## 光栅化!

## Canonical Cube to Screen

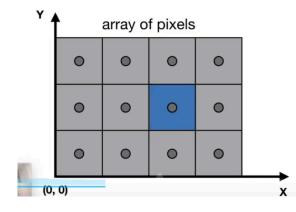
- · Defining the screen space
  - Slightly different from the "tiger book"

Pixels' indices are in the form of (x, y), where both x and y are integers

Pixels' indices are from (0, 0) to (width - 1, height - 1

Pixel (x, y) is centered at (x + 0.5, y + 0.5)

The screen covers range (0, 0) to (width, height)



这种定义方式保证了 screen covers range 是  $(0,0) \rightarrow (width, height)$ 

三角形性质很好,很方便算法处理。

抗锯齿......是必须的。

走样,混叠,其实就是采样率不够高造成的失真。

之前写教程考虑过对声信号采样率过低造成的混叠失真,没想到在这里帮助理解了。

写编曲书的时候也提前搞了傅里叶级数,呼呼。

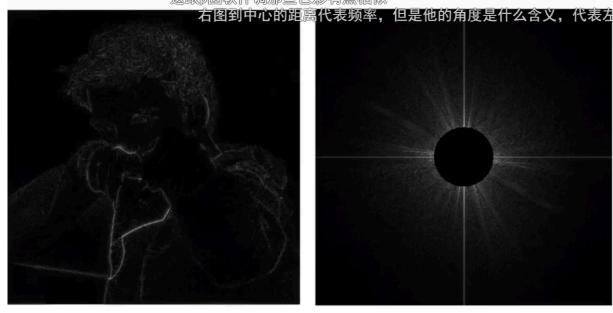


图 1 High-pass filtering

高频信息代表图像的边界,也是和之前讲的呼应上了,边界就是变化剧烈的地方嘛。



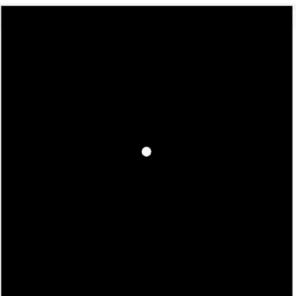


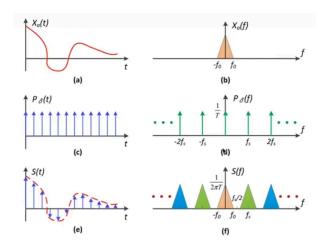
图 2 Low-pass filtering

同理只保留低频部分就会把原先的边界模糊化。

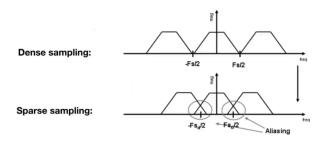
没怎么正经学过卷积,来还债楼。

时域的卷积等于频域的乘积,没想到还能 cue 到打 0I 时期学的知识点,哭了。  $O(n^2) \rightarrow O(n \log n)$ 

话说卷积定理反过来也成立。



然后就是混叠了,没想到这个概念是对频域上现象的直观描述。



增加采样率是好的......如果要增加人感官的采样率......加入终极的进化......

先做模糊,会让频谱变的很窄,就更难发生混叠,非常形象。

但把原先的三角形模糊化之后,用于描述这个三角形的函数就变得非常复杂,更别说光栅化 了。

MSAA 的做法是超采样,也即用一个机器分辨率倍数的虚拟分辨率屏幕进行采样。

FXAA 更像是机器学习/深度学习领域的一种后处理。

TAA 就是在时序上把一帧该做的 MSAA 分到多个帧去做。反正图像输出是有个缓冲池的,它应该是在缓冲区做比对、平均和替换。

DLSS 和 FXAA 有些相似。嗯哼,拼好帧。

该做作业喽。

真的懒得开虚拟机了,就用 MSYS2 工具链在 windows 上搞吧。

首先安装 opencv2

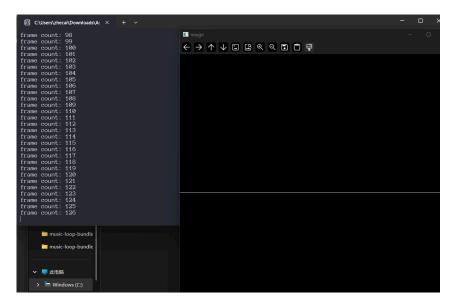
```
mingw-w64-ucrt-x86.64-openal-1.24.2-1 mingw-w64-ucrt-x86.64-openblas-0.3.29-1
mingw-w64-ucrt-x86.64-opencore-amr-0.1.6-1 mingw-w64-ucrt-x86.64-openxr-3.3.2-1
mingw-w64-ucrt-x86.64-openjpeg2-2.5.3-1 mingw-w64-ucrt-x86.64-openxr-2.5.1-1
mingw-w64-ucrt-x86.64-openjpeg2-2.5.3-1 mingw-w64-ucrt-x86.64-openxr-2.5.1-1
mingw-w64-ucrt-x86.64-pango-1.56.2-1 mingw-w64-ucrt-x86.64-protobuf-29.3-1
mingw-w64-ucrt-x86.64-python-packaging-24.2-1 mingw-w64-ucrt-x86.64-ravle-0.7.1-7
mingw-w64-ucrt-x86.64-shaderc-2025.1-1 mingw-w64-ucrt-x86.64-ravle-0.7.1-7
mingw-w64-ucrt-x86.64-shaderc-2025.1-1 mingw-w64-ucrt-x86.64-speex-1.2.1-1
mingw-w64-ucrt-x86.64-spirv-cross-1.4.309.0-1
mingw-w64-ucrt-x86.64-spirv-cross-1.4.309.0-1
mingw-w64-ucrt-x86.64-spirv-cross-1.4.309.0-1
mingw-w64-ucrt-x86.64-spirv-coll-mingw-w64-ucrt-x86.64-syt-avl-3.0.1-1
mingw-w64-ucrt-x86.64-tinyxml-2.6.2-5 mingw-w64-ucrt-x86.64-vid.stab-1.1.1-1
mingw-w64-ucrt-x86.64-vid.ca-1.3.7-4 mingw-w64-ucrt-x86.64-x265-4.1-2
mingw-w64-ucrt-x86.64-vid.ca-1.3.7-4 mingw-w64-ucrt-x86.64-x265-4.1-2
mingw-w64-ucrt-x86.64-vid.ca-1.3.7-4 mingw-w64-ucrt-x86.64-zing-3.0.5-2
mingw-w64-ucrt-x86.64-vid.ca-1.3.7-4 mingw-w64-ucrt-x86.64-zing-3.0.5-2
mingw-w64-ucrt-x86.64-vid.ca-1.3.7-6 mingw-w64-ucrt-x86.64-zing-3.2-2-any
mingw-w64-ucrt-x86.64-vid.ca-1.3.2-2-any
mingw-w64-ucrt-x86.64-vid.ca-1.1-any
4.8 mingw-w64-ucrt-x86.64-vid.ca-1.1-any
4.8 mingw-w64-ucrt-x86.64-vid.ca-1.1-any
4.8 mingw-w64-ucrt-x86.64-vid.ca-1.1-any
4.8 mi
```

图 3 1G 多,还挺大的

没想到还没在 vulkan 中绘制好三角形,已经在 games001 里要做到了......择日开启 vulkan 征途!

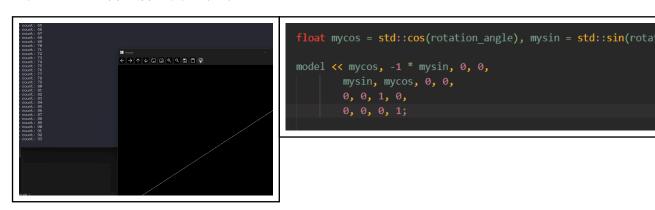
```
| Comparison | Com
```

还要安装 qt6......



好在安装完 qt6 不缺动态库了,虽然我也没找到之前缺的 dll,可能是程序动态适配了多种方案吧。接下来开始做题。

写完 model 矩阵之后可以把线旋转了。



透视投影矩阵之前没记,现搞吧。

首先是透视转正交

$$\begin{pmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & z+n & -zn \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

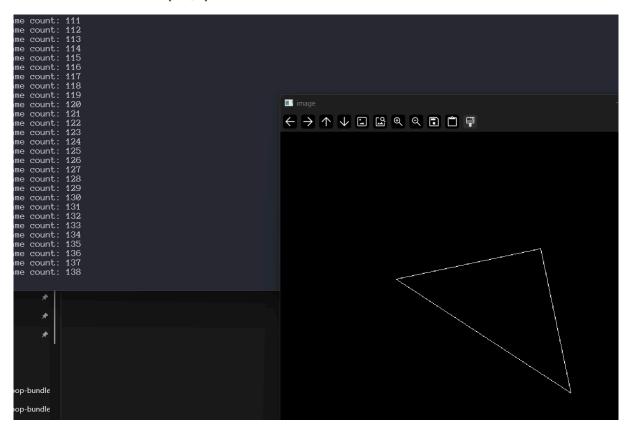
只把这个写进代码里就只有很小的一个三角形,就像是三角形离相机很远:

```
| Countrie 308 | Countrie 309 | Countrie 300 | Countrie 310 | Countrie 320 | Cou
```

## 是因为我们没有干归一化,所以要:

$$t = \|n\| \tan\left(\frac{fovY}{2}\right)$$
,  $r = t * aspect$ 

依照这个把视图转换成  $[-1,1]^3$  的立方体。



至于给的代码框架好像没啥好理解的?