作业3：贝塞尔曲线

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 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。

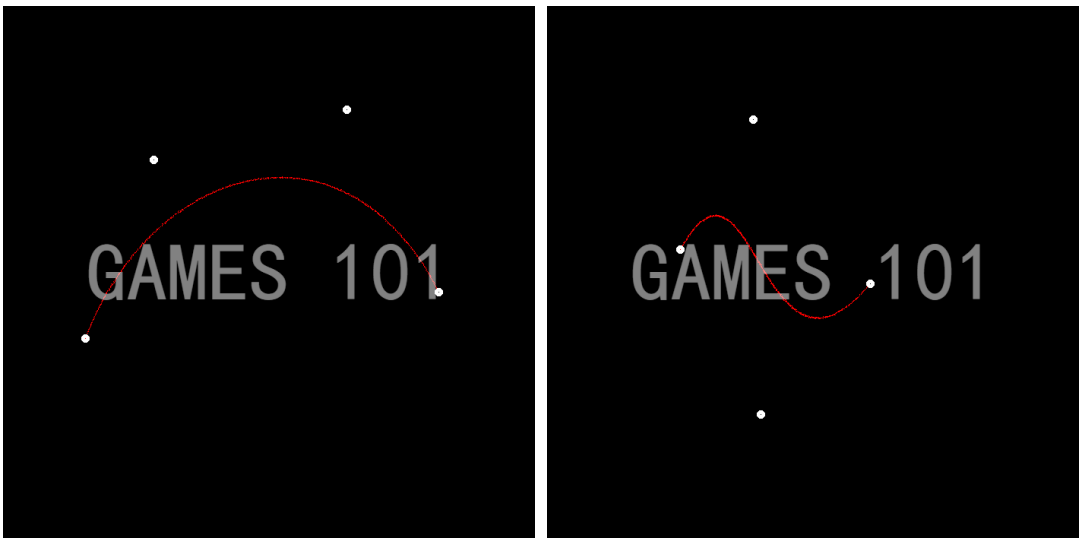
- `recursive_bezier`：该函数使用一个控制点序列和一个浮点数 `t` 作为输入，**实现 de Casteljau 算法来返回 Bézier 曲线上对应点的坐标；实现基于伯恩施坦多项式的算法返回Bézier曲线上对应点的坐标。**
这两种算法的实现细节，请参考授课ppt。

2 开始编写

在本次作业中，你会在一个新的代码框架上编写，它比以前的代码框架小很多。和之前作业相似的是，你可以选择在自己电脑的系统或者虚拟机上完成作业。请下载项目的框架代码，并使用以下命令像以前一样构建项目：

```
$ mkdir build
$ cd build
$ cmake ..
$ make
```

之后，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。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。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。