Barber_Pole

```
#include <gl\glut.h>
#include <math.h>

// For animating the rotation of the objects
float rotation = 0.0;
GLfloat PI = 3.14;
// variables to move outermost Object Frame (to move all the rendered environment)
float moveX = 0.0f;
float moveY = 0.0f;
float moveZ = 0.0f;

GLfloat rotX = 0.0f;
GLfloat rotY = 0.0f;
GLfloat rotZ = 0.0f;

//variables to move the camera
float camX = 0.0f;
float camY = 0.0f;
float camZ = 0.0f;


float x = 30;
float z = -30;




void init() {
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    glEnable(GL_DEPTH_TEST);
}

void drawAxes() {

    glBegin(GL_LINES);

    glLineWidth(1.5);

    glColor3f(1.0, 0.0, 0.0);
    glVertex3f(-20, 0, 0);
    glVertex3f(20, 0, 0);

    glColor3f(0.0, 1.0, 0.0);
    glVertex3f(0, -20, 0);
    glVertex3f(0, 20, 0);

    glColor3f(0.0, 0.0, 1.0);
    glVertex3f(0, 0, -20);
    glVertex3f(0, 0, 20);

    glEnd();
}
void DrawGrid() {
    GLfloat ext = 20.0f;
```

```
    GLfloat step = 1.0f;
    GLfloat yGrid = -0.4f;
    GLint line;

    glBegin(GL_LINES);
    for (line = -ext; line <= ext; line += step) {
        glVertex3f(line, yGrid, ext);
        glVertex3f(line, yGrid, -ext);

        glVertex3f(ext, yGrid, line);
        glVertex3f(-ext, yGrid, line);
    }
    glEnd();
}


void setLighting() {
    glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_TRUE);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);

    //setting lighting intensity and color
    GLfloat qaAmbientLight0[] = { 0.1, 0.1, 0.1, 1.0 };
    GLfloat qaDiffuseLight0[] = { 0.8, 0.8, 0.8, 1.0 };
    GLfloat qaSpecularLight0[] = { 1.0, 1.0, 1.0, 1.0 };
    glLightfv(GL_LIGHT0, GL_AMBIENT, qaAmbientLight0);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, qaDiffuseLight0);
    glLightfv(GL_LIGHT0, GL_SPECULAR, qaSpecularLight0);

    // Set the light position
    GLfloat qaLightPosition0[] = { 0.0, 1.0, -.5, 1.0 };
    glLightfv(GL_LIGHT0, GL_POSITION, qaLightPosition0);


    glEnable(GL_LIGHT1);
    //setting lighting intensity and color
    GLfloat qaAmbientLight1[] = { 0.5, 0.5, 0.5, 1.0 };
    GLfloat qaDiffuseLight1[] = { 0.4, 0.4, 0.4, 1.0 };
    GLfloat qaSpecularLight1[] = { 1.0, 1.0, 1.0, 1.0 };
    glLightfv(GL_LIGHT1, GL_AMBIENT, qaAmbientLight1);
    glLightfv(GL_LIGHT1, GL_DIFFUSE, qaDiffuseLight1);
    glLightfv(GL_LIGHT1, GL_SPECULAR, qaSpecularLight1);

    // Set the light position
    GLfloat qaLightPosition1[] = { 2.0, 1.0, 1.0, 1.0 };
    glLightfv(GL_LIGHT1, GL_POSITION, qaLightPosition1);


    glEnable(GL_LIGHT2);
    //setting lighting intensity and color
    GLfloat qaAmbientLight2[] = { 0.2, 0.2, 0.2, 1.0 };
    GLfloat qaDiffuseLight2[] = { 0.3, 0.3, 0.3, 1.0 };
    GLfloat qaSpecularLight2[] = { 1.0, 1.0, 1.0, 1.0 };
    glLightfv(GL_LIGHT2, GL_AMBIENT, qaAmbientLight2);
    glLightfv(GL_LIGHT2, GL_DIFFUSE, qaDiffuseLight2);
    glLightfv(GL_LIGHT2, GL_SPECULAR, qaSpecularLight2);

    // Set the light position
```

```
    GLfloat qaLightPosition2[] = { -5.0, 2.0, 1.0, 1.0 };
    glLightfv(GL_LIGHT2, GL_POSITION, qaLightPosition2);




}



void drawHelixStrip(GLfloat radius, GLfloat y_start, GLfloat width, GLfloat height)
{
    GLfloat angle = 0;
    GLint n = 64;
    GLfloat slope = 0.03;

    while (y_start <= height) {

        glBegin(GL_QUAD_STRIP);

        for (int i = 0; i <= n; i++) {
            angle = 2 * PI * (static_cast<float>(i) / n);
            double x = radius * cos(angle);
            double z = radius * sin(angle);
            glVertex3f(x, y_start, z);
            glVertex3f(x, y_start + width, z);

            y_start += slope;
        }

        glVertex3f(radius, y_start, 0);
        glVertex3f(radius, y_start + width, 0);

        glEnd();
    }
}

void drawCylinder(GLfloat radius, GLfloat height) {
    GLfloat angle = 0.0;
    GLint n = 64;
    GLfloat y = 0.0;

    glBegin(GL_QUAD_STRIP);
    glColor3f(1.0, 1.0, 1.0);

    for (int i = 0; i <= n; i++) {
        angle = 2 * PI * (static_cast<float>(i) / n);
        double x = radius * cos(angle);
        double z = radius * sin(angle);
        glVertex3f(x, y, z);
        glVertex3f(x, y + height, z);
    }

    glVertex3f(radius, 0, 0);
    glVertex3f(radius, height, 0);

    glEnd();
```

```
}


void drawObject() {
    GLfloat qaLightBlue[] = { 0.0, 0.74, 1.0, 1.0 };

    GLfloat qaWhite[] = { 1.0, 1.0, 1.0, 1.0 };
    GLfloat qaAsh[] = { 0.75, 0.75, 0.75, 1.0 };
    GLfloat qaRed[] = { 1.0, 0.0, 0.0, 1.0 };
    GLfloat qaBlue[] = { 0.0, 0.0, 1.0, 1.0 };
    GLfloat qaLowAmbient[] = { 0.2, 0.2, 0.2, 1.0 };
    GLfloat qaFullAmbient[] = { 1.0, 1.0, 1.0, 1.0 };

    for (int iIndex = 0; iIndex < 2; ++iIndex) {
        if (iIndex == 0) {
            glShadeModel(GL_FLAT);
        }
        else {
            glShadeModel(GL_SMOOTH);
        }

        //Set, ambient, diffuse and specular lighting. Set ambient to 20%.
        glMaterialfv(GL_FRONT, GL_AMBIENT, qaWhite);
        glMaterialfv(GL_FRONT, GL_DIFFUSE, qaWhite);
        glMaterialfv(GL_FRONT, GL_SPECULAR, qaWhite);
        glMaterialf(GL_FRONT, GL_SHININESS, 50.0);


        // Draw a sphere
         //top sphere
        glPushMatrix();
        //glColor3f(1.0, 1.0, 1.0);
        glTranslated(0.0, 8.0, 0.0);
        glutSolidSphere(1.8f, 60, 60);
        glPopMatrix();



        //bottom sphere
        glPushMatrix();
        glColor3f(1.0, 1.0, 1.0);
        glutSolidSphere(1.8f, 60, 60);
        glPopMatrix();


        glMaterialfv(GL_FRONT, GL_AMBIENT, qaAsh);
        glMaterialfv(GL_FRONT, GL_DIFFUSE, qaAsh);
        glMaterialfv(GL_FRONT, GL_SPECULAR, qaWhite);
        glMaterialf(GL_FRONT, GL_SHININESS, 50.0);

        //bottom torus
        glPushMatrix();
        glRotated(-90, 1.0, 0.0, 0.0);
        glTranslated(0.0, 0.0, 0.85);
        glColor3f(0.6, 0.6, 0.6);
        glutSolidTorus(0.2, 1.8, 10, 50);
        glPopMatrix();
```

```
        //top torus
        glPushMatrix();
        glTranslated(0.0, 7.1, 0.0);
        glRotated(-90, 1.0, 0.0, 0.0);
        glColor3f(0.6, 0.6, 0.6);
        glutSolidTorus(0.15, 1.73, 10, 50);
        glPopMatrix();



        glMaterialfv(GL_FRONT, GL_AMBIENT, qaWhite);
        glMaterialfv(GL_FRONT, GL_DIFFUSE, qaWhite);
        glMaterialfv(GL_FRONT, GL_SPECULAR, qaWhite);
        glMaterialf(GL_FRONT, GL_SHININESS, 50.0);

        glPushMatrix();
        drawCylinder(1.5, 8.0);
        glPopMatrix();

        glMaterialfv(GL_FRONT, GL_AMBIENT, qaRed);
        glMaterialfv(GL_FRONT, GL_DIFFUSE, qaRed);
        glMaterialfv(GL_FRONT, GL_SPECULAR, qaWhite);
        glMaterialf(GL_FRONT, GL_SHININESS, 50.0);
        glPushMatrix();
        glRotated(rotation, 0.0, 1.0, 0.0);
        glTranslated(0.0, -0.5, 0.0);
        glColor3f(1.0, 0.0, 0.0);
        drawHelixStrip(1.55, 0.0, 0.5, 7.5);
        glPopMatrix();


        glMaterialfv(GL_FRONT, GL_AMBIENT, qaBlue);
        glMaterialfv(GL_FRONT, GL_DIFFUSE, qaBlue);
        glMaterialfv(GL_FRONT, GL_SPECULAR, qaWhite);
        glMaterialf(GL_FRONT, GL_SHININESS, 50.0);
        glPushMatrix();
        glRotated(rotation, 0.0, 1.0, 0.0);
        glTranslated(0.0, -0.5, 0.0);
        glColor3f(0.0, 0.0, 1.0);
        drawHelixStrip(1.55, 0.95, 0.5, 7.9);
        glPopMatrix();
    }

}


void display() {

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    //Setting the modelview matrix to the identity matrix
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    glPushMatrix();
```

```
    gluLookAt(0.0 + camX, 1.0 + camY, 10.0 + camZ, 0, 0, 0, 0, 1.0, 0);
    glColor3f(1.0, 1.0, 1.0);


    glTranslatef(moveX, moveY, moveZ);
    glRotatef(rotX, 1.0f, 0.0f, 0.0f);
    glRotatef(rotY, 0.0f, 1.0f, 0.0f);
    glRotatef(rotZ, 0.0f, 0.0f, 1.0f);

    DrawGrid();

    drawAxes();

    setLighting();


    glPushMatrix();
    drawObject();

    glPopMatrix();

    glutSwapBuffers();

}

void keyboardSpecial(int key, int x, int y) {
    if (key == GLUT_KEY_UP)
        camY += 0.5;
    if (key == GLUT_KEY_DOWN)
        camY -= 0.5;

    glutPostRedisplay();
}

void keyboard(unsigned char key, int x, int y) {

    if (key == 'z')
        moveZ += 1;

    if (key == 'Z')
        moveZ -= 1;

    if (key == 'l')
        rotY -= 5.0;

    if (key == 'r')
        rotY += 1.0;

    glutPostRedisplay();

}

void Timer(int x) {
    rotation += rotation >= 360.0 ? -rotation : 2;
    glutPostRedisplay();

    glutTimerFunc(60, Timer, 1);
}
```

```
void reshape(GLsizei w, GLsizei h) {
    glViewport(0, 0, w, h);
    GLfloat aspect_ratio = h == 0 ? w / 1 : (GLfloat)w / (GLfloat)h;

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    //Define the Perspective projection frustum
    // (FOV_in_vertical, aspect_ratio, z-distance to the near plane from the camera
position, z-distance to far plane from the camera position)
    gluPerspective(120.0, aspect_ratio, 1.0, 100.0);

}

int main(int argc, char** argv) {

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_DEPTH | GLUT_RGBA);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(150, 150);
    glutCreateWindow("Helix");
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);

    // keyboard function activation
    glutKeyboardFunc(keyboard);
    glutSpecialFunc(keyboardSpecial);

    glutTimerFunc(60.0, Timer, 1);
    init();
    glutMainLoop();


    return 0;
}
```

Snow Man

```
#include <GL/glut.h>
#include <math.h>

// For animating the rotation of the objects
float teapotRotation = 0.0;

// variables to move outermost Object Frame (to move all the rendered environment)
float moveX = 0.0f;
float moveY = 0.0f;
float moveZ = 0.0f;

GLfloat rotX = 0.0f;
GLfloat rotY = 0.0f;
GLfloat rotZ = 0.0f;

//variables to move the camera
float camX = 0.0f;
float camY = 0.0f;
float camZ = 0.0f;

void init() {
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    glEnable(GL_DEPTH_TEST);
}

void drawAxes() {

    glBegin(GL_LINES);

    glLineWidth(1.5);

    glColor3f(1.0, 0.0, 0.0);
    glVertex3f(-20, 0, 0);
    glVertex3f(20, 0, 0);

    glColor3f(0.0, 1.0, 0.0);
    glVertex3f(0, -20, 0);
    glVertex3f(0, 20, 0);

    glColor3f(0.0, 0.0, 1.0);
    glVertex3f(0, 0, -20);
    glVertex3f(0, 0, 20);

    glEnd();
}
void DrawGrid() {
    GLfloat ext = 20.0f;
    GLfloat step = 1.0f;
    GLfloat yGrid = -0.4f;
    GLint line;

    glBegin(GL_LINES);
    for (line = -ext; line <= ext; line += step) {
        glVertex3f(line, yGrid, ext);
```

```
        glVertex3f(line, yGrid, -ext);

        glVertex3f(ext, yGrid, line);
        glVertex3f(-ext, yGrid, line);
    }
    glEnd();
}

void setLighting() {
    glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_TRUE);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);

    //setting lighting intensity and color
    GLfloat qaAmbientLight0[] = { 0.1, 0.1, 0.1, 1.0 };
    GLfloat qaDiffuseLight0[] = { 0.8, 0.8, 0.8, 1.0 };
    GLfloat qaSpecularLight0[] = { 1.0, 1.0, 1.0, 1.0 };
    glLightfv(GL_LIGHT0, GL_AMBIENT, qaAmbientLight0);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, qaDiffuseLight0);
    glLightfv(GL_LIGHT0, GL_SPECULAR, qaSpecularLight0);

    // Set the light position
    GLfloat qaLightPosition0[] = { 0.0, 1.0, -.5, 0.0 };
    glLightfv(GL_LIGHT0, GL_POSITION, qaLightPosition0);


    glEnable(GL_LIGHT1);
    //setting lighting intensity and color
    GLfloat paleYellow[] = { 1.0, 1.0, 0.75,1.0 };
    GLfloat qaAmbientLight1[] = { 0.1, 0.1, 0.1, 1.0 };
    GLfloat qaDiffuseLight1[] = { 0.8, 0.8, 0.8, 1.0 };
    GLfloat qaSpecularLight1[] = { 1.0, 1.0, 1.0, 1.0 };
    glLightfv(GL_LIGHT1, GL_AMBIENT, paleYellow);
    glLightfv(GL_LIGHT1, GL_DIFFUSE, qaDiffuseLight1);
    glLightfv(GL_LIGHT1, GL_SPECULAR, qaSpecularLight1);

    // Set the light position
    GLfloat qaLightPosition1[] = { 2.0, 1.0, 1.0, 1.0 };
    glLightfv(GL_LIGHT1, GL_POSITION, qaLightPosition1);

    glEnable(GL_LIGHT2);
    //setting lighting intensity and color
    GLfloat qaAmbientLight2[] = { 0.2, 0.2, 0.2, 1.0 };
    GLfloat qaDiffuseLight2[] = { 0.8, 0.8, 0.8, 1.0 };
    GLfloat qaSpecularLight2[] = { 1.0, 1.0, 1.0, 1.0 };
    glLightfv(GL_LIGHT2, GL_AMBIENT, qaAmbientLight2);
    glLightfv(GL_LIGHT2, GL_DIFFUSE, qaDiffuseLight2);
    glLightfv(GL_LIGHT2, GL_SPECULAR, qaSpecularLight2);

    // Set the light position
    GLfloat qaLightPosition2[] = { -5.0, 2.0, 1.0, 1.0 };
    glLightfv(GL_LIGHT2, GL_POSITION, qaLightPosition2);



}
```

```
void drawSnowman() {

    GLfloat qaLightBlue[] = { 0.0, 0.74, 1.0, 1.0 };

    GLfloat qaWhite[] = { 1.0, 1.0, 1.0, 1.0 };
    GLfloat qaOrange[] = { 1.0, 0.4, 0.0, 1.0 };
    GLfloat qaBlack[] = { 0.0, 0.0, 0.0, 1.0 };
    GLfloat qaLowAmbient[] = { 0.2, 0.2, 0.2, 1.0 };
    GLfloat qaFullAmbient[] = { 1.0, 1.0, 1.0, 1.0 };

    for (int iIndex = 0; iIndex < 2; ++iIndex) {
        if (iIndex == 0) {
            glShadeModel(GL_FLAT);
        }
        else {
            glShadeModel(GL_SMOOTH);
        }

        glMaterialfv(GL_FRONT, GL_AMBIENT, qaWhite);
        glMaterialfv(GL_FRONT, GL_DIFFUSE, qaWhite);
        glMaterialfv(GL_FRONT, GL_SPECULAR, qaWhite);
        glMaterialf(GL_FRONT, GL_SHININESS, 40.0);
        glPushMatrix();
        glTranslatef(0.0, 2.3, 0.0);
        glColor3f(1.0, 1.0, 1.0);
        glutSolidSphere(0.8, 20, 20);

        //left eye
        glMaterialfv(GL_FRONT, GL_AMBIENT, qaBlack);
        glMaterialfv(GL_FRONT, GL_DIFFUSE, qaBlack);
        glMaterialfv(GL_FRONT, GL_SPECULAR, qaWhite);
        glMaterialf(GL_FRONT, GL_SHININESS, 40.0);
        glPushMatrix();
        glTranslatef(-0.5, 0.2, 0.5);
        glColor3f(0.0, 0.0, 0.0);
        glutSolidSphere(0.1, 20, 20);
        glPopMatrix();

        //right eye
        glPushMatrix();
        glTranslatef(0.5, 0.2, 0.5);
        glColor3f(0.0, 0.0, 0.0);
        glutSolidSphere(0.1, 20, 20);
        glPopMatrix();

        glMaterialfv(GL_FRONT, GL_AMBIENT, qaOrange);
        glMaterialfv(GL_FRONT, GL_DIFFUSE, qaOrange);
        glMaterialfv(GL_FRONT, GL_SPECULAR, qaOrange);
        glMaterialf(GL_FRONT, GL_SHININESS, 40.0);
        glPushMatrix();
        glTranslatef(0, 0.1, 0.5);
        glColor3f(1.0, 0.4, 0.0);
        glutSolidCone(0.2, 1.0, 20, 20);
```

```
        glPopMatrix();
        glPopMatrix();

        glMaterialfv(GL_FRONT, GL_AMBIENT, qaWhite);
        glMaterialfv(GL_FRONT, GL_DIFFUSE, qaWhite);
        glMaterialfv(GL_FRONT, GL_SPECULAR, qaWhite);
        glMaterialf(GL_FRONT, GL_SHININESS, 20.0);
        glPushMatrix();
        glColor3f(1.0, 1.0, 1.0);
        glutSolidSphere(2.0, 20, 20);
        glPopMatrix();


    }

}


void multiSnowman() {

    float z = -30;
    for (int i = 0; i < 3; i++) {
        float x = -30;
        for (int j = 0; j < 3; j++) {
            glPushMatrix();
            glScalef(0.3, 0.3, 0.3);
            glTranslatef(x, 0.0, z);
            drawSnowman();
            glPopMatrix();
            x += 30;
        }
        z += 30;
    }

}



void display() {

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    //Setting the modelview matrix to the identity matrix
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    glPushMatrix();

    gluLookAt(3.0 + camX, 3.0 + camY, 5.0 + camZ, 0, 0, 0, 0, 1.0, 0);
    glColor3f(1.0, 1.0, 1.0);
    //glTranslatef(0.0, 0.0, -5.0);

    glTranslatef(moveX, moveY, moveZ);
    glRotatef(rotX, 1.0f, 0.0f, 0.0f);
    glRotatef(rotY, 0.0f, 1.0f, 0.0f);
    glRotatef(rotZ, 0.0f, 0.0f, 1.0f);

    DrawGrid();
```

```
    drawAxes();
    setLighting();

    glPushMatrix();
    glRotatef(teapotRotation, 0.0, 1.0, 0.0);
    drawSnowman();
    glPopMatrix();

    /*
    glPushMatrix();
    multiSnowman();
    glPopMatrix();
    */
    glPopMatrix();

    glutSwapBuffers();

}

void keyboardSpecial(int key, int x, int y) {
    if (key == GLUT_KEY_UP)
        camY += 0.5;
    if (key == GLUT_KEY_DOWN)
        camY -= 0.5;
    if (key == GLUT_KEY_LEFT)
        camZ += 0.5;
    if (key == GLUT_KEY_RIGHT)
        camZ -= 0.5;


    glutPostRedisplay();
}

void keyboard(unsigned char key, int x, int y) {

    if (key == 'z')
        moveZ += 1;

    if (key == 'Z')
        moveZ -= 1;

    if (key == 'x')
        moveX += 1;

    if (key == 'X')
        moveX -= 1;

    if (key == 'y')
        moveY += 1;

    if (key == 'Y')
        moveY -= 1;

    if (key == 'l')
        rotY -= 5.0;

    if (key == 'r')
```

```
        rotY += 5.0;



    glutPostRedisplay();

}

//Rotating teapot
void Timer(int x) {
    teapotRotation += teapotRotation >= 360.0 ? -teapotRotation : 2;
    glutPostRedisplay();

    glutTimerFunc(60, Timer, 1);
}

void reshape(GLsizei w, GLsizei h) {
    glViewport(0, 0, w, h);
    GLfloat aspect_ratio = h == 0 ? w / 1 : (GLfloat)w / (GLfloat)h;

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    //Define the Perspective projection frustum
    // (FOV_in_vertical, aspect_ratio, z-distance to the near plane from the camera
position, z-distance to far plane from the camera position)
    gluPerspective(120.0, aspect_ratio, 1.0, 100.0);

}

int main(int argc, char** argv) {

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_DEPTH | GLUT_RGBA);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(150, 150);
    glutCreateWindow("Rotating Teapot");
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);

    // keyboard function activation
    glutKeyboardFunc(keyboard);
    glutSpecialFunc(keyboardSpecial);

    glutTimerFunc(60.0, Timer, 1);
    init();
    glutMainLoop();


    return 0;
}
```