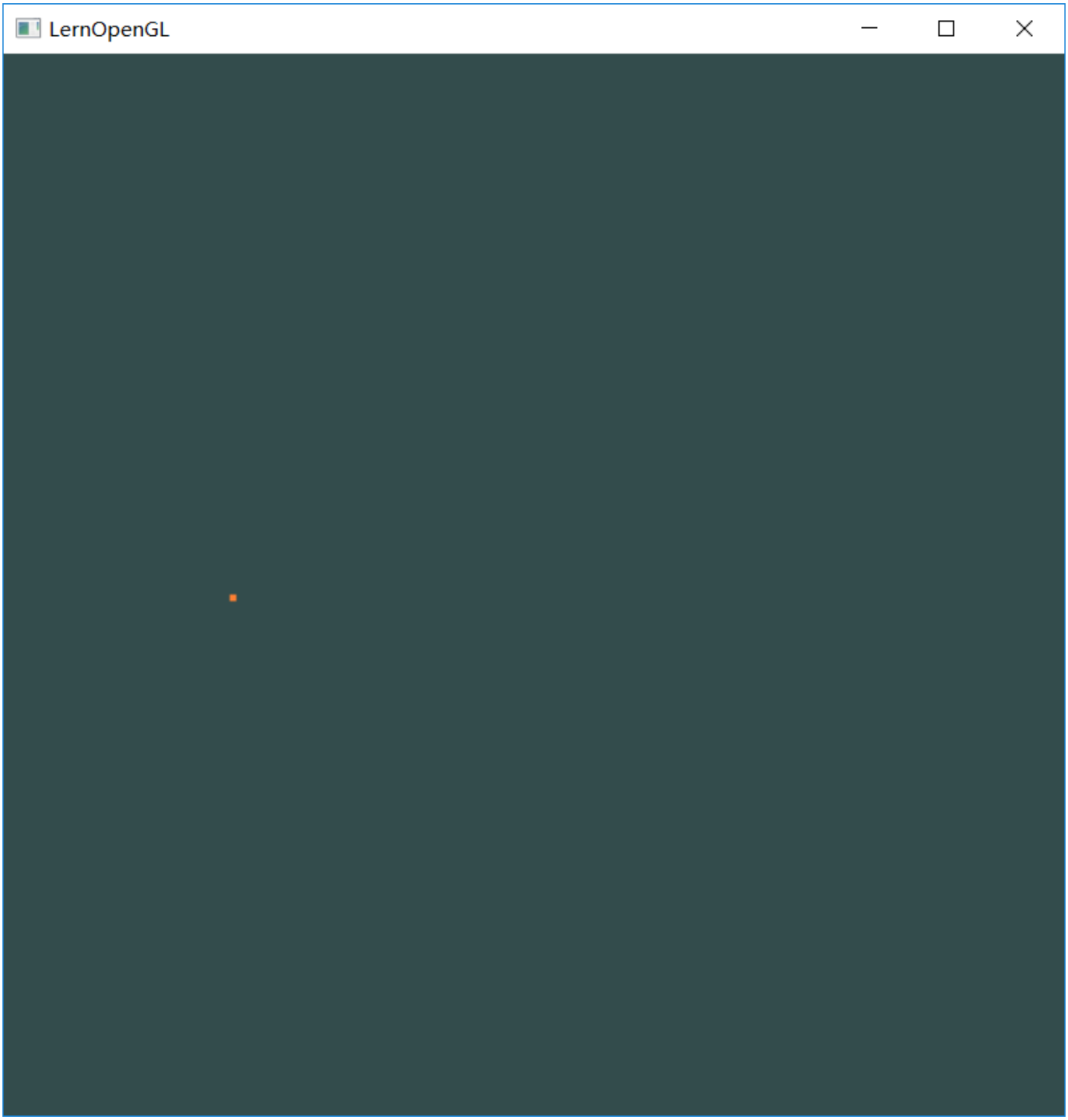
实验报告

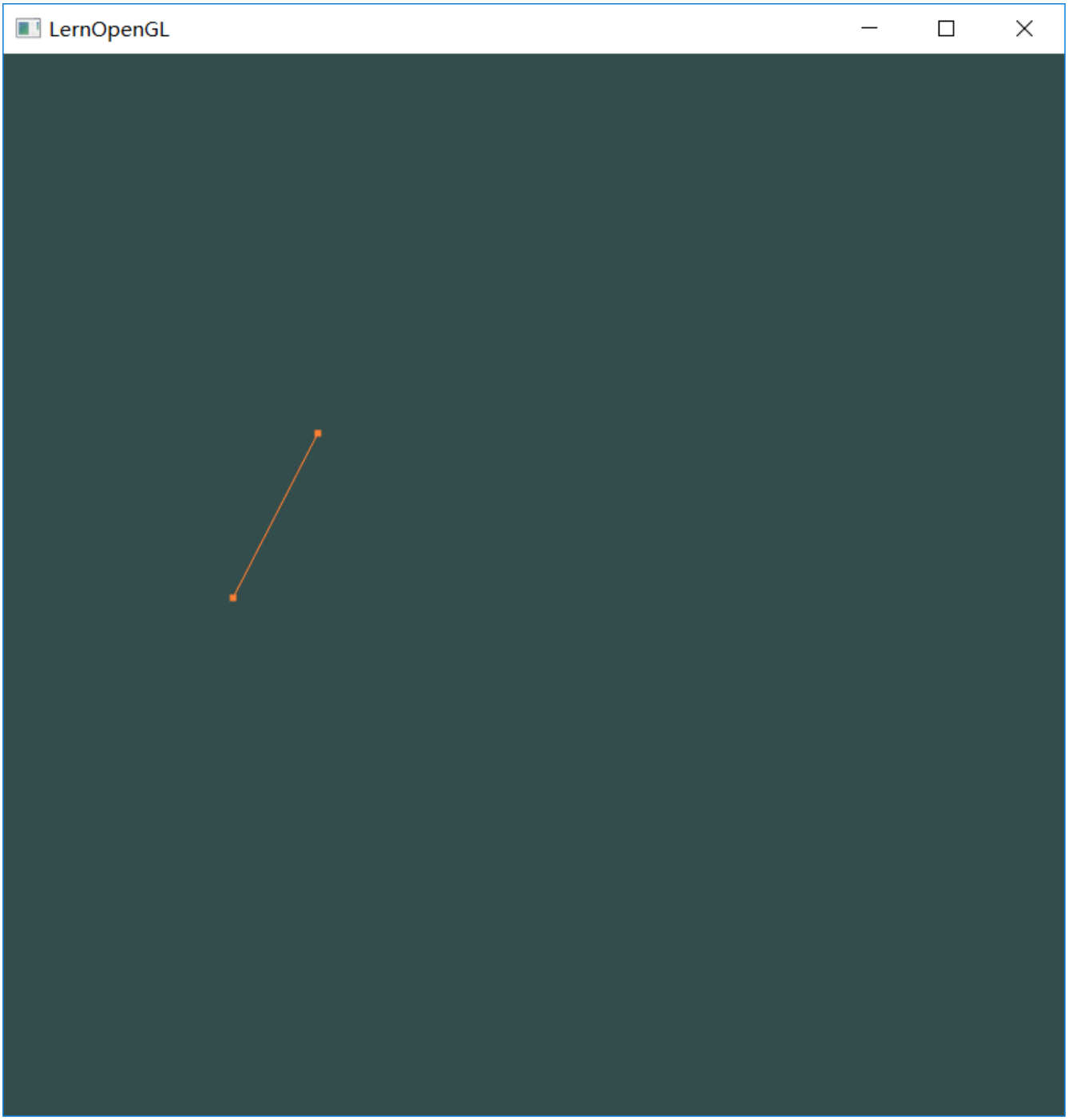
Basic

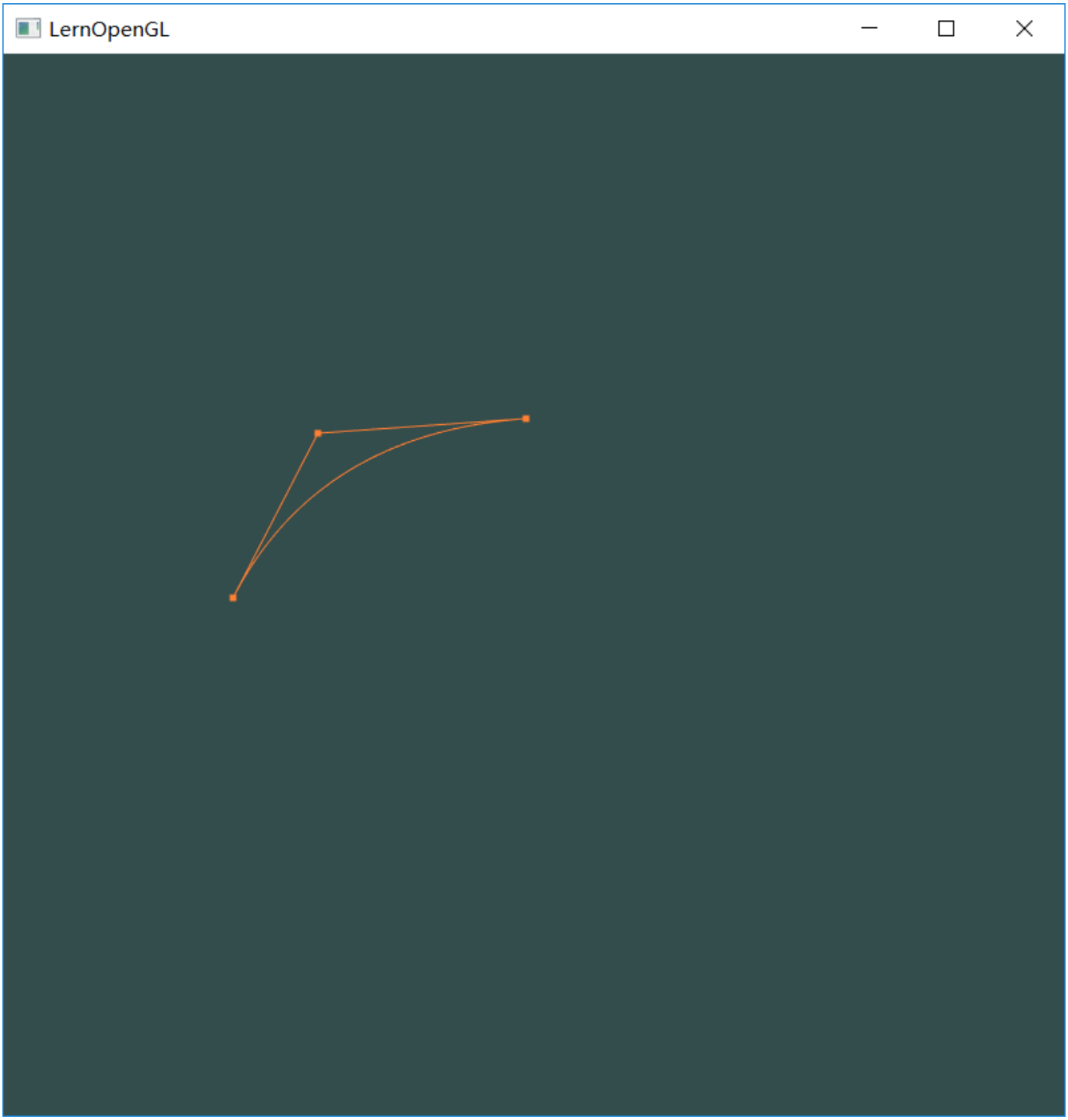
要求

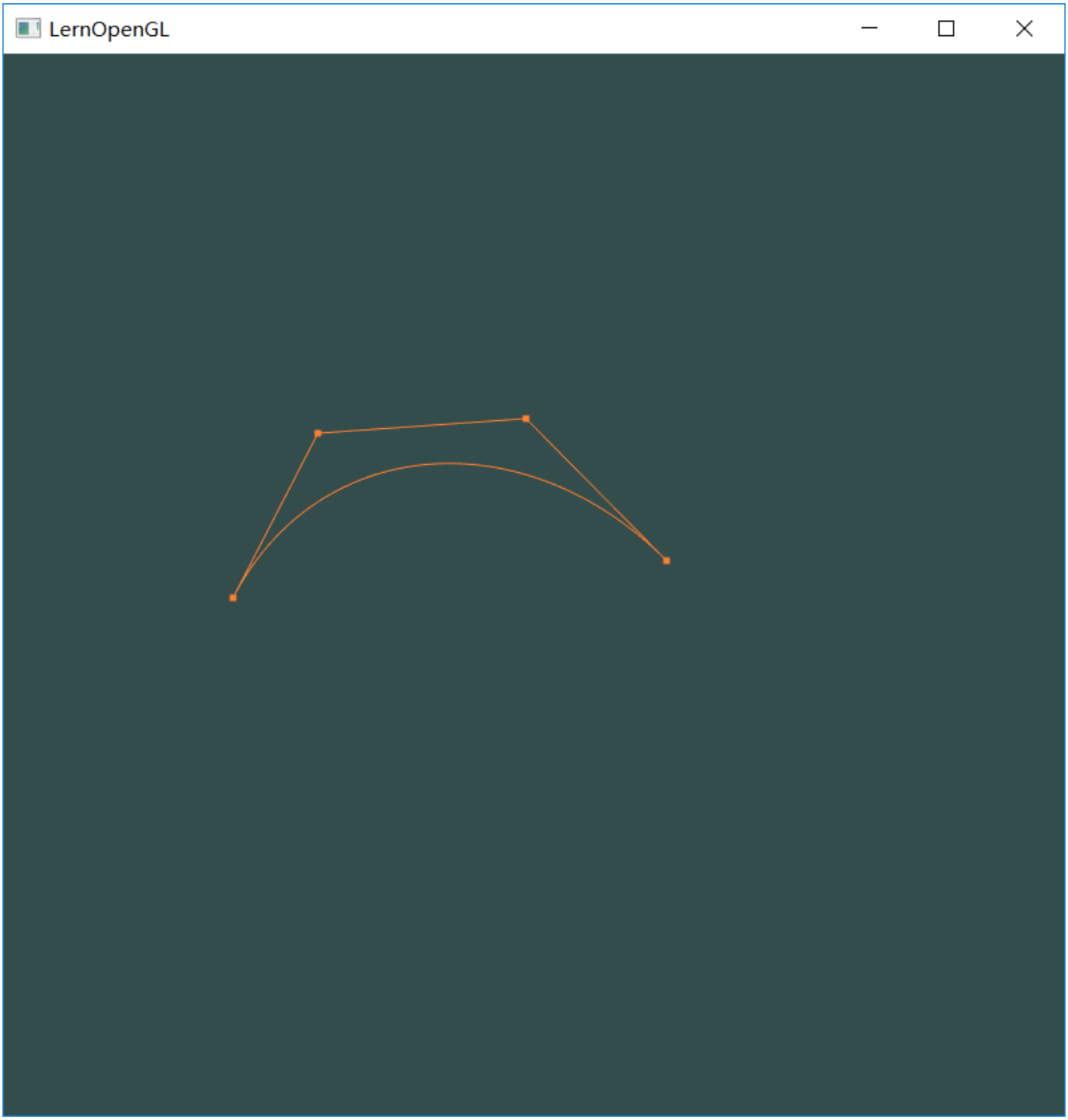
1. 用户能通过左键点击添加Bezier曲线的控制点，右键点击则对当前添加的最后一个控制点进行消除
2. 工具根据鼠标绘制的控制点实时更新Bezier曲线。

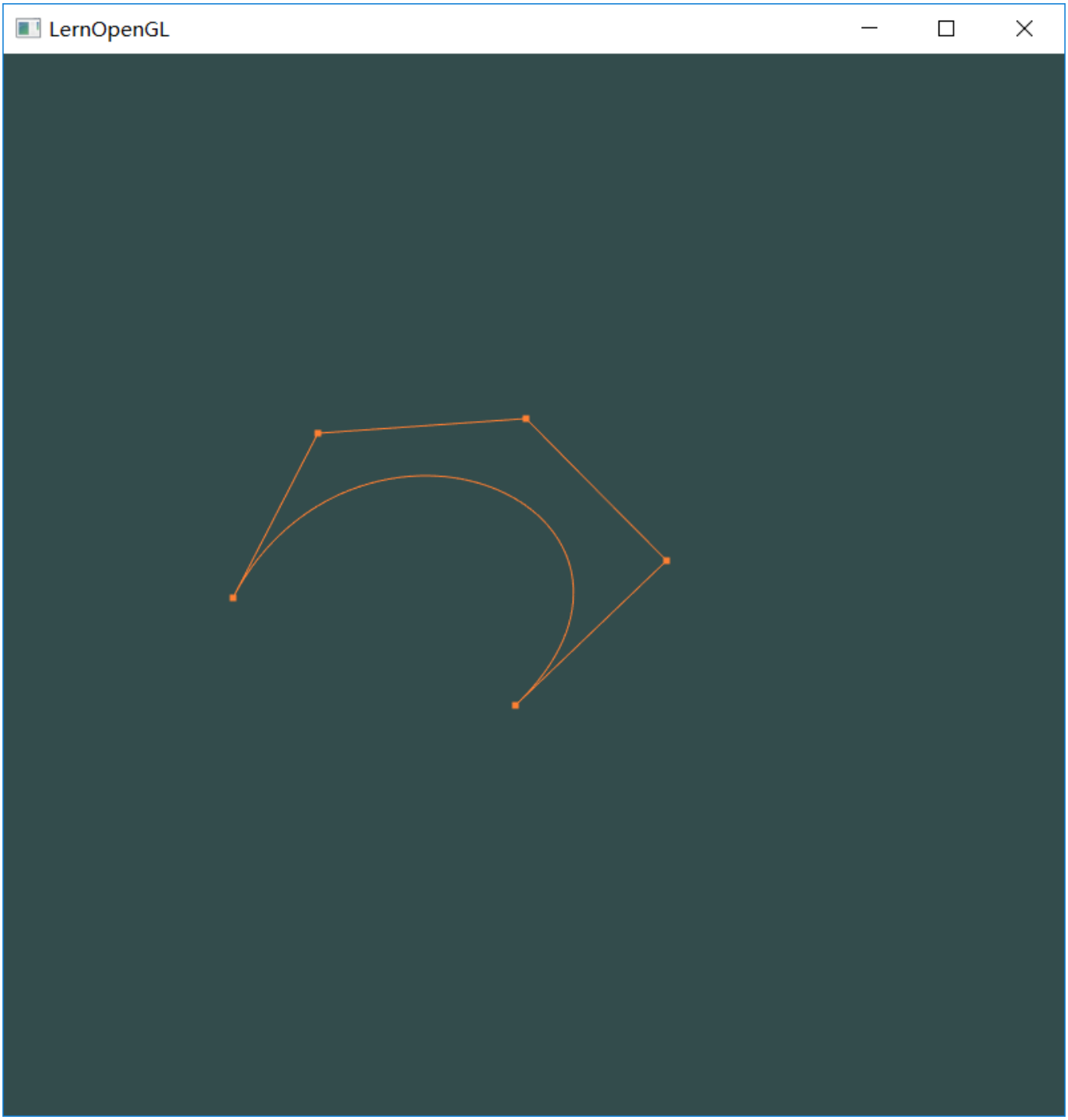
实验结果


LernOpenGL


LernOpenGL


LernOpenGL


LernOpenGL



实现原理

1. 点击监听函数
    1. `void mouseButtonCallBack(GLFWwindow* window, int button, int action, int mods)`
    2. 获取屏幕坐标，并存到控制点的信息中

```
glfwGetCursorPos(window, &nowPosX, &nowPosY);
nowPosX = (-1) + (nowPosX / SCR_WIDTH) * 2;
nowPosY = (-1)*((-1) + (nowPosY / SCR_HEIGHT) * 2);
glm::vec3 nowPos((float)startX, (float)startY, 0.0f);
pos.push_back(nowPos);
```

3. 将控制点的信息存到缓存中

```
double verticesLine[] = {
    nowPosX, nowPosY, 0.0f,
    nowPosX, nowPosY, 0.0f
};
//printf("%f %f\n", windowPosX, windowPosY);
glBindBuffer(GL_ARRAY_BUFFER, lineVBO[lineVBO.size() - 1]);
glBufferData(GL_ARRAY_BUFFER, sizeof(verticesLine), verticesLine, GL_STATIC_DRAW);
glBindVertexArray(lineVAO[lineVAO.size() - 1]);
glVertexAttribPointer(0, 3, GL_DOUBLE, GL_FALSE, 3 * sizeof(double), (void*)0);
glEnableVertexAttribArray(0);
//解绑
glBindBuffer(GL_ARRAY_BUFFER, 0);
glBindVertexArray(0);
```

4. 将控制点和点之间的连线画出

```
if (lineVAO.size() == 1) {
    glBindVertexArray(lineVAO[0]);
    glPointSize(5);
    glDrawArrays(GL_POINTS, 0, 1);
}
else {
    for (vector<unsigned int>::iterator iter = lineVAO.begin(); iter !=
lineVAO.end(); ++iter)
    {
        glBindVertexArray(*iter);
        glDrawArrays(GL_LINE_STRIP, 0, 2);
        glPointSize(5);
        glDrawArrays(GL_POINTS, 0, 2);
    }
}
```

5. 如果点击的是右键，将最后一个点从存储的控制点中删除，并更新存入缓存中的信息（由于我是每一条线存在一个VAO和VBO中的，因此只需要删除最后一个即可）

```
pos.pop_back();
lineVAO.pop_back();
lineVBO.pop_back();
```

6. 如果点击的是右键，对应新的控制点，重新绘制曲线

2. Bezier曲线
    1. 计算公式：$B_{i,n}(t) = C_n^i t^i (1-t)^{n-1}$，$Q_{i,n}(t) = \sum_{i=0}^{n} P_i B_{i,n}(t)$其中，$t \in [0,1]$
    2. 做法：
        1. 让t以0.01的步长取0-1之间的数，通过上面写出的公式分别计算B和Q的值，将点存起来（由于模拟的方式是两点间连线，因此中间的点要存多一次）

```
double t = 0;
```

```
for (int i = 0; i <= 100; i++) {
    float x = 0.0f, y = 0.0f;
    for (int j = 0; j < pos.size(); j++) {
        float c = getCombinatorialNumber(pos.size() - 1, j);
        x += c * pow(t, j)*pow(1 - t, pos.size() - 1 - j)*pos[j].x;
        y += c * pow(t, j)*pow(1 - t, pos.size() - 1 - j)*pos[j].y;
    }
    bezierPos.push_back(x);
    bezierPos.push_back(y);
    bezierPos.push_back(0.0f);
    if (i != 0 && i != 100) {
        bezierPos.push_back(x);
        bezierPos.push_back(y);
        bezierPos.push_back(0.0f);
    }
    t += 0.01;
}
```

2. 将上面存储的点存入缓存中

```
glBindBuffer(GL_ARRAY_BUFFER, bezierVBO);
glBufferData(GL_ARRAY_BUFFER, bezierPos.size() * sizeof(float),
bezierPos.data(), GL_STATIC_DRAW);
glBindVertexArray(bezierVAO);
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(float), (void*)0);
glEnableVertexAttribArray(0);
//解绑
glBindBuffer(GL_ARRAY_BUFFER, 0);
glBindVertexArray(0);
```

3. 将Bezier曲线绘制出来

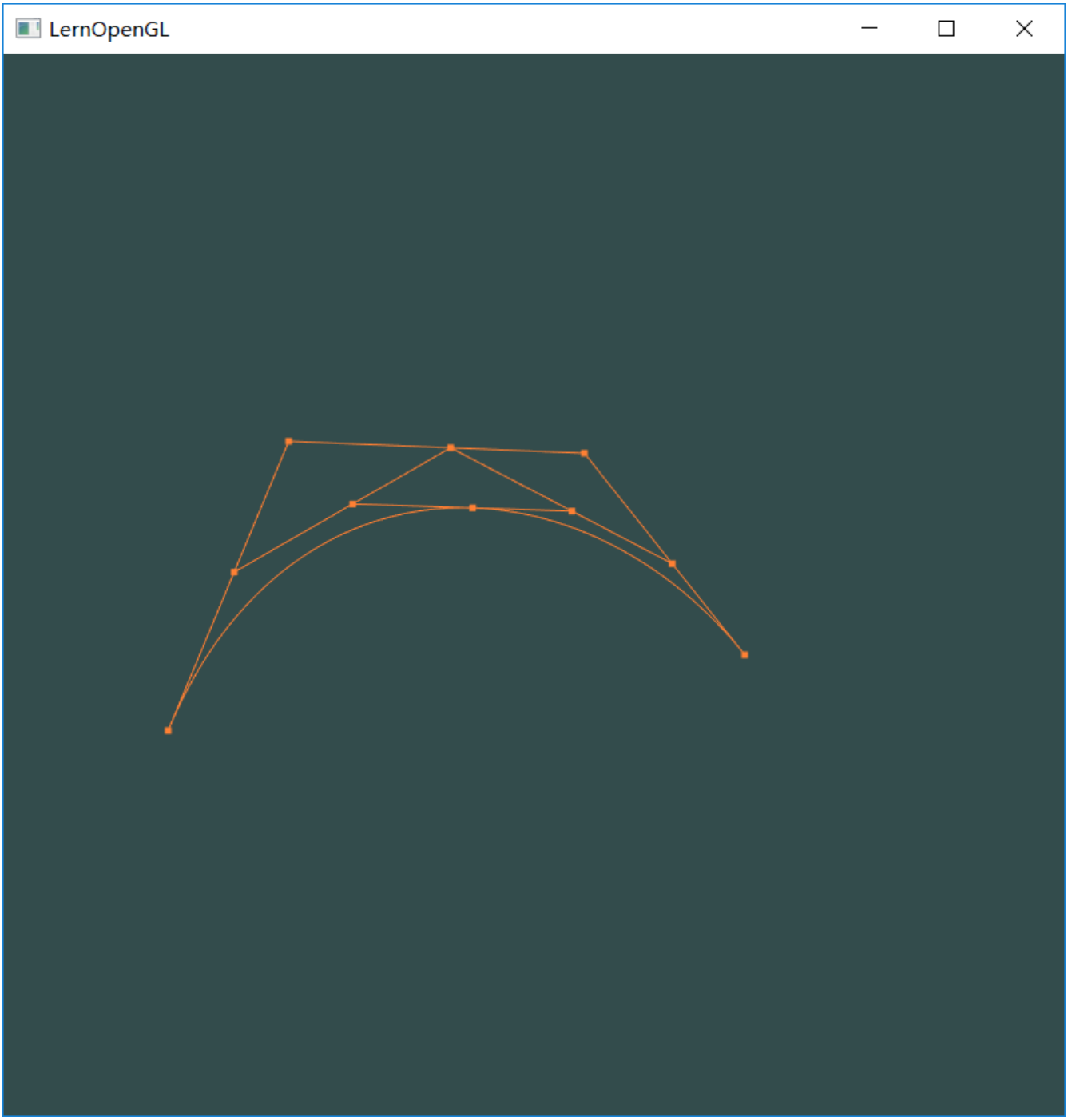
```
glBindVertexArray(bezierVAO);
glDrawArrays(GL_LINES, 0, 200);
```

Bonus

要求

动态地呈现Bezier曲线的生成过程。

实验截图



实现原理

1. 在点击键盘`s`键后，开始显示动画，并存储点击时间，同时点击不再有响应

```
if (glfwGetKey(window, GLFW_KEY_S) == GLFW_PRESS) {
    if (!isShow) {
        isShow = true;
        currTime = glfwGetTime();
    }
}
```

2. 通过`glfwGetTime`获取当前时间，通过经过时间获得当前的t

```
double t = (glfwGetTime() - currTime)*0.08;
```

3. 在存储的控制点间的连线处，获得t等分点，将这些点的信息存储到一个顶点的vector1中。同时将这一层的顶点信息存到另一个vector2中。

```
for (int i = 0; i < pos.size() - 1; i++) {
    float x = (pos[i + 1].x - pos[i].x)*t + pos[i].x;
    float y = (pos[i + 1].y - pos[i].y)*t + pos[i].y;
    linePos.push_back(x);
    linePos.push_back(y);
    linePos.push_back(0.0f);
    if (i != 0 && i != pos.size() - 2) {
        linePos.push_back(x);
        linePos.push_back(y);
        linePos.push_back(0.0f);
    }
    testPos.push_back(glm::vec3(x,y,0.0f));
}
```

4. 获取上一层顶点连线的t等分点，并将这些信息存到vector1中，同时，遍历完上一层的顶点后，清空vector2，将这一层的信息存到vector2中。不断重复这一步，直到上一层的顶点个数为2个。

```
while (testPos.size() > 1) {
    vector<glm::vec3> test2Pos;
    for (int i = 0; i < testPos.size() - 1; i++) {
        float x = (testPos[i + 1].x - testPos[i].x)*t + testPos[i].x;
        float y = (testPos[i + 1].y - testPos[i].y)*t + testPos[i].y;
        linePos.push_back(x);
        linePos.push_back(y);
        linePos.push_back(0.0f);
        if (i != 0 && i != testPos.size() - 2) {
            linePos.push_back(x);
            linePos.push_back(y);
            linePos.push_back(0.0f);
        }
        test2Pos.push_back(glm::vec3(x, y, 0.0f));
    }
    testPos.clear();
    testPos = test2Pos;
}
```

5. 将vector1中的顶点信息存到缓存中

```
glBindBuffer(GL_ARRAY_BUFFER, showVBO);
glBufferData(GL_ARRAY_BUFFER, linePos.size() * sizeof(float), linePos.data(),
GL_STATIC_DRAW);
glBindVertexArray(showVAO);
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(float), (void*)0);
glEnableVertexAttribArray(0);
//解绑
glBindBuffer(GL_ARRAY_BUFFER, 0);
glBindVertexArray(0);
glBindVertexArray(showVAO);
```

6. 绘制顶点和顶点间的连线

```
glBindVertexArray(showVAO);
glDrawArrays(GL_LINES, 0, linePos.size()/3);
glPointSize(5);
glDrawArrays(GL_POINTS, 0, linePos.size() / 3);
```

7. 当$t \geq 1$时，将停止动画，恢复点击屏幕的响应

```
if (t >= 1) {
    isShow = false;
}
```