# 作业 2：Bézier 曲线

## 1 总览

Bézier 曲线是一种用于计算机图形学的参数曲线。在本次作业中，你需要实现 de Casteljau 算法来绘制由**多个控制点**表示的 Bézier 曲线。

## 2 代码框架

在本次作业中，你需要修改 main.cpp 中的的函数：

- **bezier**：该函数实现绘制 Bézier 曲线的功能。它使用一个控制点序列和一个 OpenCV::Mat 对象作为输入，没有返回值。它会使 t 在 0 到 1 的范围内进行迭代，并在每次迭代中使 t 增加一个微小值。对于每个需要计算的 t，将调用另一个函数 recursive_bezier，然后该函数将返回在 Bézier 曲线上 t 处的点。最后，将返回的点绘制在 OpenCV::Mat 对象上。

  **注：你可以使用 window.at<cv::Vec3b>(point.y, point.x)[1] = 255 将绿色的线画在屏幕上。**

- **recursive_bezier**：该函数使用一个控制点序列和一个浮点数 t 作为输入，实现 de Casteljau 算法来返回 Bézier 曲线上对应点的坐标。

## 3 算法

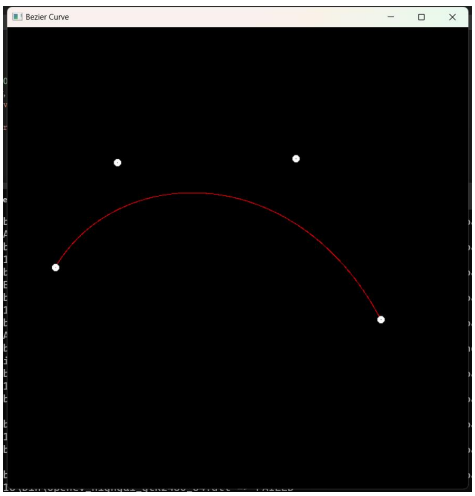
De Casteljau 算法说明如下：

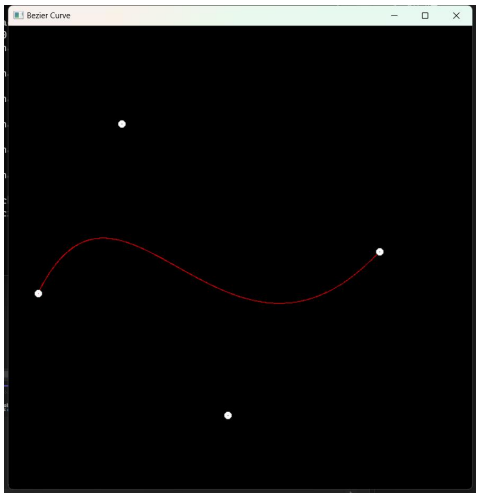
- 考虑一个 p0, p1, ... pn 为控制点序列的 Bézier 曲线。首先，将相邻的点连接起来以形成线段。
- 用 t : (1 - t) 的比例细分每个线段，并找到该分割点。
- 得到的分割点作为新的控制点序列，新序列的长度会减少 1。
- 如果序列只包含一个点，则返回该点并终止。否则，使用新的控制点序列并 转到步骤 1。使用 [0,1] 中的多个不同的 t 来执行上述算法，你就能得到相应的 Bézier 曲线。

## 4 开始编写

运行时，程序将打开一个黑色窗口。现在，你可以点击屏幕选择点来控制 Bézier 曲线。程序将等待你在窗口中选择 4 个控制点，然后它将根据你选择的控制点来自动绘制 Bézier

曲线。代码框架中提供的实现通过使用**多项式方程**来计算 Bézier 曲线并绘制为红色。两张控制点对应的 Bézier 曲线如下所示：

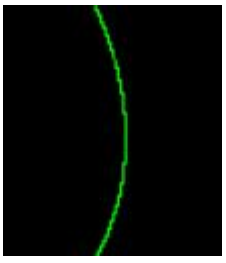


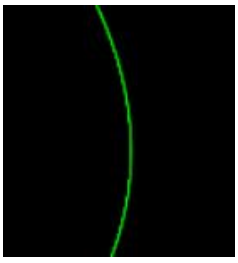



在确保代码框架一切正常后，就可以开始完成你自己的实现了。注释掉 main 函数中 while 循环内调用 naive_bezier 函数的行，并取消对 bezier 函数的注释。要求你的实现将 Bézier 曲线绘制为绿色。

如果要确保实现正确，请同时调用 naive_bezier 和 bezier 函数，如果实现正确，则两者均应写入大致相同的像素，因此该曲线将表现为黄色。如果是这样，你可以确保实现正确。在这之后，你需要使用不同数量的控制点 **(大于四个)**，来查看不同的 Bézier 曲线。

最后，你需要实现对 Bézier 曲线的反走样。因为用零散的像素点来描述曲线，会因为点之间的距离或一些错位导致走样问题。所以对于一个曲线上的点，不只把它对应于一个像素，你需要根据到像素中心的距离来考虑与它相邻的像素的颜色。


走样


2x2 反走样

## 5 评分与提交

评分：

- 正确地提交所有必须的文件，且代码能够编译运行。
- De Casteljau 算法：对于给定的控制点，你的代码能够产生正确的 Bézier 曲线。
- 实现对 Bézier 曲线的反走样。

提交：

当你完成作业后，请清理你的项目，在你的文件夹中需要包含所有的程序文件（无论是否修改)。同时，请提交实验结果的图片与添加一个 README 文件写下完成情况，并简要描述你在各个函数中实现的功能。最后，将上述内容打包，并用“学号_姓名_作业 2.zip”的命名方式提交。