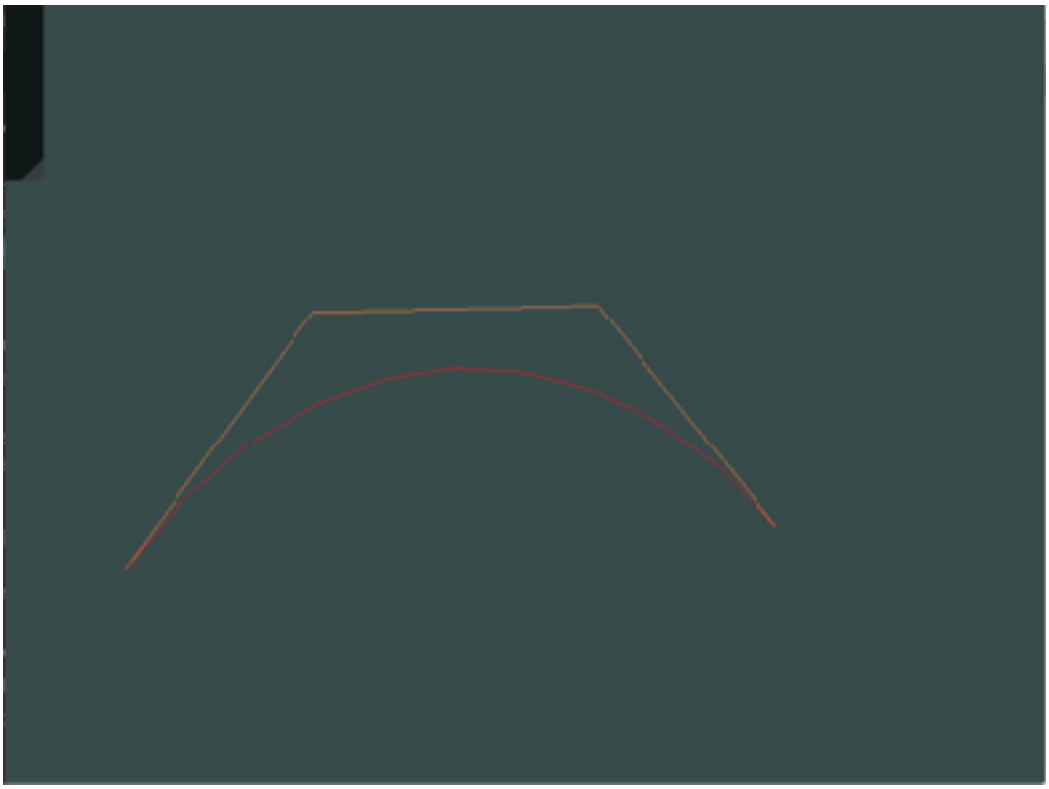
Homework 8

姓名：张家侨　　学号：15331390

Basic：

1.用户能在工具屏幕上画4个点（使用鼠标点击），然后工具会根据这4个点拟合出一条Bezier Curve（按照画点的顺序）。

运行结果：



实现：

要实现在屏幕上画四个点，然后根据这四个点进行连线形成折线，然后在折线的基础上拟合出一条Bezier Curve，大致可以分为以下几个步骤：

（1）实现鼠标点击画点

（2）根据鼠标画的点将点逐个连起来形成折线

（3）利用Bezier的计算方法，画出4个点的Bezier Curve

首先是实现鼠标的点击，并记录点的位置，glfw库中已经提供了鼠标的点击的回调函数接口glfwSet-MouseButtonCallback，通过设置回调函数，我们就能轻易监听鼠标的点击时间。

然后，鼠标点击后，我们还需要获取鼠标点击的屏幕位置，这里glfw也有相应的回调函数接口glfw-SetCursorPosCallback，用于用于监听指针的移动位置，每帧更新其位置，用两个变量分别存取X和Y坐标，然后在点击的时候，将坐标赋予点的坐标即可。

当然，我们在做上述工作时还必须将点的坐标保留下来，因为作业要求是固定4个点，所以我就用4个全局变量来存储以及后面的划线

有了点的坐标，利用opengl的绘画函数glDrawArrays，并将参数设为GL_LINE_STRIP，就能将4个点的原始折线画出。

最后，就是根据算法，利用4个点的位置信息，拟合出三次的Bezier Curve。下面看下三次的计算公式：

$$\mathbf{B}(t) = \mathbf{P}_0(1-t)^3 + 3\mathbf{P}_1 t(1-t)^2 + 3\mathbf{P}_2 t^2(1-t) + \mathbf{P}_3 t^3 \ , t \in [0,1]$$

由上可知，整段曲线的坐标，都可以利用上面公式得到，将t由0变化到1，我们只需手动设置变化的幅度，即设置划线点个数，在这里我设置10个点进行划线，即t的变化幅度是0.1。

```
for (double t = 0.0f; t <= 1.0f; t += 0.1)
{
    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;
    Q[0] = a1 * vertices[0] + a2 * vertices[3] + a3 * vertices[6] + a4 * vertices[9];
    Q[1] = a1 * vertices[1] + a2 * vertices[4] + a3 * vertices[7] + a4 * vertices[10];
    point[i * 3]     = Q[0];
    point[i * 3 + 1] = Q[1];
    i++;
}
glBufferData(GL_ARRAY_BUFFER, sizeof(point), point, GL_STATIC_DRAW);
glDrawArrays(GL_LINE_STRIP, 0, 11);
```

有了拟合曲线的点坐标后，利用glDrawArrays函数，我们就能将曲线显示出来。

2.用户画完第一条Bezier Curve之后，可以调整4个点的位置。工具会根据调整位置后的点实时更新曲线的样子。
运行结果：

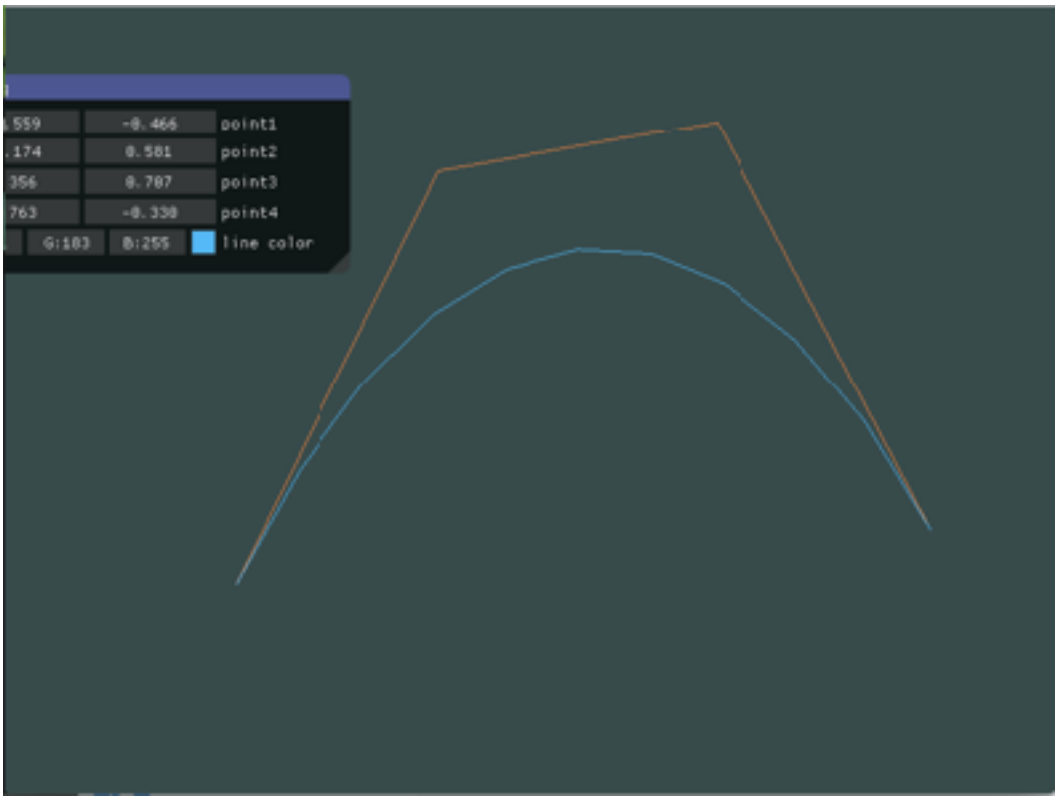
实现：

这里的实现还是挺简单的，因为点的坐标我是用全局变量保存的，所以只要我设置GUI调整四个全局变量的值，然后在渲染循环中更新Buffer的值即可实现。

Bonus：

1.在GUI里添加菜单栏，用户可以选择Bezier Curve的颜色。

运行结果：



实现：

这里的实现只需针对段渲染器里的颜色进行设置，然后用GUI进行调节值。

2.用户画点时，可以把画出的某个点消除。

实现：

这里的实现我通过监听键盘的**D键**，然后可以按画点的反顺序逐一消除已经画的点，就可以方便用户重新画点。