# Homework 8 - Bezier Curve

## 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

### 1. 求贝塞尔曲线

贝塞尔曲线的基本公式：

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

曲线上的每个点都是在按照上述基本公式生成的线段的不断迭代后的最后一个点，比如有p0，p1，p2，p3四个点，求t=T时的贝塞尔曲线上的点，则：

- 根据基本公式生成p0，p1在t=T时的中间点p4，根据基本公式生成p1，p2在t=T时的中间点p5，根据基本公式生成p2，p3在t=T时的中间点p6.
- 同理，根据基本公式生成p4，p5在t=T时的中间点p7，根据基本公式生成p5，p6在t=T时的中间点p8.
- 同理，根据基本公式生成p7，p8在t=T时的中间点p9.
- p9即为t=T时贝塞尔曲线上的点

因此，可以将计算曲线的过程中生成的点分为三类：

- 点击产生的点（p0，p1，p2，p3）：clickPoints
- 基本公式计算出的端点（p0，p1，p2，p3，p4，p5，p6，p7，p8）：endPoints
- 所求的贝塞尔曲线上的点（p9）：drawPoints

当clickPoints有n个点时，端点生成时可以分为（n-1）层，在上述例子中就有三层：（0，1，2，3）（4，5，6）（7，8），而计算时只需要前三层即可。

根据上述逻辑，基本公式：

```
glm::vec2 getInterPoint(glm::vec2 p0, glm::vec2 p1, float t) {
    float x = (1 - t) * p0.x + t * p1.x;
    float y = (1 - t) * p0.y + t * p1.y;
    return glm::vec2(x, y);
}
```

求贝塞尔曲线代码：

```
//t
for (float t = 0; t < 1.001f; t += 0.001f) {
    //会有clickPointNum-1层进行计算
    int layerNum = clickPointNum - 1;
    //endPoints中的下标
    int current_index = 0;
    //每层endPoints中点的数量，第一层数量与clickPoints数量相同，每层数量-1
    int point_num_in_layer = clickPointNum;
```

```
    for (int i = 0; i < layerNum; i++) {
        point_num_in_layer--;
        for (int j = 0; j < point_num_in_layer; j++) {
            glm::vec2 p0 = endPoints[current_index];//p0
            glm::vec2 p1 = endPoints[current_index + 1];//p1
            //根据基本公式求p0和p1之间的点
            endPoints.push_back(getInterPoint(p0, p1, t));
            endPointNum++;
            current_index++;
        }
        //下标跳到下一层的第一个点
        current_index++;
    }
    //最后一层求的点就是贝塞尔曲线上的点
    drawPoints.push_back(endPoints[endPointNum - 1]);
    drawPointNum++;

    //endPoints初始化
    initEndPoints();
}
```

这样贝塞尔曲线上的点就都在drawPoints中，画时全部放进顶点数组中即可。

## 2. 鼠标响应与实时更新

鼠标响应函数：

```
//鼠标点击事件
void mouse_callback(GLFWwindow* window, int button, int action, int mods) {
    if (action == GLFW_PRESS) switch (button)
    {
    //鼠标左键
    case GLFW_MOUSE_BUTTON_LEFT:
        //将点击的鼠标位置加入到clickPoints中
        clickPointNum++;
        clickPoints.push_back(mousePos);
        //更新贝塞尔曲线
        getBezier();
        ...
        break;
    //鼠标右键
    case GLFW_MOUSE_BUTTON_RIGHT:
        if (clickPointNum > 0) {
            //pop最后加入的clickPoint
            clickPointNum--;
            clickPoints.pop_back();
            //更新贝塞尔曲线
            getBezier();
            ...
        }
        break;
    default:
```

```
            return;
        }
        return;
    }

    //鼠标位置事件
    void pos_callback(GLFWwindow* window, double xpos, double ypos) {
        //注意将鼠标位置转化为[-1, 1]的坐标位置
        mousePos.x = ((2*xpos) / width) - 1;
        mousePos.y = 1 - ((2*ypos) / height);
    }
```

## Bonus

1. 可以动态地呈现Bezier曲线的生成过程。

动态呈现生成过程，其实就是把计算过程中的endPoints（去除前几个clickPoints）连接的线段画出来，然后按照一定频率更新t，相应的线段们就会被画出来。

如果将需要的点称为dynamicPoints（例子中的p4，p5，p6，p7，p8），顶点数组为vertics_dynamic。

### 1. 求dynamicPoints

主要过程就是求endPoints，和上面的类似。

```
//只有两个clickPoints时不需要动态生成
if (clickPointNum <= 2)
    return;
//dynamicPoints的数量
int dynamicPointNum = (clickPointNum / 2.0f - 1)*(clickPointNum + 1);
//顶点数组
float* vertics_dynamic = new float[dynamicPointNum * 3];
//层数是n-2
int layerNum = clickPointNum - 2;
//endPoints下标
int current_index = 0;
//顶点数组下标
int dynamic_index = 0;
//每层endPpints的点的数量
int point_num_in_layer = clickPointNum;

for (int i = 0; i < layerNum; i++) {
    point_num_in_layer--;
    for (int j = 0; j < point_num_in_layer; j++) {
        //求中间点pi
        glm::vec2 p0 = endPoints[current_index];
        glm::vec2 p1 = endPoints[current_index + 1];
        glm::vec2 pi = getInterPoint(p0, p1, dynamic_t);
        //将pi坐标放入顶点数组中，z轴为0
        vertics_dynamic[dynamic_index * 3] = pi.x;
        vertics_dynamic[dynamic_index * 3 + 1] = pi.y;
        vertics_dynamic[dynamic_index * 3 + 2] = 0.0f;
```

```
            endPoints.push_back(pi);
            dynamic_index++;
            current_index++;
        }
        current_index++;
    }
    initEndPoints();
```

### 2. 按层画线

并不是将数组中所有点连起来，而是将每层的点连起来：

```
layerNum = clickPointNum - 2;
//每层的点数
point_num_in_layer = clickPointNum - 1;
//每层的第一个点的下标
current_index = 0;
for (int i = 0; i < layerNum; i++) {
    //白色
    shader.use();
    shader.setVec3("spriteColor", glm::vec3(1.0f, 1.0f, 1.0f));

    //画出数组中每层的点连成的线段
    glBindVertexArray(VAO_dynamic);
    glDrawArrays(GL_LINE_STRIP, current_index, point_num_in_layer);

    current_index += point_num_in_layer;
    point_num_in_layer--;
}
```

### 3. 定时更新

```
//clock_num为更新频率，dynamic_t为求基本公式时用到的t，[0，1]更新完后又会回到0。
if (clock > clock_num) {
    if (dynamic_t <= 0.99) {
        dynamic_t += 0.01;
        clock = 0;
    }
    else {
        dynamic_t = 0;
        clock = 0;
    }
}
else {
    clock++;
}
```

上述过程均在一个函数中，在main的主loop中调用这个函数即可。

效果见gif文件。