

Bezier 曲线公式 (P1-4 为控制点)

$$P(t) = (1-t)^3 P_1 + 3t(1-t)^2 P_2 + 3t^2(1-t) P_3 + t^3 P_4$$

写成矩阵形式

$$P(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} P_1 & P_2 & P_3 & P_4 \end{bmatrix} \begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix}$$

等式左端第一个由控制点组成的矩阵我们称"Geometry matrix", 第二个称"Spline matrix(Bernstein)"控制曲线的类型如 Bernstein, 第三个称"Canonical monomial basis"。由此我们可以给出一般曲线的公式

$$Q(t) = GBT(t) = \text{Geometry} \cdot \text{SplienBasis} \cdot \text{PowerBasis} \cdot T(t)$$


```
// Q(t)
glm::vec2 Q(double t) {
    int n = point.size() - 1;
    double x = 0, y = 0;
    double u = 1, v = pow((1 - t), n);
    for (int i = 0; i <= n; i++) {
        x += C(n, i) * u * v * point[i].x;
        y += C(n, i) * u * v * point[i].y;
        u *= t;
        v /= (1 - t);
    }
    return glm::vec2(x, y);
}
```

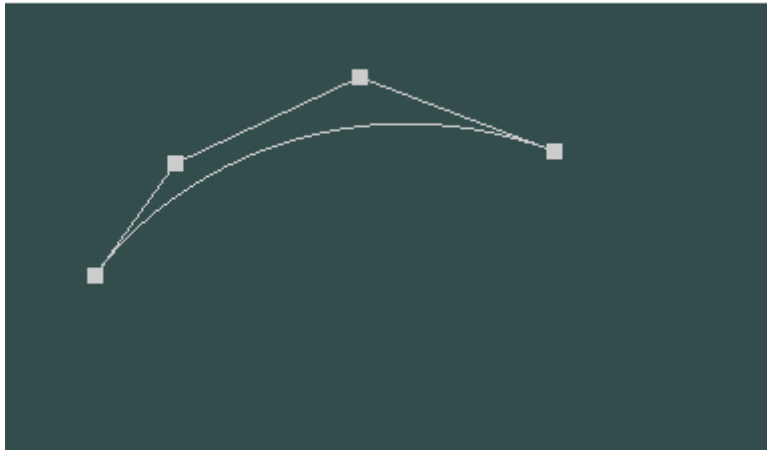
Bi(t)的通用算法公式

$$B_i^n(t) = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i}$$

```
// C(n, k)
int C(int n, int k) { return factorial(n) / (factorial(k) * (factorial(n - k))); }
// n!
int factorial(int n) {
    if (n <= 1) return 1;
    return n * factorial(n - 1);
}
```


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。

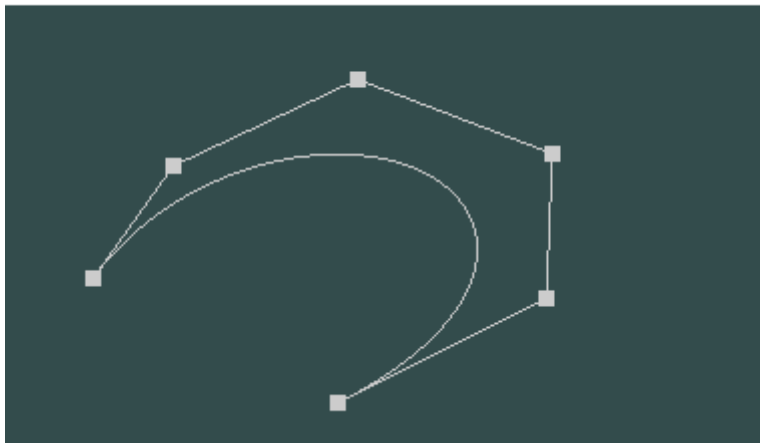
 LearnOpenGL



左键*4 ->


->

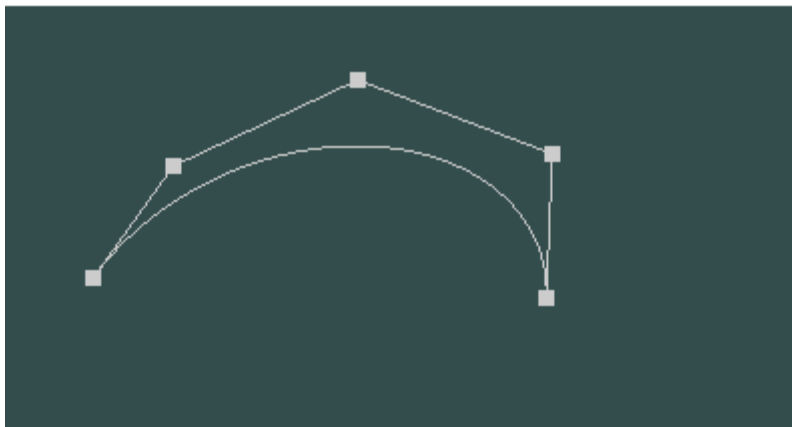
 LearnOpenGL



左键*2 ->

->

 LearnOpenGL



右键*1 ->