Practical 7: - Bezier Curve

Code:

#include <iostream>

#include <stdlib.h>

#include <GL/glut.h>

#include <math.h>

using namespace std;

class Point {

public:

float x, y;

void setxy(float x2, float y2) {

x = x2; y = y2;

}

const Point & operator-(const Point &rPoint) {

x = rPoint.x;

y = rPoint.y;

return \*this;

}

};

int factorial(int n) {

if (n <= 1) return 1;

else return n \* factorial(n - 1);

}

float binomial\_coff(float n, float k) {

float ans;

ans = factorial(n) / (factorial(k) \* factorial(n - k));

return ans;

}

Point abc[20];

int SCREEN\_HEIGHT = 500;

int points = 0;

int clicks = 4;

void myInit() {

glClearColor(1.0, 1.0, 1.0, 0.0);

glColor3f(0.0, 0.0, 0.0);

glPointSize(3);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, 640.0, 0.0, 500.0);

}

void drawDot(int x, int y) {

glBegin(GL\_POINTS);

glVertex2i(x, y);

glEnd();

glFlush();

}

void drawLine(Point p1, Point p2) {

glBegin(GL\_LINES);

glVertex2f(p1.x, p1.y);

glVertex2f(p2.x, p2.y);

glEnd();

glFlush();

}

Point drawBezier(Point PT[], double t) {

Point P;

P.x = pow((1 - t), 3) \* PT[0].x + 3 \* t \* pow((1 - t), 2) \* PT[1].x + 3 \* (1 - t) \* pow(t, 2) \* PT[2].x + pow(t, 3) \* PT[3].x;

P.y = pow((1 - t), 3) \* PT[0].y + 3 \* t \* pow((1 - t), 2) \* PT[1].y + 3 \* (1 - t) \* pow(t, 2) \* PT[2].y + pow(t, 3) \* PT[3].y;

return P;

}

Point drawBezierGeneralized(Point PT[], double t) {

Point P;

P.x = 0; P.y = 0;

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

P.x += binomial\_coff((float)(clicks - 1), (float)i) \* pow(t, (double)i) \* pow((1 - t), (clicks - 1 - i)) \* PT[i].x;

P.y += binomial\_coff((float)(clicks - 1), (float)i) \* pow(t, (double)i) \* pow((1 - t), (clicks - 1 - i)) \* PT[i].y;

}

return P;

}

void myMouse(int button, int state, int x, int y) {

if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN) {

abc[points].setxy((float)x, (float)(SCREEN\_HEIGHT - y));

points++;

drawDot(x, SCREEN\_HEIGHT - y);

if (points == clicks) {

glColor3f(0.2, 1.0, 0.0);

for (int k = 0; k < clicks - 1; k++) {

drawLine(abc[k], abc[k + 1]);

}

Point pl = abc[0];

for (double t = 0.0; t <= 1.0; t += 0.02) {

Point p2 = drawBezierGeneralized(abc, t);

cout << pl.x << " , " << pl.y << endl;

cout << p2.x << " , " << p2.y << endl;

cout << endl;

drawLine(pl, p2);

pl = p2;

}

glColor3f(0.0, 0.0, 0.0);

points = 0;

}

}

}

void myDisplay() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(640, 500);

glutInitWindowPosition(100, 150);

glutCreateWindow("Bezier Curve");

glutMouseFunc(myMouse);

glutDisplayFunc(myDisplay);

myInit();

glutMainLoop();

return 0;

}

Output:



Click Four Points on the Window