光栅化作业报告

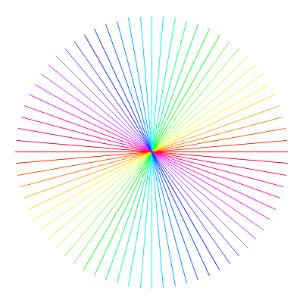
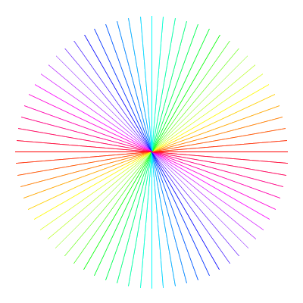
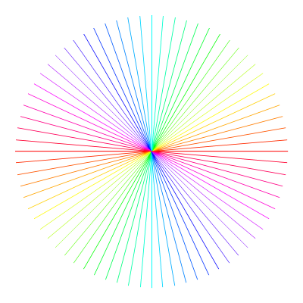
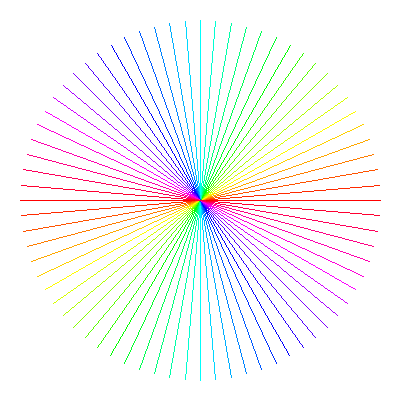
2015011308 计53 唐适之

**概要**

本作业实现了直线画线算法，及其三种反走样算法——SSAA（提高分辨率后使用OpenCV缩小）、Sampling（区域采样）、Kernel（加权区域采样）。程序输出不同颜色不同方向的直线，以进行全面的比较。为了处理直线相交的情况，本程序绘制一定灰度的像素时，实际上是以该灰度为比例，将当前绘制的颜色与已绘制的背景，即以前绘制的直线，进行混合。

**效果**

输出效果如下（从左至右依次为朴素算法、SSAA、Sampling、Kernel）：



组图1: 完整效果

为展示反走样效果，将原图局部放大后效果如下（顺序同上）：



组图2: 局部效果

可以看出，三种反走样算法均实现了反走样效果。其中，SSAA与Sampling表现相似，这是因为使用OpenCV在缩小图片时实际上是采用的Sampling算法。Kernel算法绘制的直线的间断感比Sampling算法绘制的小，体现了加权的优势。

另外从效率上看，若记直线长度为n，采样精度为d（每个像素划分成d2个小格），通过预处理权值矩阵的局部和，程序实现了O(n·d)的Sampling和Kernel算法，而SSAA算法的复杂度为O(n·d2)，效率不如Sampling和Kernel算法。本程序可以输出实际运行时间，从中也可以看出Sampling和Kernel只稍微比朴素算法慢，而SSAA要慢数十到一百倍。

原始输出图像请见output文件夹。

**具体实现**

本程序首先实现了Draw类作为朴素算法，其中draw函数将直线镜像翻转为斜率∈[0,1]的情况，交给drawImpl执行具体算法。DrawWidth、DrawSampling、DrawKernel分别直接或间接继承Draw类，覆盖其drawImpl函数以执行SSAA、Sampling和Kernel算法。最后Canvas类负责与OpenCV交互以输出。

具体代码见src文件夹，此外还有Doxygen导出的各函数和类的说明，见doc/doxygen/index.html。