作业8实验报告

实验要求

Basic:

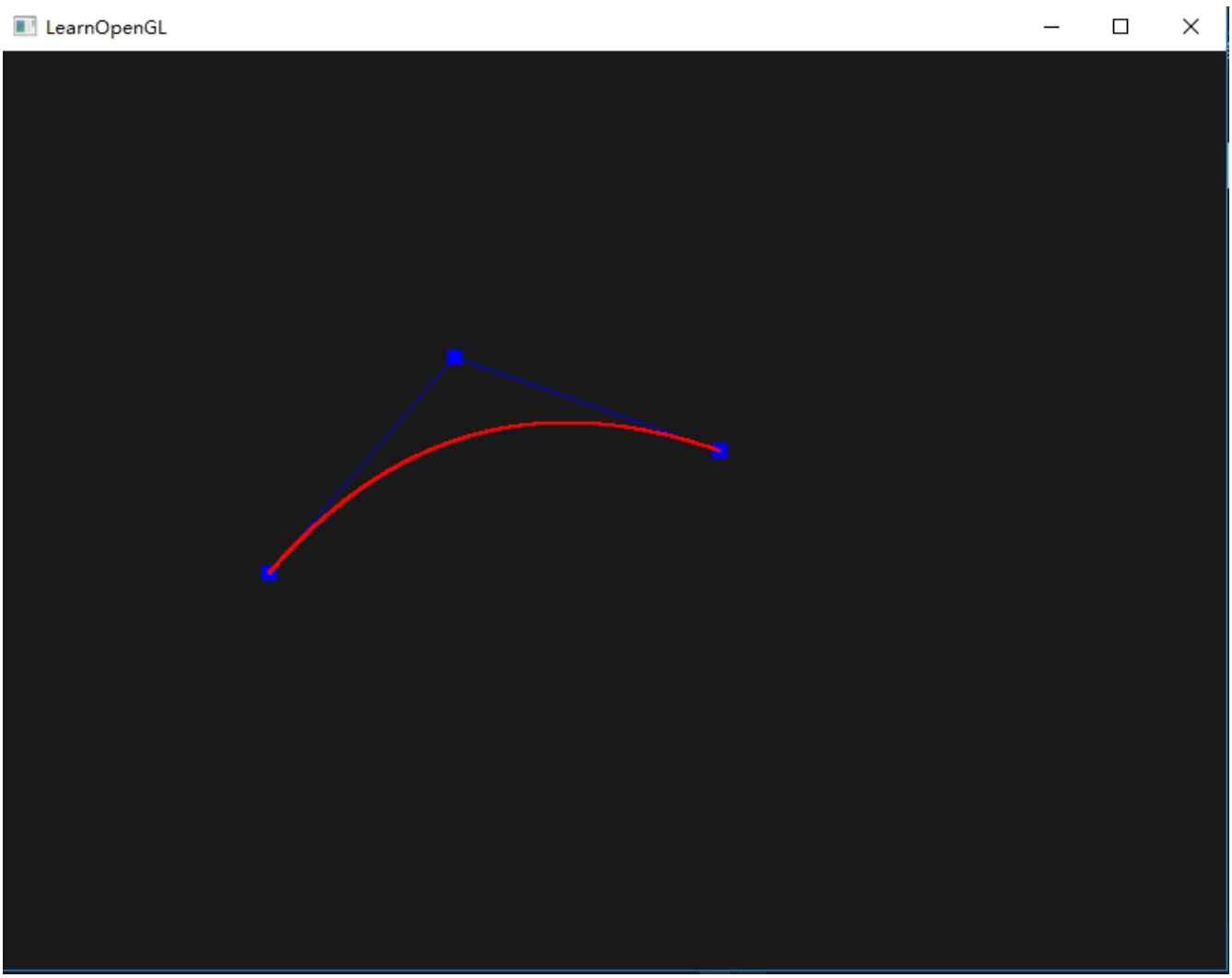
1. 用户能通过左键点击添加Bezier曲线的控制点，右键点击则对当前添加的最后一个控制点进行消除
2. 工具根据鼠标绘制的控制点实时更新Bezier曲线。

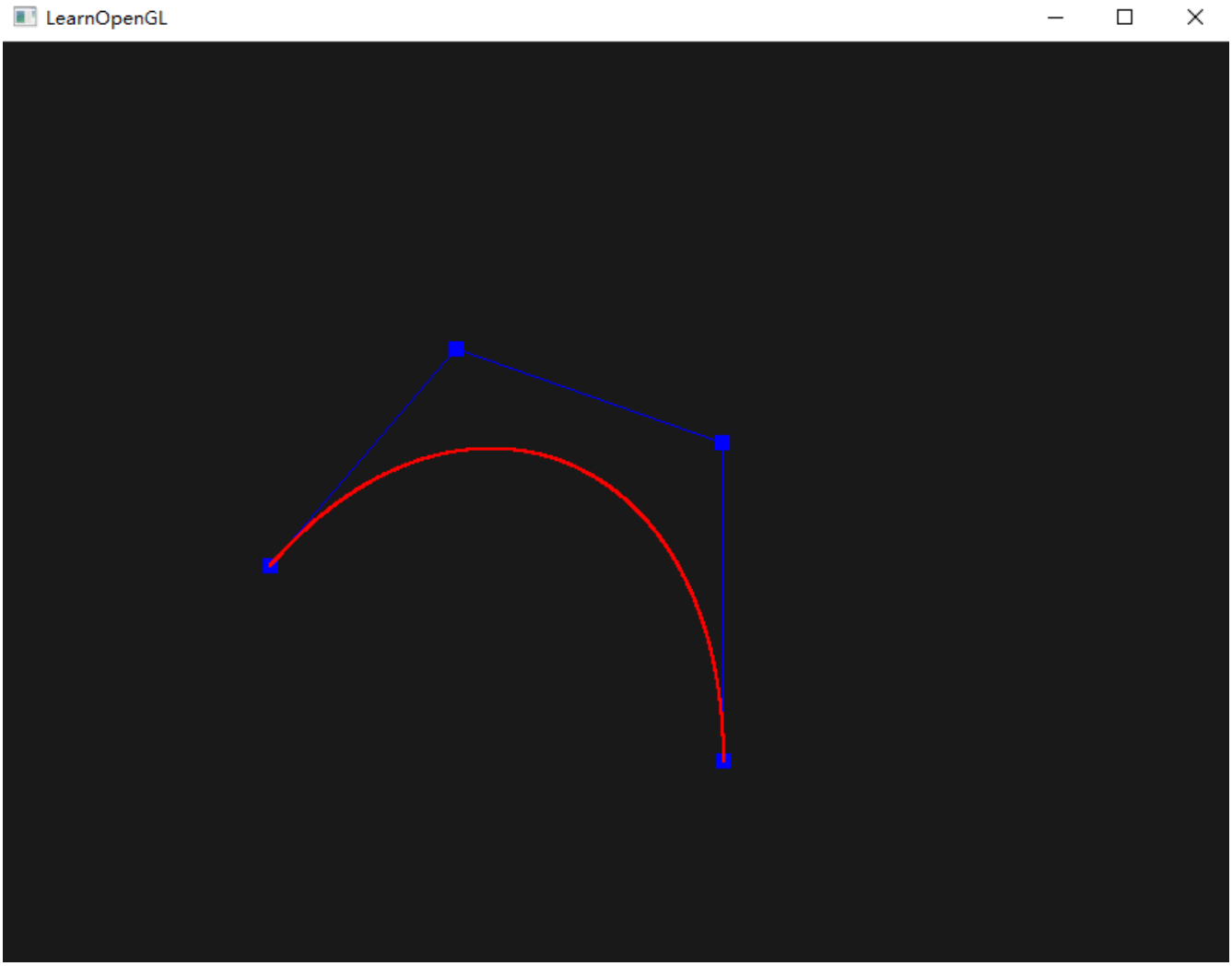
   Hint: 大家可查询捕捉mouse移动和点击的函数方法

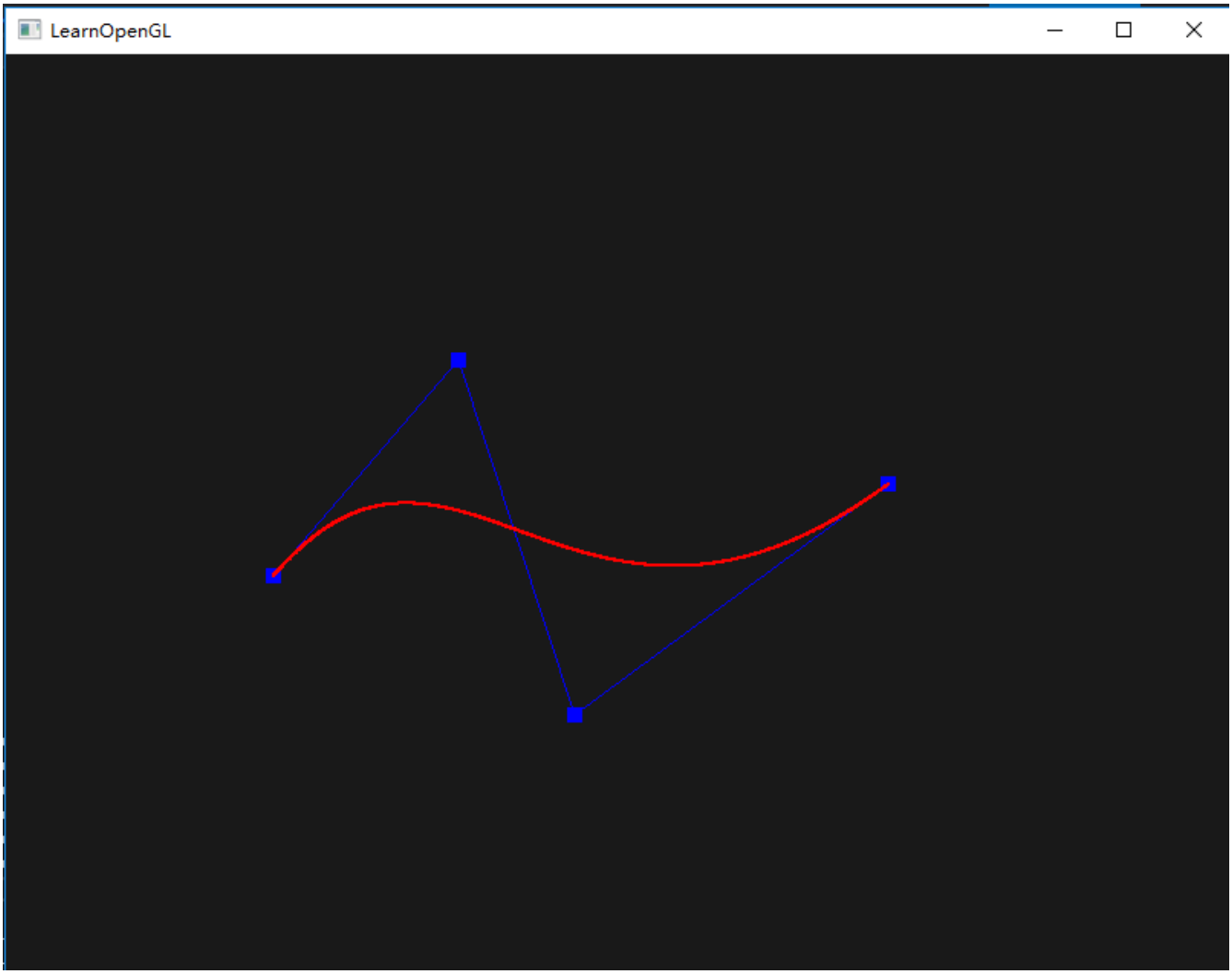
Bonus:

1. 可以动态地呈现Bezier曲线的生成过程。

实验截图

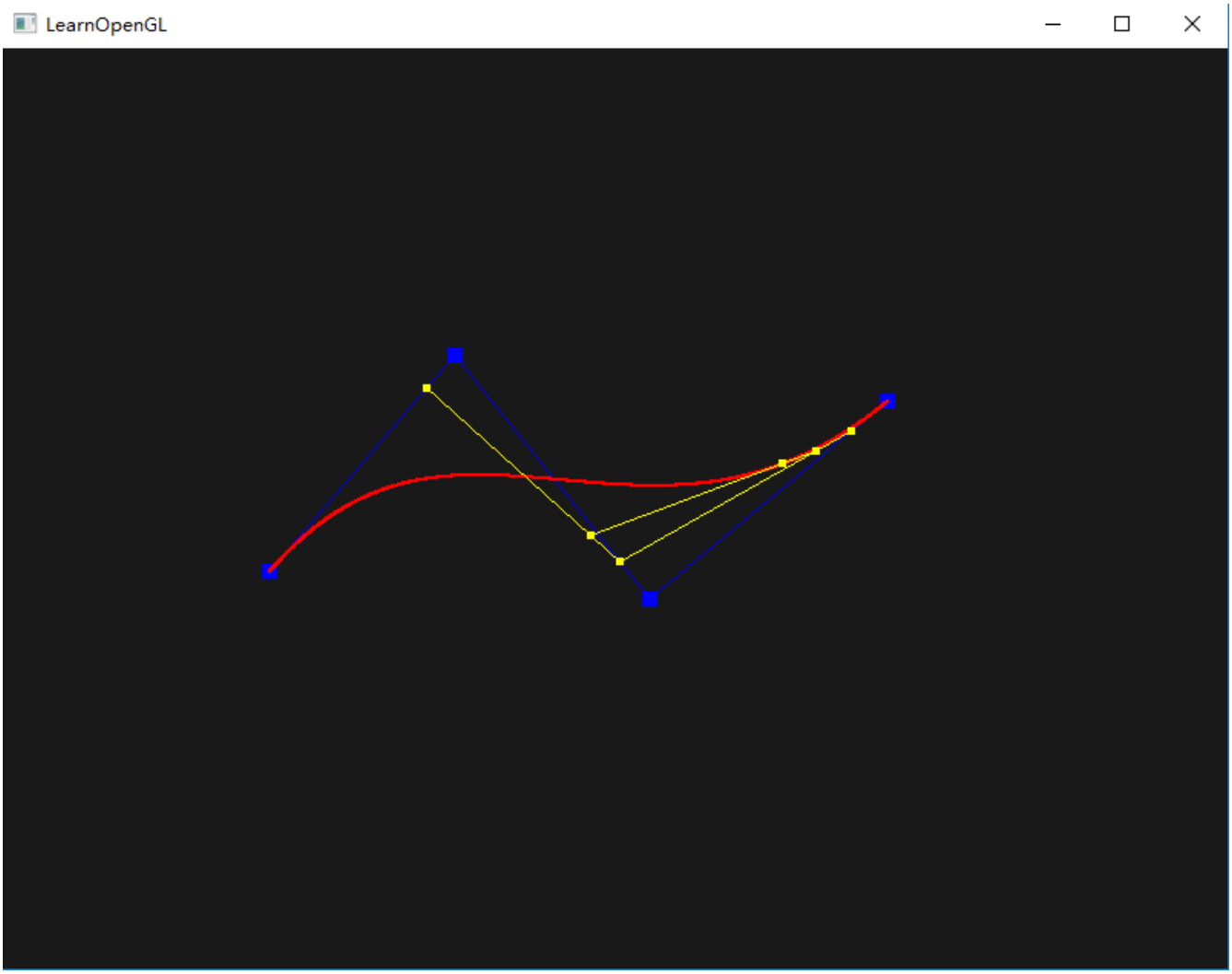



LearnOpenGL



bonus: 点击S键时，则可以显示曲线动态实现过程


LearnOpenGL



关键代码

1. 鼠标位置监听函数

```
void mouse_pos_callback(GLFWwindow* window, double x, double y) {
    mouse_x = x;
    mouse_y = y;
}
....
glfwSetCursorPosCallback(window, mouse_pos_callback);
```

2. 鼠标左右键监听函数

```
void mouse_btn_callback(GLFWwindow* window, int button, int action, int mods) {
    if (action == GLFW_PRESS) {
        change = true;
        t_count = 0.0f;
        Point p(mouse_x - SCR_WIDTH / 2.0f, SCR_HEIGHT / 2.0f - mouse_y);
        switch (button) {
        case GLFW_MOUSE_BUTTON_LEFT:
            ctrlPoints.push_back(p);
            break;
```

```
        case GLFW_MOUSE_BUTTON_RIGHT:
            if (!ctrlPoints.empty()) {
                ctrlPoints.pop_back();
            }
            break;
        default:
            break;
        }

    }
}
glfwSetMouseButtonCallback(window, mouse_btn_callback);
```

3. 点绘制函数

```
// 绘制控制点
        float *ctrlPointVertices = createVertices(ctrlPoints, 0.0f, 0.0f, 1.0f);
        // VAO, VBO
        glGenVertexArrays(1, &ctrlVAO);
        glGenBuffers(1, &ctrlVBO);
        glBindVertexArray(ctrlVAO);
        glBindBuffer(GL_ARRAY_BUFFER, ctrlVBO);
        glBufferData(GL_ARRAY_BUFFER, ctrlPoints.size() * 6 * sizeof(GLfloat),
ctrlPointVertices, GL_STREAM_DRAW);
        glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(GLfloat),
(GLvoid*)0);
        glEnableVertexAttribArray(0);
        glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(GLfloat),
(GLvoid*)(3 * sizeof(float)));
        glEnableVertexAttribArray(1);
        // 绘制
        glPointSize(10.0f);
        glDrawArrays(GL_POINTS, 0, ctrlPoints.size());
        glPointSize(1.0f);
        glDrawArrays(GL_LINE_STRIP, 0, ctrlPoints.size());
        glBindBuffer(GL_ARRAY_BUFFER, 0);
        glBindVertexArray(0);
        if (ctrlPointVertices != NULL) {
            delete ctrlPointVertices;
            ctrlPointVertices = NULL;
        }
```

4. bezierCurve算法函数

```
void getCurvePoints(vector<Point> ctrlPoints, int size) {
    curvePoints.clear();
    for (float i = 0; i <= frequency; ++i) {
        float t = i / frequency;
        vector<Point> temp;
        Point p(0, 0);
        for (int i = 0; i < size; ++i) {
            p.x = ctrlPoints[i].x * (1 - t) + ctrlPoints[i + 1].x * t;
```

```
            p.y = ctrlPoints[i].y * (1 - t) + ctrlPoints[i + 1].y * t;
            temp.push_back(p);
        }
        while (temp.size() > 1) {
            vector<Point> next;
            for (int i = 0; i < temp.size() - 1; ++i) {
                p.x = temp[i].x * (1 - t) + temp[i + 1].x * t;
                p.y = temp[i].y * (1 - t) + temp[i + 1].y * t;
                next.push_back(p);
            }
            temp = next;
        }
        curvePoints.push_back(temp[0]);
    }
}
```

实验思考

贝塞尔曲线

Bézier curve本质上是由调和函数（Harmonic functions）根据控制点 （Control points）插值生成。其参数方程如下：

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

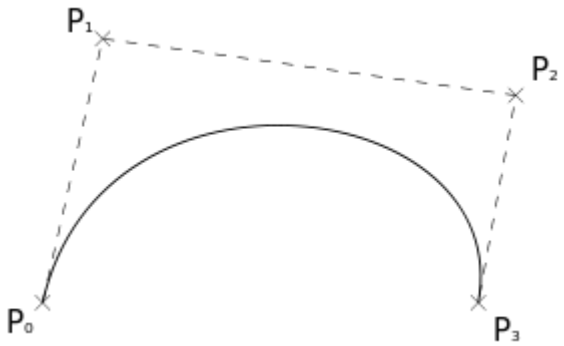
上式为n次多项式，具有$n+1$项。其中，$Pi(i=0,1\ldots n)$表示特征多边 形的$n+1$个顶点向量；$Bi,n(t)$为伯恩斯坦（Bernstein）基函数，其多项式表示为：

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

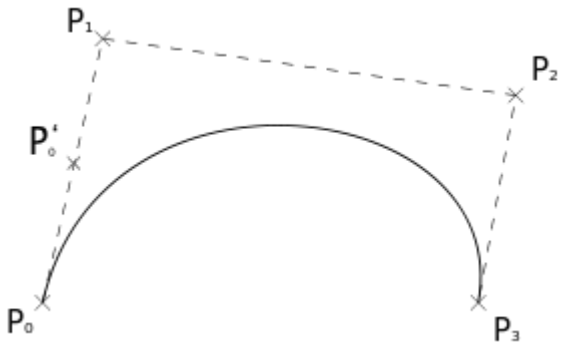
$$B(t)=\sum_{i=0}^{n}\binom{n}{i}Pi(1-t)^{n-i}t^i=\binom{n}{0}P0(1-t)^nt^0+\binom{n}{1}P1(1-t)^{n-1}t^1+\ldots+\binom{n}{n-1}Pn-1(1-t)^1t^{n-1}+\binom{n}{n}Pn(1-t)^0t^n,t\in[0,1]$$

绘制贝塞尔曲线

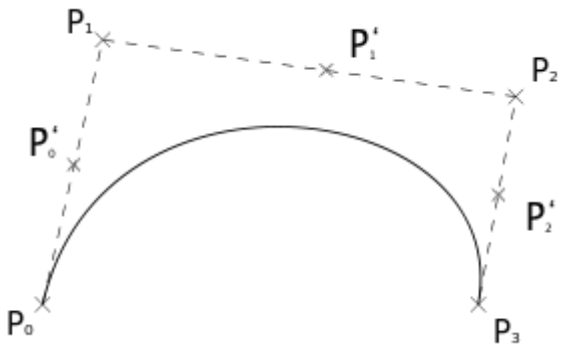
以三阶贝塞尔曲线为例，包括4个控制点，分别为$\mathbf{P_0}$，$\mathbf{P_1}$，$\mathbf{P_2}$，$\mathbf{P_3}$。



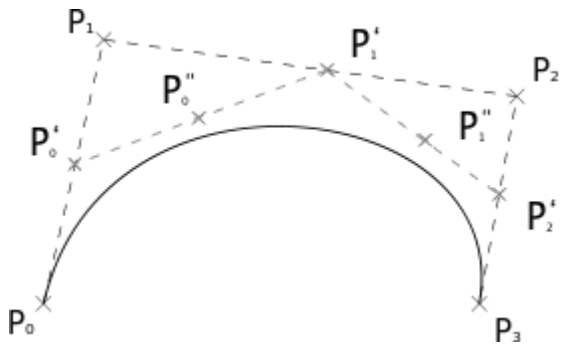
1. 四个控制点通过先后顺序进行连接，形成了三条线段，也就是上图中的P_0P_1，P_1P_2，P_2P_3，然后通过一个参数**t**，参数的值等于线段上某一个点距离起点的长度除以线段长度。就比如P_0P_1线段上有一个点P_0'，此时**t**的值就是P_0P_0'/P_0P_1，其中P_0'位置如下图所示。



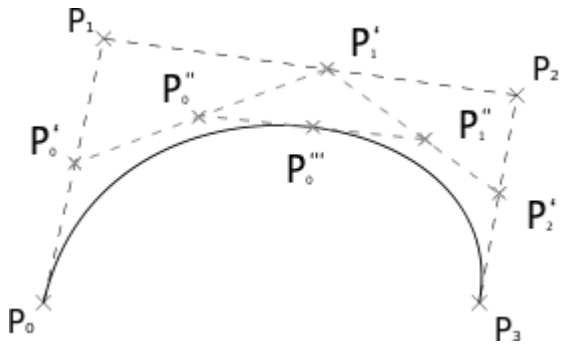
2. 接下来对每一条线段做同样的操作，得到三个控制点P_0',P_1',P_2'，如下图所示。



3. 然后对这三个控制点重复第1步操作，得出两个控制点P_0'',P_1''，如下图所示。



4. 最后再使用同样的方法可以得到，最终的一个点P_0'''，如下图所示，此时这个点就是贝塞尔曲线上的一个点。



通过控制**t**的值，由 0 增加至 1，就绘制出了一条由起点P_0至终点P_1的贝塞尔曲线。

而根据这个方式，我们只要记录之前得出的点，连成直线，这样我们就可以实现贝塞尔曲线的动态实现过程了。