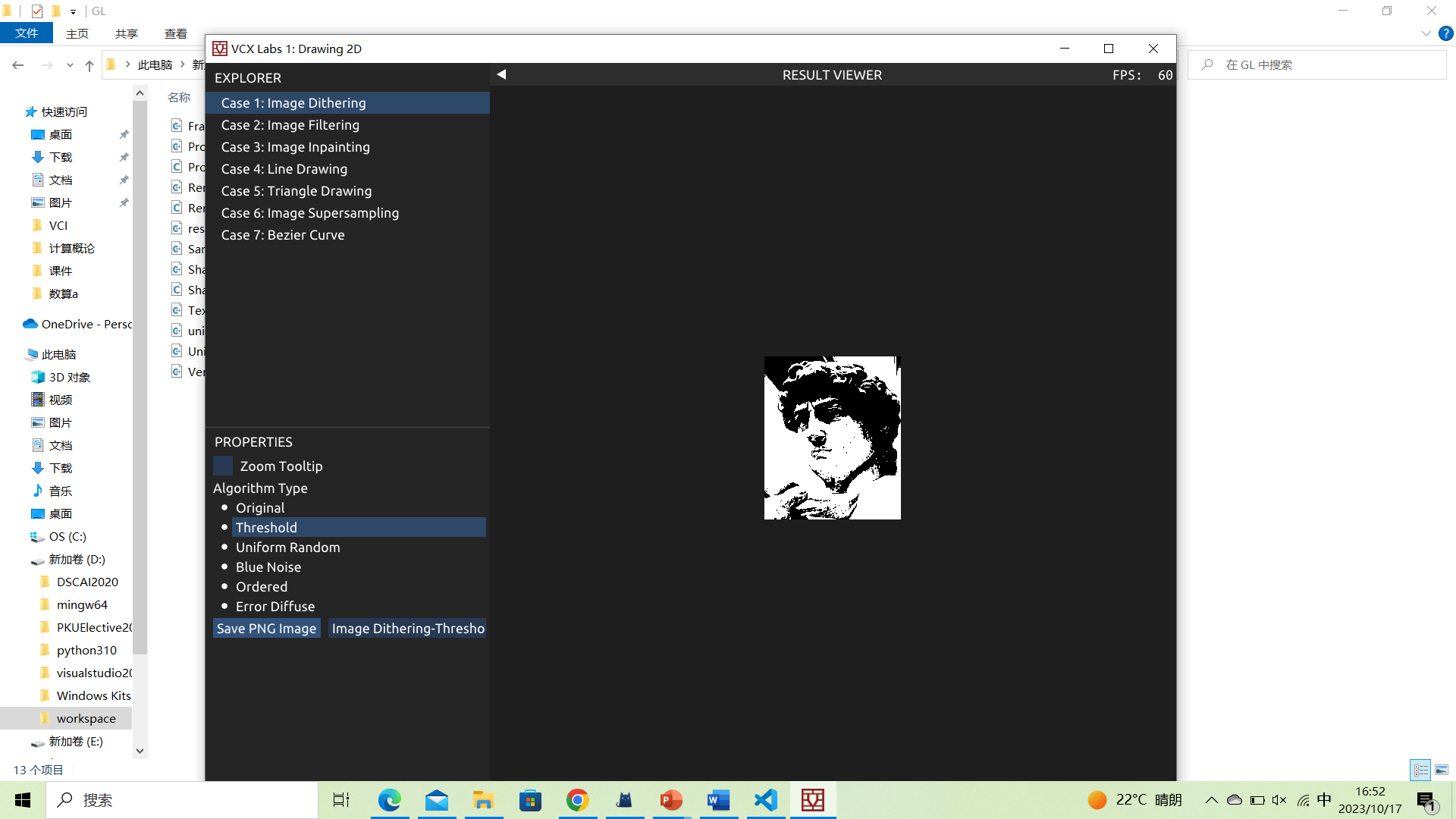
**VCL Lab1**

唐一早 2100012613

**Task 1: Image Dithering**

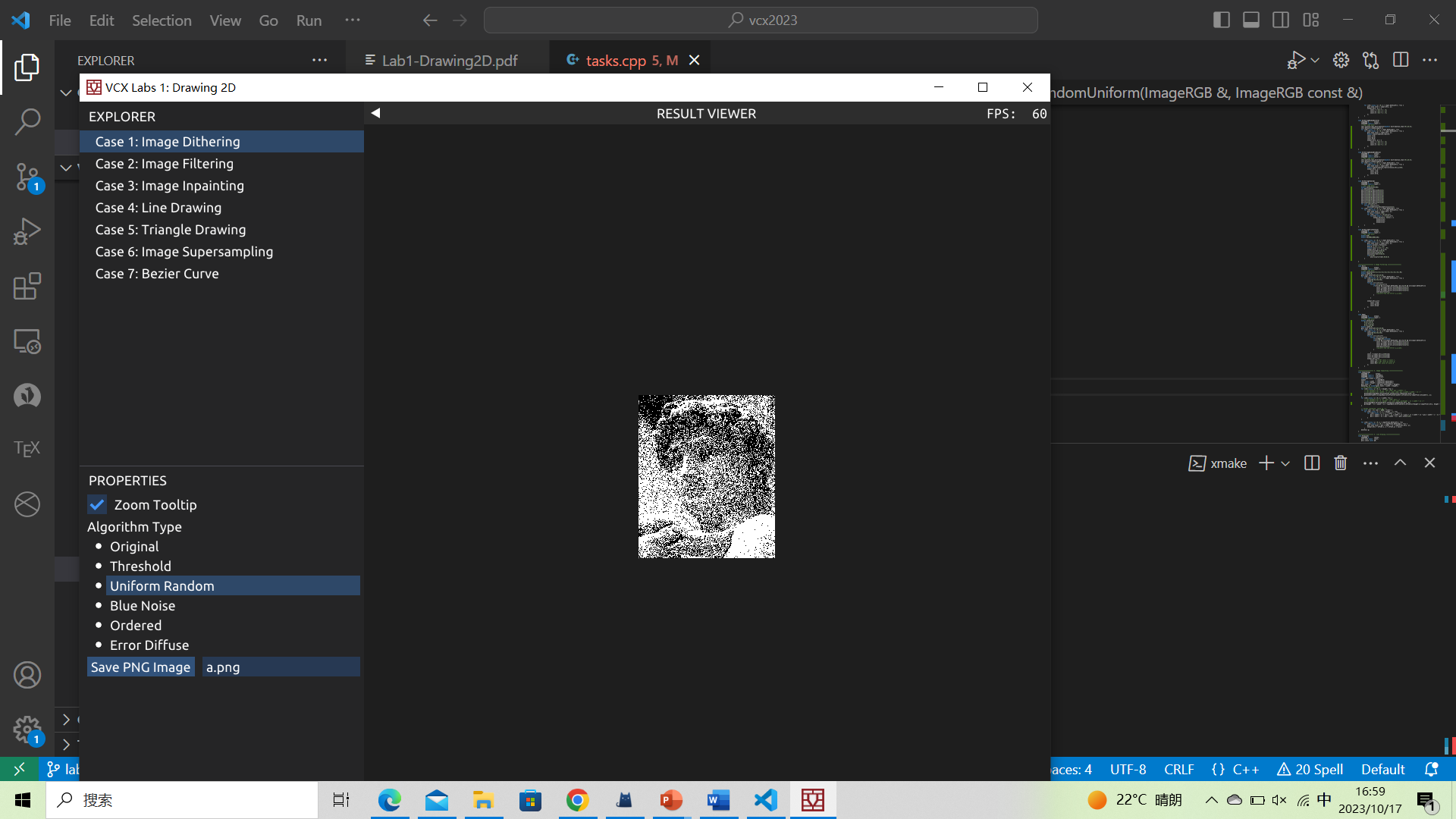
* 1. Threshold

Color>0.5时向上取1，否则向下取0。



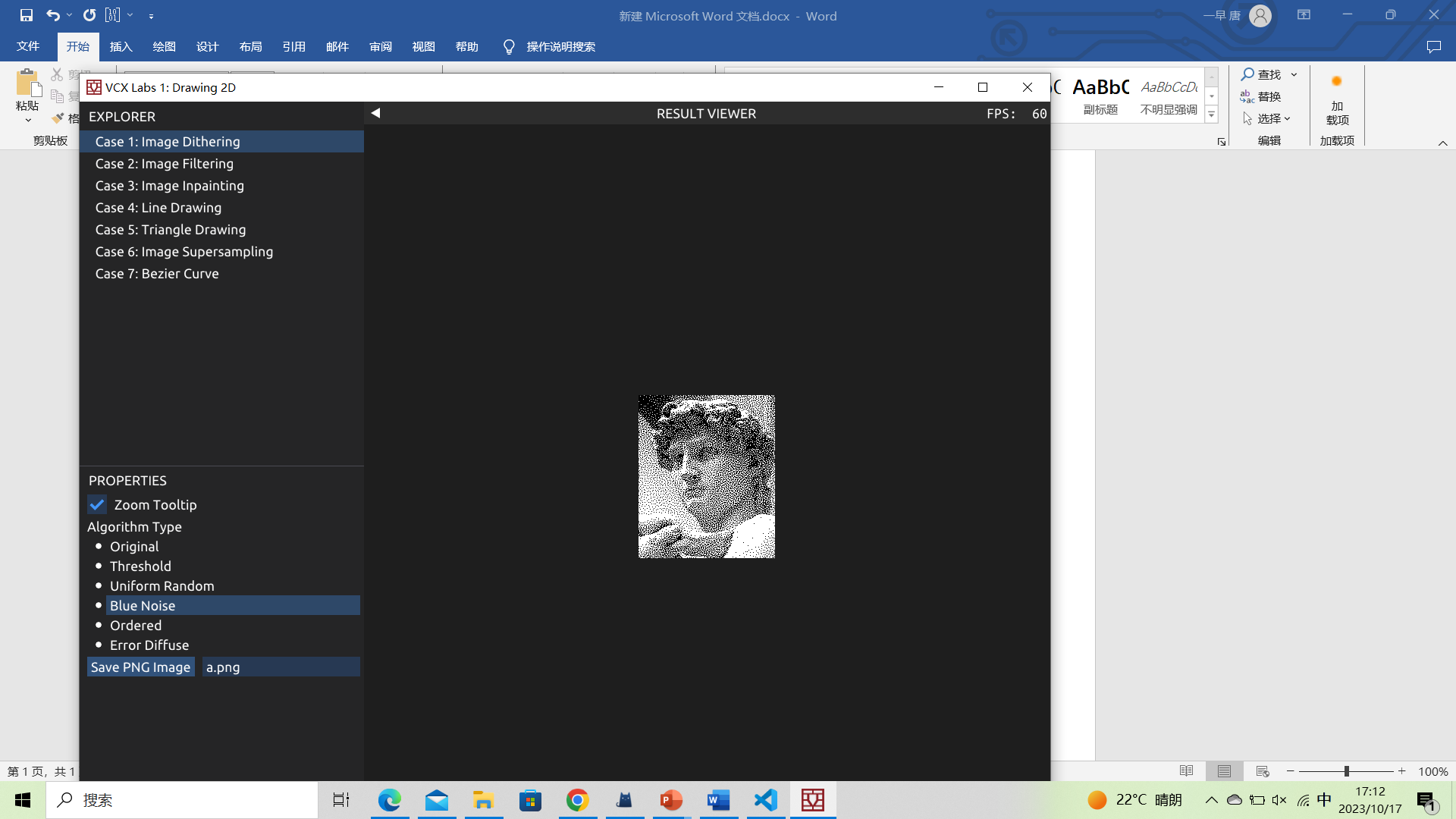
* 1. Uniform Random

用std::uniform\_real\_distribution <double> distribution\_real(-0.5,0.5) 构造(-0.5,0.5)的随机变量，让每一个像素加上。再threshold即可。



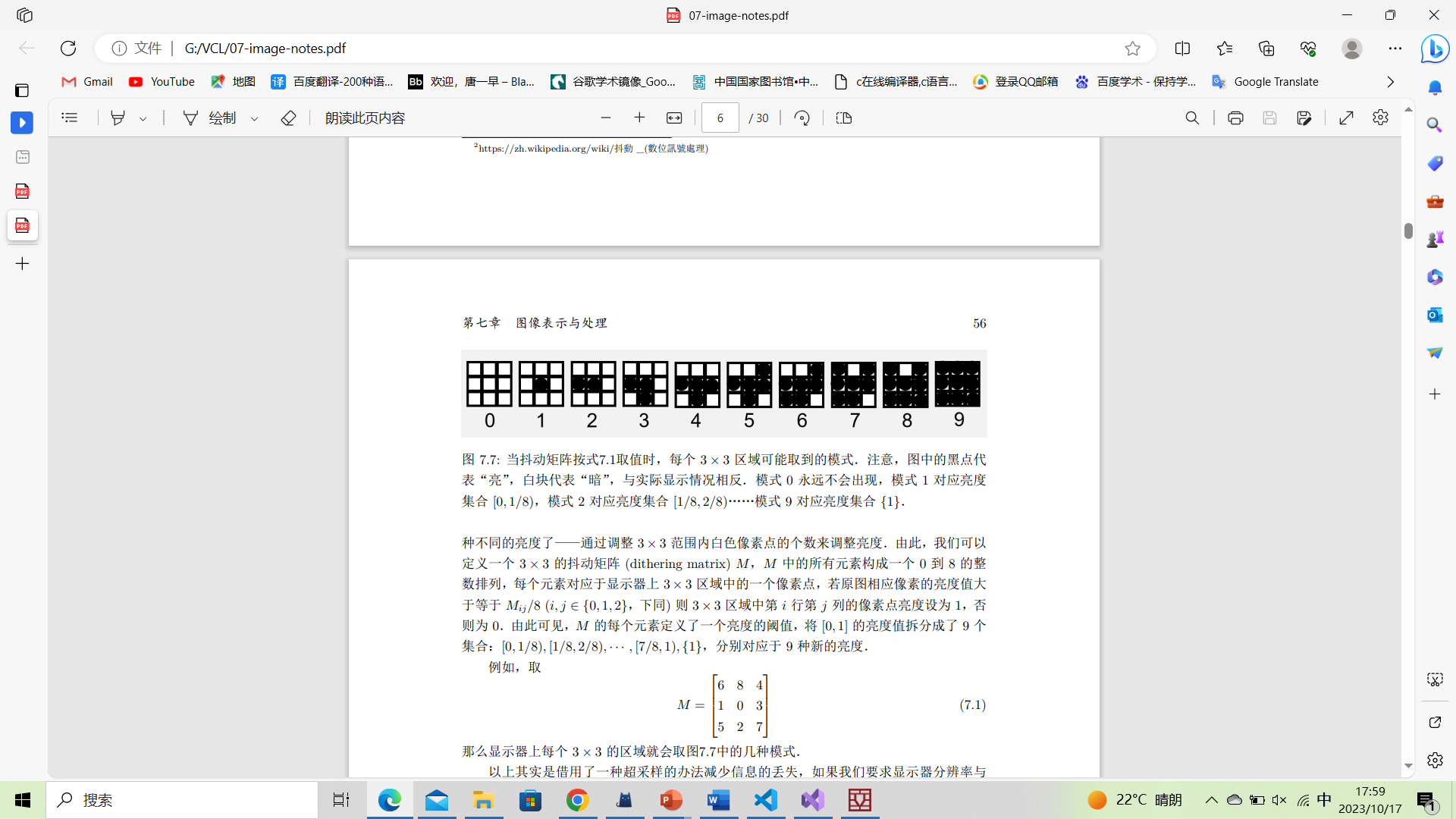
* 1. Blue Noise Random

给每个像素加上蓝噪声对应像素-0.5，（蓝噪声变化范围0~1，减0.5后正好变为-0.5~0.5），之后再threshold。



* 1. Ordered

先构造9个3 × 3 的抖动矩阵，将[0, 1]的亮度值拆分成了9个集合：[0, 1/9), [1/9, 2/9), · · · , [8/9, 1]，分别对应于 9 种新的亮度。然后遍历原图片，将矩阵复制到新图片的每个像素里即可。9个抖动矩阵如下。

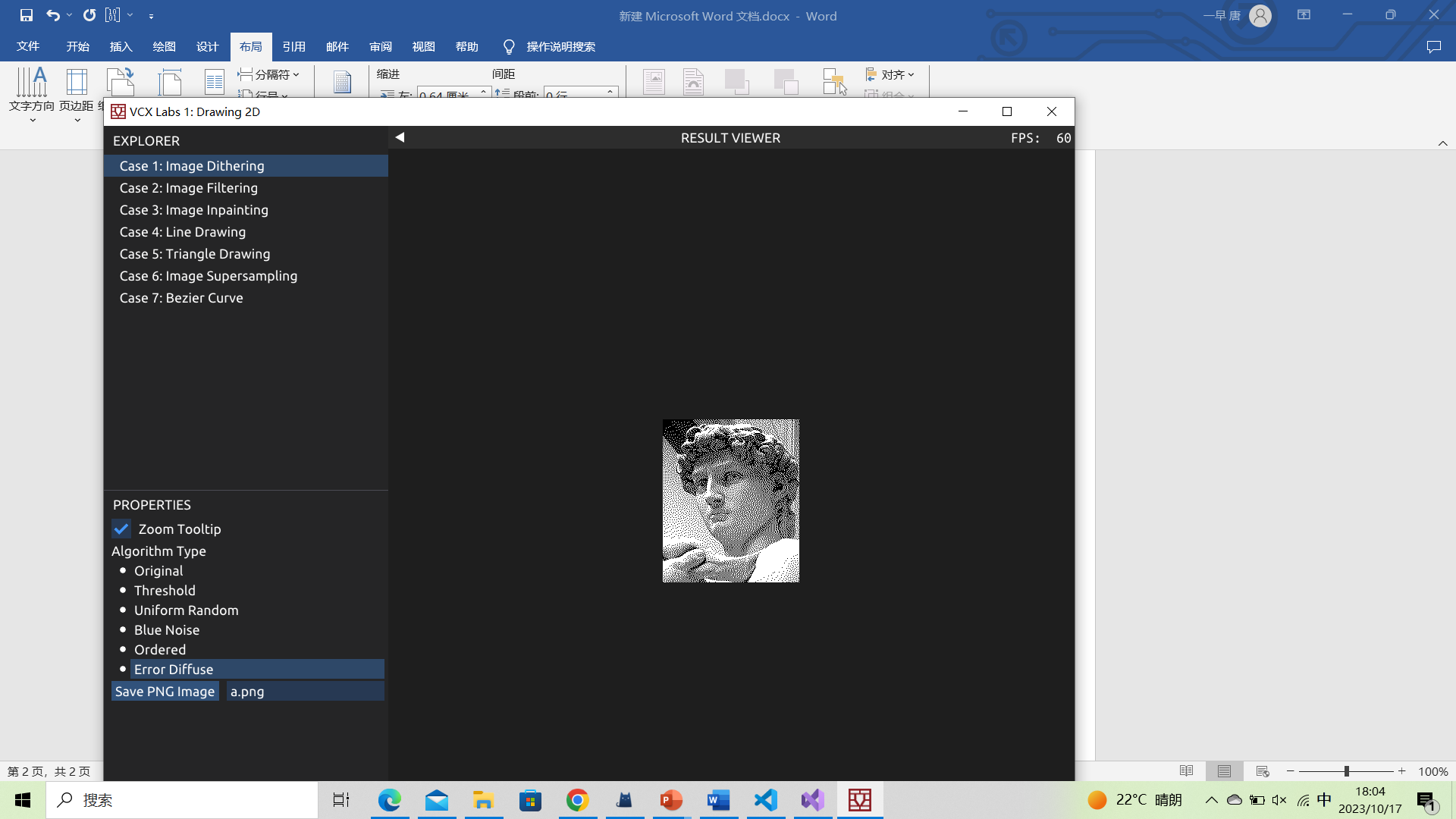


白色代表亮度0，黑色代表亮度1。



* 1. Error Diffuse

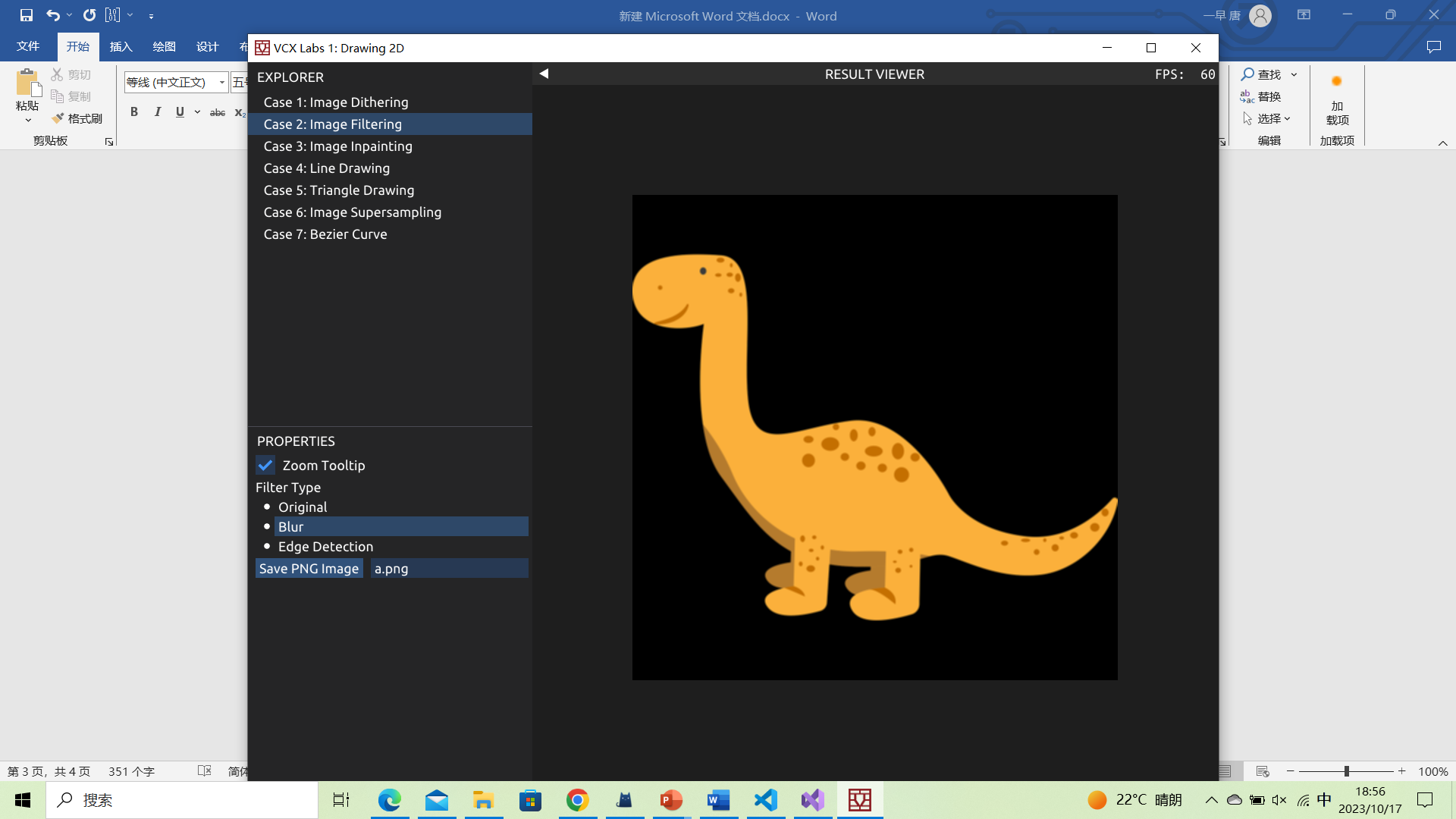
从左到右、从上到下依次处理每一个像素点，将当前像素点的亮度以0.5为阈值进行二值化，将二值化前的亮度减去二值化后的亮度作为误差值，然后将误差值分成占比分别为 7/16, 3/16, 5/16, 1/16 的四个部分，并分别加到右方、左下方、下方、右下方四个像素的亮度值.



**Task 2: Image Filtering**

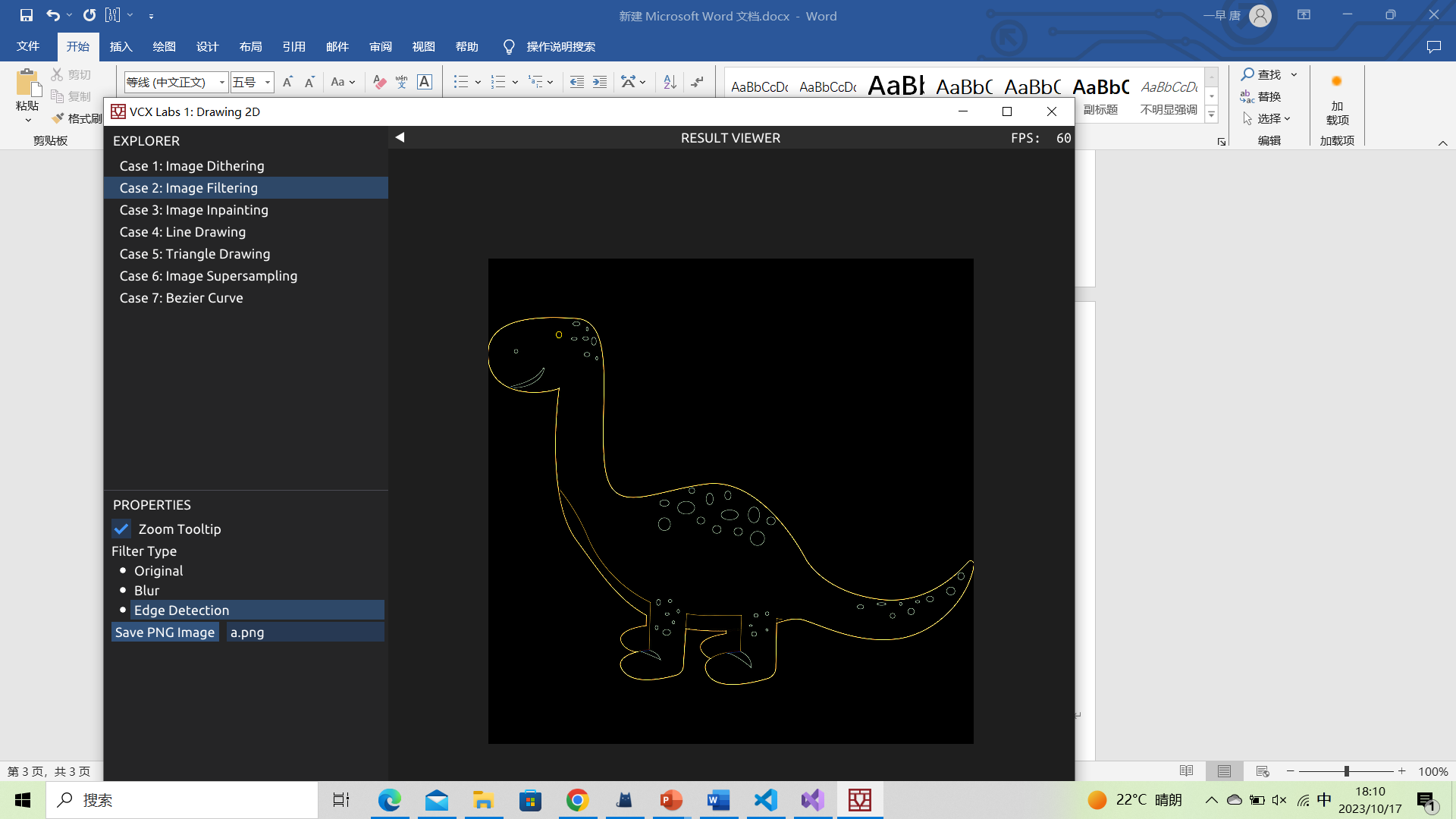
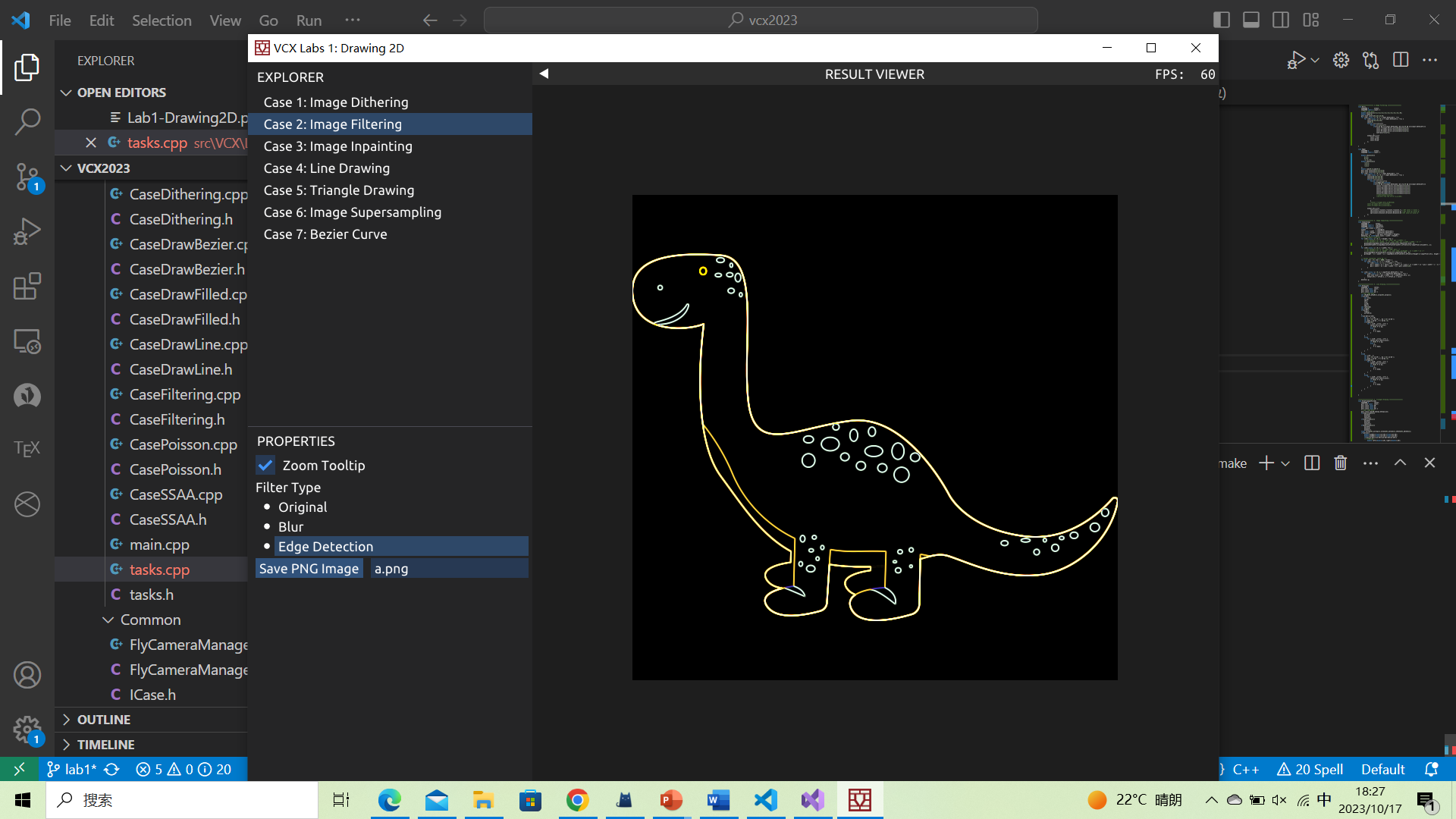
* 1. Blur

使用线性滤波器模糊，边界也用平均进行模糊。



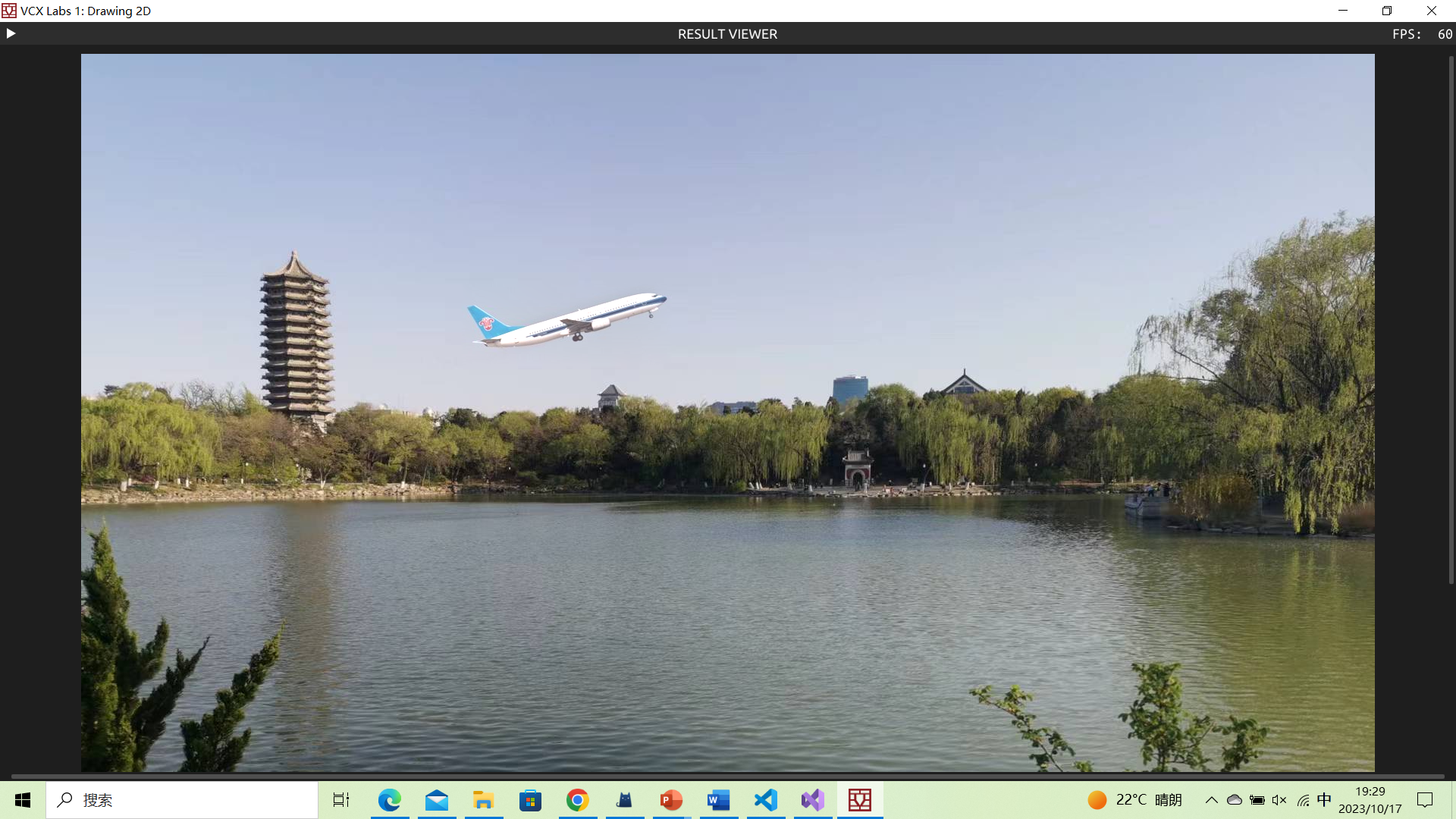
* 1. Edge

先尝试使用Laplace滤波器（如左图），效果较差，亮度不足，且线条很细。后改用Sobel算子，非线性滤波器。 分别使用和计算，最后计算，得到右图。

** **

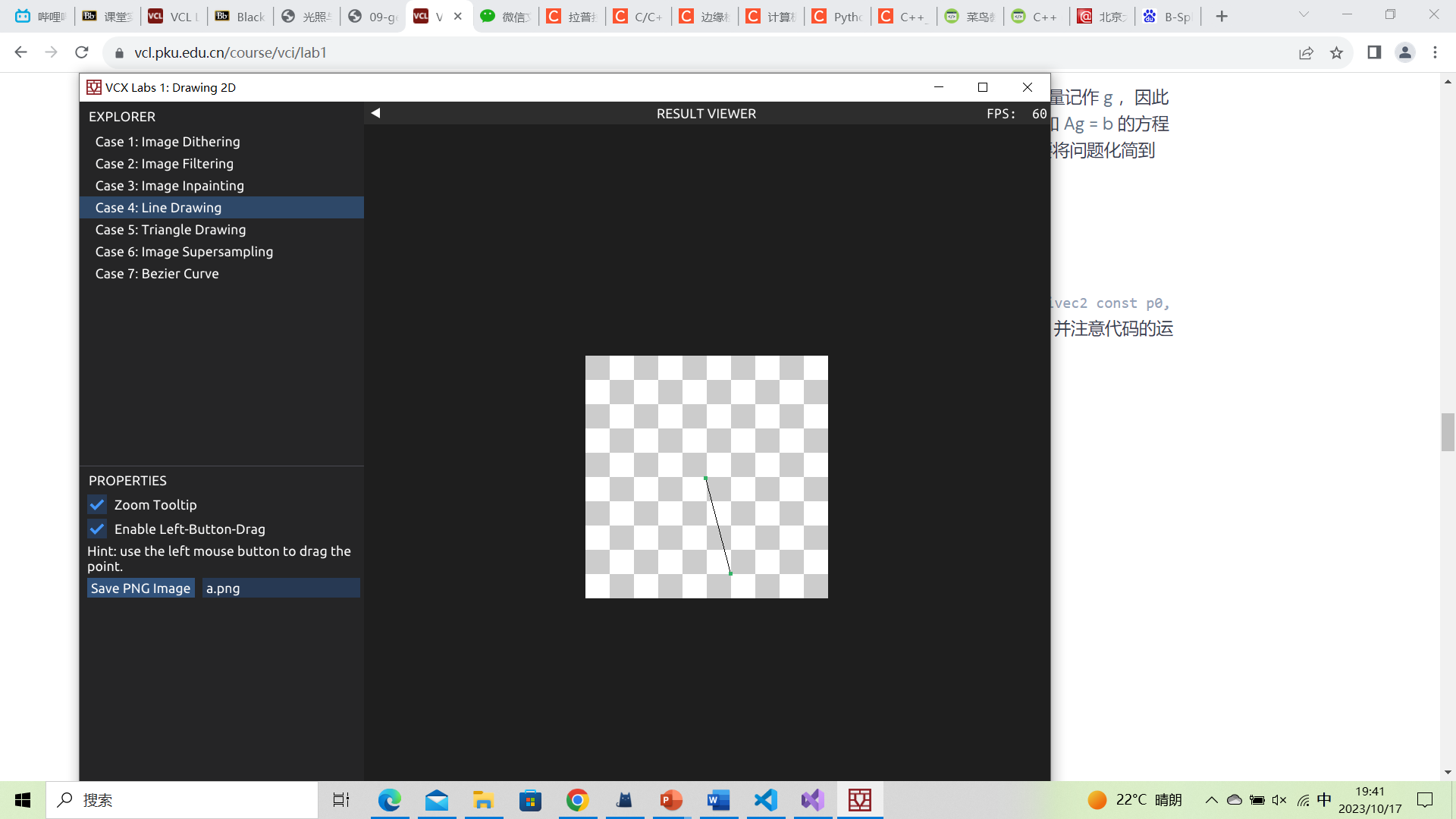
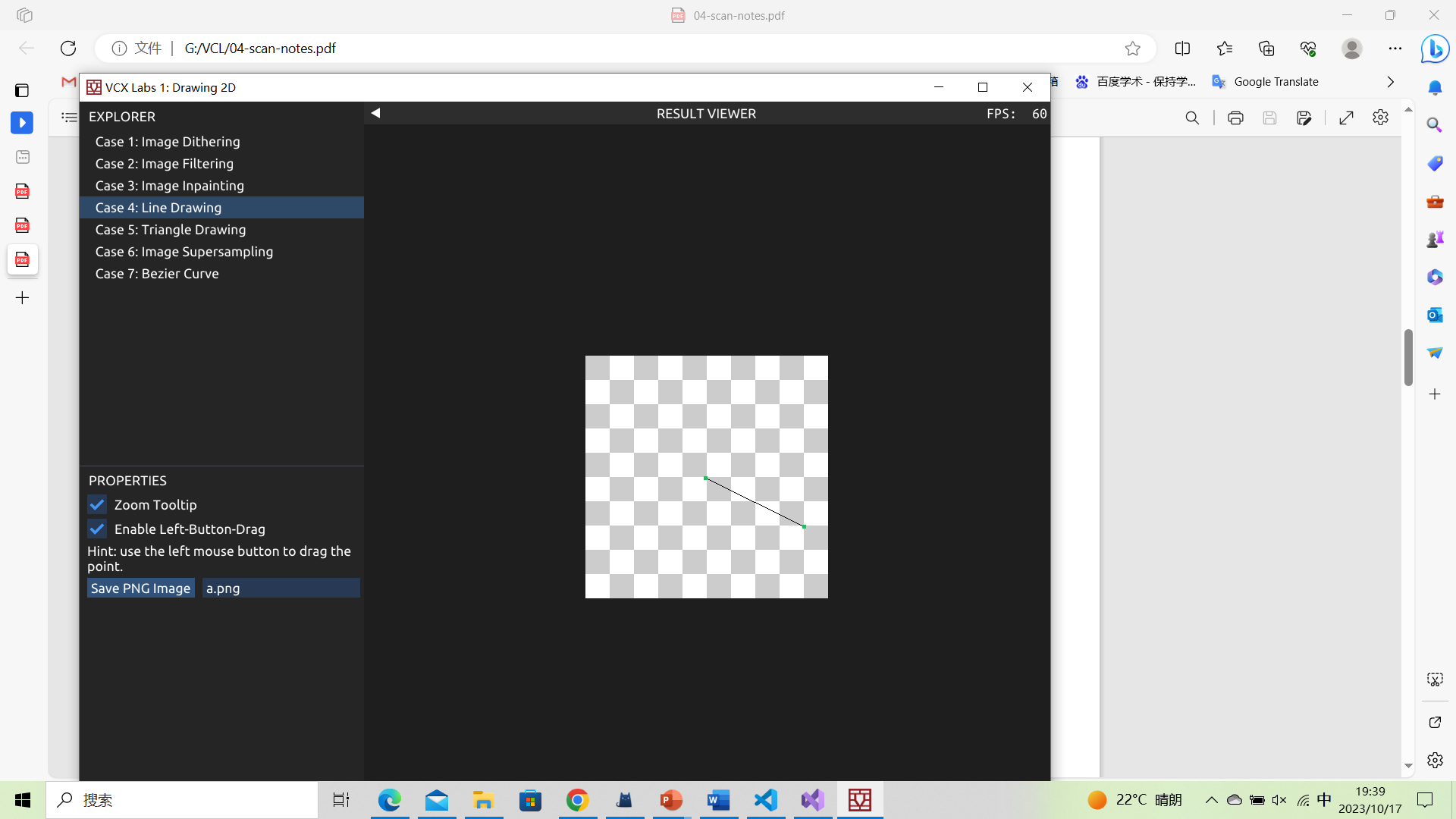
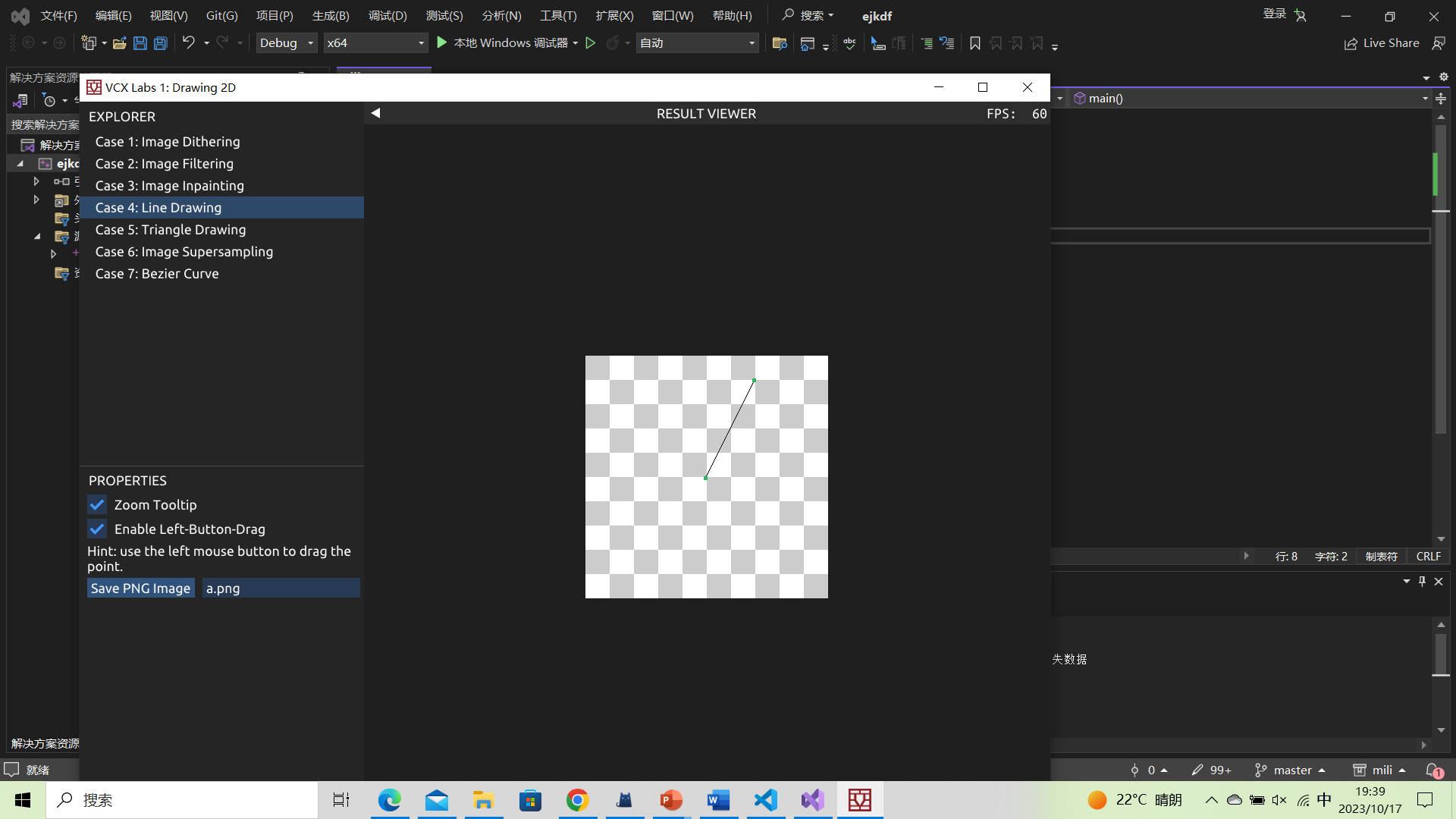
**Task 3: Image Inpainting**

因为边缘需要平滑过渡，所以对于边界上的点g[y\*width+x]=inputBack.At(offset[0]+x,offset[1]+y)-inputFront.At(x, y)。然后使雅克比 (Jacobi) 迭代计算线性方程组。



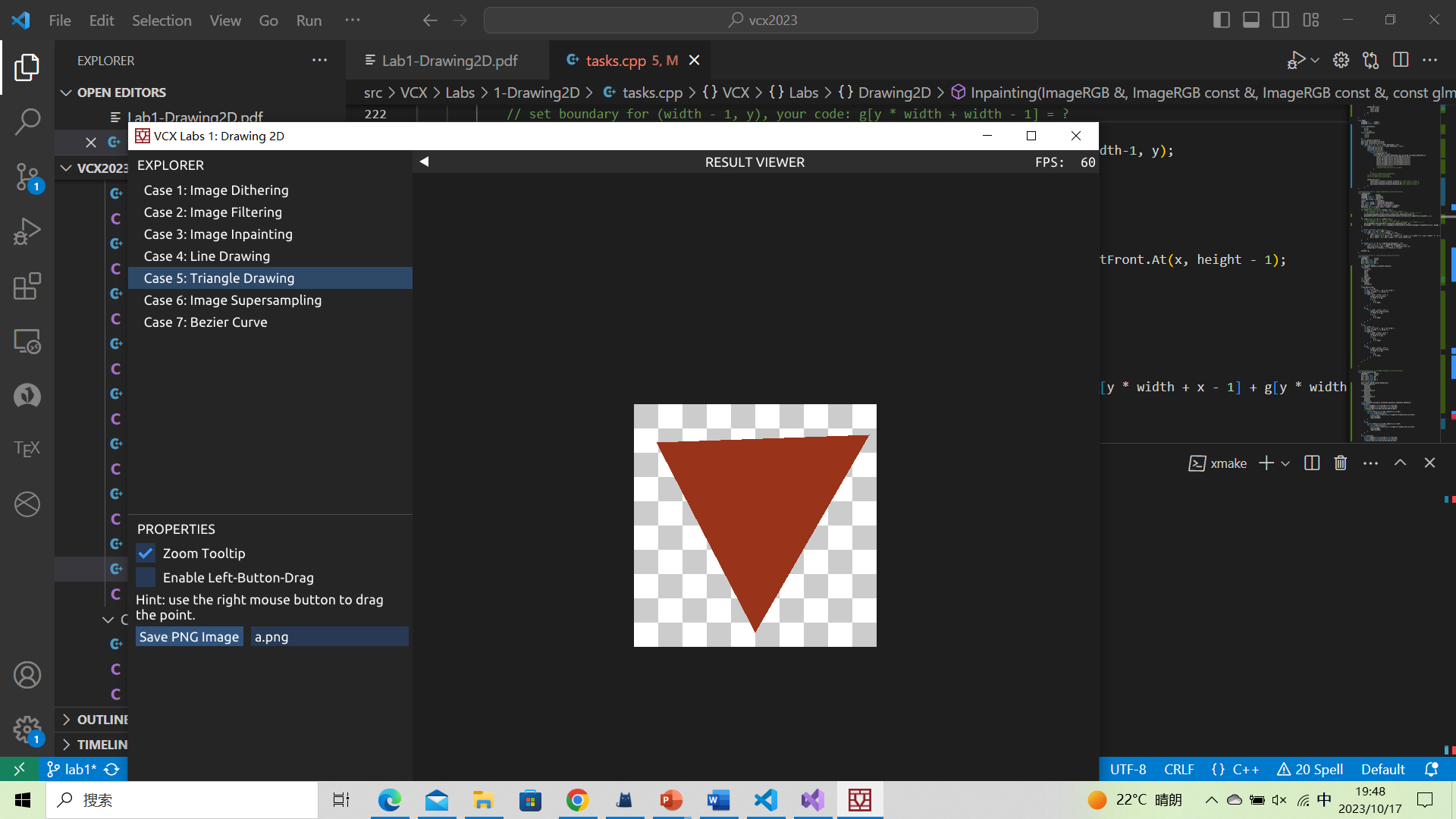
**Task 4: Line Drawing**

将所有结束点在起始点左边的直线，起始点与结束点反转，使结束点一定在起始点右边。将所有结束点在起始点下方的情况，沿以起始点为原点的x轴翻折，使结束点一定在起始点右上方，最后将所有斜率大于1的直线对y=x翻折，使得直线斜率一定小于等于1。最后使用Bresenham直线算法。画出直线。



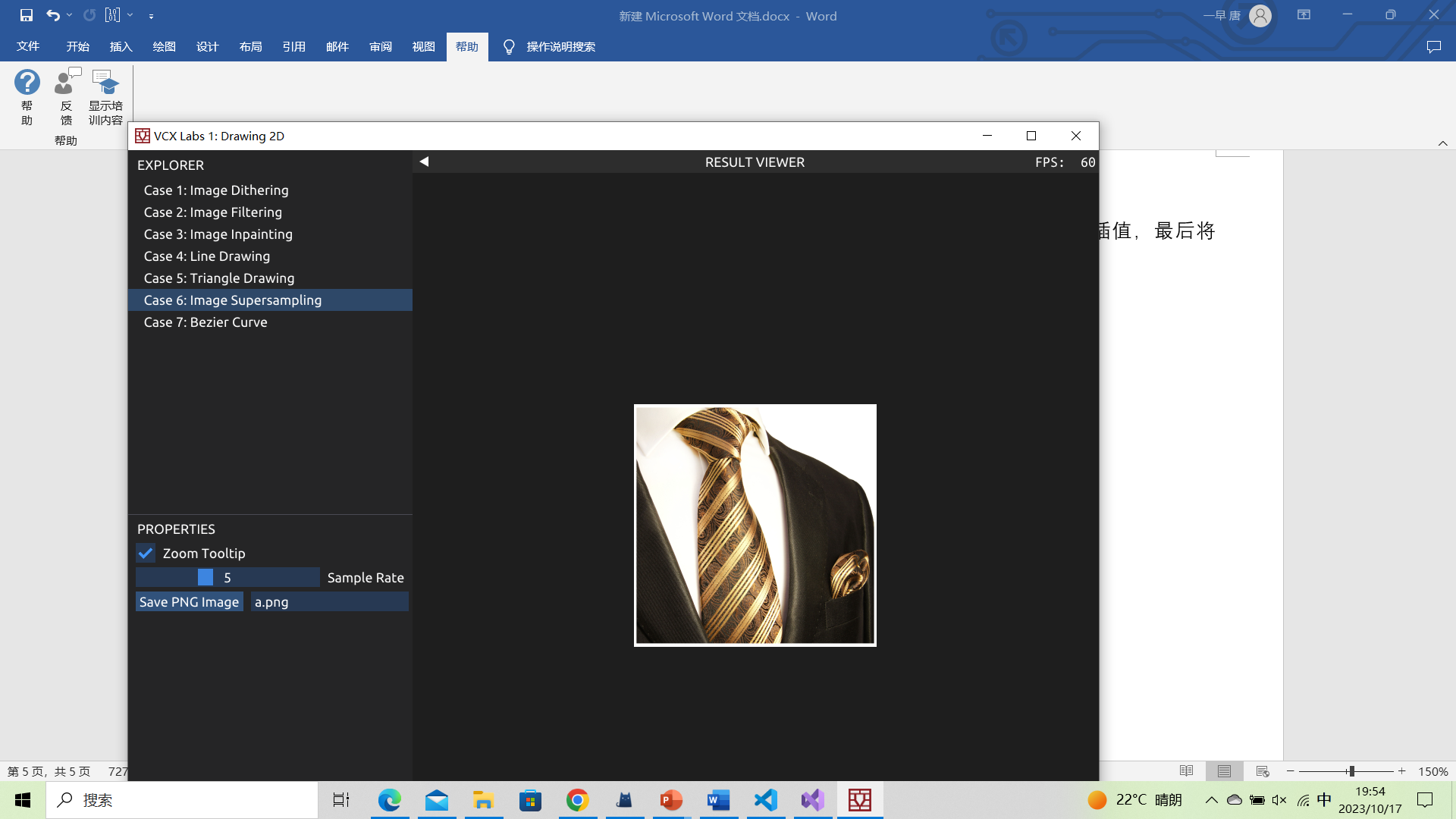
**Task 5: Triangle Drawing**

将三个点按高度大小排列，使用扫描线算法和DDA算法，找出每一次扫描的两端点并填色。最后处理一下两点高度相同的特殊情况即可。



**Task 6: Image Supersampling**

在原图上进行rate\*rate倍的超采样，对没有落在整数点的点找到相邻近的四个点进行线性插值，最后将rate\*rate个像素平均到一个像素上输出即可。下图为5倍超采样反锯齿。



**Task 7: Bezier Curve**

对曲线上每个点计算。一共4个点n=3。

