Trần Hà Duy – 22IT054 – Lab 05

**Code**

#include <GL/glut.h>

#include <cmath>

// Function to draw a cylinder

void drawCylinder(float radius, float height, int slices) {

float angleStep = 2.0f \* M\_PI / slices;

glBegin(GL\_QUAD\_STRIP);

for (int i = 0; i <= slices; ++i) {

float angle = i \* angleStep;

float x = cos(angle) \* radius;

float z = sin(angle) \* radius;

glVertex3f(x, 0.0f, z);

glVertex3f(x, height, z);

}

glEnd();

}

// Function to draw a cone

void drawCone(float radius, float height, int slices) {

float angleStep = 2.0f \* M\_PI / slices;

// Base of the cone

glBegin(GL\_TRIANGLE\_FAN);

glVertex3f(0.0f, 0.0f, 0.0f);

for (int i = 0; i <= slices; ++i) {

float angle = i \* angleStep;

float x = cos(angle) \* radius;

float z = sin(angle) \* radius;

glVertex3f(x, 0.0f, z);

}

glEnd();

// Sides of the cone

glBegin(GL\_TRIANGLE\_FAN);

glVertex3f(0.0f, height, 0.0f); // Cone's apex

for (int i = 0; i <= slices; ++i) {

float angle = i \* angleStep;

float x = cos(angle) \* radius;

float z = sin(angle) \* radius;

glVertex3f(x, 0.0f, z);

}

glEnd();

}

// Function to draw a sphere

void drawSphere(float radius, int slices, int stacks) {

glutSolidSphere(radius, slices, stacks);

}

// Function to draw an ellipsoid

void drawEllipsoid(float rx, float ry, float rz, int slices, int stacks) {

for (int i = 0; i <= stacks; ++i) {

float phi = M\_PI \* i / stacks;

glBegin(GL\_TRIANGLE\_STRIP);

for (int j = 0; j <= slices; ++j) {

float theta = 2.0f \* M\_PI \* j / slices;

float x = rx \* sin(phi) \* cos(theta);

float y = ry \* cos(phi);

float z = rz \* sin(phi) \* sin(theta);

glVertex3f(x, y, z);

}

glEnd();

}

}

// Function to draw a toroid

void drawTorus(float innerRadius, float outerRadius, int slices, int stacks) {

glutSolidTorus(innerRadius, outerRadius, slices, stacks);

}

// Function to draw a Bezier curve

void drawBezierCurve(float controlPoints[4][3]) {

glMap1f(GL\_MAP1\_VERTEX\_3, 0.0, 1.0, 3, 4, &controlPoints[0][0]);

glEnable(GL\_MAP1\_VERTEX\_3);

glBegin(GL\_LINE\_STRIP);

for (int i = 0; i <= 100; ++i) {

float t = (float)i / 100.0f;

glEvalCoord1f(t);

}

glEnd();

}

// Function to draw a Bezier surface

void drawBezierSurface(float controlPoints[4][4][3]) {

glMap2f(GL\_MAP2\_VERTEX\_3, 0.0, 1.0, 3, 4, 0.0, 1.0, 12, 4, &controlPoints[0][0][0]);

glEnable(GL\_MAP2\_VERTEX\_3);

glMapGrid2f(30, 0.0, 1.0, 30, 0.0, 1.0);

glEvalMesh2(GL\_FILL, 0, 30, 0, 30);

}

// Display function

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

glTranslatef(0.0f, 0.0f, -15.0f);

// Cylinder

glPushMatrix();

glTranslatef(-8.0f, 0.0f, 0.0f);

glColor3f(0.0f, 1.0f, 0.0f);

drawCylinder(1.0f, 3.0f, 36);

glPopMatrix();

// Cone

glPushMatrix();

glTranslatef(-5.0f, 0.0f, 0.0f);

glColor3f(1.0f, 0.0f, 0.0f);

drawCone(1.0f, 3.0f, 36);

glPopMatrix();

// Sphere

glPushMatrix();

glTranslatef(-2.0f, 0.0f, 0.0f);

glColor3f(0.0f, 0.0f, 1.0f);

drawSphere(1.0f, 36, 18);

glPopMatrix();

// Ellipsoid

glPushMatrix();

glTranslatef(1.0f, 0.0f, 0.0f);

glColor3f(1.0f, 0.5f, 0.0f);

drawEllipsoid(1.0f, 1.5f, 0.5f, 36, 18);

glPopMatrix();

// Toroid

glPushMatrix();

glTranslatef(4.0f, 0.0f, 0.0f);

glColor3f(0.5f, 0.0f, 0.5f);

drawTorus(0.3f, 1.0f, 36, 18);

glPopMatrix();

// Bezier curve

float bezierControlPoints[4][3] = {

{-2.0, -1.0, 0.0}, {-1.0, 2.0, 0.0},

{1.0, -2.0, 0.0}, {2.0, 1.0, 0.0}

};

glPushMatrix();

glTranslatef(-4.0f, -3.0f, 0.0f);

glColor3f(1.0f, 1.0f, 0.0f);

drawBezierCurve(bezierControlPoints);

glPopMatrix();

// Bezier surface

float bezierSurfaceControlPoints[4][4][3] = {

{{-1.5, -1.5, 0.0}, {-0.5, -1.5, 0.0}, {0.5, -1.5, 0.0}, {1.5, -1.5, 0.0}},

{{-1.5, -0.5, 0.0}, {-0.5, -0.5, 2.0}, {0.5, -0.5, 2.0}, {1.5, -0.5, 0.0}},

{{-1.5, 0.5, 0.0}, {-0.5, 0.5, 2.0}, {0.5, 0.5, 2.0}, {1.5, 0.5, 0.0}},

{{-1.5, 1.5, 0.0}, {-0.5, 1.5, 0.0}, {0.5, 1.5, 0.0}, {1.5, 1.5, 0.0}}

};

glPushMatrix();

glTranslatef(4.0f, -3.0f, 0.0f);

glColor3f(0.5f, 0.5f, 1.0f);

drawBezierSurface(bezierSurfaceControlPoints);

glPopMatrix();

glFlush();

glutSwapBuffers();

}

// Initialization function

void init() {

glEnable(GL\_DEPTH\_TEST);

glClearColor(0.0, 0.0, 0.0, 0.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45.0, 1.0, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

// Main function

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowSize(800, 600);

glutCreateWindow("LAB 05 - Curves and Surfaces");

init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**Kết quả:**

