20337025+崔璨明+HW5

姓名	学号	专业
崔璨明	20337025	计算机科学与技术

Task 1

根据Bézier曲线的定义来计算给定t值($t \in [0,1]$)下的Bézier曲线上的点。

根据Bézier 曲线的定义公式,其中 P_i 为第i个控制顶点:

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

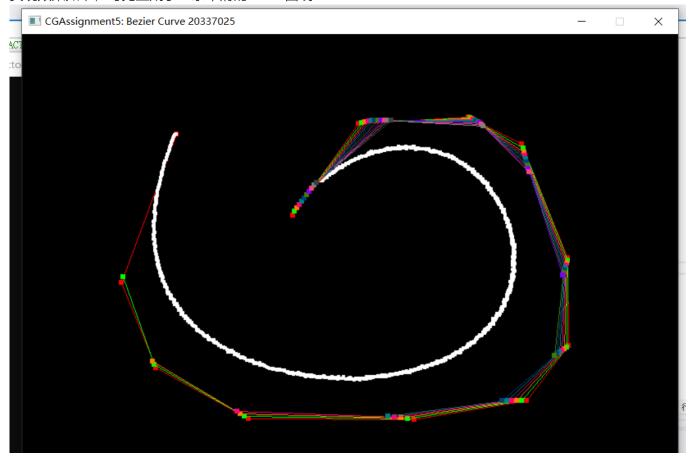
一共有n+1个控制顶点。上式是一个n次多项式,一共n+1项,每个控制点对应一项。多项式系数 $B_{i,n}(t)$ 是 Bernstein基函数:

$$B_{i,n}(t) = rac{n!}{i!(n-i)!} t^i (1-t)^{(n-i)}, i=0,1,\cdots,n$$

根据公式编写代码如下, getFactorial 函数用于计算阶乘:

```
long long BezierCurve::getFactorial(int n)const
{
        long long res = 1;
        for (int i = 1; i <= n; i++) {
                res *= i;
        return res;
}
Point2D BezierCurve::implementTask1(const std::vector<Point2D> &points, const double &t)const
{
        //Task1: implement Bezier curve generation algorithm accroding to the definition
        int n = points.size() - 1;
        auto p = Point2D(0, 0);
        for (int i = 0; i \leftarrow n; i++)
                double BezierValue= getFactorial(n) / (getFactorial(i) * getFactorial(n - i)) *
std::pow(t, i) * std::pow(1 - t, n - i);
                p += points[i] * BezierValue;
        }
        return p;
}
```

实现效果如下,可见生成了一条平滑的Bézier曲线:



Task 2

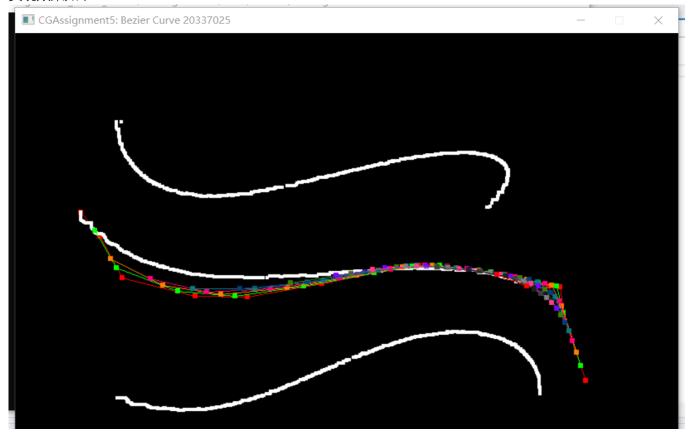
Task 2、实现更为简单、直观的de Casteljau算法来生成给定t值 $(t\in[0,1])$ 下的Bézier曲线上的点。 de Casteljau算法的核心伪代码如下:

- 1. 考虑一个 p_0, p_1, \ldots, p_n 为控制点序列的Bézier曲线,首先将相邻的点连接起来形成线段;
- 2. 用t:(1-t)的比例划分每个线段,用线性插值法找到分割点;
- 3. 对所有的线段执行上述操作,得到的分割点作为新的控制点序列,新序列的数量会减少一个;
- 4. 如果新的序列只包含一个点,则返回该点,终止迭代过程。否则,使用新的控制点序列并转到步骤1,如此迭代下去。

根据算法编写代码:

```
Point2D BezierCurve::implementTask2(const std::vector<Point2D> &points, const double &t) const
{
    //Task2: implement de Casteljau algorithm for Bezier curve
    // Note: you should use Point2D::lerp().
    std::vector<Point2D>* iter_points = new std::vector<Point2D>(points);
    do
    {
        std::vector<Point2D>* temp = new std::vector<Point2D>({});
        int size = (*iter_points).size();
```

实现效果如下:



Task 3

谈谈你对Bézier曲线的理解, Bézier曲线的缺点是什么?

答: Bezier曲线的初衷是用尽可能少的数据表示出复杂的图形。具有许多重要的特点,如各项系数之和为1、系数对称性、曲线的起始点和终点对应第一个和最后一个控制点、凸包性、几何不变性等等,是计算机图形图像造型的基本工具,也是图形造型运用得最多的基本线条之一。贝塞尔曲线上的所有控制点、节点均可编辑。这种"智能化"的矢量线条为艺术家提供了一种理想的图形编辑与创造的工具。

Bézier曲线的缺点主要有:需要计算阶乘,当控制点增多时,计算复杂度也会大大增加,计算效率便会下降,但可以进行改进,如储存计算的中间结果等等。除此之外,其特征多边形的顶点离开得很远时不利于精确控