2.0),
        -R * cos((float)(j + 1) / q * M_PI / 2.0) * sin(2.0 * (float)i / p * M_PI));
    glColor3f((float)rand() / RAND_MAX, (float)rand() / RAND_MAX, (float)rand() / RAND_MAX);
    glVertex3f(R * cos((float)j / q * M_PI / 2.0) * cos(2.0 * (float)i / p * M_PI),
        -R * sin((float)j / q * M_PI / 2.0),
        -R * cos((float)j / q * M_PI / 2.0) * sin(2.0 * (float)i / p * M_PI));

}
glEnd();
```

Additionally, each point is drawn with a randomly generated color.

Helix

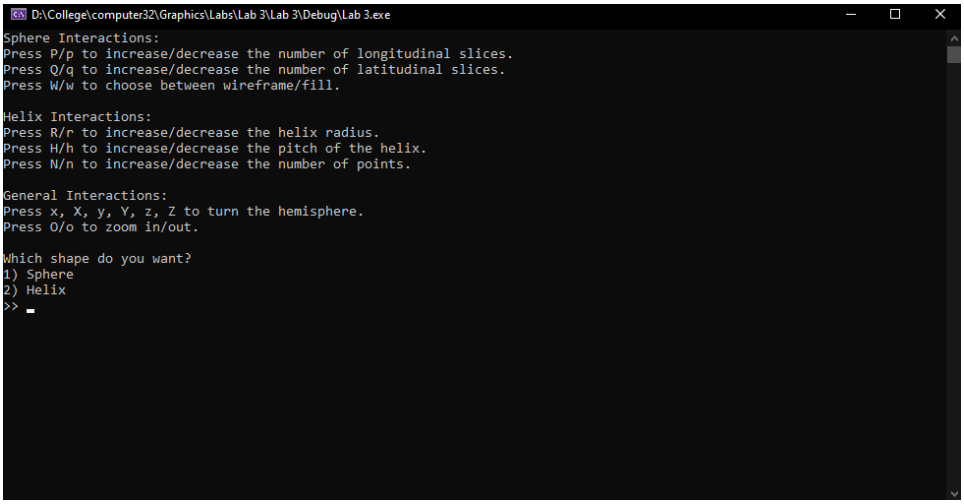
```
case helix:
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_LINE_STRIP);
    for (float t = -5 * M_PI; t <= 5 * M_PI; t += M_PI / n) {
        glColor3f((float)rand() / RAND_MAX, (float)rand() / RAND_MAX, (float)rand() / RAND_MAX);
        glVertex3f(r * cos(t), r * sin(t), h * t - 20.0);
    }
    glEnd();

    break;
```

For the helix implementation, we simply implemented a loop that plots points using the equation mentioned in the lab file.

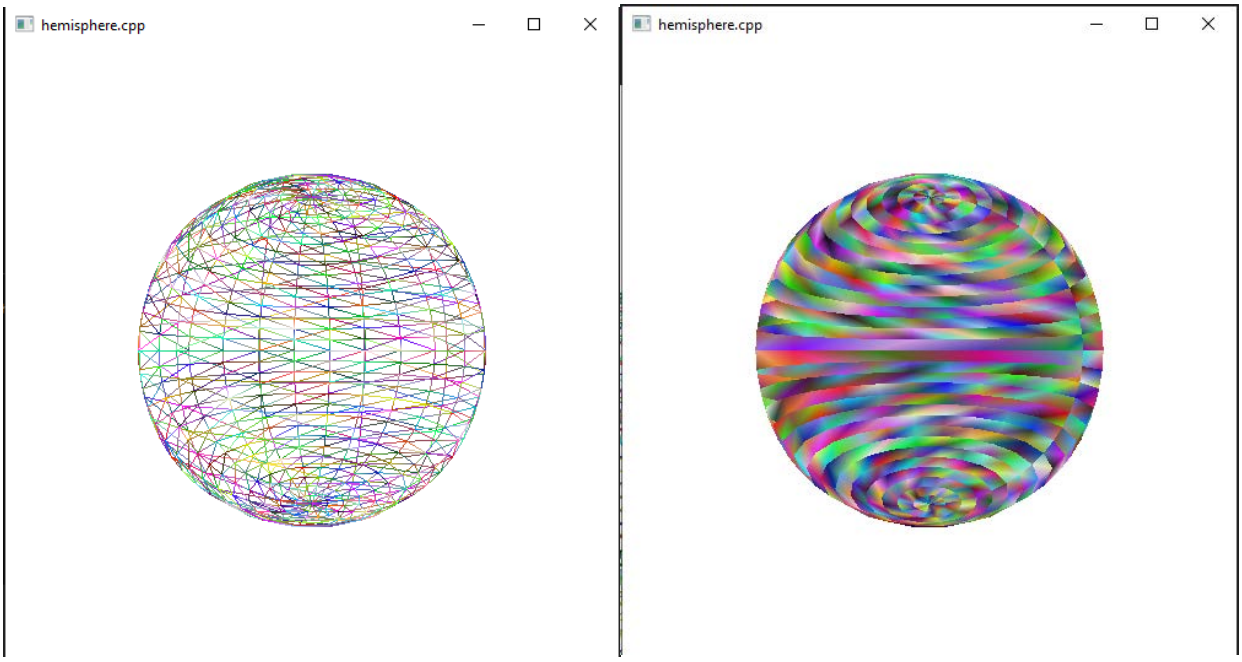
Note: you might need to zoom in/out to fully see the helix.

Example Of Running Code

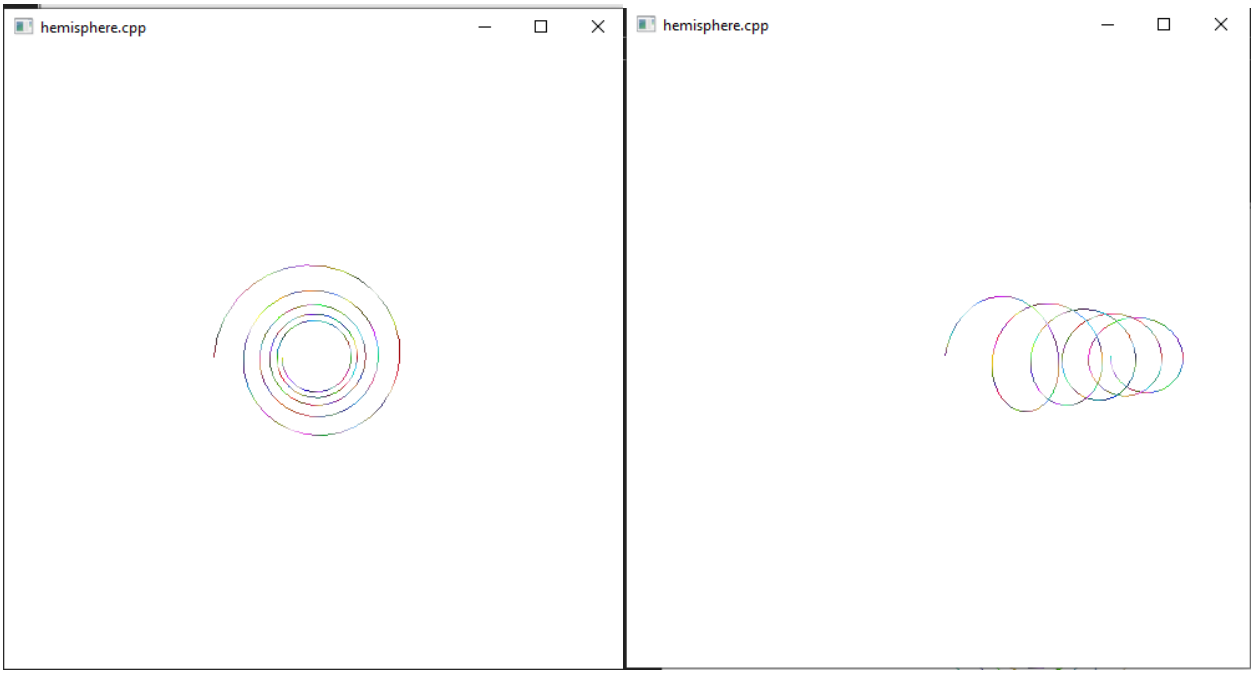
Main Window



Sphere



HELIX



CHALLENGES

The main challenge was implementing the helix itself and finding the initial values of the variables.