《计算机图形学》

实验报告

（作业八）

学 院 名 称：	数据科学与计算机学院
专业（班级）：	软件工程
学 生 姓 名：	江炎鸿
学　　号：	16340094
时　　间：	2019 年 5 月 29 日

作业八 ：Bezier Curve

一、作业要求

1. 用户能通过左键点击添加Bezier曲线的控制点，右键点击则对当前添加的最后一个控制点进行消除。

2. 工具根据鼠标绘制的控制点实时更新Bezier曲线。

加分项：

1. 可以动态地呈现Bezier曲线的生成过程。

二、具体实现

此次作业的实现思路是利用公式算出曲线上的点的坐标，然后通过渲染所有的点来实现曲线。至于动态生成，通过插值算出中间点的坐标然后连线即可。

1、首先鼠标位置的监听和实现鼠标点击事件

```
void cursor_position_callback(GLFWwindow* window, double x, double y)
{
    //将位置储存在全局变量中
    xpos = (int)x;
    ypos = SCR_HEIGHT - (int)y;
    return;
}

void mouse_button_callback(GLFWwindow* window, int button, int action, int mods)
{
    if (action == GLFW_PRESS) {
        switch (button)
        {
        case GLFW_MOUSE_BUTTON_LEFT:
            //将tn设为大于1的值来停止动态生成
            tn = 2;
            //添加点
            points.push_back(Point(xpos, ypos));
            break;
        case GLFW_MOUSE_BUTTON_RIGHT:
            //将tn设为大于1的值来停止动态生成
            tn = 2;
            //删除点
            if (points.size() > 0) {
                points.pop_back();
            }
            break;
        default:
            return;
        }
    }
}
```

2、使用布雷斯汉算法绘制直线的函数

```
void swap(int& num1, int& num2)
{
    int temp = num1;
    num1 = num2;
    num2 = temp;
}

void getLinePoint(int x1, int y1, int x2, int y2)
{
    int temp_x = abs(x1 - x2);
    int temp_y = abs(y1 - y2);

    //斜率大于1时进行坐标变换
    bool change = (temp_x < temp_y) ? true : false;
    if (change) {
        swap(x1, y1);
        swap(x2, y2);
        swap(temp_x, temp_y);
    }

    //判断斜率是否为正
    bool pos = ((x1 - x2) * (y1 - y2) >= 0) ? true : false;

    int x = (x1 < x2) ? x1 : x2;
    int y = (x1 < x2) ? y1 : y2;
    int p = 2 * temp_y - temp_x;
    for (int i = 0; i < temp_x + 1; i++) {

        if (change) {
            line.push_back(resPoint((float)y, (float)(x + i)));
        }
        else {
            line.push_back(resPoint((float)(x + i), (float)y));
        }

        if (p > 0) {
            //斜率为正递增，为负递减
            y = pos ? y + 1 : y - 1;
            p = p + 2 * temp_y - 2 * temp_x;
        }
        else {
            p = p + 2 * temp_y;
        }
    }
}
```

3、通过标记的点算出曲线的函数

```
//求阶乘
long factorial(int num) {
    if (num == 0)
        return 1;
    long temp = num;
    for (int i = 1; i < num; i++) {
        temp = temp * i;
    }
    return temp;
}

void setCurve()
{
    //points中储存的是标记的点
    float tempX;
    float tempY;
    int n = points.size() - 1;
    for (float t = 0; t <= 1; t += 0.001) {
        tempX = 0.0f;
        tempY = 0.0f;
        for (int i = 0; i <= n; i++) {
            tempX += (factorial(n) / (factorial(i) * factorial(n - i)) * pow(1 - t, n - i) * pow(t, i) * points[i].x);
            tempY += (factorial(n) / (factorial(i) * factorial(n - i)) * pow(1 - t, n - i) * pow(t, i) * points[i].y);
        }
        line.push_back(resPoint(tempX, tempY));
    }
}
```

4、通过递归实现动态生成曲线的过程，其中tn随时间增大

```
void createCurve(vector<Point> temp)
{
    int n = temp.size();
    if (n <= 2)
        return;

    vector<Point> tempuse;
    for (int i = 0; i < n - 2; i++) {
        int tempX1 = (int)((1 - tn) * temp[i].x + tn * temp[i + 1].x);
        int tempY1 = (int)((1 - tn) * temp[i].y + tn * temp[i + 1].y);
        int tempX2 = (int)((1 - tn) * temp[i + 1].x + tn * temp[i + 2].x);
        int tempY2 = (int)((1 - tn) * temp[i + 1].y + tn * temp[i + 2].y);
        getLinePoint(tempX1, tempY1, tempX2, tempY2);
        tempuse.push_back(Point(tempX1, tempY1));
        if (i == n - 3) {
            tempuse.push_back(Point(tempX2, tempY2));
            int x1 = (int)((1 - tn) * tempX1 + tn * tempX2);
            int y1 = (int)((1 - tn) * tempY1 + tn * tempY2);
            vector<Point> sem;
            sem.push_back(Point(x1, y1));
            setPoint(sem);
        }
    }
    //setPoint的作用是进行将坐标点附近的点进行渲染使肉眼可以直观的看见点
    setPoint(tempuse);
    createCurve(tempuse);
}
```

5、具体调用过程

```
while (!glfwWindowShouldClose(window))
{
    processInput(window);
    vector<resPoint> s;
    line.swap(s);
    //将标记的点可视化
    setPoint(points);

    //两个点以上时绘制直线，三个点以上时绘制曲线
    if (points.size() > 1) {
        for (int i = 0; i < points.size() - 1; i++) {
            getLinePoint(points[i].x, points[i].y, points[i + 1].x, points[i + 1].y);
        }
        if (points.size() > 2) {
            setCurve();
            if (tn == 2) {
                temppoints = points;
                tn = 0;
            }
        }
    }

    //随时间增大tn
    nowtime = glfwGetTime();
    if (nowtime - beftime > 0.001) {
        beftime = nowtime;
        tn += 0.001;
    }

    //tn在0到1之间时渲染生成的过程
    if (tn < 1) {
        createCurve(temppoints);
    }
```

三、实现效果

截图如下，完整效果见同目录下gif动图。