《计算机图形学》实验 8 实验报告

学生	刘沅昊	学号	15331220
学院	数据科学与计算机学院	年级专业	16 级软件工程 (数字媒体技术)

实验内容：

Homework

Basic:

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。

Hint: 大家可查询捕捉mouse移动和点击的函数方法

Bonus:

1. 可以动态地呈现Bezier曲线的生成过程。

实验结果：

具体效果请查看 record.gif

Basic：

N 阶 Bezier 曲线通式是

$$\mathbf{B}(t)=\sum_{i=0}^{n}\binom{n}{i}\mathbf{P}_i(1-t)^{n-i}t^i=\mathbf{P}_0(1-t)^n+\binom{n}{1}\mathbf{P}_1(1-t)^{n-1}t+\cdots+\mathbf{P}_n t^n\ ,\ t\in[0,1]$$

因为高阶 Bezier 曲线计算量也比较大，所以放在 shader 里面计算会比较好，我在本次作业里面实现的也最高支持 5 阶 Bezier 曲线绘制，也就是允许最多添加 5 个控制点。Bezier 曲线需要注意的是每一项的系数是一个组合数，这个需要计算出来，然后求和可以用一个循环就能实现。

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**

首先，需要用到函数获取鼠标在窗口的坐标位置，使用函数 glfwGetCursorPos()就可以获得鼠标在窗口的位置。然后，判断附近有没有已有的控制顶点，如果有，那就可以长按拖动该控制顶点，如果没有，就添加一个新的控制点。

```
295     if (button == GLFW_MOUSE_BUTTON_LEFT) {
296         // add one point on the canvas  && move the selected points
297         if (action == GLFW_PRESS) {
298             isLeftButtonPressed = true;
299             if (p.end() == getNearPoints(xpos, ypos, 180) && pointNum < 5) {
300                 // add the selected point
301                 addPoint(xpos, ypos);
302                 pointNum++;
303             }
304         }
305 
306         if (action == GLFW_RELEASE) {
307             currPointIter = p.end();
308             isLeftButtonPressed = false;
309         }
310     }
```

点击右键就对最后添加的控制点进行消除。

```
312     if (button == GLFW_MOUSE_BUTTON_RIGHT && action == GLFW_PRESS) {
313         // delete the last point
314         for (auto iter = p.rbegin(); iter != p.rend(); ++iter) {
315             if (*iter != glm::vec3(-100.0f, -100.0f, -100.0f)) {
316                 *iter = glm::vec3(-100.0f, -100.0f, -100.0f);
317                 pointNum--;
318                 break;
319             }
320         }
321     }
```

2. 工具根据鼠标绘制的控制点实时更新 Bezier 曲线

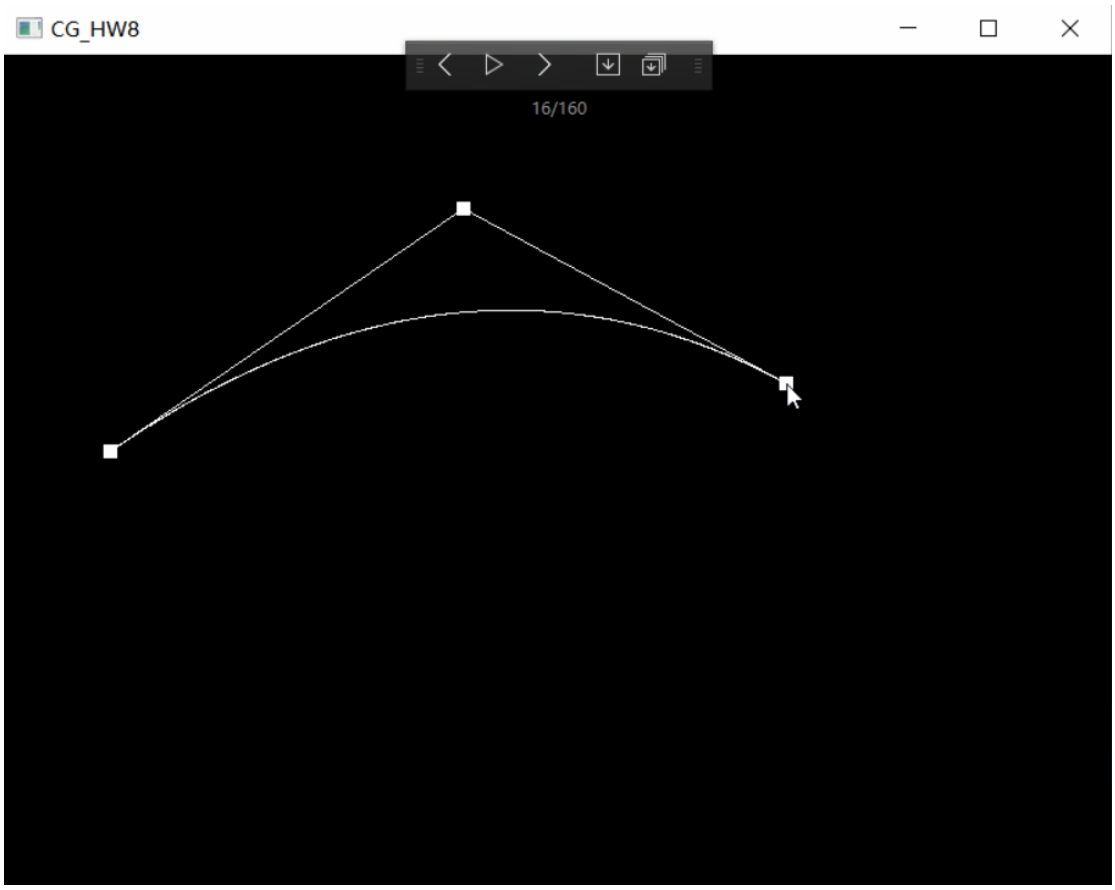
每添加一个控制顶点就能绘制出当前的 Bezier 曲线，所以要在渲染循环里面一直调用绘制 Bezier 曲线的函数。

Bobus：

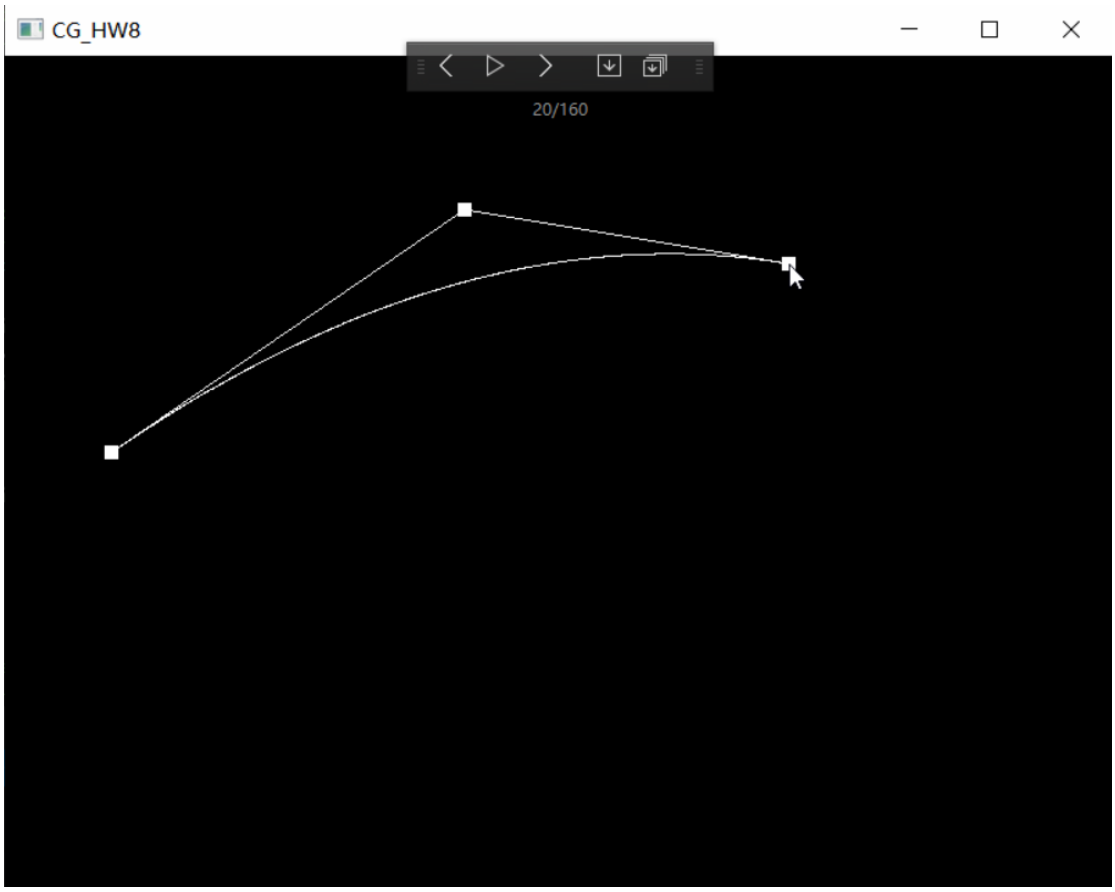
1. 可以动态的呈现 Bezier 曲线的生成过程

```
207         // bonus
208         if (isBonus == 1) {
209             int layer = pointNum - 1;
210             cout << t << endl;
211             vector<glm::vec3> tp = p;
212             for (int i = layer; i >= 1; i--) {
213                 tp = getPointsInEachLayers(tp, i);
214                 glBindVertexArray(pVAO);
215                 glBindBuffer(GL_ARRAY_BUFFER, pVBO);
216                 auto pointData = controlPoints2dataVector(tp);
217                 glBufferData(GL_ARRAY_BUFFER, pointData.size() * sizeof(GLfloat), pointData.data(), GL_STATIC_DRAW);
218                 glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(GLfloat), (void*)0);
219                 glEnableVertexAttribArray(0);
220                 glBindBuffer(GL_ARRAY_BUFFER, 0);
221                 pointShader.use();
222                 glPointSize(5.0f);
223                 glDrawArrays(GL_POINTS, 0, pointData.size() / 3);
224                 glDrawArrays(GL_LINE_STRIP, 0, pointData.size() / 3);
225                 glBindVertexArray(0);
226             }
227         }
```

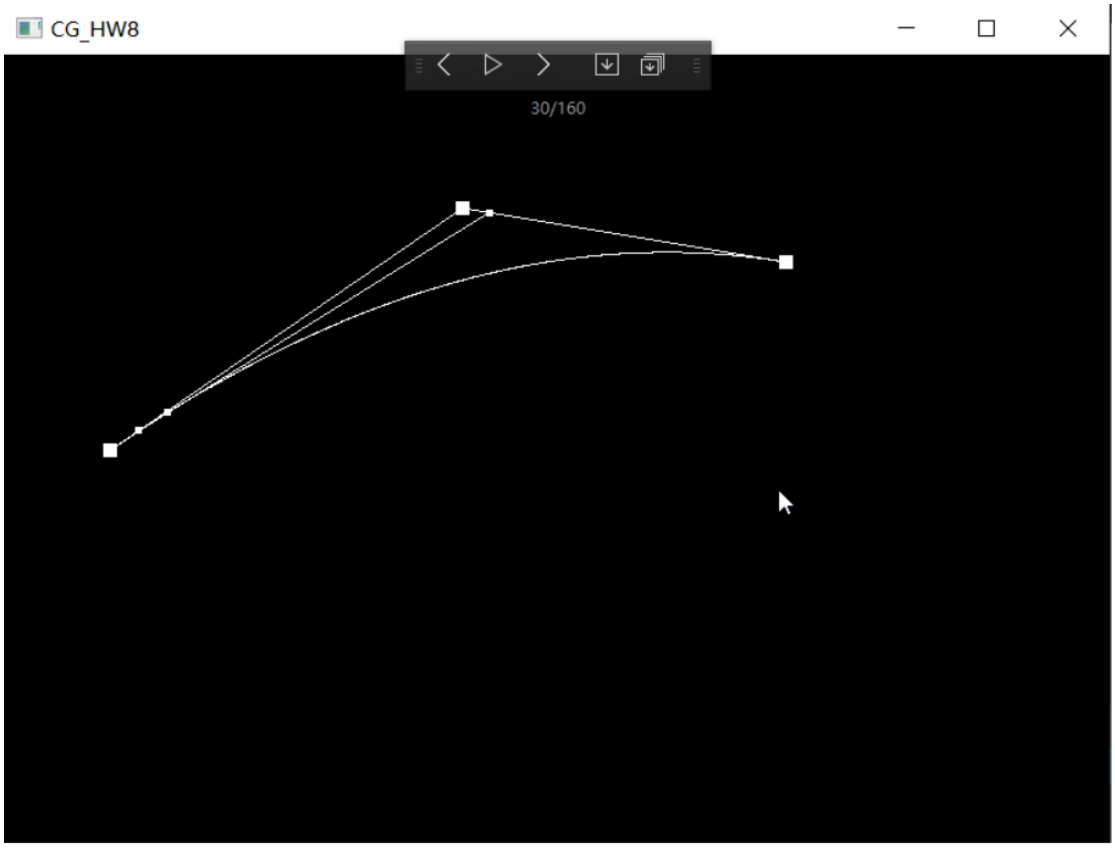
我设置长按 R 可以动态绘制 Bezier 曲线。原理是我将 Bezier 曲线分成了很多层，每一层根据变量 t 算出当前层的插值点，层数等于控制点数减 1，用一个循环就能将每一层的插值点找到，然后将点和点之间的连线画出来就实现动态绘制 Bezier 曲线生成。



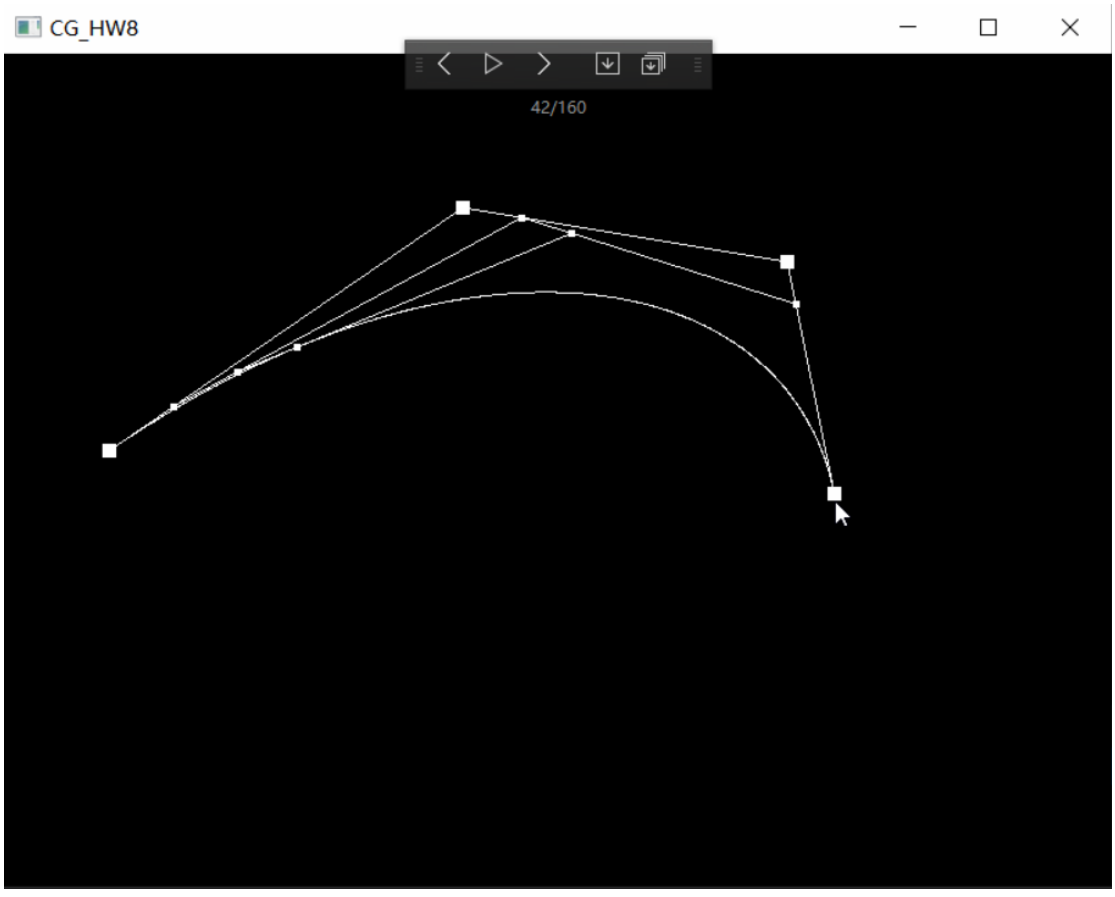
可以长按拖动控制顶点



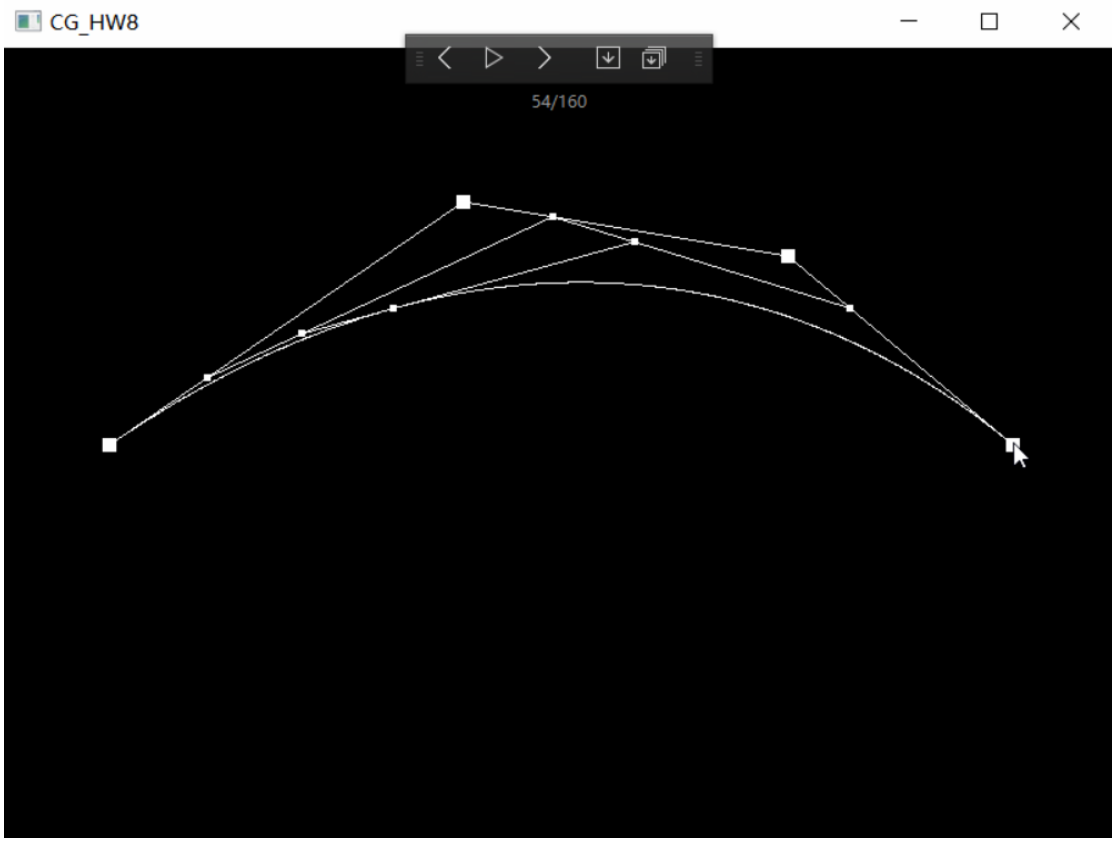
长按 R 动态绘制 Bezier 曲线



添加新的控制顶点，绘制过程接着之前的进行



绘制中也可以拖动控制顶点



最大支持 5 阶 Bezier 曲线，此后就不能再添加了。