Assignment no. 07

/\* Name:- Yash Gawandare

CRN no:- 24113064

Roll no:- 4064

Division :-SE IT

Batch :- S4 \*/

**Bezier Curve :**

#include<GL/glut.h>

#include<math.h>

#include<iostream>

using namespace std;

int points[4][2];

int clicks = 0;

bool curveDrawn = false;

void drawPoint(GLfloat x, GLfloat y)

{

    glColor3f(1.0, 0.0, 0.0);

    glPointSize(3.5f);

    glBegin(GL\_POINTS);

    glVertex2f(x, y);

    glEnd();

    glFlush();

}

void drawLine(GLfloat x1, GLfloat y1, GLfloat x2, GLfloat y2)

{

    glColor3f(1.0, 0.0, 0.0);

    glBegin(GL\_LINES);

    glVertex2f(x1, y1);

    glVertex2f(x2, y2);

    glEnd();

    glFlush();

}

void midPointSubDivision(GLfloat x1, GLfloat y1, GLfloat x2, GLfloat y2, GLfloat x3, GLfloat y3, GLfloat x4, GLfloat y4)

{

    GLfloat xAB = (x2 + x1) / 2;

    GLfloat yAB = (y2 + y1) / 2;

    GLfloat xBC = (x3 + x2) / 2;

    GLfloat yBC = (y3 + y2) / 2;

    GLfloat xCD = (x4 + x3) / 2;

    GLfloat yCD = (y4 + y3) / 2;

    GLfloat xABC = (xAB + xBC) / 2;

    GLfloat yABC = (yAB + yBC) / 2;

    GLfloat xBCD = (xBC + xCD) / 2;

    GLfloat yBCD = (yBC + yCD) / 2;

    GLfloat xABCD = (xABC + xBCD) / 2;

    GLfloat yABCD = (yABC + yBCD) / 2;

    if (((x2 - x1)\*(x2 - x1) + (y2 - y1)\*(y2 - y1)) > 4 ||

        ((x3 - x2)\*(x3 - x2) + (y3 - y2)\*(y3 - y2)) > 4 ||

        ((x4 - x3)\*(x4 - x3) + (y4 - y3)\*(y4 - y3)) > 4)

    {

        midPointSubDivision(x1, y1, xAB, yAB, xABC, yABC, xABCD, yABCD);

        midPointSubDivision(xABCD, yABCD, xBCD, yBCD, xCD, yCD, x4, y4);

    }

    else

    {

        drawLine(x1, y1, xABCD, yABCD);

        drawLine(xABCD, yABCD, x4, y4);

    }

}

void init()

{

    glClearColor(1.0, 1.0, 1.0, 1.0);

    glClear(GL\_COLOR\_BUFFER\_BIT);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluOrtho2D(0, 600, 0, 600);

}

void display()

{

    glClear(GL\_COLOR\_BUFFER\_BIT);

    glColor3f(1.0, 0.0, 0.0);

    if (curveDrawn)

    {

        for (int i = 0; i < 4; ++i)

        {

            drawPoint(points[i][0], points[i][1]);

        }

    }

    glFlush();

}

void mouse(int btn, int state, int x, int y)

{

    if (state == GLUT\_DOWN && btn == GLUT\_LEFT\_BUTTON)

    {

        points[clicks][0] = (GLfloat)x;

        points[clicks][1] = (GLfloat)(600 - y);

        clicks++;

        drawPoint((GLfloat)x, (GLfloat)(600 - y));

        if (clicks == 4)

        {

            drawLine(points[0][0], points[0][1], points[1][0], points[1][1]);

            drawLine(points[1][0], points[1][1], points[2][0], points[2][1]);

            drawLine(points[2][0], points[2][1], points[3][0], points[3][1]);

            midPointSubDivision(points[0][0], points[0][1],

                                 points[1][0], points[1][1],

                                 points[2][0], points[2][1],

                                 points[3][0], points[3][1]);

            curveDrawn = true; // important to refresh display

        }

    }

}

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

{

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowPosition(100, 100);

    glutInitWindowSize(600, 600);

    glutCreateWindow("Bezier Curve using Mid-Point Subdivision");

    init();

    glutDisplayFunc(display);

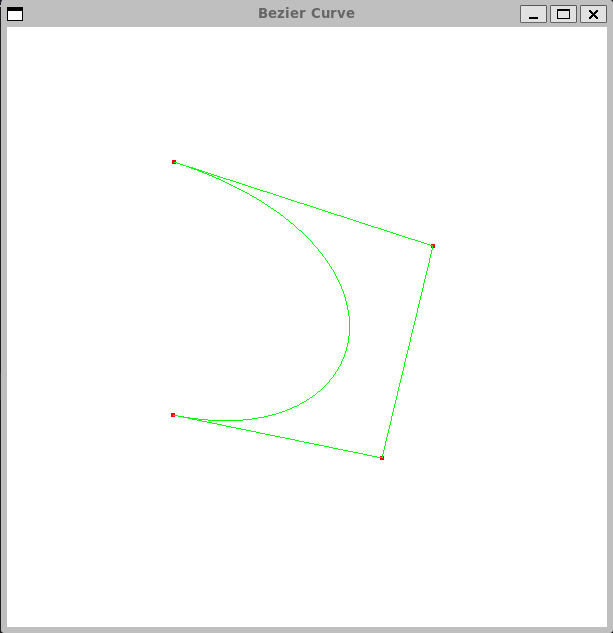
    glutMouseFunc(mouse);

    glutMainLoop();

    return 0;

}

**Output :**

****

**Koch curve :**

#include <GL/glut.h>

#include <cmath>

#include <iostream>

using namespace std;

int iterations;

void koch(GLfloat x1, GLfloat y1, GLfloat x2, GLfloat y2, int iter) {

    if (iter == 0) {

        glVertex2f(x1, y1);

        glVertex2f(x2, y2);

        return;

    }

    GLfloat dx = (x2 - x1) / 3;//another option x3=x1+(x2-x1)/3;

    GLfloat dy = (y2 - y1) / 3;

    GLfloat x3 = x1 + dx;

    GLfloat y3 = y1 + dy;

    GLfloat x5 = x2 - dx;

    GLfloat y5 = y2 - dy;

    // Coordinates for the peak of the triangle

    GLfloat x4 = 0.5 \* (x1 + x2) - sqrt(3.0) \* (y1 - y2) / 6;

    GLfloat y4 = 0.5 \* (y1 + y2) - sqrt(3.0) \* (x2 - x1) / 6;

    koch(x1, y1, x3, y3, iter - 1);

    koch(x3, y3, x4, y4, iter - 1);

    koch(x4, y4, x5, y5, iter - 1);

    koch(x5, y5, x2, y2, iter - 1);

}

void display() {

    glClear(GL\_COLOR\_BUFFER\_BIT);

    glColor3f(1.0, 0.0, 1.0);

    glBegin(GL\_LINES);

// Equilateral triangle points

GLfloat x1 = -0.5f, y1 = -0.3f;

GLfloat x2 = 0.5f, y2 = -0.3f;

GLfloat x3 = 0.0f, y3 = 0.5f;

koch(x1, y1, x2, y2, iterations);

koch(x2, y2, x3, y3, iterations);

koch(x3, y3, x1, y1, iterations);

glEnd();

glFlush();

}

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

cout << "Enter number of iterations: ";

cin >> iterations;

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(600, 600);

glutInitWindowPosition(100, 100);

glutCreateWindow("Koch Snowflake");

glClearColor(1.0, 1.0, 1.0, 1.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-1, 1, -1, 1);

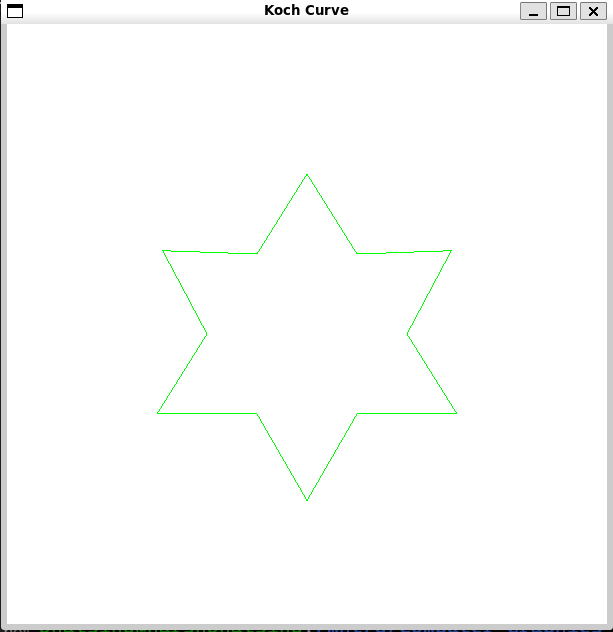
glutDisplayFunc(display);

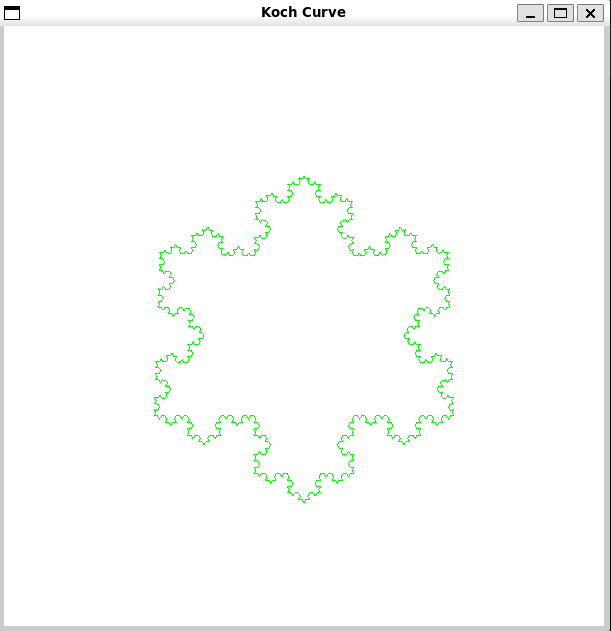
glutMainLoop();

return 0;

}

**Output :**

****

****