CG 实验报告

胡俊豪 181240020@smail.nju.edu.cn

1 准备工作

1.1 Ubuntu 18.04 虚拟机

我们在 VirtualBox 上安装了 Ubuntu 18.04，64 位系统，50GB 内存，4 个核心。

1.2 Anaconda

我们在安装好的 Ubuntu18.04 上安装了 Anaconda3，并创建了名为 graphics 的虚拟环境，使用的 python 版本和各种包的版本与实验要求一致，分别为：python 3.7.4，numpy 1.18.1，pillow 7.0.0，pyqt 5.9.2。

我们选择的 IDE 是 Anaconda 自带的 Spyder IDE。

1.3 初试代码

我们将实验网站上的示例代码下载下来，放入自己搭建好的环境，并成功运行。在 gui 和 cli 两个界面下进行了探索，对整个实验的要求和最后的结果有了一个更加深刻的理解。

2 cg_algorithm.py

在这个章节，我们将对 cg_algorithm.py 文件中涉及到的每一个函数，所用到的每一种算法的原理进行分析和对比。

2.1 draw_line

2.1.1 DDA

DDA 算法又称数值差分分析，它在一个坐标轴上以单位增量对线段离散取样（$\Delta x = 1$ 或 $\Delta y = 1$），确定在另一个坐标轴上最靠近线段路径的对应整数值。取样方向决定于直线斜率绝对值的大小：斜率绝对值小于 1 时，在 x 方向取样，计算 y 方向的位置坐标；斜率

绝对值大于 1 时，在 y 方向取样，计算 x 方向的位置坐标。增量取决于线段生成方向与坐标轴方向的关系：二者相同取 $+1$，相反取 -1。

DDA 算法比直接使用直线方程快，但是求解每个像素点位置的时候都必须进行取整操作和浮点运算，这会导致误差的累计，以及消耗大量运算时间。

要优化 DDA，可以将增量乘以一个常数，把所有运算变成整数操作。

2.1.2 Bresenham

Bresenham 提出一种算法。根据光栅扫描原理，线段离散取样过程中的每一个放置位置上只可能有两个像素接近于线路经。考虑两候选像素于实际路劲间的偏移关系，选择更近的一个像素作为线的一个离散点。

需要注意的是，水平线，垂直线以及对角线可以直接装入帧缓冲器，无需进行画线算法处理。

Bresenham 算法相比于 DDA 的优点是：它同样可以用作除了线段以外的其他图形的绘制。

2.1.3 进一步优化

在使用 DDA 或者 Bresenham 算法进行线段绘制的时候，免不了会讨论各种各样繁琐冗余的情况。比如斜率大于一，斜率小于一，斜率为正，斜率为负，从左到右画，还是从右到左画等等。为了统一画线方式，我们试图采取以下方法进行代码优化。

首先，对于任何线段，我们都从左往右画；其次，对于任何线段，我们首先把它的左端点移到原点，并带着右端点做整体平移；接下来，我们试图把这根直线对称到第一象限，斜率小于一的八分之一部分，在那里进行描点；最后，把所描的点，对称回原来直线所在的八分之一部分，并按照原来的轨迹，把所有点平移回原来的位置。

这样一来，无论是哪种直线，我们只关心如何画好第一象限斜率小于一的那八分之一部分的直线，就可以了。