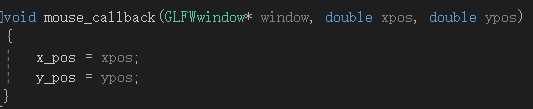
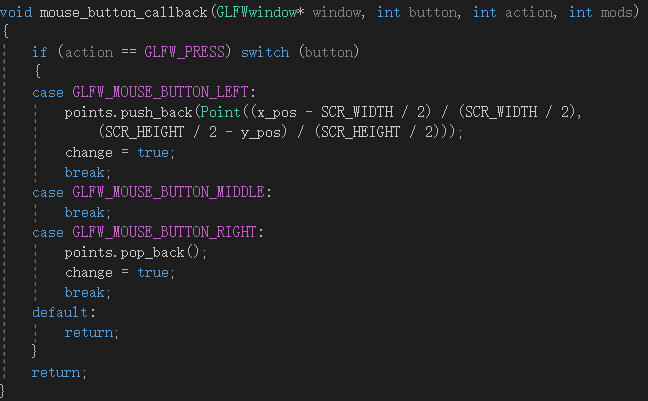
**Basic:**

先实现mouse的监听：

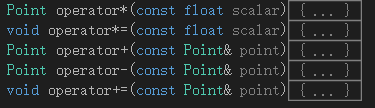
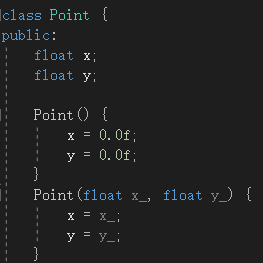






当点击鼠标左键时，将当前鼠标的x，y坐标对屏幕大小进行处理，将其化为[-1,1]区间范围内，然后存储到一个类Point中，将Point放到一个vector<Point> points里，points这个容器用以存储控制点。而点击鼠标右键时，将最后一个point弹出。

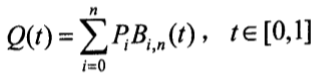
**类Point的实现：**



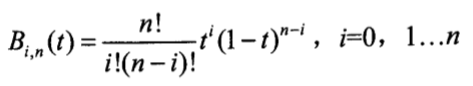
Point存储点的坐标，并且重载了一些会用到的运算符。

**Bezier Curve的实现：**

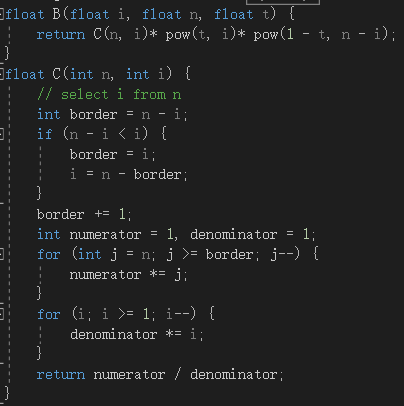
又控制点生成的曲线公式如下：



其中，B的公式为（左部份为组合，即n中取i）：

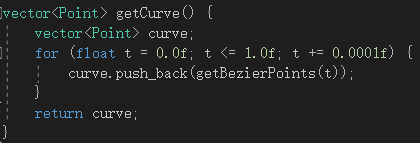


代码实现：

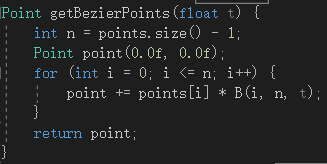


其中，C函数即公式左半部分，排列组合n中取i。而B函数需要参数t。

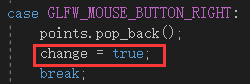
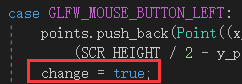
当要生成Bezier曲线时，需要t以一定步长从0移动到1，从而不断生成多个插值点（步长越小，插值点越多），用一个函数getCurve() 来获得描绘曲线所需的插值点：



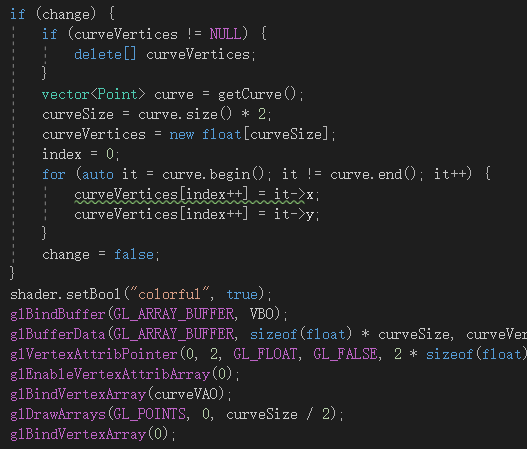
对于每一个t，都调用一个函数来获得插值点Q(t)，根据上面的公式实现：



最后将获得的点放入VAO中进行渲染即可，当控制点数量发生变化（鼠标click）时将change变量置为true：



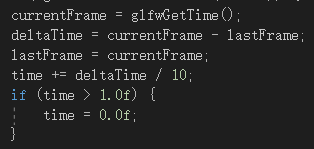
当change为true时，就重新调用getCurve获得新的曲线，将其转化为数组并进行渲染：



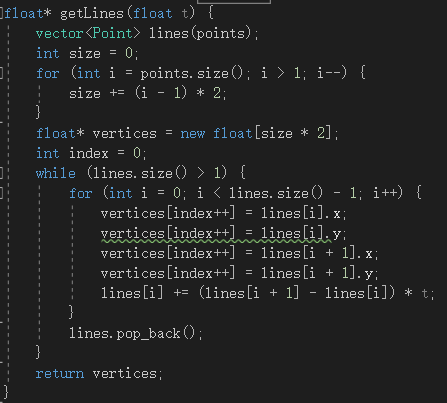
以上即完成了Bezier曲线的实时更新。

**Bonus:**

要求动态呈现曲线的生成过程，就需要用到时间相关的变量，用一个变量time来控制t从0到1的速度：



用一个getLines(float t) 函数来获得每个时间点t的需要渲染的辅助线段：



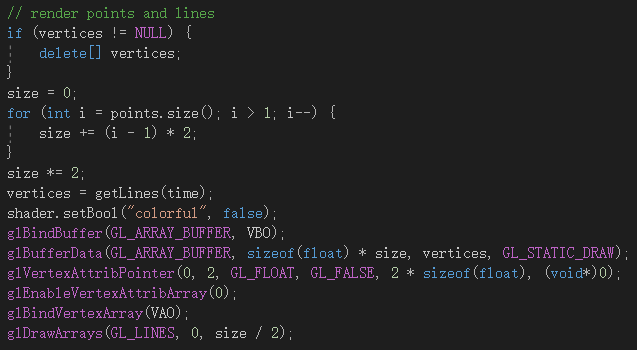
原理很简单，就是每一步都将控制点更新，新的点为：

**P0’ = (P1 – P0) \* t + P0**

因为渲染线段需要起点和终点两个点，因此每次更新前都将P[i] 和P[i+1]放到数组中，然后才对P[i]进行更新；最后将vector中的最后一个点去掉。

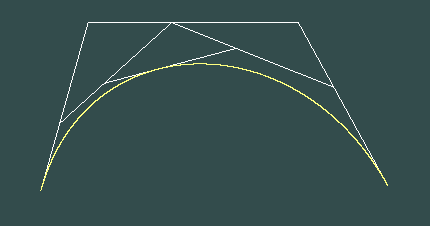
当vector中只剩下一个点时，更新结束，返回得到的数组。

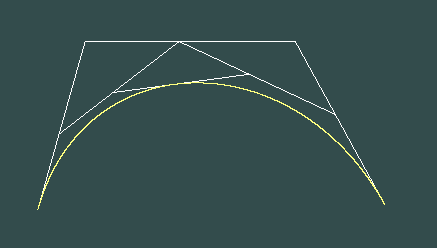
进行渲染：



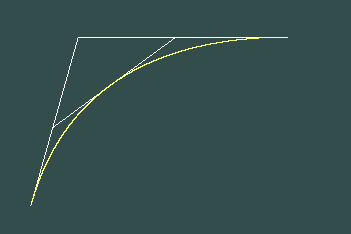
**结果展示:**

鼠标左键点四个控制点，黄色线即为Bezier Curve，会展示曲线的生成过程：





点右键，去掉最后一个点，更新Bezier曲线：



更多展示请看mp4。