**SCHOOL OF COMPUTER SCIENCE**

**UNIVERSITY OF PETROLEUM AND ENERGY STUDIES**

**DEHRADUN, UTTARAKHAND**



**COMPUTER GRAPHICS**

**LABORATORY FILE**

**(2024-2025)**

**For**

**Vth Semester**

**Submitted To: Submitted By:**

Mr. Dinesh Akshat Negi

Assistant Professor 500106533(SAP ID)

[Vth Semester] R2142220414(Roll No.)

School of Computer Science B.Tech. CSF (Batch-1)

**LAB EXPERIMENT – 7**

**Drawing Bezier Curves**

**[Virtual GLUT based demonstration]**

1. Write a program to draw a cubic spline.

#include <GL/freeglut.h>

#include <vector>

#include <iostream>

#include <cmath>

struct Point {

float x, y;

};

// Define a vector to store control points for the cubic spline

std::vector<Point> controlPoints(4);

// Function to interpolate points for a cubic spline

Point cubicSpline(float t, Point p0, Point p1, Point p2, Point p3) {

float a = (1 - t) \* (1 - t) \* (1 - t);

float b = 3 \* t \* (1 - t) \* (1 - t);

float c = 3 \* t \* t \* (1 - t);

float d = t \* t \* t;

return {

a \* p0.x + b \* p1.x + c \* p2.x + d \* p3.x,

a \* p0.y + b \* p1.y + c \* p2.y + d \* p3.y

};

}

// Function to render the cubic spline

void renderSpline() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0, 0.0, 0.0); // Red color for the spline

glBegin(GL\_LINE\_STRIP);

for (float t = 0; t <= 1; t += 0.01) {

Point p = cubicSpline(t, controlPoints[0], controlPoints[1], controlPoints[2], controlPoints[3]);

glVertex2f(p.x, p.y);

}

glEnd();

glFlush();

}

void init() {

glClearColor(1.0, 1.0, 1.0, 1.0);

glOrtho(0.0, 500.0, 0.0, 500.0, -1.0, 1.0);

}

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

std::cout << "Enter 4 control points for the cubic spline (x y):\n";

for (int i = 0; i < 4; ++i) {

std::cout << "Point " << i + 1 << ": ";

std::cin >> controlPoints[i].x >> controlPoints[i].y;

}

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutCreateWindow("Cubic Spline - Akshat Negi");

init();

glutDisplayFunc(renderSpline);

glutMainLoop();

return 0;

}

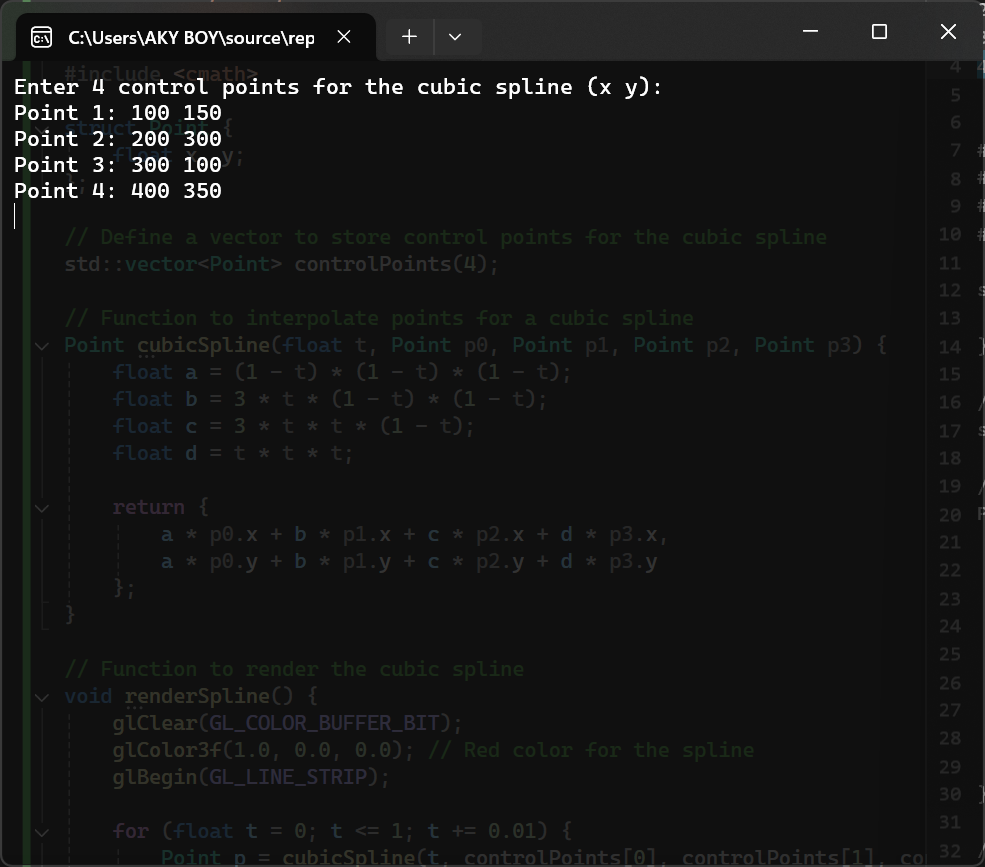
4 control points for the cubic spline (x y):

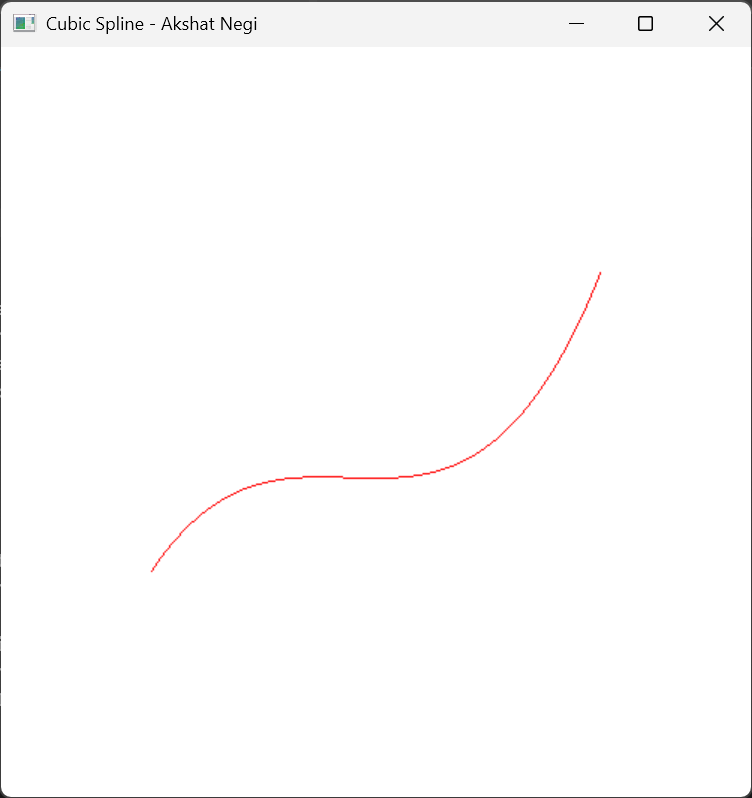
100 150

200 300

300 100

400 350





1. WAP to draw a Bezier curve.

*# Take necessary values as input from the user like degree of the Bezier curve.*

#include <GL/freeglut.h>

#include <iostream>

#include <vector>

struct Point {

float x, y;

};

std::vector<Point> controlPoints;

// Function to calculate Bezier point using De Casteljau's algorithm

Point bezierPoint(float t, const std::vector<Point>& points) {

std::vector<Point> temp = points;

for (int j = 1; j < points.size(); ++j) {

for (int i = 0; i < points.size() - j; ++i) {

temp[i].x = (1 - t) \* temp[i].x + t \* temp[i + 1].x;

temp[i].y = (1 - t) \* temp[i].y + t \* temp[i + 1].y;

}

}

return temp[0];

}

// Function to render the Bezier curve

void renderBezierCurve() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.0, 0.0, 1.0); // Blue color for Bezier curve

glBegin(GL\_LINE\_STRIP);

for (float t = 0; t <= 1; t += 0.01) {

Point p = bezierPoint(t, controlPoints);

glVertex2f(p.x, p.y);

}

glEnd();

glFlush();

}

void init() {

glClearColor(1.0, 1.0, 1.0, 1.0);

glOrtho(0.0, 500.0, 0.0, 500.0, -1.0, 1.0);

}

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

int degree;

std::cout << "Enter the degree of the Bezier curve: ";

std::cin >> degree;

controlPoints.resize(degree + 1);

std::cout << "Enter the control points:\n";

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

std::cout << "Point " << i + 1 << " (x y): ";

std::cin >> controlPoints[i].x >> controlPoints[i].y;

}

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutCreateWindow("Bezier Curve - Akshat Negi");

init();

glutDisplayFunc(renderBezierCurve);

glutMainLoop();

return 0;

}

SAMPLE INPUTS

Enter the degree of the Bezier curve: 2

Enter the control points:

Point 1 (x y): 100 100

Point 2 (x y): 250 400

Point 3 (x y): 400 100

Enter the degree of the Bezier curve: 3

Enter the control points:

Point 1 (x y): 50 50

Point 2 (x y): 150 400

Point 3 (x y): 350 400

Point 4 (x y): 450 50

Enter the degree of the Bezier curve: 4

Enter the control points:

Point 1 (x y): 50 50

Point 2 (x y): 100 400

Point 3 (x y): 250 300

Point 4 (x y): 400 400

Point 5 (x y): 450 50

