VCX-lab1 报告

陈冲寒 2000012946

1. Dithering算法

(1) Uniform random

使用rand生成随机数，对10000取模再除以10000，减去0.5，可以得到[-0.5, 0.5]大致的均匀分布

对每个像素加上该均匀分布输出图片

(2) BlueNoise

对每个像素加上读取的蓝噪声并减去0.5来实现扰动，再输出图片

(3) Ordered

用一个数组`int pos[10][2]`存储10个阶层灰度对应3*3图像中，比上一个阶层多出的像素坐标

实现时，将输入像素灰度值*10向下取整得到对应图像编号t，然后依次把pos[1~t]的坐标点亮

(4) ErrorDiffuse

对每一个像素，将其产生的误差向右下方扩散后，将扩散的误差存储在右下方各个像素的output里面。

当准备处理该像素数，将输入的值与output中已经累积的误差相加再做处理。

由于该lab实现的ImageRGB只能存储[0, 1]的值，对于负数误差和大于1的误差无法存储。这里使用`output.r`存储正数误差，`output.g`存储负数误差，并且在存储误差时缩小10倍来防止误差超过1

在读取误差时使用`(output.r-output.g)*10`来获得真实的误差值。

2. Image Filtering

(1) Blur

使用卷积核

$$\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

在`0 <= x < input.GetSizeX() - 2`，`0 <= y < input.GetSizeY() - 2`位置应用卷积核

得到大小为`(input.GetSizeX() - 2) * (input.GetSizeY() - 2)`的输出图片

(2) Edge

分别使用以下两个卷积核来得到对x的偏导和对y的偏导

$$\begin{bmatrix} -1 & 0 & -1 \\ -2 & 0 & -2 \\ -1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

梯度的大小代表边界，而梯度模长为$\sqrt{\left(\frac{\partial s}{\partial x}\right)^2 + \left(\frac{\partial s}{\partial y}\right)^2}$

则在该两个卷积核应用完后，将结果平方加和再开方，得到输出图像的像素值。

3. Image Inpainting

覆盖上去的飞机图片为$f(x,y)$，对应背景位置为$g(x,y)$，最后得到输出像素为$g(x,y)+f(x,y)$

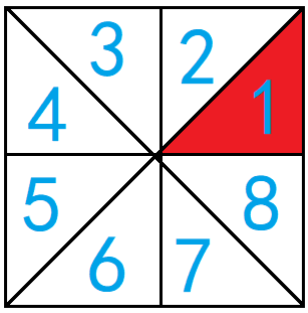
则边界位置需保证填充后与背景原图一致，即$g(x,y)+f(x,y)=back(x,y)$

则边界部分$g(x,y)=back(x,y)-f(x,y)$

剩余部分使用Jacobi迭代使得图片变得平滑

4. Line Drawing

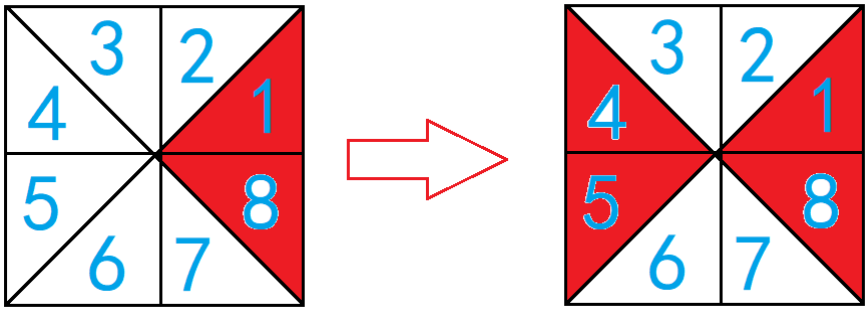
对于$x_0<x_1$，$0<slope<1$时（即如下图象限），使用课上讲的Bresenham算法。



首先考虑扩展到第8象限，此时dy为负数，则将dy取反为正数，并修改`y++`为`y--`即可。

使用变量`flagy = y0 < y1 ? 1 : -1`来判断dy是否为正，画直线需要修改y时使用`y+=flagy`

在考虑扩展到4、5象限，即dx为负数的情况（如下图）



增添变量`flagx=x0 < x1 ? 1 : -1`来判断，并将dx取反为正数，修改x时使用`x+=flagx`

在考虑扩展到3、2、6、7像素，即斜率大于1的情况

此时只需要在最开始交换x、y坐标，并在输出前交换回来即可。

5. Triangle Drawing

首先将三个点按照x坐标从小到大排序为q_0,q_1,q_2

首先绘制x坐标在q_0,q_1之间的点，计算直线q_0q_2,q_0q_1的斜率

对于q_0,q_1之间每一个x坐标，通过斜率计算出两条直线的y坐标，将两个y坐标之间的点涂上颜色。

对于x坐标在q_1,q_2之间的点同理

6. Supersampling

对于输出图像的坐标(x,y)，计算其在输入的大图上的坐标(浮点数)：$(x',y')=(x,y)\times k$

k大图对于小图的比例

在(x', y')坐标附近进行$rate \times rate$的采样，即将输出图的像素拆成$rate \times rate$大小，每一个小格子的长度为$1/rate$，对应大图的长度为$k/rate$

由于$rate \times rate$的方格要保证(x', y')在中心位置，则这个方格左上角的位置为
$(x', y') - k/rate * [(rate - 1)/2] * (1, 1)$

则方格中$[i, j]$对应的大图坐标为
$(x', y') + (k/rate * [i - (rate - 1)/2], k/rate * [j - (rate - 1)/2])$

四舍五入读取这个坐标的颜色数据

将方格里每个采样取平均，得到最终输出图像的颜色。

7. 贝塞尔曲线

首先将给的points复制一份方便实现

循环n次

每次令`points[i]=(1-t)*points[i]+t*points[i+1]`

则最终points[0]即为该点的坐标。