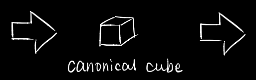
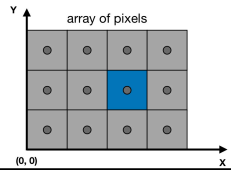
Orthographic
Perspective



多边形转化为屏幕上的像素值

- What is a screen?
  - An array of pixels
  - Size of the array: resolution
  - A typical kind of raster display
- Raster == screen in German
  - Rasterize == drawing onto the screen
- Pixel (FYI, short for "picture element")
  - For now: A pixel is a little square with uniform color
  - Color is a mixture of (red, green, blue)

- 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

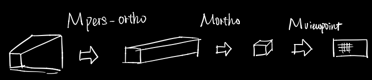
Pixels' indices are from (0, 0) to (width - 1, height - 1)

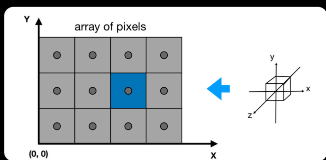
Pixel (x, y) is centered at (x + 0.5, y + 0.5)

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

Defining Screen Space

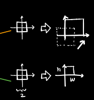
Canonical Cube to Screen





- Irrelevant to z
- Transform in xy plane : $[-1, 1]^2$ to $[0, width] \times [0, height]$
- Viewport transform matrix

$$M_{viewport} = \begin{pmatrix} \frac{width}{2} & 0 & 0 & \frac{width}{2} \\ 0 & \frac{height}{2} & 0 & \frac{height}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



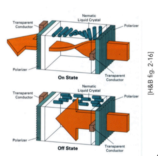
LCD

LCD (Liquid Crystal Display) Pixel

Principle: block or transmit light by twisting polarization

Illumination from backlight (e.g. fluorescent or LED)

Intermediate intensity levels by partial twist



[H&B fig. 2-16]

液晶扭曲造成光线偏振
以显示该像素内容

Rasterization

Drawing to raster display

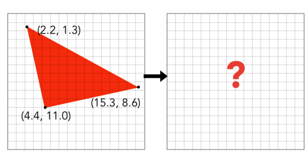
Meshes
- polygon 
- triangle 

why triangles ?

- Most basic polygon → can break up other shapes
- Unique properties
  → guaranteed to be planer
  → well-defined interior
  → well-defined method for interpolating values at vertices over triangle
  barycentric interpolation

Triangle

What Pixel Values Approximate a Triangle?



Input: position of triangle vertices projected on screen

Output: set of pixel values approximating triangle

Sampling

a simple approach 利用像素中心对屏幕空间采样

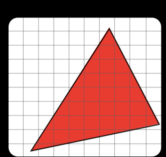
Evaluating a function at a point is sampling.

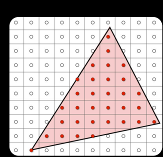
We can discretize a function by sampling.

```
for (int x = 0; x < xmax; ++x)
    output[x] = f(x);
```

Sampling is a core idea in graphics.

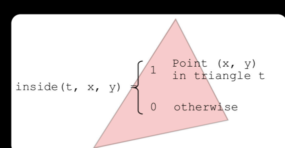
We sample time (1D), area (2D), direction (2D), volume (3D) ...



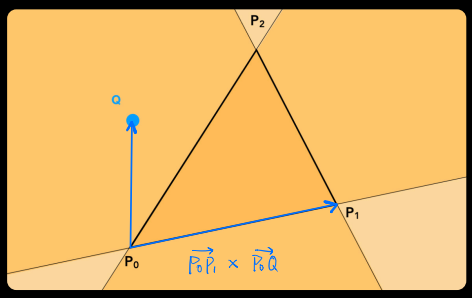


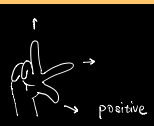
定义函数

判断像素中心是否在图形内

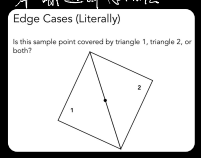


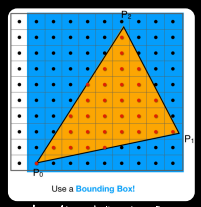

```
for (int x = 0; x < xmax; ++x)
  for (int y = 0; y < ymax; ++y)
    image[x][y] = inside(tri,
                         x + 0.5,
                         y + 0.5);
```



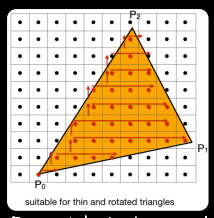


本课选择忽略





Axis Aligned Bounding Box
AABB

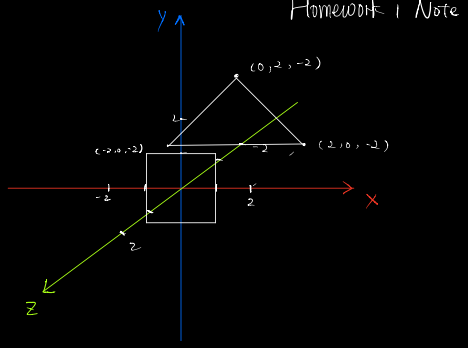


Incremental triangle traversal

RGBG

绿色密度最大 → 绿色过于九眼睛那剑

Homework 1 Note



Antialiasing 抗锯齿 <反走样>

Sampling is ubiquitous in computer graphics

Sampling artifact < error / mistakes / inaccuracies >

Artifacts due to sampling - "Aliasing"

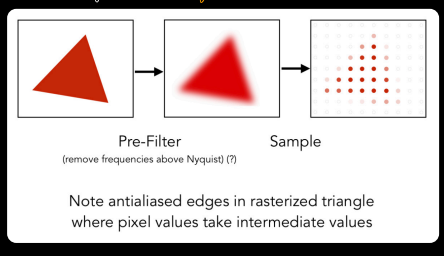
- Jaggies – sampling in space
- Moire – undersampling images
- Wagon wheel effect – sampling in time
- [Many more] …

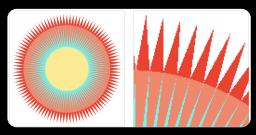
Behind the Aliasing Artifacts

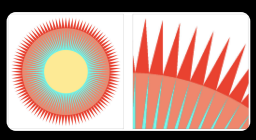
- Signals are changing too fast (high frequency), but sampled too slowly

☆

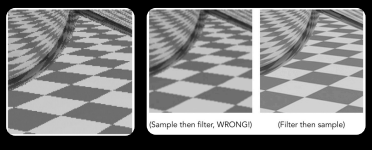
Antialiasing idea: blurring < pre-filtering > before sampling







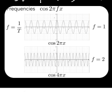
BLURRED ALIASING



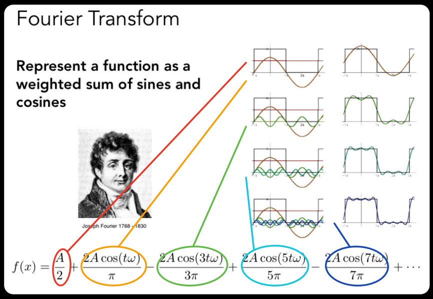
Filter + Sample

Fourier Transform 傅里叶级数展开与傅里叶变换

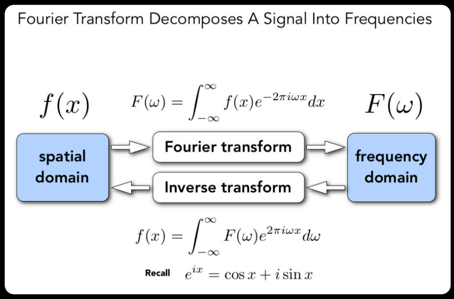
频率与频域



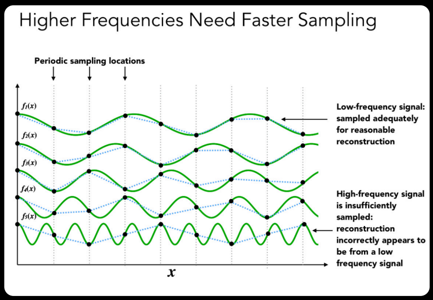
△ 周期 = $\frac{1}{频率}$

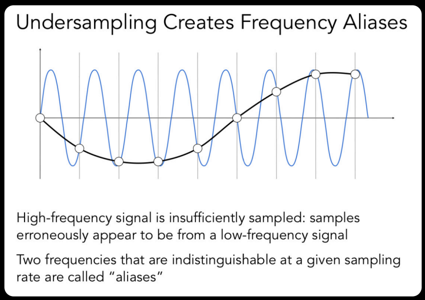


频率递增 (t)



通过存储频率以之后复原





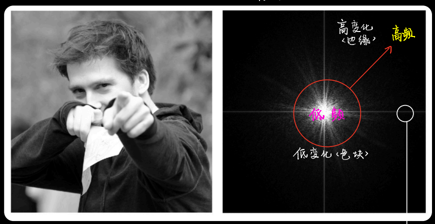
高频率函数难以精确采样 ⟶ 类似于走样

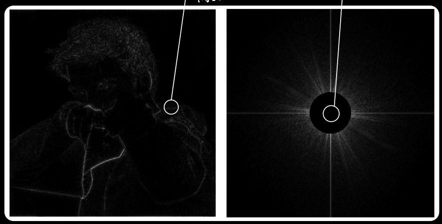
Filtering = Getting rid of certain frequency contents

亮度代表信息量

突出剧烈变化的高频→边缘

滤除低频波以突出边缘

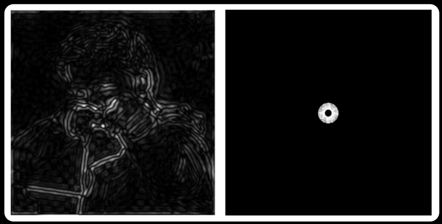




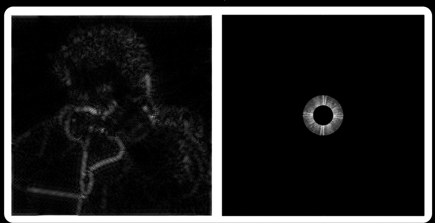
时域 ⇨ 频率

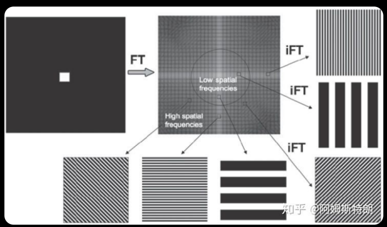
图像边界之间完全无相似 因而产生剧烈频率

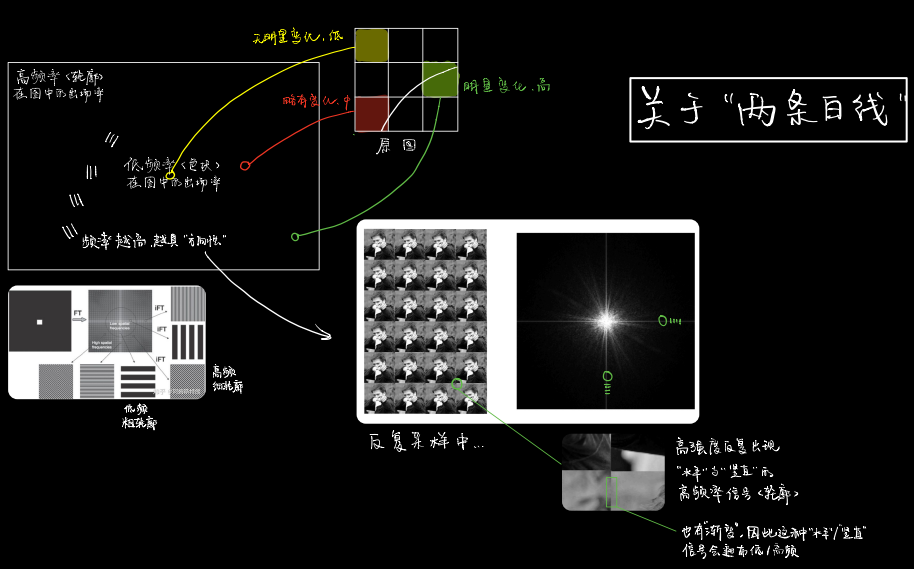
High Pass Filter



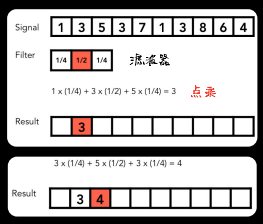
Low Pass Filter <blur>







Convolution 卷积



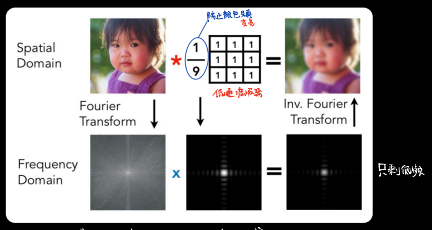
Convolution in the spatial domain is equal to multiplication in the frequency domain, and vice versa

Option 1:

- Filter by convolution in the spatial domain

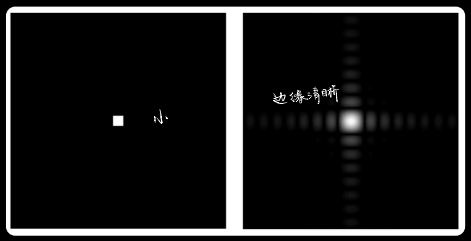
Option 2:

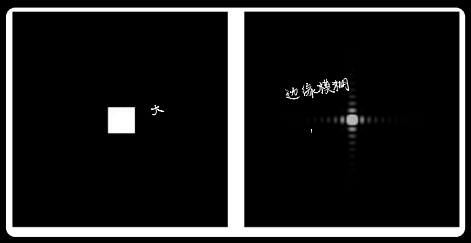
- Transform to frequency domain (Fourier transform)
- Multiply by Fourier transform of convolution kernel
- Transform back to spatial domain (inverse Fourier)



时域上的卷积 = 频域上的乘积

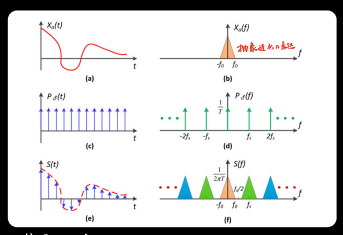
filtering = Convolution 卷积 = Averaging 模糊 平均



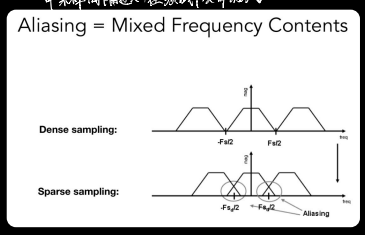


Sampling = Repeating Frequency Contents

频域中的频率与时域中的间隔为倒数关系，因此时域中采样间隔越大，在频域中反体现为↓



信号(a)乘以冲激函数得到采样(舍了点)



产生走样

How can we reduce aliasing error

反走样　先模糊　后采样

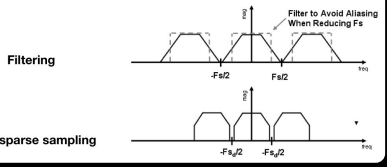
Option 1: Increase sampling rate

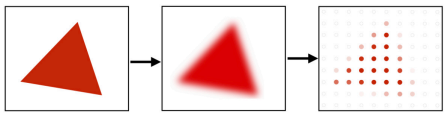
- Essentially increasing the distance between replicas in t Fourier domain
- Higher resolution displays, sensors, framebuffers…
- But: costly & may need very high resolution

Option 2: Antialiasing

- Making Fourier contents "narrower" before repeating
- i.e. Filtering out high frequencies before sampling

Antialiasing = Limiting, then repeating





Pre-Filter (remove frequencies above Nyquist)　Sample

Note antialiased edges in rasterized triangle where pixel values take intermediate values

A 1 pixel-width box filter (low pass, blurring)

Spatial Domain　Frequency Domain

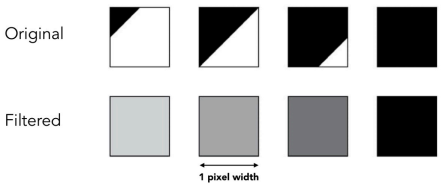
用低通滤波器进行模糊化处理

Averaging values in pixel area

Solution:

- **Convolve** f(x,y) by a 1-pixel box-blur
  - Recall: convolving = filtering = averaging
- **Then sample** at every pixel's center

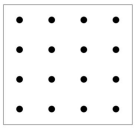
In rasterizing one triangle, the average value inside a pixel area of f(x,y) = inside(triangle,x,y) is equal to the area of the pixel covered by the triangle.



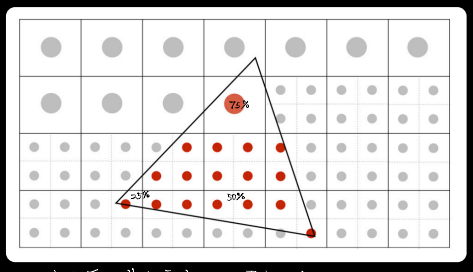
Multisample Antialiasing　MSAA

Approximate the effect of the 1-pixel box filter by sampling multiple locations within a pixel and averaging their values:

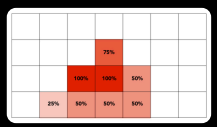
将单个像素细分
分别采样
并计算出像素值。



4x4 supersampling



① 分割
② 计算
③ 输出<百分比>



现实中采用其它更有效的图案分割

Fast Approximate AA　FXAA

图像后期处理 → 处理已有锯齿的图

△ 寻找有锯齿的边界并计算

Temporal AA　TAA

与时间相关 /'tem'porel/

处理静态时，观测同一像素不同时间的值，并将相近值近似至下一帧

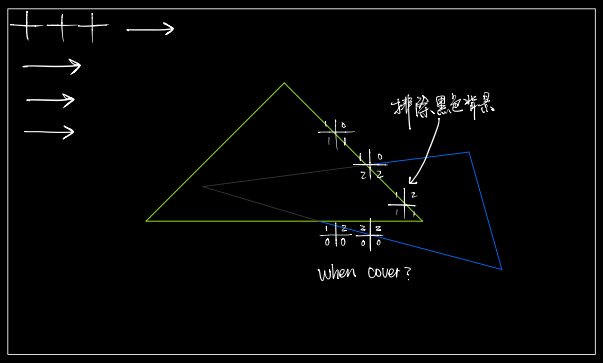
Super resolution / sampling　超分辨率

from low resolution to high resolution 小图放大

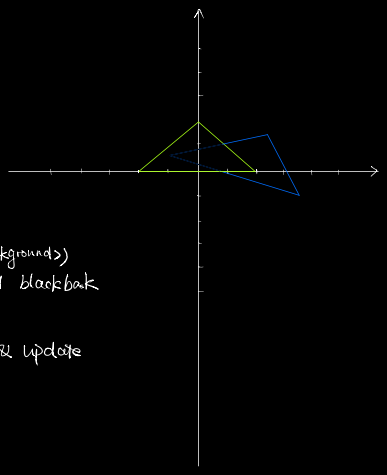
Still "not enough sampling" problem

DLSS Deep learning super-sampling 猜测→补细节

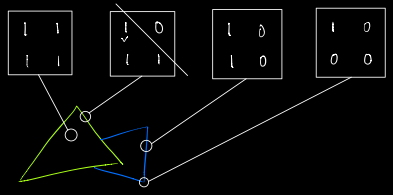
HW2 : Super-sampling



Z-buffering



if in-side triangle
  if (buffering = -1 (background))
    update & discard blackbak
  else
    mix two color & update
  end.



if all-in triangle
  cover
if mixed with black || mixed with other color
  Z-buffer higher
    replace former color
  else
    mix with other color