**Bezier curve**

#include<bits/stdc++.h>

#include <stdio.h>

#include<iostream>

#include<GL/glut.h>

#include<math.h>

//This is a point class, used to store the coordinates of the point

class Point

{

public:

int x, y;

void setxy(int \_x, int \_y)

{

x = \_x;

y = \_y;

}};

//Number of points

static int POINTSNUM = 0;

//Used to store a collection of points, because the Bezier curves with 4 points are drawn, so the array size is 4

static Point points[4];

//Initialization function

void init(void)

{

glClearColor(0.0, 0.0, 0.0,0); //Set the background to black

glColor3f(1.0, 1.0, 1.0); //The drawing color is white

glPointSize(4.0); //The size of the set point is 2\*2 pixels

glMatrixMode(GL\_PROJECTION); // Set the appropriate matrix

glLoadIdentity(); // is a non-parameter valueless function, its function is to replace the current matrix with a 4×4 identity matrix

//In fact, it is to initialize the current matrix.

//That is to say, no matter how many matrix transformations have been performed before, after the execution of this command, the current matrix will be restored to an identity matrix, which is equivalent to no matrix transformation state

gluOrtho2D(0.0, 600.0, 0.0, 480.0); //Parallel projection, the four parameters are x, y range

}

//Draw points

void setPoint(Point p)

{

glBegin(GL\_POINTS);

glVertex2f(p.x, p.y);

glEnd();

glFlush();

}

// draw a straight line

void setline(Point p1, Point p2)

{

glBegin(GL\_LINES);

glVertex2f(p1.x, p1.y);//Set vertex coordinates

glVertex2f(p2.x, p2.y);

glEnd();

glFlush();//Empty the buffer

}

// Draw Bezier curve

Point setBezier(Point p1, Point p2, Point p3, Point p4, double t)

{

Point p;

double a1 = pow((1 - t), 3);

double a2 = pow((1 - t), 2) \* 3 \* t;

double a3 = 3 \* t\*t\*(1 - t);

double a4 = t \* t\*t;

p.x = a1 \* p1.x + a2 \* p2.x + a3 \* p3.x + a4 \* p4.x;

p.y = a1 \* p1.y + a2 \* p2.y + a3 \* p3.y + a4 \* p4.y;

return p;

}

//display function

void display()

{

//glClear(GL\_COLOR\_BUFFER\_BIT);

//glFlush();

}

// mouse event

void mymouseFunction(int button, int state, int x, int y)

{

if (state == GLUT\_DOWN) // If the mouse is pressed, the left and right buttons are not distinguished

{

points[POINTSNUM].setxy(x, 480 - y); // When looking for the coordinates of the mouse point here

// Set the color of the point, draw the point

glColor3f(1.0, 0.0, 0.0);

setPoint(points[POINTSNUM]);

// Set the color of the line, draw the line

glColor3f(1.0, 0.0, 0.0);

if (POINTSNUM > 0) setline(points[POINTSNUM - 1], points[POINTSNUM]);

//If 4 bezier curves are reached, the counter will be cleared afterwards

if (POINTSNUM == 3)

{

//Draw Bezier curve

glColor3f(1.0, 1.0, 0.0); // Set the color of the Bezier curve

Point p\_current = points[0]; //Set as starting point

for (double t = 0.0; t <= 1.0; t += 0.05)

{

Point P = setBezier(points[0], points[1], points[2], points[3], t);

setline(p\_current, P);

p\_current = P;

}

POINTSNUM = 0;

} else

{ POINTSNUM++; } }}

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

{

glutInit(&argc, argv); //Fixed format

glutInitDisplayMode(GLUT\_RGB | GLUT\_SINGLE); //Cache mode

glutInitWindowSize(600, 480); //The size of the display box

glutInitWindowPosition(100, 100); //Determine the position of the upper left corner of the display box

glutCreateWindow("Bezier curve");

init(); // Initialize

glutMouseFunc(mymouseFunction); // Add mouse event

glutDisplayFunc(display); // execute display

glutMainLoop(); //Enter the GLUT event processing loop

return 0;

}