可视计算与交互概论 Lab1 报告

邱荻 2000012852

2022 年 10 月 14 日

1 介绍

这次 Lab 中，我实现了可视计算与交互概论课程前 8 讲中介绍的几种重要的算法或思想，包括 Dithering 算法，Image Filter 算法，Poisson Editing 算法，Bresenham 直线算法，Scan Converting 三角形算法，Supersampling 算法以及 de Casteljau 算法。

2 实现思路和结果

2.1 Uniform Random

给每个像素加上 [−0.5,0.5] 中均匀分布的随机扰动，然后使用 Threshold 算法。用 rand() / (double) (RANDMAX)-0.5 产生 [-0.5,0.5] 中的随机数。

图 1: Uniform Random

2.2 Blue Noise Random

给每个像素加上蓝噪声的随机扰动，然后使用 Threshold 算法。把每个像素都加上所给蓝噪声然后减去 0.5 就可以了。

图 2: Blue Noise Random

2.3 Ordered

使用课件 P24 给的 3x3 的有规律的黑白像素分布表示原图的一个灰度像素。[0,0.1) 的像素值用 0 号 3x3 黑白像素，[0.1,0.2) 用 1 号……[0.9,1] 用 9 号

图 3: Ordered（实际图片比是其他图片的三倍大）

2.4 Error Diffuse

将舍入的时候被剥夺的像素值以 7/16,3/16,5/16,1/16 的比例分配给了下面的像素、右上的像素、右边的像素、右下的像素，也就是离该像素最近且还没有被舍入处理过的像素（我是外层循环是 x 里层循环是 y）。

图 4: Error Diffuse

2.5 Blur

每个像素变为自己和自己周边像素的平均值。如四个角的地方就是四个像素平均，边上就是六个像素平均，里面的就是九个像素平均。

图 5: Blur

2.6 Edge

先用竖着的 Sobel Filter 进行卷积操作检测竖边，再用横着的 Sobel Filter 进行卷积操作检测横边，最后每个像素值等于每次操作的值的绝对值之和。

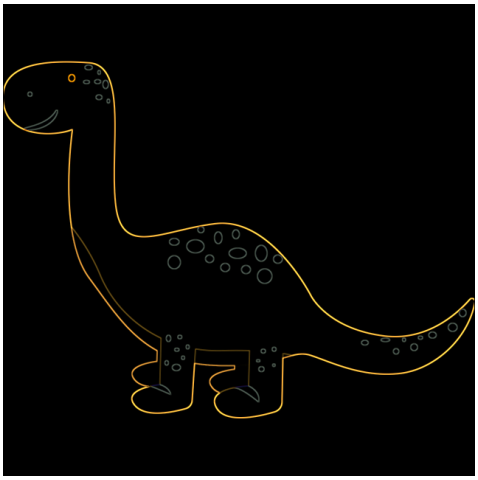


图 6: 边缘提取

2.7 Image Inpainting

这个函数用的时间比较久。一开始根据课件内容和网上资料我误认为 g 表示的是目标图像的像素值，但是在这样的想法下，后面的雅可比迭代解线性方程组就没有 divv，也就是没有常数项，并且最后 color = g[y * width + x] + inputFront.GetAt(x, y) 加了一项也十分奇怪，并且最终结果偏白，也就是像素值偏高。后来在助教的提示和点醒下，我明白了 g 是目标图像与飞机图像的像素差值，这下子前面的疑惑一下子解释得通了：“+ inputFront.GetAt(x, y)” 是因为之前减掉了，而雅可比迭代解方程组因为 g 是差值，所以就相当于插入的图片是一张纯色图，也就是 divv=0。最终结果如下：

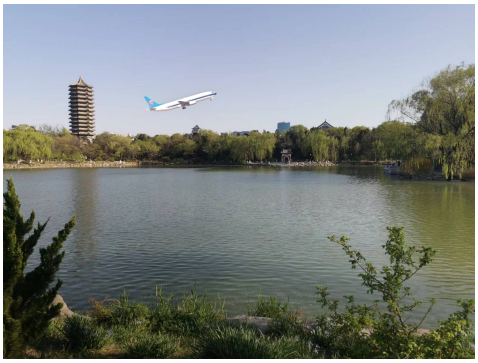
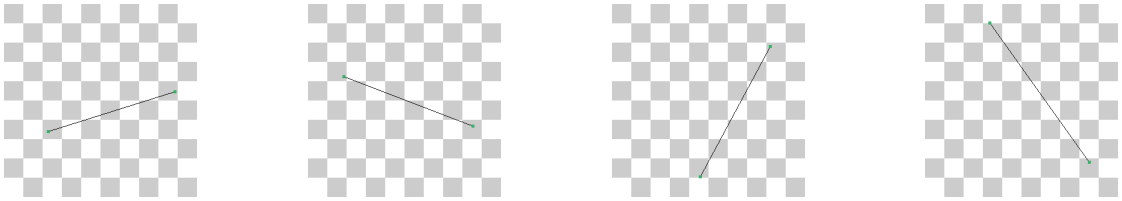


图 7: Image Inpainting

2.8 Line Drawing

用 Bresenham 算法绘制直线。先特判斜率不存在的情况，再按照课堂上给出的示例写出 slope 在 0 到 1 之间的情况，然后根据对称性写出其他的情况。如斜率在 1 到正无穷就是把 x 和 y 在相应的地方调换，斜率在-1 到 0 的情况就是在相应的地方把 x 变为-x，斜率小于-1 的情况就是先把 x 和 y 在相应的地方调换，然后 x 在相应的地方变为-x。最终实现结果如下：



(a) pic1. (b) pic2. (c) pic3. (d) pic4.

图 8: Lines

2.9 Triangle Drawing

思路是先特判三角形退化为直线的情况，直接画直线（因为后面有除以一个数的地方，担心分母变成 0 的时候有一个特判）。然后如果确定是三角形，先找出三个点中最高（y 最小）、次高、最矮（y 最大）的点，然后从次高的点画一条水平线把三角形切成两半，先画上面一半，再画下面一半，如果上面是平的或者下面是平的，就只画另一半。画法是横着扫描，一行一行地画。最终结果如下：

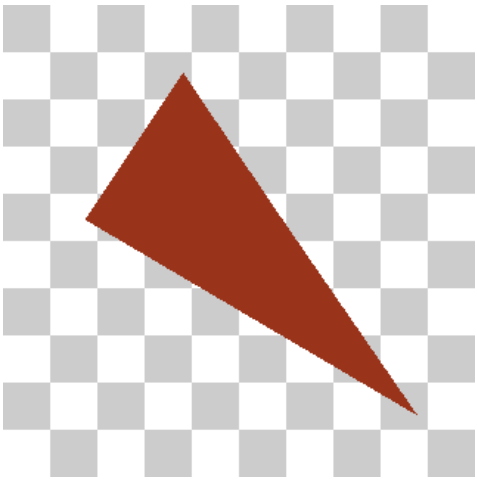


图 9: Triangle Drawing

2.10 Supersample

input 是 2500*2500，output 是 320*320, 为了应对大图缩小时的走样，在 output 的一个像素对应的 input 的若干个像素中取 rate 个点，把它们的像素值取平均，得到 output 的该个像素。这里我参数设置如下：output 的 (x,y) 对应 input 的以 (125/16*x，125/16*y) 为左上角的 8*8 个像素，在这 64 个像素里采样。部分结果如下（仅挑选了有代表性的展示）：

图 10: rate==1

图 11: rate==5

图 12: rate==12

2.11 CalculateBezierPoint

直接使用公式：$\mathrm{B(t)}=\sum_{i=0}^{n}\begin{pmatrix} n \\ i \end{pmatrix} Pi(1-t)^{n-i}t^{i} = \begin{pmatrix} n \\ 0 \end{pmatrix} P0(1-t)^{n}t^{0} + \begin{pmatrix} n \\ 1 \end{pmatrix} P1(1-t)^{n-1}t^{1} + \ldots + \begin{pmatrix} n \\ n-1 \end{pmatrix} Pn-1(1-t)^{1}t^{n-1} + \begin{pmatrix} n \\ n \end{pmatrix} Pn(1-t)^{0}t^{n}, t \in [0,1]$

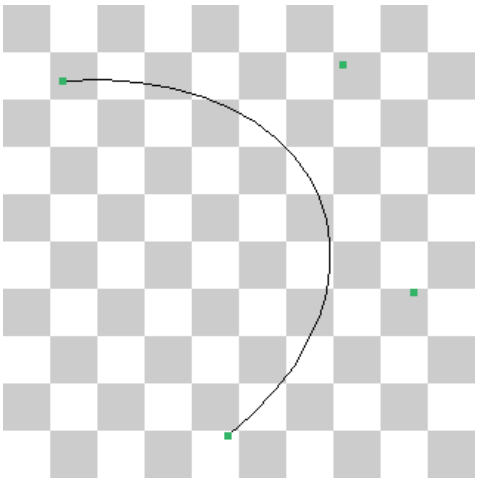


图 13: Bezier Curve

3 总结

整体来说我认为我的这个 lab 完成得结果很不错，肉眼判断几乎和给出的示例结果图相同。这个 lab 设计得很好，框架搭得很好，让我体验很好，和课堂内容紧密相关，要用的知识点在课件的哪一面都直接告诉我们了，十分贴心，让我在环境搭建和知识点的学习上省去了很多不必要的麻烦。通过对知识点的学习、琢磨与理解之后，我可以较为顺利地动手实践，感谢助教团队和 lab 小组设计出的这么好的 lab，我感觉收获很大，很能学到东西。