实验 4：Bézier 曲线

要求：

- **独立完成**
- **在截止时间之前提交**

---

1 总览

Bézier 曲线是一种用于计算机图形学的参数曲线。在本次实验中，你需要实现 de Casteljau 算法来绘制由 4 个控制点表示的 Bézier 曲线(当你正确实现该算法时，你可以支持绘制由更多点来控制的 Bézier 曲线)。

你需要修改的函数在提供的 main.cpp 文件中。

• 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。

• 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。

在本次实验中，你会在一个新的代码框架上编写，编译成功后，你可以通过使用以下命令运行给定代码 ./BezierCurve。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。

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。

你也可以尝试修改代码并使用不同数量的控制点，来查看不同的 Bézier 曲线。

2 算法

De Casteljau 算法说明如下：

1. 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。
2. 用 t : (1 - t) 的比例细分每个线段，并找到该分割点。
3. 得到的分割点作为新的控制点序列，新序列的长度会减少一。
4. 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。

3 提高项

实现对 Bézier 曲线的反走样。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。