计算机图形学

作业四：贝塞尔曲线

数据科学与计算机学院　17大数据与人工智能

17341015　陈鸿峥

一、 实验原理

三次贝塞尔(Bezier)曲线的参数方程如下

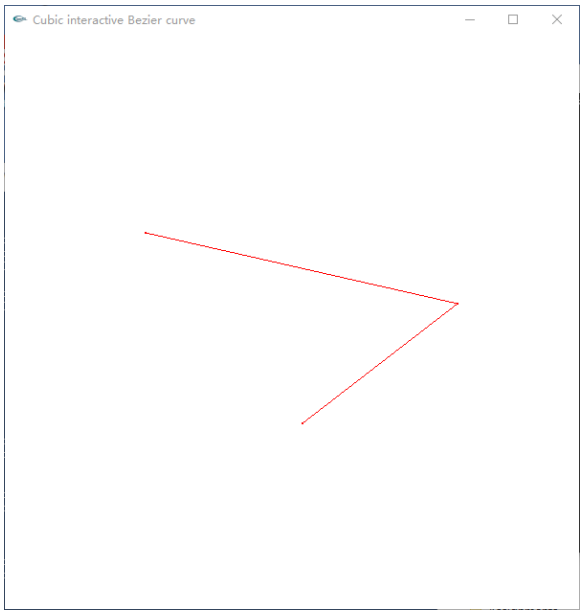
$$B(t) = P_0(1-t)^3 + 3P_1t(1-t)^2 + 3P_2t^2(1-t) + P_3t^3,\ t \in [0,1]$$

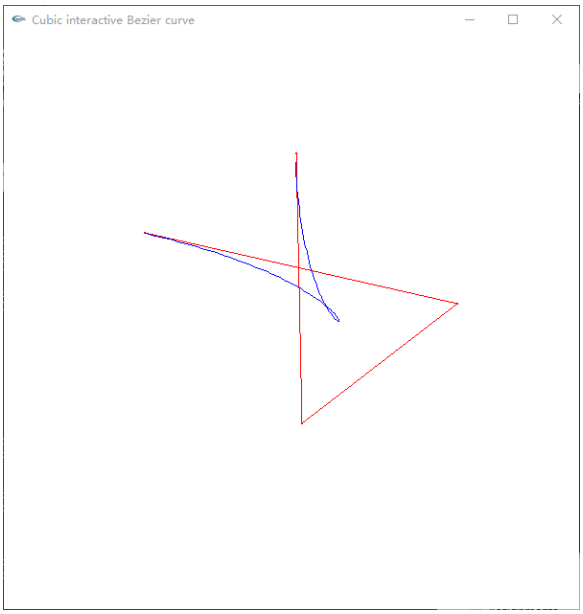
通过不断调整t的值，并且连接新的点与上一个点的线段，只要t的间隔足够小，最终即可得到一条平滑的曲线。

二、 实验结果

只需在当前目录下双击`bezier.exe`执行文件打开，即可运行。通过鼠标点击屏幕上的空白位置，可产生红色点，当得到四个点时会自动连接产生贝塞尔曲线。并且每产生四个点都会自动执行清屏功能。

为了突出展示效果，这里我采用了白色背景、红色直线及蓝色贝塞尔曲线的设置，效果如下图所示。





附录 A. 源代码

```
#include <windows.h> // must be the first one to be included!
#include <GL/glut.h>
#include <math.h>
#include <stdio.h>

#define WIN_WIDTH 600
#define WIN_HEIGHT 600

class Point
{
public:
    Point() : x(0), y(0) {}
    Point(int px, int py) {
        set(px,py);
    }
    void set(int px, int py) {
        this->x = px;
        this->y = py;
    }
    int x, y;
};

static int num_points = 0;
static Point points[4];

void init(void)
{
    glClearColor(1.0, 1.0, 1.0, 0); // set bg color -> black
    glColor3f(0.0,0.0,0.0); // drawing color -> white
    glPointSize(2.0);
    // be careful: need to set projection!
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0,WIN_WIDTH,0.0,WIN_HEIGHT,1,-1);
}

void drawPoint(Point p) {
    glBegin(GL_POINTS);
    glVertex2f(p.x, p.y);
    glEnd();
    glFlush();
}

void drawLine(Point p1, Point p2) {
```

```
45      glBegin(GL_LINES);
46      glVertex2f(p1.x,p1.y);
47      glVertex2f(p2.x,p2.y);
48      glEnd();
49      glFlush();
50  }
51
52  Point drawBezier(Point p1, Point p2, Point p3, Point p4, double t) {
53      // B(t) = P_0 (1-t)^3 + 3P_1 t(1-t)^2 + 3P_2 t^2(1-t) + P_3 t^3, t\in[0,1]
54      double a1 = pow((1 - t), 3);
55      double a2 = 3 * t * pow((1 - t), 2);
56      double a3 = 3 * pow(t, 2) * (1 - t);
57      double a4 = pow(t, 3);
58      Point p(a1*p1.x + a2*p2.x + a3*p3.x + a4*p4.x,
59              a1*p1.y + a2*p2.y + a3*p3.y + a4*p4.y);
60      return p;
61  }
62
63  void myDisplay()
64  {
65      glClear(GL_COLOR_BUFFER_BIT);
66      glFlush();
67  }
68
69  void mouseKicked(int button, int state, int x, int y) {
70      if (state == GLUT_DOWN)
71      {
72          // be careful that y increases from top to bottom
73          points[num_points].set(x,WIN_HEIGHT-y);
74
75          // draw point
76          glColor3f(1.0,0.0,0.0); // red
77          if (num_points == 0) // clear the previous curve
78              myDisplay();
79          drawPoint(points[num_points]);
80
81          // draw line
82          glColor3f(1.0,0.0,0.0); // red
83          if (num_points > 0)
84              drawLine(points[num_points-1], points[num_points]);
85
86          // update num_points
87          if (num_points == 3) {
88
89              glColor3f(0.0,0.0,1.0); // blue
```

```
90
91              // draw curve in small segements
92              Point p_curr = points[0];
93              for (double t = 0.0; t <= 1.0; t += 0.01)
94              {
95                  Point p_new = drawBezier(points[0], points[1], points[2], points[3],
                        ↪ t);
96                  drawLine(p_curr, p_new);
97                  p_curr = p_new;
98              }
99
100             num_points = 0;
101         } else {
102             num_points++;
103         }
104     }
105 }
106
107 int main(int argc, char *argv[])
108 {
109     glutInit(&argc, argv);
110     glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE);
111     glutInitWindowPosition(100, 100);
112     glutInitWindowSize(WIN_WIDTH, WIN_HEIGHT);
113     glutCreateWindow("Cubic interactive Bezier curve");
114     printf("Please click left button of mouse to input control point of Bezier
        ↪ curve!\n");
115
116     init();
117     glutMouseFunc(mouseKicked);
118     glutDisplayFunc(myDisplay);
119     glutMainLoop();
120     return 0;
121 }
```

编译指令如下：

```
g++ bezier.cpp -I.\include -L.\lib -lglut32 -lopengl32 -o bezier.exe
```