Computer Graphics Lab 1 & 2 Report

一、实验内容

1.多边形填充扫描线算法与加权反走样的实现。

2.二维线段裁剪 Cohen-Sutherland 算法的实现。

3.二维线段裁剪 Liang-Barsky 算法的实现。

二、测试环境

OS: openSUSE 42.2 64Bit

CPU: Intel Core i7-6500U @ 2.50GHz

RAM: 8GB DDR4 2133MHz

Compiler Version: g++ (GCC) 6.3.0

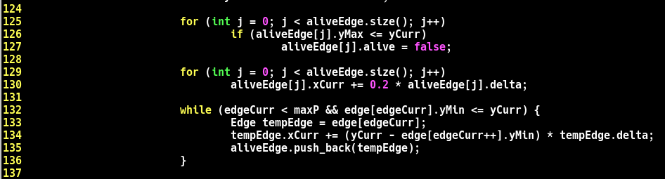
三、实现过程

1. 多边形填充扫描线算法与加权反走样:

以下是图形属性设置:

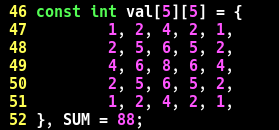


maxX和maxY决定图形的长和宽，maxP是随机生成的点的个数。该程序随机生成maxP个点，依次连接构成多边形（不一定是简单多边形），然后使用扫描线算法填充。



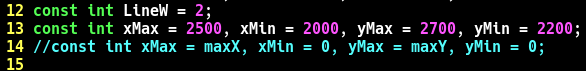
使用vector实现活性边表，同时使用LazyTag实现O(1)删除。

以下是使用高斯分布函数计算得来的反走样加权矩阵:

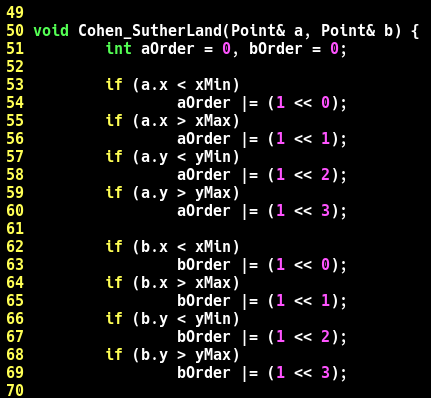


该程序将一个像素细分成5\*5的矩阵，并对其进行填充，而后根据其填充情况决定该像素的灰度值。

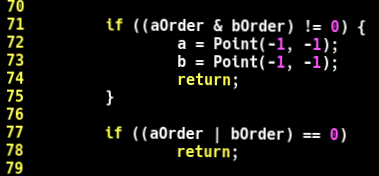
1. 二维裁剪的 Cohen-Sutherland 算法



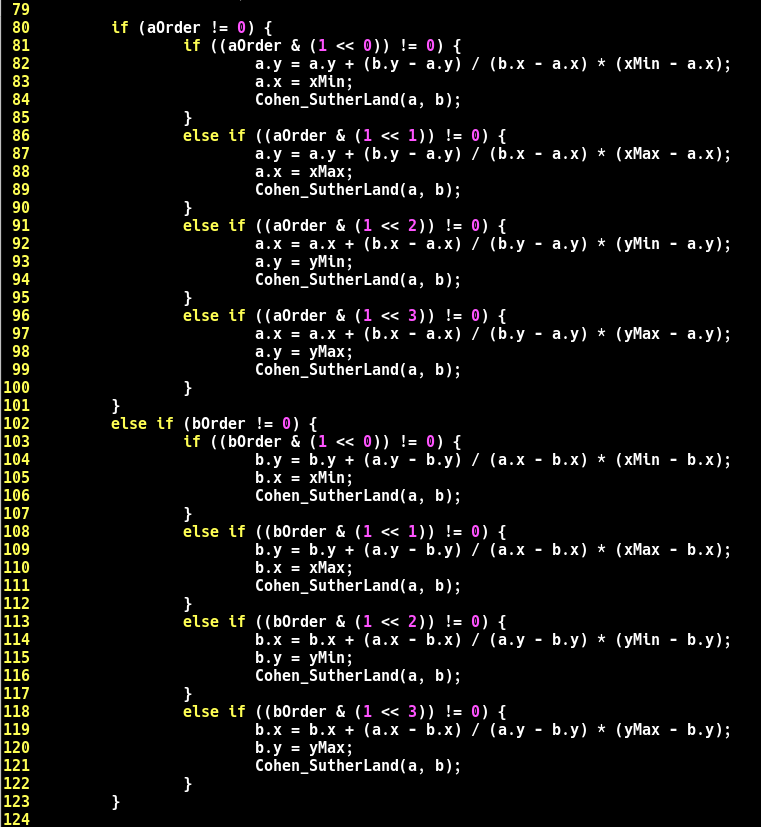
使用LineW规定线段宽度，(xMin, xMax)和(yMin, yMax)规定了裁剪框的属性。



根据线段端点a, b的位置信息将其编号。

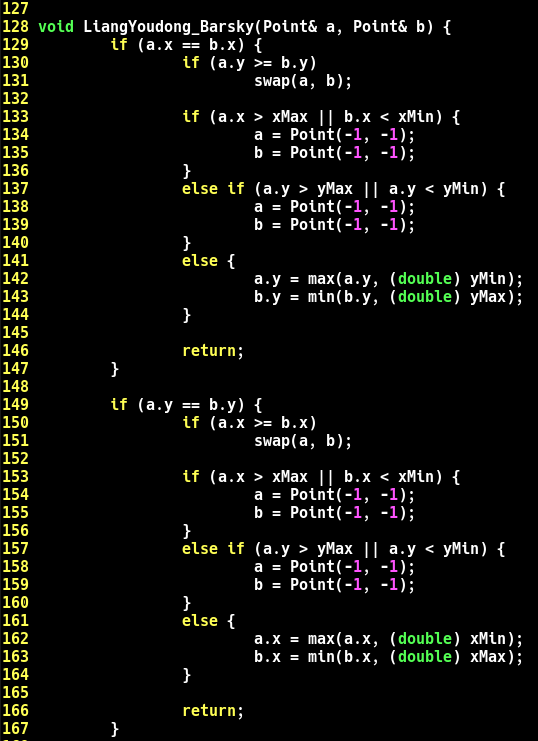


首先判断线段是否完全在裁剪框内外。

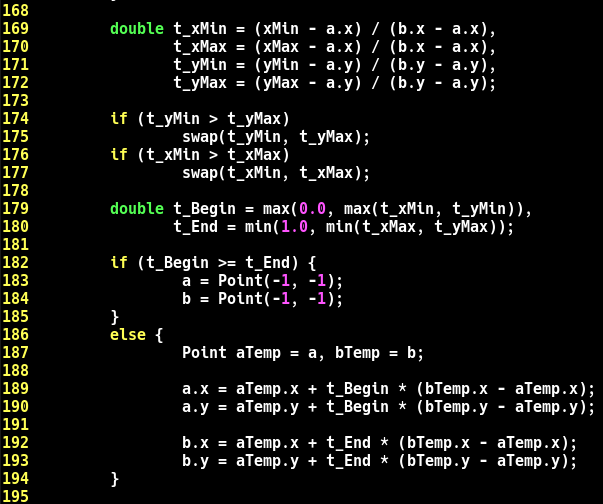


判断线段哪一端点在裁剪框之外，将其裁剪后递归调用CS算法。

1. 二维裁剪的 Liang-Barsky 算法



首先处理水平和竖直的直线。



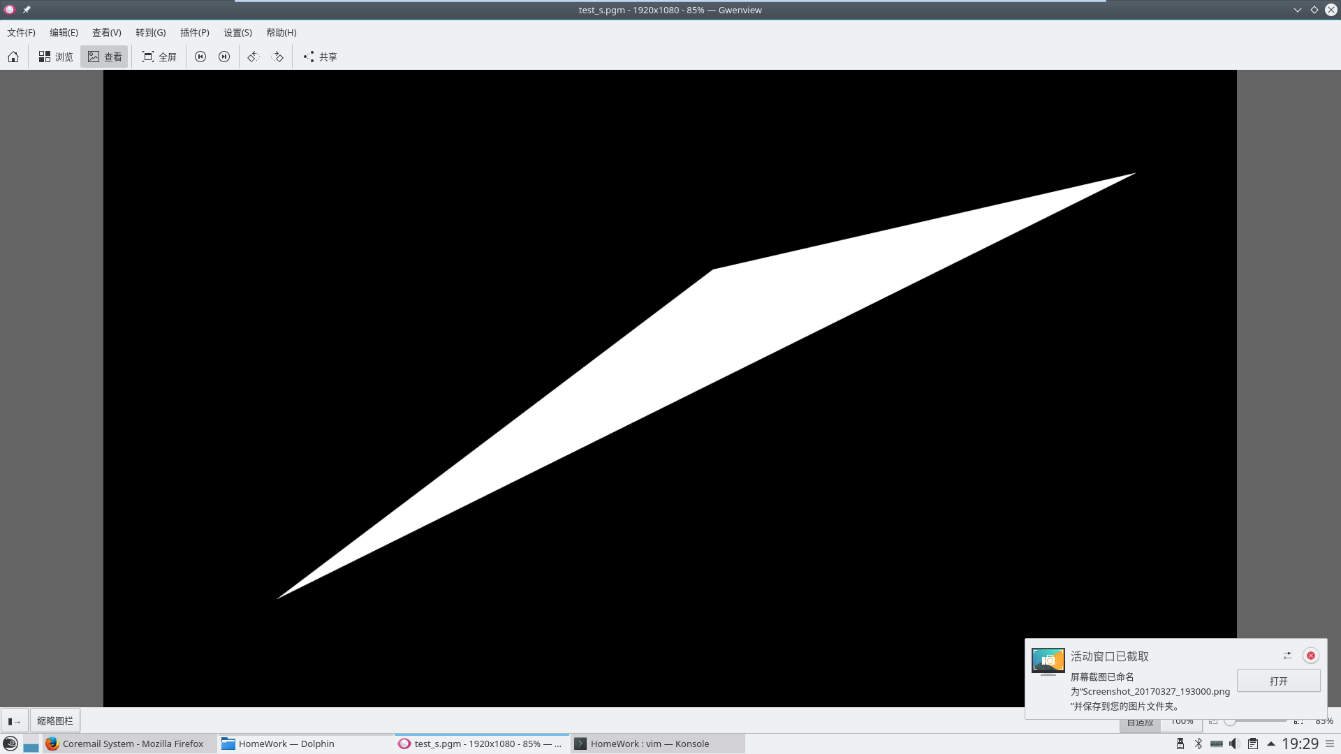
而后将其转化为参数表示，并求出在裁剪框内的参数范围。

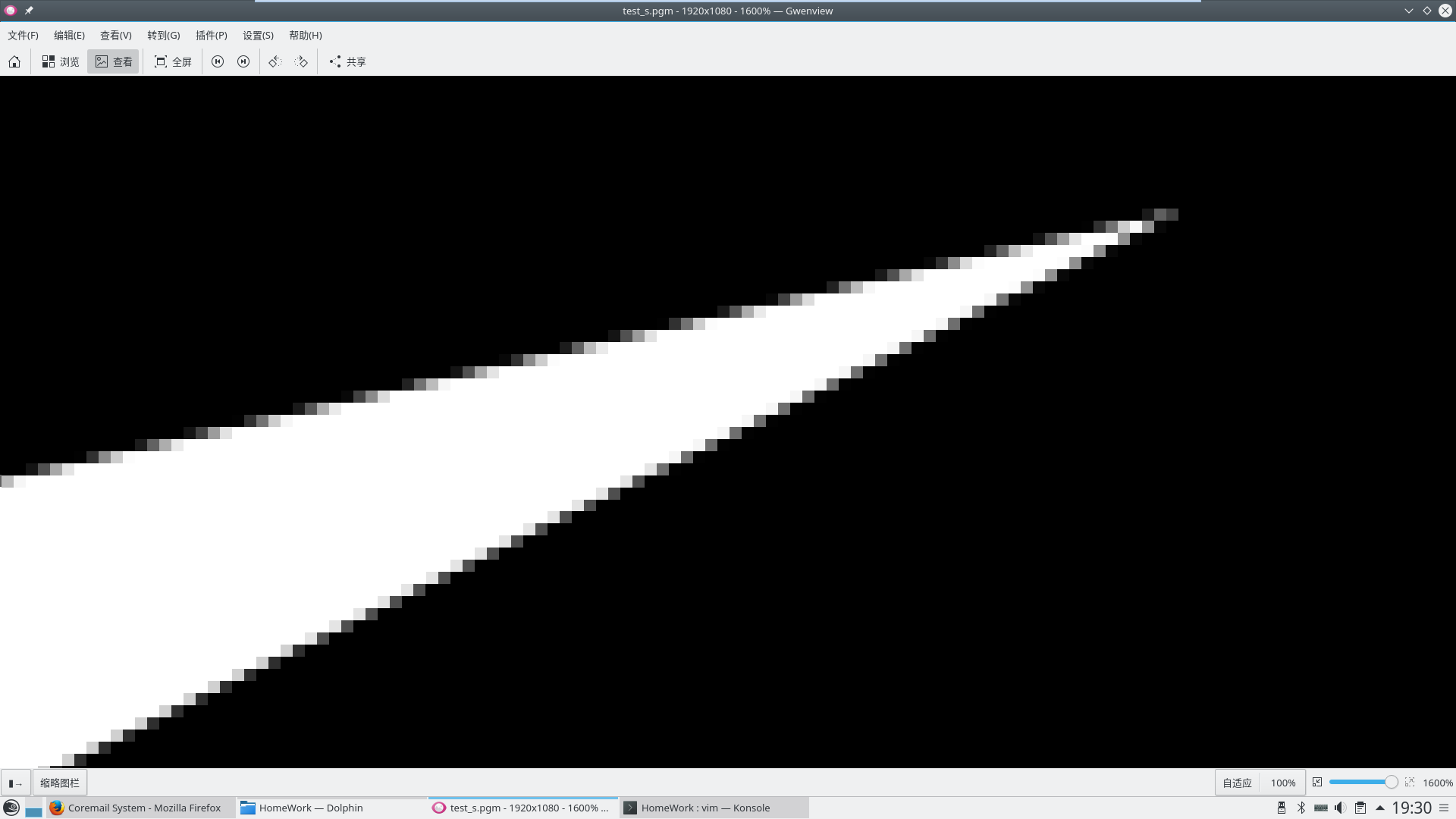
四、结果示例

注：输出以 PGM 灰度图片格式编码，所提交程序均会自动随机取点绘图。

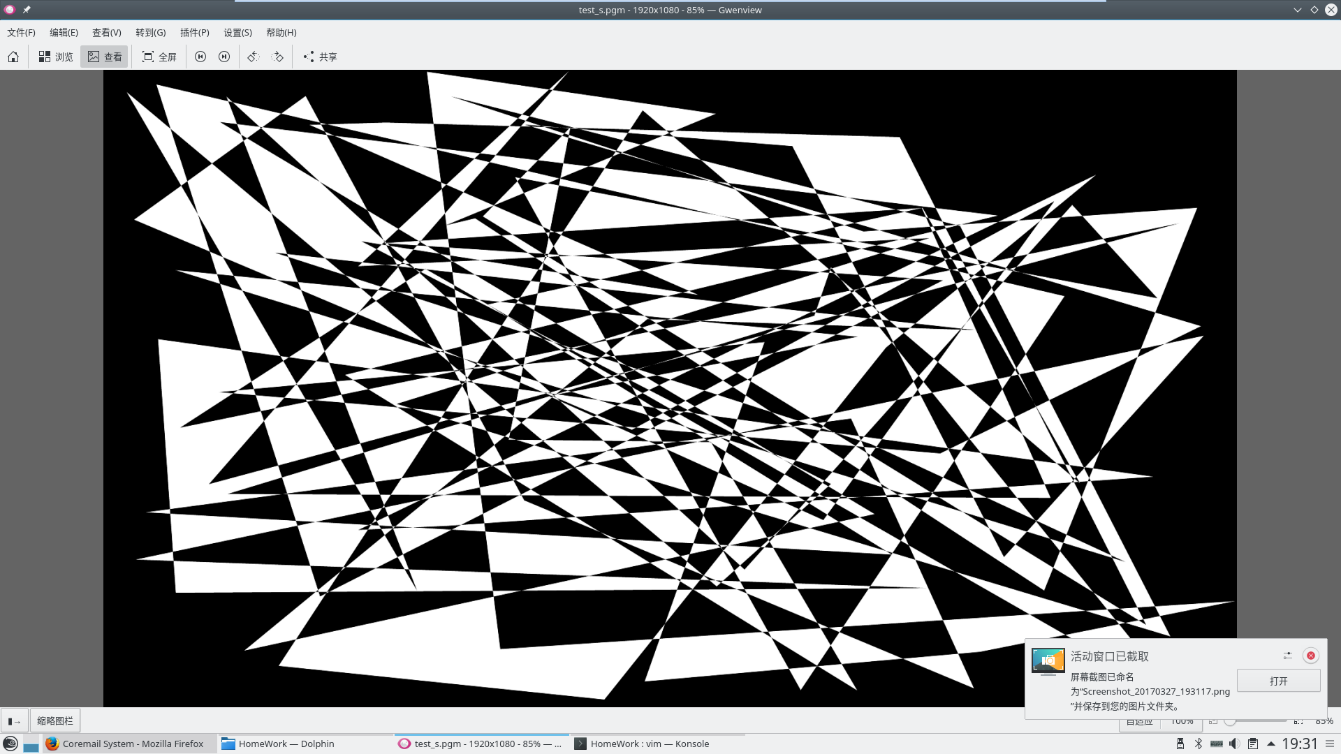
1.多边形填充与反走样

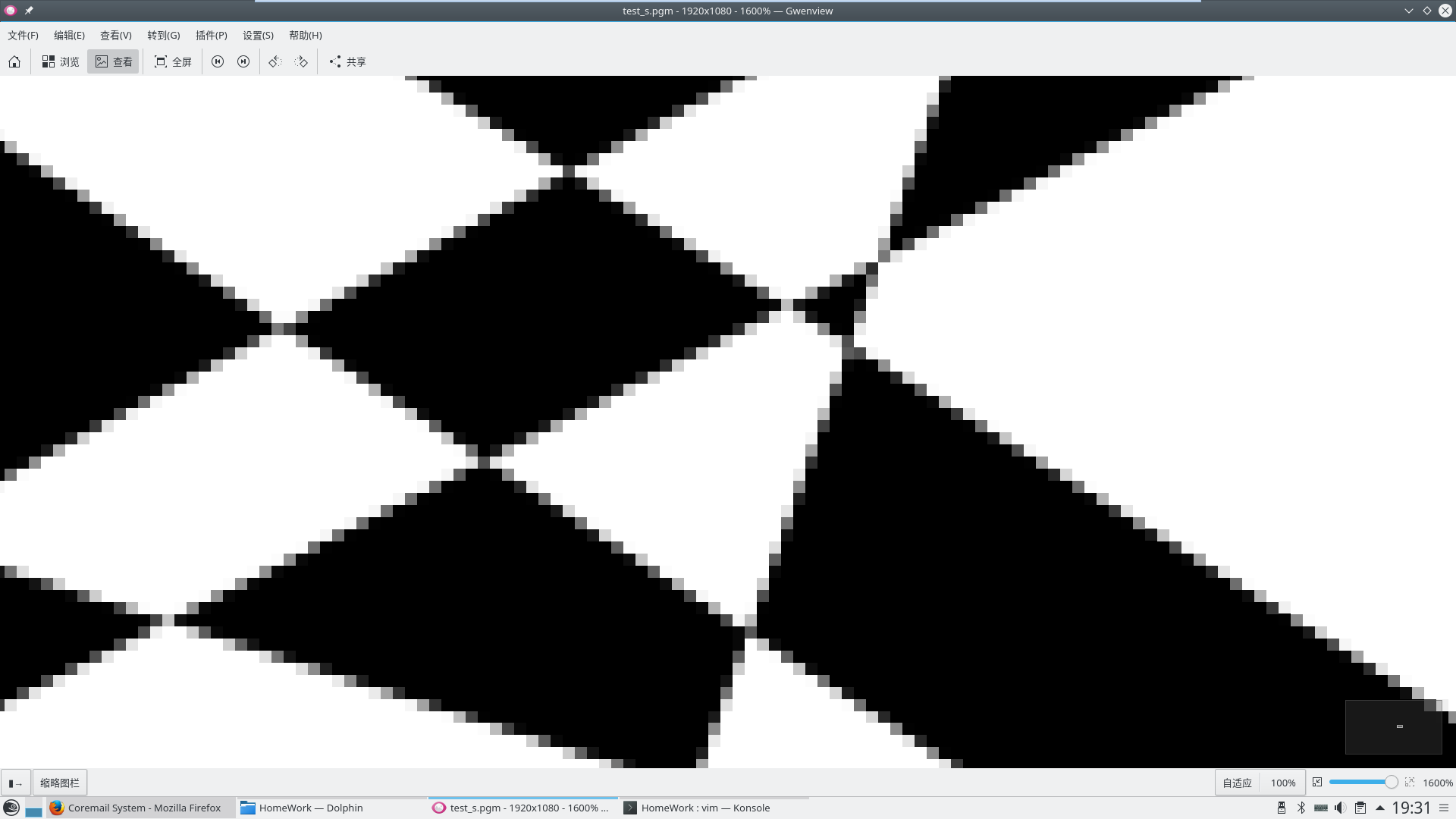
随机三角形：



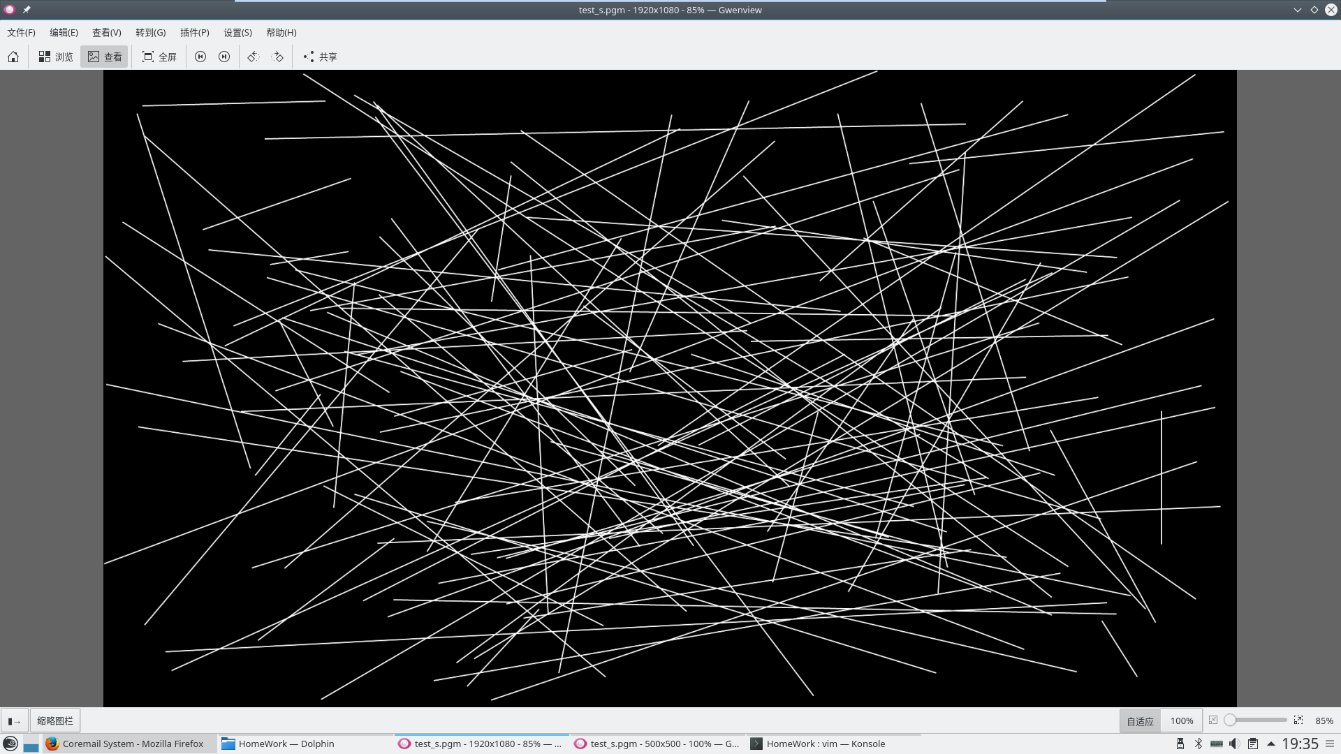


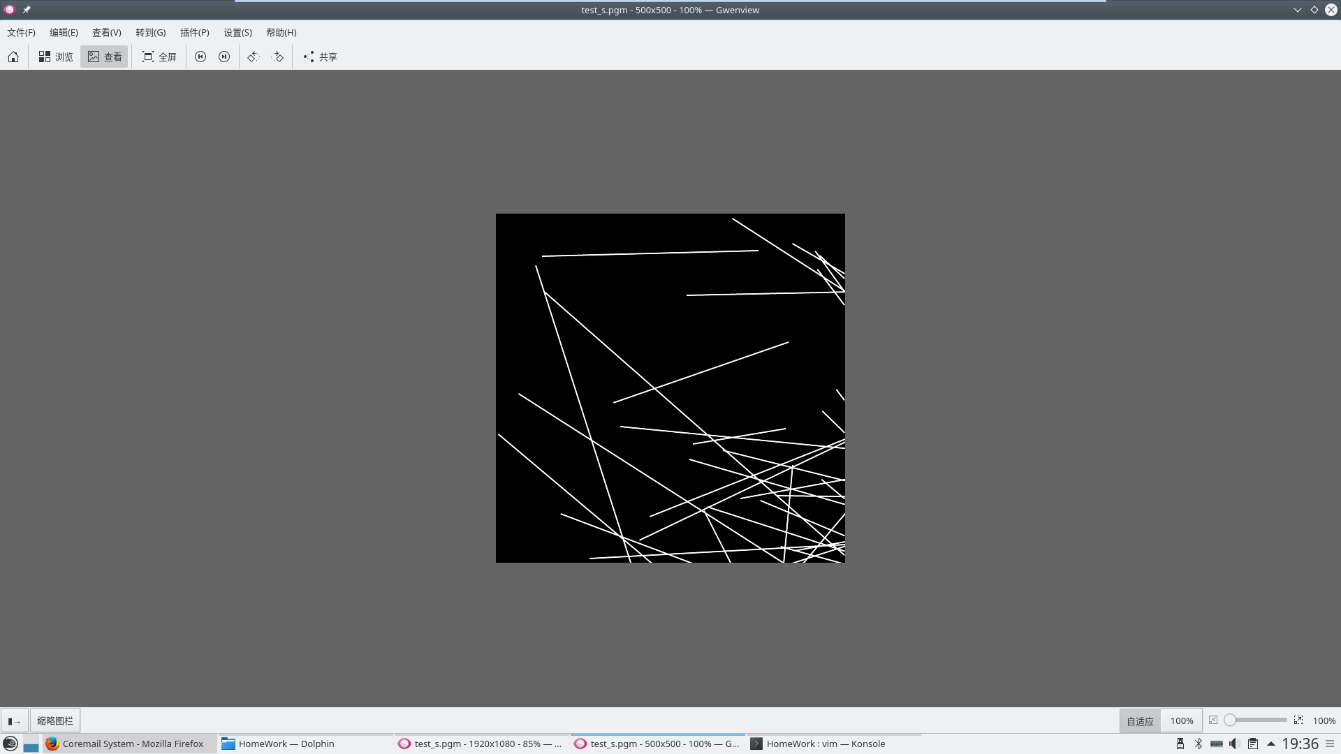
随机 100 边形：





2.二维线段裁剪的 Cohen-Sutherland 算法





3.二维线段裁剪的 Liang-Barsky 算法  
