作业2

算法：采用Liang-Barsky算法（from *Computer Graphics Principles and Practice Second Edition in C,* by Foley, Figure 3.45）

函数：CMyController::Clip(Line& line)

没有自己写算法的原因和我的一些思考：

主要是没有想出简单而有效的新算法。对于作业上的编码，我没有想出好的应用。以下是我的思考：这个编码的最大特点是排列有序，而且是连续的。这让我首先想到了数组：能否填写一个数组使得依据这个编码作为索引可以取得想要的关于改点的信息，首先想到的是由此得到该点的Cohen-Sutherland编码，但由于以如此的空间代价进取的这点时间上进步（由于没有测试，只能说可能是进步），不够划算。而后想到建立一张二维表格，但由于很多在clip框上的边界条件事实上并无太大作用，所以25\*25的二维表格将会有很多浪费。于是，我想了一个改进算式（其中sign3受到另一位同学启发）：

**k = sign(x-x1)-sign(x2-x)+sign3(y-y1)-sign3(y2-y)**

其中：

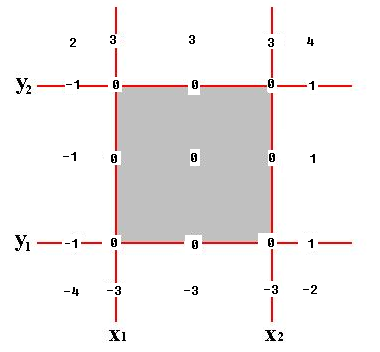
sign(x) = 0 if x < 0,

sign(x) = 1 if x >= 0;

sign3(x) = 0 if x < 0,

sign3(x) = 3 if x >= 0;

如上得到的数值分布如下：



该分布把边框和交点的无用数值都省略了，只剩下最有用的9个数值。

由这个算式继续思考，如果有一个9\*9的表格可以查找两个端点对应的情况，如果能加以利用还是可以有不错的效果的。我当时的想法是将两个端点均不在2, 4, -2, -4区域内的情况统一作为“基础情况”。在这种情况下，均直接算出：

如果是顶点在-1, 1或3, -3这种情况，直接对于左右两边或上下两边裁减；

如果均在0内，简单接受；

如果顶点在类似3, 1的分布，先判断右上顶点在直线的哪一侧，从而可以得知直线是否可以被直接排除，如果不能，直接与右边和上边做裁减，得到最终裁减线。

上述方法需要对于每一个点的分布(a, b)知道如下条件：参与裁减的边，参与判断的顶点，判断方向直接排除的条件（代入直线方程后有符号距离大于0还是小于0），而这些都可以放在一个表格中，甚至上述3类情况的处理函数也可以以函数指针的形式放在里面。

对于不在上述“基础情况”范围内的，只需要在表格中填入另外的裁减边做最多两次裁减即可成为基础情况，例如顶点分别在1和2，只需要关于左边界限裁减一次即可；对于2, -4这样的，直接排除即可；对于-2, 2这样的，分别对于上下边裁减一次即可。对于回到基础情况的，在此调用裁减即可得到结果。

上述方法，虽然可行，但代码量太大，基本属于穷举，而且实现复杂，加上至少2次的函数调用，巨大的空间消耗（那个表格的空间，虽然是一次性开销），故最后还是被抛弃。

最后便决定采用我较为喜欢的简洁的Liang-Barsky算法作为演示。