Lab Sheet No : 03
Index No : 19APP3936
Name : W.H.S.N.Rathnaweera
Date : 03.02.2025

Q1.

```
#ifdef __APPLE_CC__
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#endif
#include <math.h>
#include <stdbool.h>

float angle = 0.0f;
int windowWidth = 800, windowHeight = 600;
bool isHovered = false;
bool clipEnabled = false;
float aspectRatio = 1.0f;
GLdouble clipPlane[] = { 1.0, 0.0, 0.0, -0.5 };

void resize(int w, int h) {
    if (h == 0) h = 1;
    aspectRatio = (float)w / (float)h;
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if (aspectRatio >= 1.0f) {
        gluPerspective(45.0f, aspectRatio, 0.1f, 100.0f);
    } else {
        gluPerspective(45.0f / aspectRatio, aspectRatio, 0.1f, 100.0f);
    }
    glMatrixMode(GL_MODELVIEW);
    windowWidth = w;
    windowHeight = h;
}

void drawTorus(float innerRadius, float outerRadius, int numSides, int numRings) {
    for (int i = 0; i < numRings; i++) {
        float ringAngle = 2.0f * M_PI * i / numRings;
        float nextRingAngle = 2.0f * M_PI * (i + 1) / numRings;
```

```
        glBegin(GL_QUAD_STRIP);
        for (int j = 0; j <= numSides; j++) {
            float sideAngle = 2.0f * M_PI * j / numSides;
            float x1 = (outerRadius + innerRadius * cos(sideAngle)) * cos(ringAngle);
            float y1 = (outerRadius + innerRadius * cos(sideAngle)) * sin(ringAngle);
            float z1 = innerRadius * sin(sideAngle);
            float x2 = (outerRadius + innerRadius * cos(sideAngle)) * cos(nextRingAngle);
            float y2 = (outerRadius + innerRadius * cos(sideAngle)) * sin(nextRingAngle);
            float z2 = innerRadius * sin(sideAngle);

            glPushMatrix();
            glScalef(1.0f / aspectRatio, 1.0f, 1.0f);



            if (isHovered)
                glColor3f(0.0f, 0.0f, 1.0f);
            else
                glColor3f((cos(sideAngle) + 1) / 2, (sin(sideAngle) + 1) / 2, (cos(ringAngle) + 1) / 2);

            glVertex3f(x1, y1, z1);
            glVertex3f(x2, y2, z2);
            glPopMatrix();
        }
        glEnd();
    }
}

void display() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glTranslatef(0.0f, 0.0f, -5.0f);
    glPushMatrix();
    glRotatef(angle, 1.0f, 1.0f, 0.0f);

    if (clipEnabled) {
        glEnable(GL_CLIP_PLANE0);
        glClipPlane(GL_CLIP_PLANE0, clipPlane);
    } else {
        glDisable(GL_CLIP_PLANE0);
    }

    drawTorus(0.5f, 1.5f, 50, 50);
```

```
    glPopMatrix();
    glutSwapBuffers();
}

void update(int value) {
    angle += 2.0f;
    if (angle > 360) angle -= 360;
    glutPostRedisplay();
    glutTimerFunc(16, update, 0);
}

void highlightOnHover(int x, int y) {
    float nx = (float)x / windowWidth * 2.0f - 1.0f;
    float ny = -(float)y / windowHeight * 2.0f + 1.0f;

    if (sqrt(nx * nx + ny * ny) < 0.5) {
        isHovered = true;
    } else {
        isHovered = false;
    }
}

void mouseMotion(int x, int y) {
    highlightOnHover(x, y);
}

void keyboard(unsigned char key, int x, int y) {
    if (key == 'c' || key == 'C') {
        clipEnabled = !clipEnabled;
    }
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(windowWidth, windowHeight);
    glutCreateWindow("Rotating Torus");
    glEnable(GL_DEPTH_TEST);
    glutDisplayFunc(display);
    glutReshapeFunc(resize);
    glutMotionFunc(mouseMotion);
    glutKeyboardFunc(keyboard);
```

```
    glutTimerFunc(25, update, 0);
    glutMainLoop();
    return 0;
}
```