计算机图形学 HW4-1 报告

贝塞尔曲线的绘制

1 开发环境

本作业的开发环境是MinGW，完成作业后使用vs 2017编译并提交release版。 基于GLFW第三方库进行窗口管理。

2 程序实现说明

2.1 概述

由于本程序在OPENGL核心模式下进行编程，因此实现的方法有一些复杂，下面解释一下程序的细节。

2.2 贝塞尔曲线上的点坐标的计算

2.2.1 关键函数 get_point

该函数的作用是计算（x1,y1) (x2,y2) 两点间比例为t处的点的坐标。 该函数在计算贝塞尔曲线上的点的坐标时需要多次调用。

```
void get_point(float x1, float y1, float x2, float y2, float t, float & x, float & y){
    x = x1 + (x2-x1)*t;
    y = y1 + (y2-y1)*t;
}
```

2.2.2 计算三阶贝塞尔曲线上的点的坐标

在计算的过程中，遵循三阶贝塞尔曲线的定义进行计算。对于已知的四个控制点，对每两个控制点，根据当时的比例 `t` ，计算出三个点(x11,y11),(x22,y22),(x33,y33)。对这三个点，进行类似的计算，得到两个点(x111,y111),(x222,y222)，最后对这两个点再计算比例为t处的点的坐标，即为贝塞尔曲线上的一点。当t从0渐渐累加到1的时候，就能够计算出贝塞尔曲线上的所有点。

```
void init(float x1, float y1, float x2, float y2, float x3, float y3, float x4, float y4){
     // ......
```

```
            float t = 0;
        float interval = 0.001;
        while (t <= 1){
            float x11,y11;
            float x22,y22;
            float x33,y33;
            get_point(x1, y1, x2, y2, t, x11, y11);
            get_point(x2, y2, x3, y3, t, x22, y22);
            get_point(x3, y3, x4, y4, t, x33, y33);

            float x111, y111;
            float x222, y222;
            get_point(x11, y11, x22, y22, t, x111, y111);
            get_point(x22, y22, x33, y33, t, x222, y222);

            float x1111, y1111;
            get_point(x111, y111, x222, y222, t, x1111, y1111);

            vertices[3*this->num_point] = x1111;
            vertices[3*this->num_point+1] = y1111;
            vertices[3*this->num_point+2] = 0;
            this->num_point++;
            t += interval;
        }
    // .......
}
```

2.3 场景Scene类的实现

在本程序中，最关键的设计为Scene类的设计。该类提供如下接口，在main.cpp中只需调用 `add_point` 往场景中增加点，就能够画点，并根据场景当前的点的数量自动判断是否需要画线，以及绘制贝塞尔曲线。

```
/**
 * @brief 场景类，用于显示点和线
 *
 */
class Scene{
    /**
     * @brief 使用当前的顶点数据和索引数据更新缓冲区
     * 若到达四个点，则更新本地贝塞尔曲线数据
     */
    void update();

    /**
     * @brief 在场景中增加一个点
     */
    void add_point(float x, float y, float z = 0);

    /**
     * @brief 如果场景中有四个点及以上，则视为满
     */
    int is_full();

    /**
     * @brief 如果场景没有点，作为空
     */
    int is_empty();
```

```
    /**
     * @brief 绘图
     *
     * 若该场景有三个点或者更少，则只画点
     * 一旦到达四个点，更改为画线模式并且调用贝塞尔曲线对象的draw函数
     */
    void draw();
}
```

其中，由于在 `main.cpp` 中的主渲染循环中只调用了draw函数，下面简单解释下draw函数的实现。实现的逻辑很简单，如果该场景没有点，则不进行绘图，如果场景中有1-3个点，则只画这三个点；一旦到达四个点，该函数会做一下两件事情

1. 将绘图模式更改为 `GL_LINE_STRIP` , 以便画出连接的直线
2. 调用贝塞尔曲线对象 `this->my_bezier.draw();` 绘制贝塞尔曲线，该曲线的相关数据在调用draw函数前已经通过调用 `Scene.update` 函数更新了。

```
    /**
     * @brief 绘图
     *
     * 若该场景有三个点或者更少，则只画点，否则画线
     *
     */
    void draw(){
        if (this->is_empty()){
            return ;
        }
        else if(this->is_full()){
            /* 如果满了就画线 */
            glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
            glBindVertexArray(this->VAO);
            // glDrawElement 这个函数画不了点
            glDrawArrays(GL_LINE_STRIP, 0, this->num_point);
            this->my_bezier.draw();
        }
        else {
            /* 如果没有满就画点 */
            glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
            glBindVertexArray(this->VAO);
            // glDrawElement 这个函数画不了点
            glDrawArrays(GL_POINTS,0, this->num_point);
        }
    }
```

2.4 鼠标交互的实现

在用户交互的程序编写中，我设计了两个回调函数，他们分别有其重要的用途。

在鼠标进行移动的时候，会调用 `curse_pos_callback` ，该函数能够将鼠标当前所在位置转换为opengl坐标系的坐标，并存储在两个全局变量 `curse_x` 和 `curse_y` 中。

```
/**
 * @brief 鼠标移动的回调函数
 *
 * @param window
 * @param x
 * @param y
 */
void curse_pos_callback(GLFWwindow *window, double x, double y)
{
    // std::cout << "(pos:" << x << "," << y << ")" << std::endl;
    curse_x = 2*(x / SCR_WIDTH - 0.5);
    curse_y = 2*(0.5 - y / SCR_HEIGHT);
}
```

在鼠标进行点击的时候，会调用 `mouse_button_callback` 函数，该函数能够将当前鼠标所在的坐标作为场景中的一个新点，增加进场景中。

```
/**
 * @brief 鼠标点击的回调函数
 *
 * @param window
 * @param button
 * @param action
 * @param mods
 */
void mouse_button_callback(GLFWwindow* window, int button, int action, int mods)
{
    if (button == GLFW_MOUSE_BUTTON_LEFT && action == GLFW_PRESS)
        s1.add_point(curse_x, curse_y);
}
```

3 程序运行方法

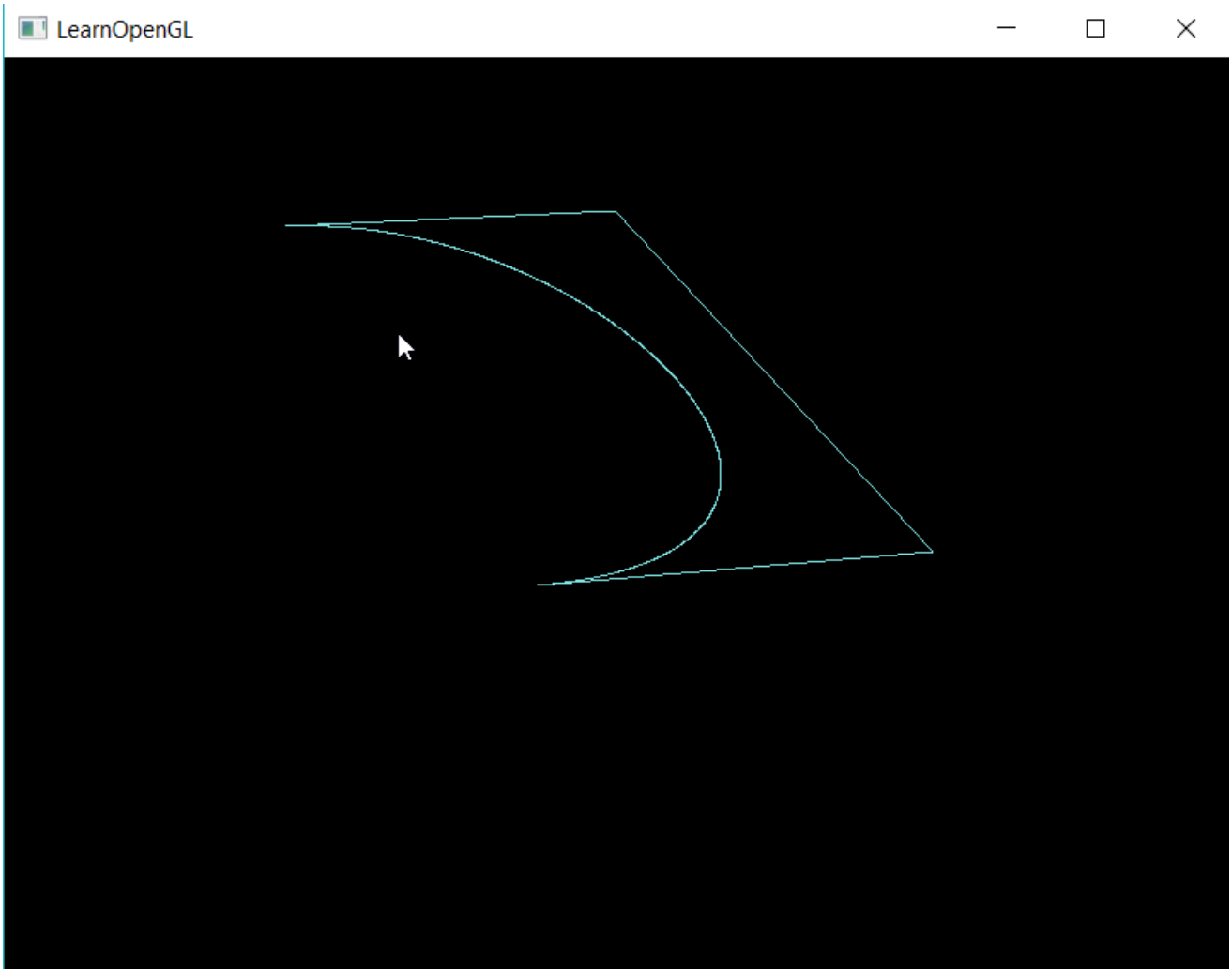
3.1 编译方法

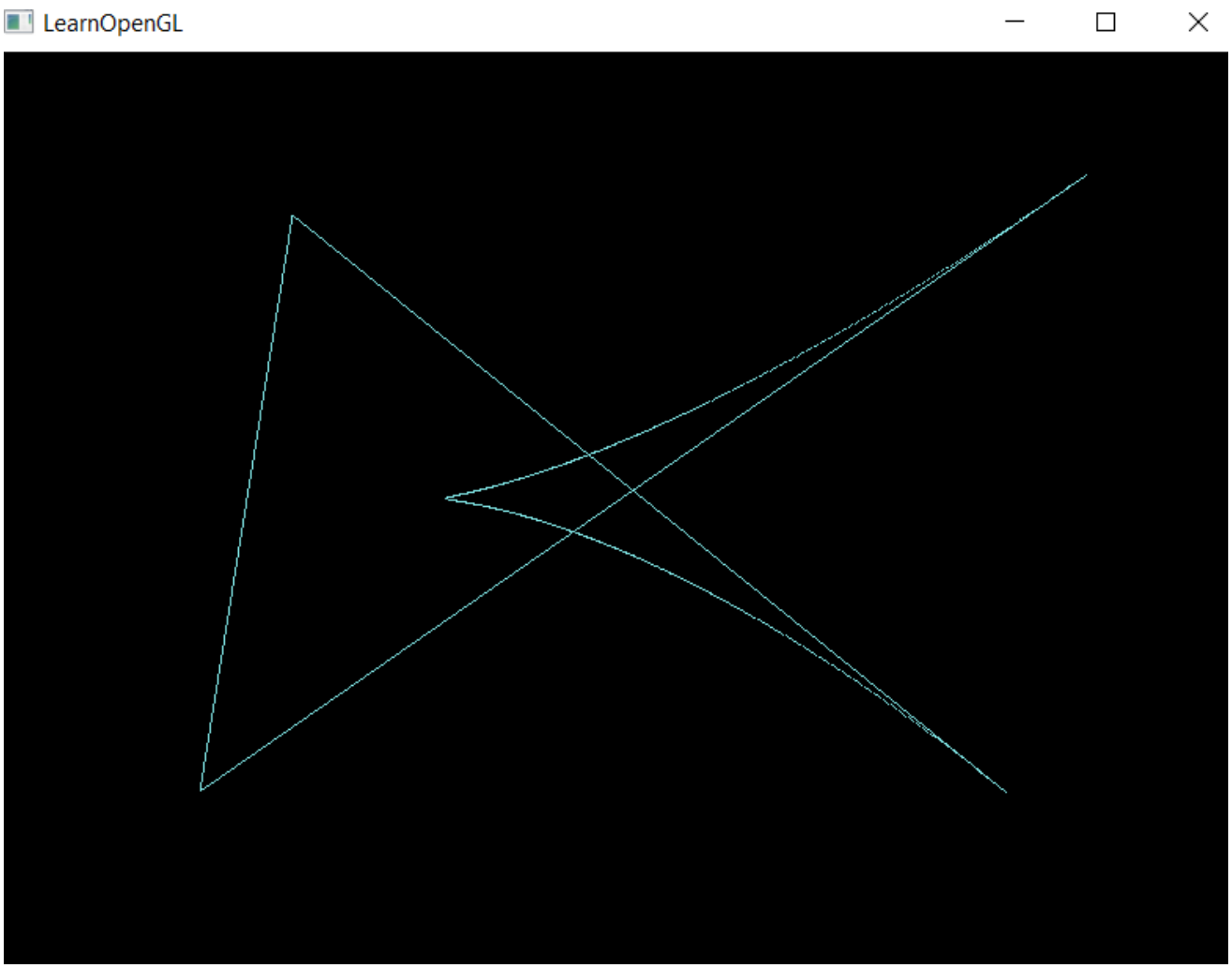
在代码目录中，可使用cmake工具进行编译。

3.2 运行方法

使用鼠标在软件界面中进行点击，每点击四个点即可画出一条由该四个控制点绘制的三阶贝塞尔曲线。

4 程序运行截图


LearnOpenGL



5 程序运行录屏

可见作业目录下的 `HW4-1-bezier.mp4`