厦門大學

实验 6

Bézier Curve 贝塞尔曲线

姓　　名：雷昱

学　　号：22920202204666

学　　院：信息学院

专　　业：软件工程

年　　级：2020 级

二〇二二年 **6** 月 **13** 日

实验 6、Bézier Curve 贝塞尔曲线

建议阅读资料：

（1） 课件

贝塞尔曲线

学习要求：

（1） 掌握贝塞尔曲线和曲面的生成算法

Task1. 使用 OpenGL 画点和画线功能，实现贝塞尔曲线生成算法。

1. 自行设置 3 个控制点，利用 de Casteljau 生成贝塞尔曲线。要求生成曲线形成过程的**动画**。

完成，设计了 Bezier 曲线类，进行相应点的计算

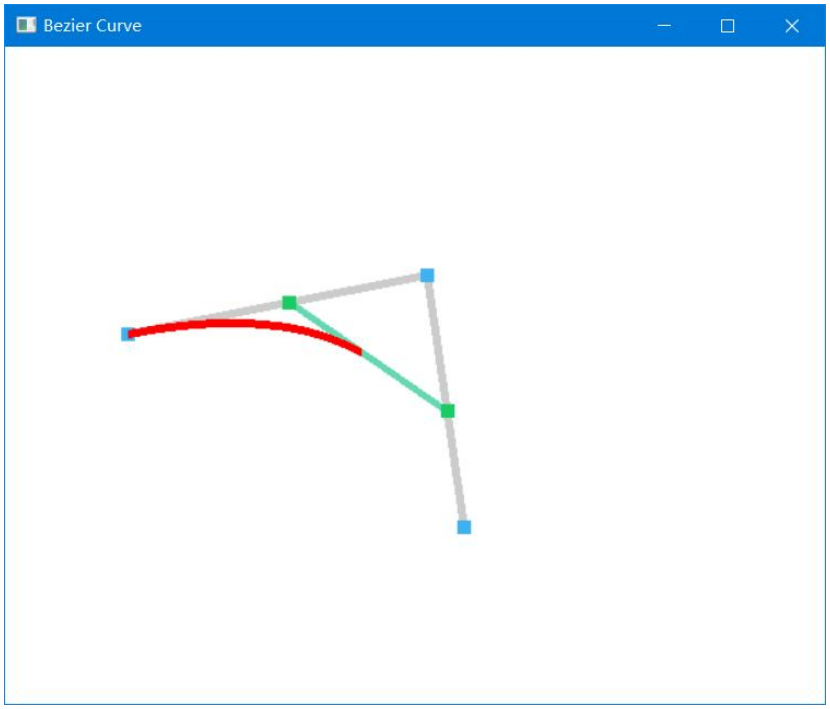
```
class Bezier
{
public:
    Bezier();
    Point       bezierpoint(GLdouble);
    bool        addPoint(Point);
    void        deletPoint();
    void        create(GLdouble);
    Point       lastpoint(void);
private:
    Point       points[POINT_MAX];
    GLdouble    param[POINT_MAX];
    GLint       cnt;
};
```

```
Point Bezier::bezierpoint(GLdouble t)
{
    Point p(0, 0);
    for (int i = 0; i < cnt; ++i)
    {
        param[i] = pow((1 - t), cnt - i - 1)
            * pow(t, i)
            * calculate(i, cnt - 1);
    }
    for (int i = 0; i < cnt; ++i)
    {
        p.x += param[i] * points[i].x;
        p.y += param[i] * points[i].y;
    }
    return p;
}
```

贝塞尔曲线绘制

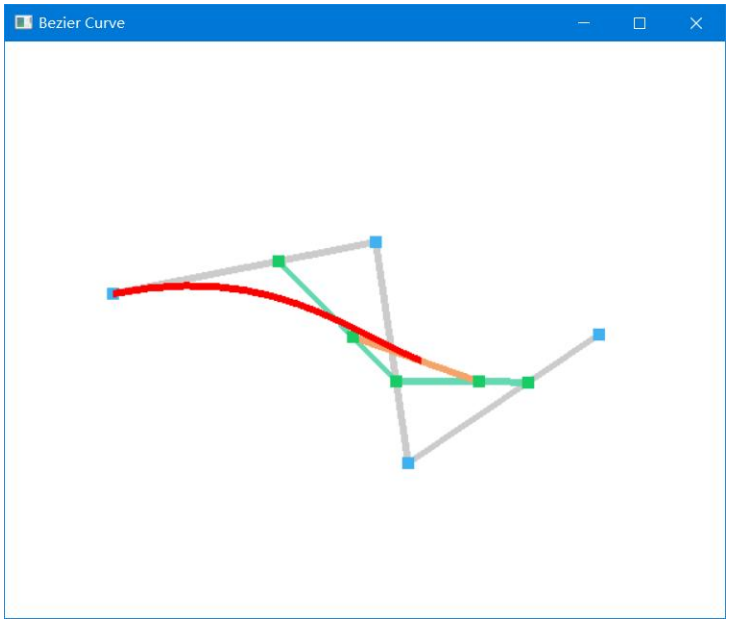
```
// 绘制贝塞尔曲线
if (cnt < 2) return;
Point pTemp[POINT_MAX][POINT_MAX];
for (int i = 0; i < cnt - 2; ++i)
{
    pTemp[0][i].x = points[i].x * (1 - temp) + points[i + 1].x * temp;
    pTemp[0][i].y = points[i].y * (1 - temp) + points[i + 1].y * temp;
    pTemp[0][i + 1].x = points[i + 1].x * (1 - temp) + points[i + 2].x * temp;
    pTemp[0][i + 1].y = points[i + 1].y * (1 - temp) + points[i + 2].y * temp;
    glColor3f(0.4f, 0.85f, 0.7f);
    setLine(pTemp[0][i], pTemp[0][i + 1]);
    glColor3f(0.1f, 0.8f, 0.4f);
    setPoint(pTemp[0][i]); setPoint(pTemp[0][i + 1]);
}
```

见如下截图

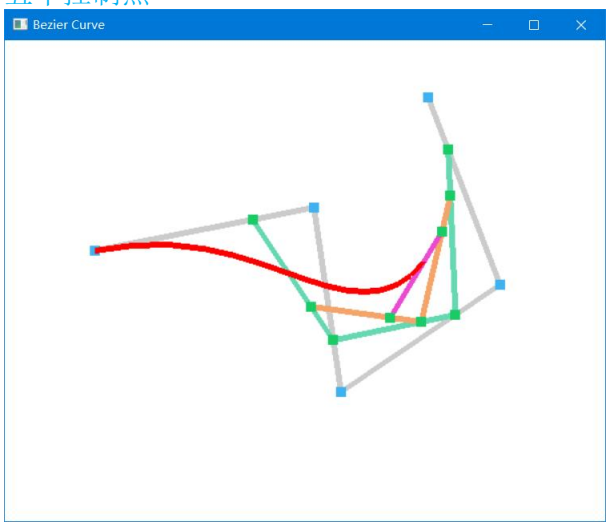


2. 在上述基础上，分别增加控制点数为 4、5、6、7、8，并生成相应的曲线形成**动画**。
   完成，见如下截图

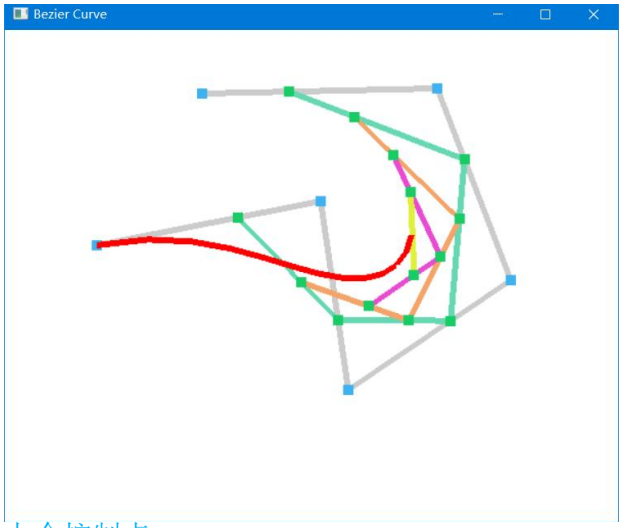
四个控制点



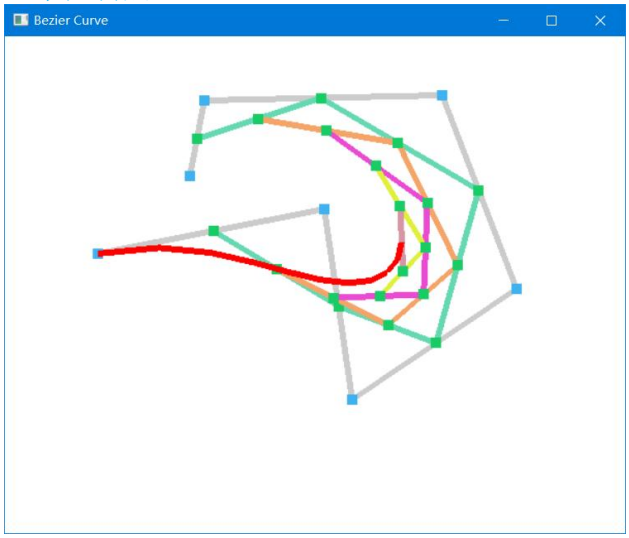
五个控制点



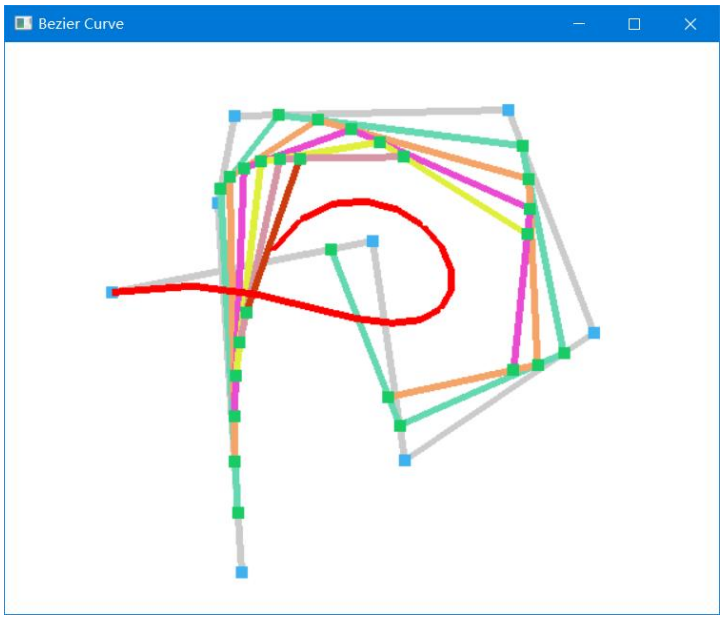
六个控制点



七个控制点



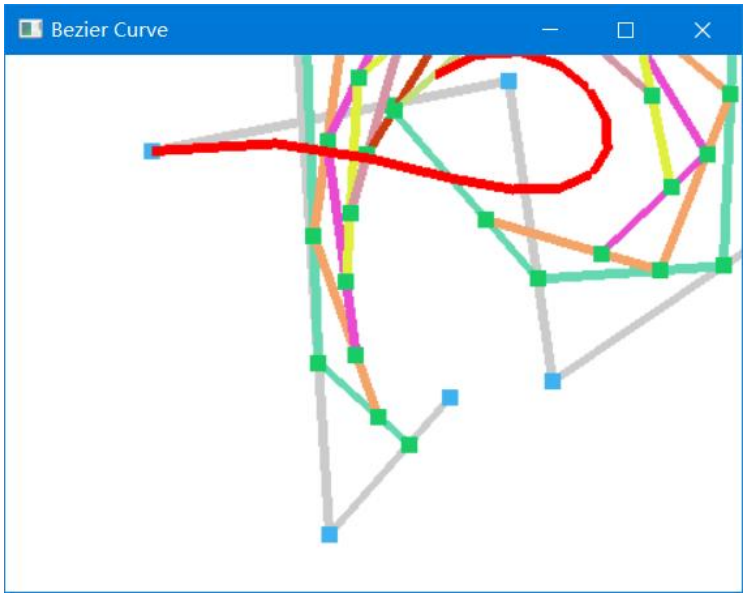
八个控制点



3. 自行完善功能，例如提供控制点的选取或移动功能；

完成，见如下截图

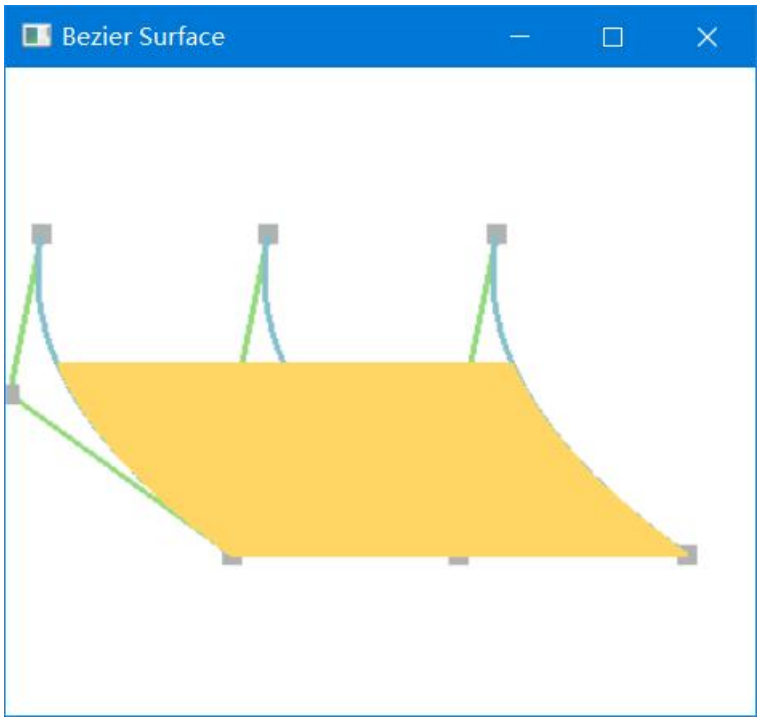
可以进行鼠标控制点插入并逐个增加贝塞尔曲线，且窗口图像可以任意变换大小。



Task2. 使用 OpenGL 画点、画线和画面功能，实现贝塞尔曲面生成算法。

1. 自行设置 3*3 个控制点，利用 de Casteljau 生成贝塞尔曲面。要求生成曲面形成过程的**动画。**

完成，见如下截图



核心代码

```
// 画出所有点和折线
for (int i = 0; i < cnt; ++i){
    for (int j = 0; j < cnt - 1; ++j){
        glColor3ub(149, 219, 125);
        setLine(points[i][j], points[i][j + 1]);
        glColor3f(0.7f, 0.7f, 0.7f);
        setPoint(points[i][j]);
        setPoint(points[i][j + 1]);
    }
}
```

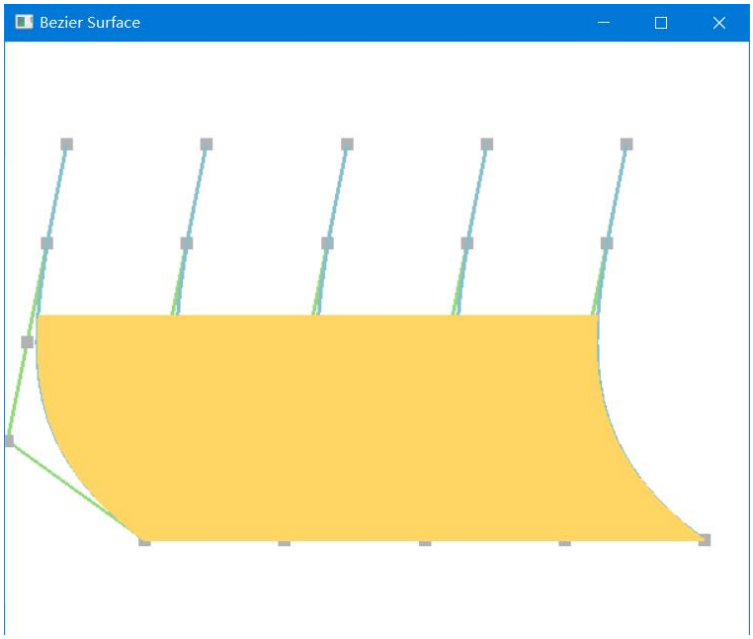
```
// 绘制贝塞尔曲线
for (int i = 0; i < cnt; ++i){
    Point pLast = points[i][0];
    for (GLdouble j = 0.0; j <= 1.05; j += 0.05)
    {
        Point pNow = bezierpoint(j, i);
        glColor3ub(133, 192, 206);
        setLine(pLast, pNow);
        pLast = pNow;
    }
}
```

```
// 贝赛尔曲面绘制
for (double f = 0.0; f <= temp; f += 0.003){
    Point ptemps[POINT_MAX];
    for (int i = 0; i < cnt; ++i)
    {
        ptemps[i] = bezierpoint(f, i);
    }
    Point pLast = ptemps[0];
    for (GLdouble j = 0.0; j <= 1.05; j += 0.05)
    {
        glLineWidth(3);
        Point pNow = bezierpoint(j, ptemps);
        glColor4ub(255, 214, 100, 0.2f);
        setLine(pLast, pNow);
        pLast = pNow;
    }
}
glFlush();
```

控制点移动

```
bool Bezier::hit(GLfloat point_x, GLfloat point_y)
{
    for (int i = 0; i < cnt; ++i)
    {
        for (int j = 0; j < cnt; ++j)
        {
            if (abs(point_x - points[i][j].x) <= POINT_SIZE / 2 + 3 && abs(point_y - points[i][j].y) <= POINT_SIZE / 2 + 3)
            {
                selectI = i, selectJ = j;
                return true;
            }
        }
    }
    return false;
}
```

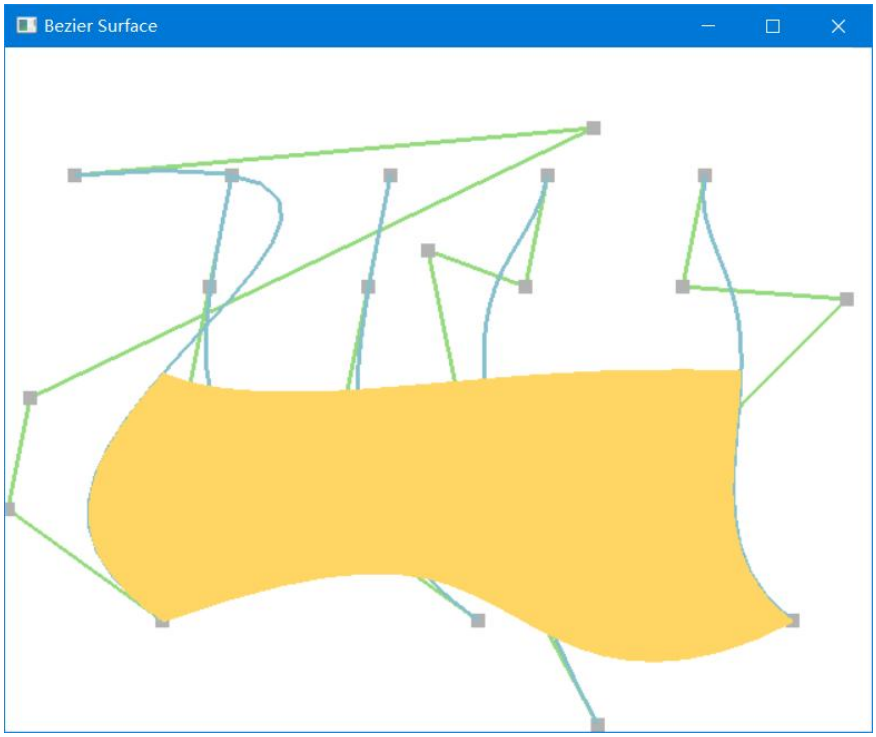
2. 在上述基础上，分别增加控制点数为 4*4、5*5 并生成相应的曲面形成**动画**。
完成，见如下截图



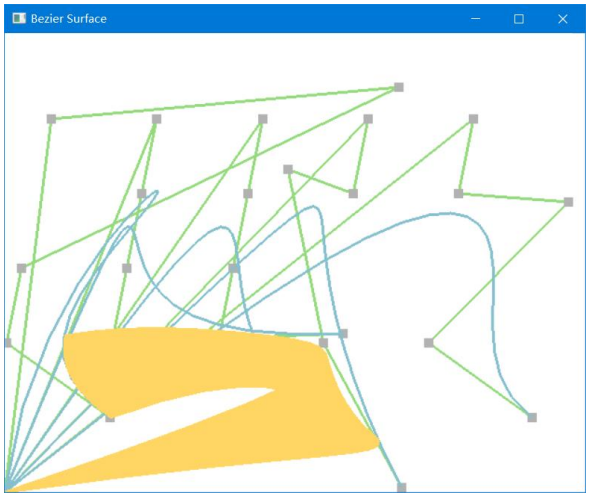
3. 自行完善功能，例如提供控制点的选取或移动功能；改善生成图形美观程度。
完成，见如下截图

可以拖动控制点进行贝塞尔曲面重新绘制

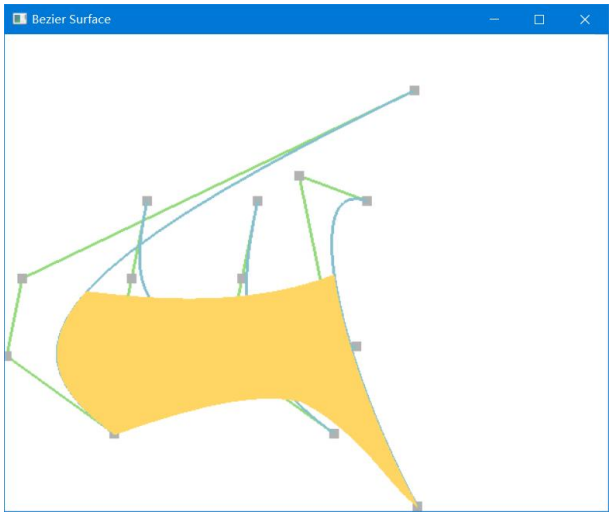
按键键盘“a”和“d”可以增加控制点的数量。最多为 6*6 最少为 2*2

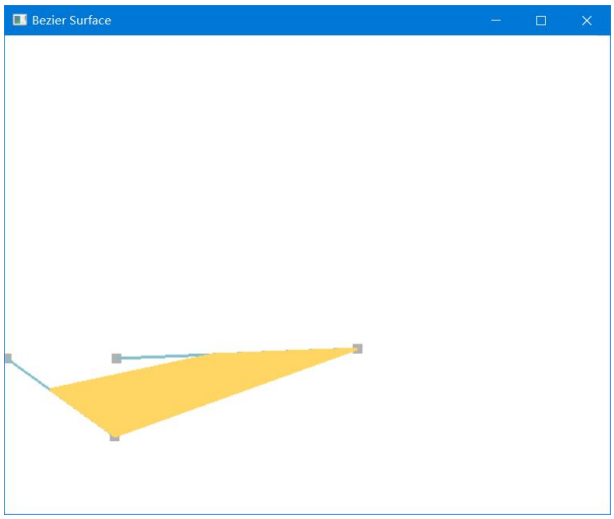


按键”a”增加控制点



按键”d”减少控制点





作业提交说明：

本次实验共有 2 个任务。

提交方式为：*将源文件、可执行文件、实验报告*放到一个文件夹中，命名为“您的学号_姓名”，打包上传到 ftp 服务器中相应目录下。请确保提交的可执行文件可以运行（打分的重要依据）。本次实验作业的提交截止日期为 6 月 15 日 23：59 分。

特别说明：本次实验 2 个任务都需要提交。您需要提供一个完整的文档（不限格式），逐条说明您完成每一条任务的具体情况。