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

Bezier curve:

Q(t)表示绘制曲线上的点的坐标，Pi 表示控制点坐标，Bi,n(t)表示伯恩斯坦基函数，

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

Opengl 实现：

```
int controlvertexNum = 0;
const int maxcontrolNum = 10;
const int curveVetexTotalNum = 400;
Point2D  controlVertices[maxcontrolNum];
float  vertices[(curveVetexTotalNum + maxcontrolNum) * 3] = { 0 };

//mouse
float mx;
float my;
```

全局变量：目前控制点数目，最大数目、曲线预期绘制点数、存放控制点的矩阵、存放曲线点的矩阵、鼠标的 x、y 坐标

```
// first, configure the cube's VAO (and VBO)
unsigned int VBO, cubeVAO;
glGenVertexArrays(1, &cubeVAO);
glGenBuffers(1, &VBO);

glBindBuffer(GL_ARRAY_BUFFER, VBO);
glBufferData(GL_ARRAY_BUFFER, sizeof(float)*((curveVetexTotalNum + controlvertexNum) * 3), vertices, GL_STATIC_DRAW);

glBindVertexArray(cubeVAO);

// position attribute
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(float), (void*)0);
glEnableVertexAttribArray(0);

// render the cube
glBindVertexArray(cubeVAO);
if (controlvertexNum > 0) {
    glDrawArrays(GL_POINTS, 0, curveVetexTotalNum+controlvertexNum);
}
```

在 while 循环内每一次迭代都重新提交数据到缓存上

```
void mouse_button_callback(GLFWwindow* window, int button, int action, int mods)
{
    if (action == GLFW_PRESS) switch (button)
    {
    case GLFW_MOUSE_BUTTON_LEFT:
        addControlVertex();
        break;
    case GLFW_MOUSE_BUTTON_RIGHT:
        deleteControlVertex();
        break;
    default:
        return;
    }
    return;
}
```

鼠标回调，监听鼠标事件，左右击分别调用增加和删除控制点函数

```
void cursor_position_callback(GLFWwindow* window, double x, double y) {
    mx = float((x-SCR_WIDTH/2)/(SCR_WIDTH/2));
    my = float(-(y - SCR_HEIGHT / 2) / (SCR_HEIGHT / 2));
}
```

鼠标坐标回调，获取鼠标当前坐标并调整映射到 opengl 视口的坐标系统上：屏幕中央是(0,0)且 y 正轴向上

```
void addControlVertex() {
    //std::cout<<("add")<<std::endl;
    //std::cout << ("Mouse position:(") << mx << ", " << my << ")" << std::endl;
    if (controlvertexNum < maxcontrolNum) {
        controlVertices[controlvertexNum++].set(mx,my);
        updateCurveVertex();
    }
}
void deleteControlVertex() {
    //std::cout << ("delete") << std::endl;
    //std::cout << ("Mouse position:(") << mx << ", " << my << ")" << std::endl;
    if (controlvertexNum > 0) {
        controlvertexNum--;
        updateCurveVertex();
    }
}
```

增加控制点和删除控制点，先增加控制点，然后更新曲线的点

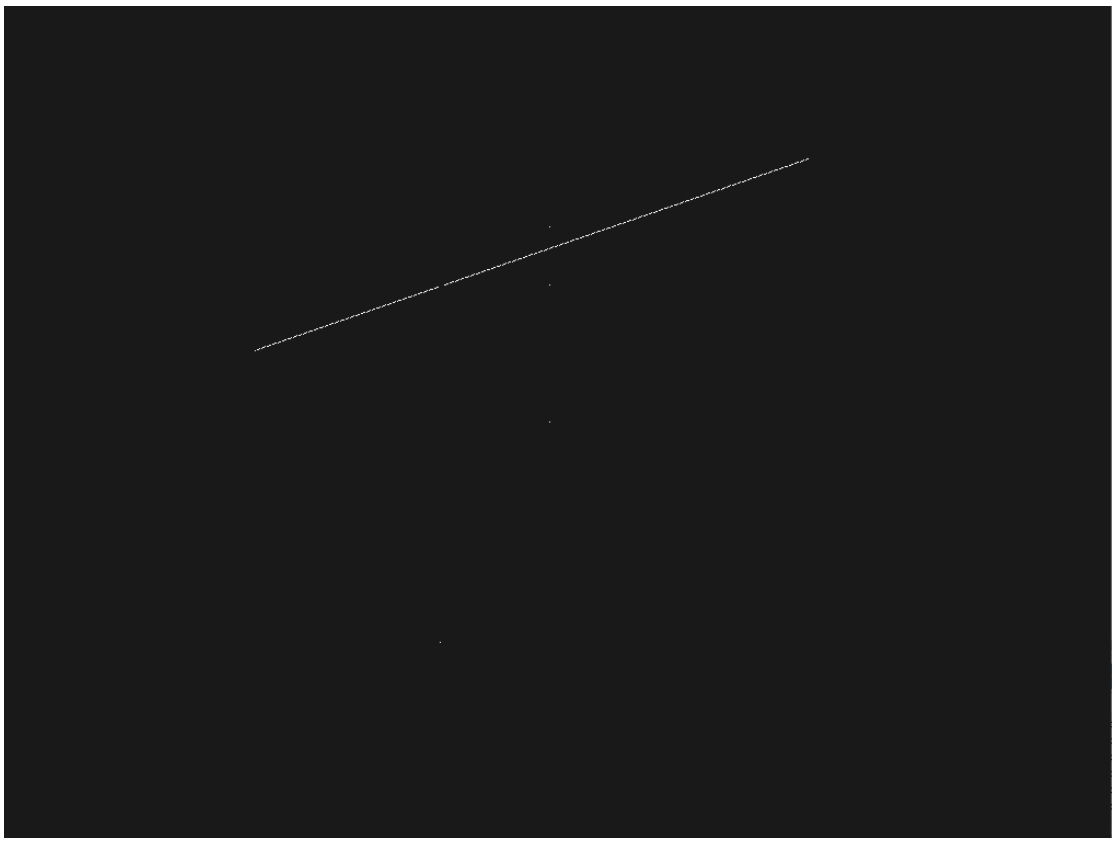
```
void updateCurveVertex() {
    for (int i = 0; i < curveVetexTotalNum; i++) {
        double t = (1/float(curveVetexTotalNum)) * i;
        double x=0,y=0;
        for (int j = 0; j < controlvertexNum; j++) {
            double b = Bernstein(j,controlvertexNum-1,t);
            x += double(controlVertices[j].x) * b;
            y += double(controlVertices[j].y) * b;
            //std::cout << ("Iteration position:(") << x << ", " << y << ")" << std::endl;
        }
        vertices[i * 3] = float(x);
        vertices[i * 3 + 1] = float(y);
        vertices[i * 3 + 2] = 0.0f;
        std::cout << ("Final vertex position:(") << x << ", " << y << ")" << std::endl;
    }
    for (int i = 0; i < controlvertexNum; i++) {
        vertices[curveVetexTotalNum + i * 3] = controlVertices[i].x;
        vertices[curveVetexTotalNum + i * 3 + 1] = controlVertices[i].y;
        vertices[curveVetexTotalNum + i * 3 + 2] = controlVertices[i].z;
    }
}
```

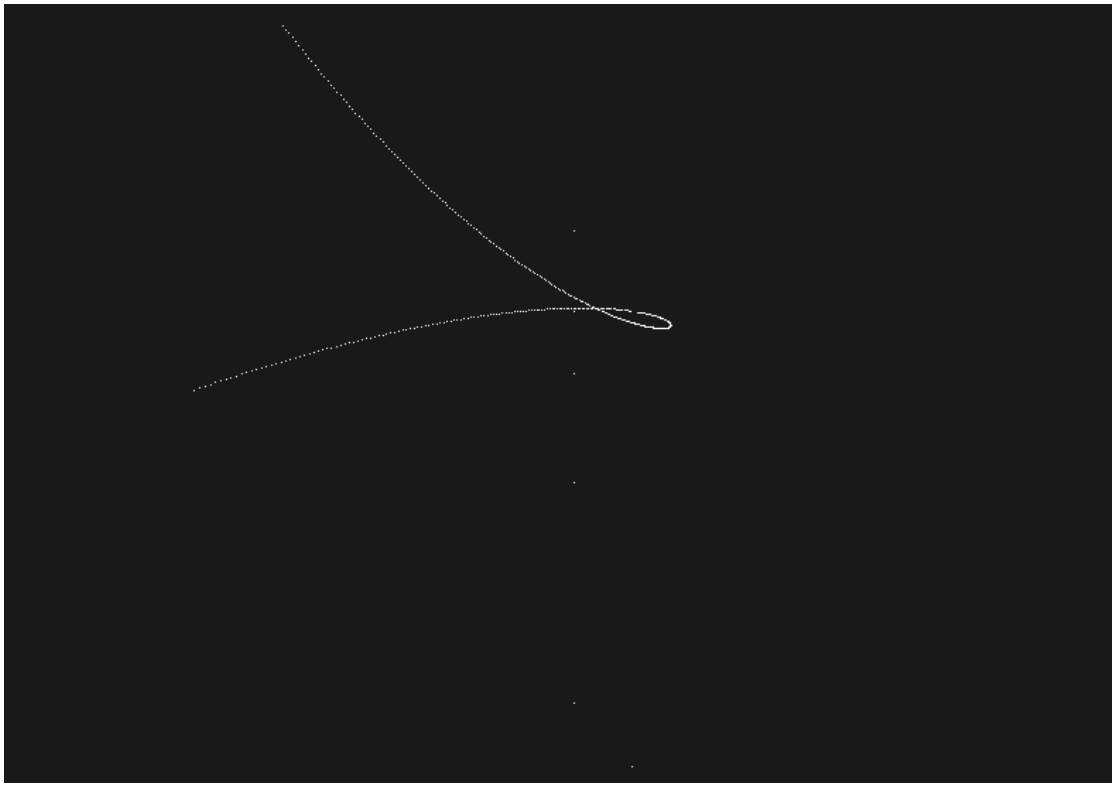
更新曲线的点：根据上述调和函数求每一个点的值，t∈(0,1/总曲线点数,1),最后绘制控制点

```
double stairsMultiply(int n) {
    if (n == 0||n == 1)
        return 1;
    int result = 1;
    for (int i = 2; i <= n; i++) {
        result *= i;
    }
    return double(result);
}
double m_pow(float base, int exp) {
    if (exp == 0) {
        return 1;
    }
    else {
        return pow(base, exp);
    }
}
double Bernstein(int i, int n, float t) {
    return (stairsMultiply(n) / stairsMultiply(i) / stairsMultiply(n - i)) * m_pow(t, i) * m_pow(1 - t, n - i);
}
```

分别是阶乘、幂运算、伯恩斯坦基函数

结果：





具体见 gif