BASIC：

动态效果见附视频文件 hw8.mp4。

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
   要求已实现，效果见视频文件。
2. 工具根据鼠标绘制的控制点实时更新 Bezier 曲线
   要求已实现，效果见视频文件。

Bonus

1. 可以动态地呈现 Bezier 曲线的生成过程
   要求已实现，效果见视频文件。

代码

附代码文件（本次作业的核心代码在 hw8.cpp 文件内）

1. Bezier 曲线的绘制方式

```
if (flush)
{
    for (size_t i = 0; i < curveSize; i++)
    {
        float tempt = (float)i / (float)curveSize;
        float x = 0, y = 0;
        for (size_t j = 0; j < sizeOfControlVec; j++)
        {
            int n = sizeOfControlVec - 1;
            float proportion = factorial(n) / (factorial(j)*factorial(n - j))*pow(tempt, j)*pow(1 - tempt, n - j);
            x += controlVec[j].first*proportion;
            y += controlVec[j].second*proportion;
        }
        curveVec[2 * i] = x;
        curveVec[2 * i + 1] = y;
    }
    flush = false;
}
```

$$Q(t)=\sum_{i=0}^{n}P_iB_{i,n}(t),\quad t\in[0,1]$$

$$B_{i,n}(t)=\frac{n!}{i!(n-i)!}t^{i}(1-t)^{n-i}\ ,\quad i=0,\quad 1\ldots n$$

直接根据 Bezier 曲线的参数方程采样计算出曲线上的点。

2. 动态呈现曲线与控制点的关系

主要思路获取当前时间，并映射到以上函数的自变量 t，使其循环从 0 变化到 1，每次绘制时逐段按比值插值求出下一阶的控制点，然后使用 draw 函数将每阶段求出的控制点连线。

```
for (size_t i = 0; i < sizeOfControlVec; i++)
{
    float *temp = new float[2 * (sizeOfControlVec - i)];
    for (size_t j = 0; j < sizeOfControlVec - i; j++)
    {
        if (i == 0)
        {
            temp[2 * j] = controlVec[j].first;
            temp[2 * j + 1] = controlVec[j].second;
        }
        else
        {
            temp[2 * j] = (1 - t)*sideVec[2 * j] + t * sideVec[2 * (j + 1)];
            temp[2 * j + 1] = (1 - t)*sideVec[2 * j + 1] + t * sideVec[2 * (j + 1) + 1];
        }
    }
    draw(temp, radius, sizeOfControlVec - i);
    delete[] sideVec;
    sideVec = temp;
}
```

Draw 连线函数

```
void draw(float *vertices, float radius, int sizeOfVec) {
    for (size_t i = 0; i < sizeOfVec; i++)
    {
        float vec[8];
        // 右上
        vec[0] = vertices[2 * i] + radius;
        vec[1] = vertices[2 * i + 1] + radius;
        // 右下
        vec[2] = vertices[2 * i] + radius;
        vec[3] = vertices[2 * i + 1] - radius;
        // 左上
        vec[4] = vertices[2 * i] - radius;
        vec[5] = vertices[2 * i + 1] + radius;
        // 左下
        vec[6] = vertices[2 * i] - radius;
        vec[7] = vertices[2 * i + 1] - radius;
        unsigned int indices[6] = {
            0, 1, 2,
            1, 2, 3
        };
        glBufferData(GL_ARRAY_BUFFER, sizeof(vec), vec, GL_DYNAMIC_DRAW);
        glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(indices), indices, GL_STATIC_DRAW);
        glVertexAttribPointer(0, 2, GL_FLOAT, GL_FALSE, 2 * sizeof(float), (void*)0);
        glEnableVertexAttribArray(0);
        glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_INT, 0);
        if (i != 0)
        {
            float lineVec[4];
            // 前点
            lineVec[0] = vertices[2 * (i - 1)];
            lineVec[1] = vertices[2 * (i - 1) + 1];
            // 当前点
            lineVec[2] = vertices[2 * i];
            lineVec[3] = vertices[2 * i + 1];
            glBufferData(GL_ARRAY_BUFFER, sizeof(lineVec), lineVec, GL_DYNAMIC_DRAW);
            glVertexAttribPointer(0, 2, GL_FLOAT, GL_FALSE, 2 * sizeof(float), (void*)0);
            glEnableVertexAttribArray(0);
            glDrawArrays(GL_LINES, 0, 2);
        }
    }
}
```