第4讲：光栅化

中国海洋大学 周佳然

上次课程内容

- 二维 & 三维变换
  - 二维组合变换
  - 由二维变换推广到三维变换
- 观测变换
  - 视图变换
  - 投影变换（正交投影、透视投影）
  - 视口变换

本次课程内容

- 什么是采样？
- 什么是走样？
- 反走样
- 遮挡/可见性

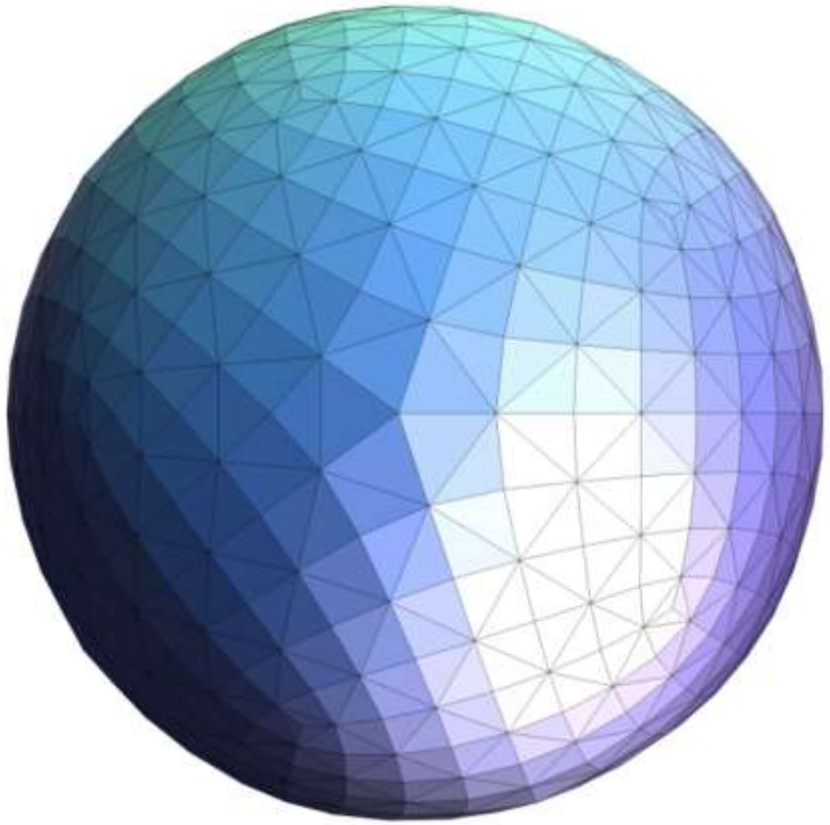
光栅化

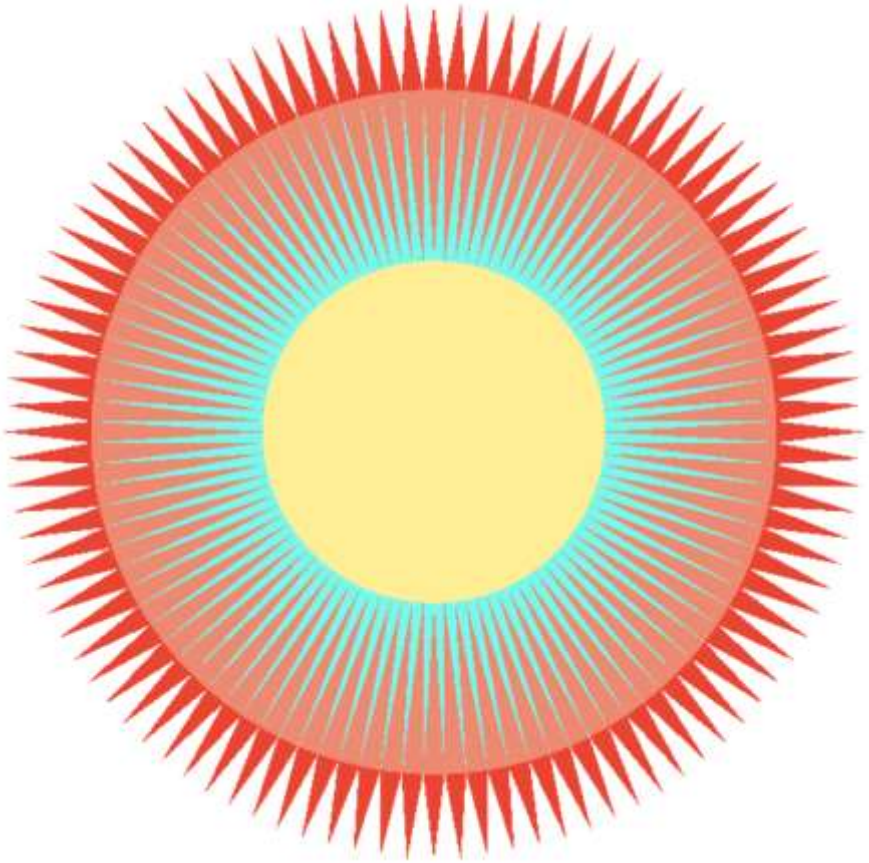
- 光栅（Raster）=德语中的屏幕
  - 光栅化(Rasterize)：画在屏幕上
- 像素（Pixel:picture element）
  - 我们这里的像素指统一颜色的小正方形
  - 颜色可以用RGB形式表示

多边形网格（Polygon Meshes）

- 光栅化：多边形网格 -> 像素

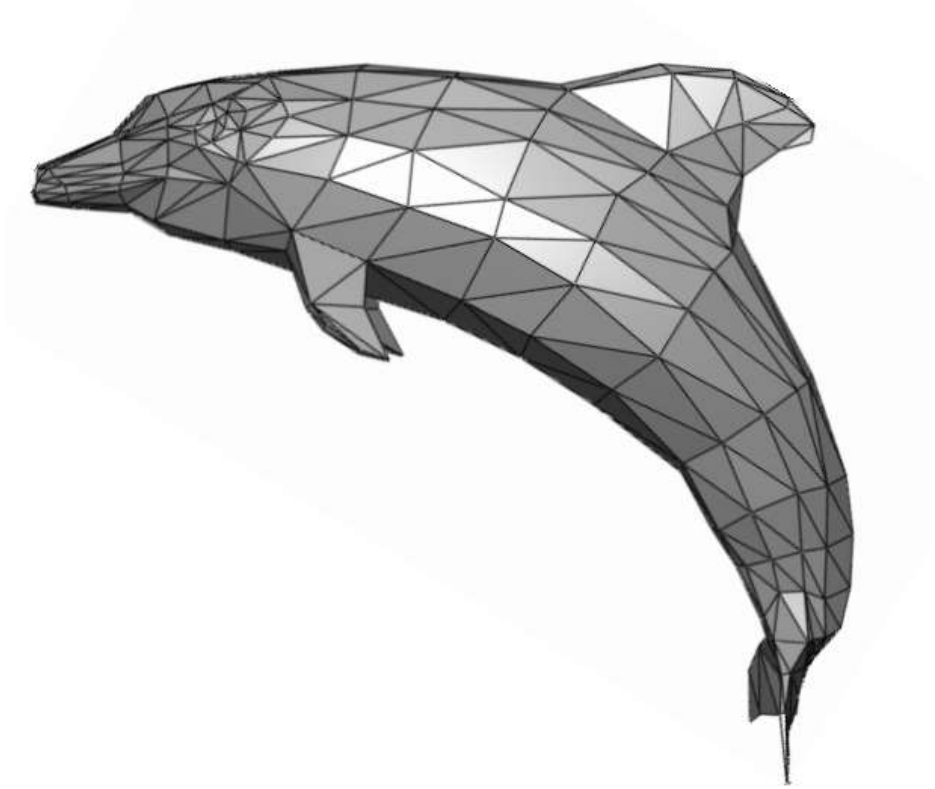
三角网格（Triangle Meshes）



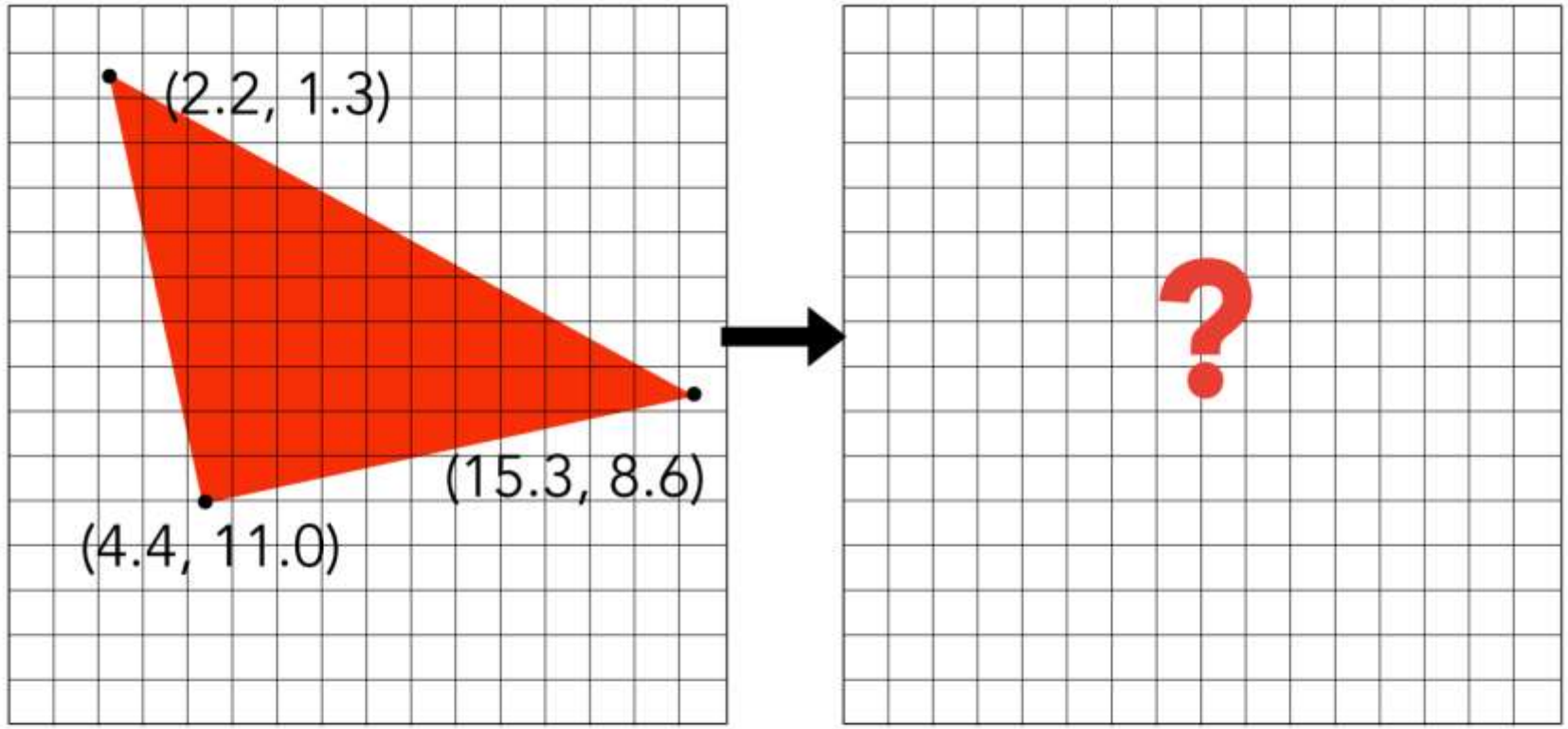


三角网格

- 三角网格是最基本的形状元素
  - ➢ 最简单的多边形
  - ➢ 其它多边形可以拆分为三角形
  - ➢ 独特的性质：
    - ✓ 三角形一定是平面的
    - ✓ 内外定义清晰
    - ✓ 对内部的任意点方便做插值（重心坐标）



如何利用像素来近似一个三角形？



Input: position of triangle vertices projected on screen

Output: set of pixel values approximating triangle

采样（sampling）

- 最简单的方法来做光栅化：采样

- 什么是采样？

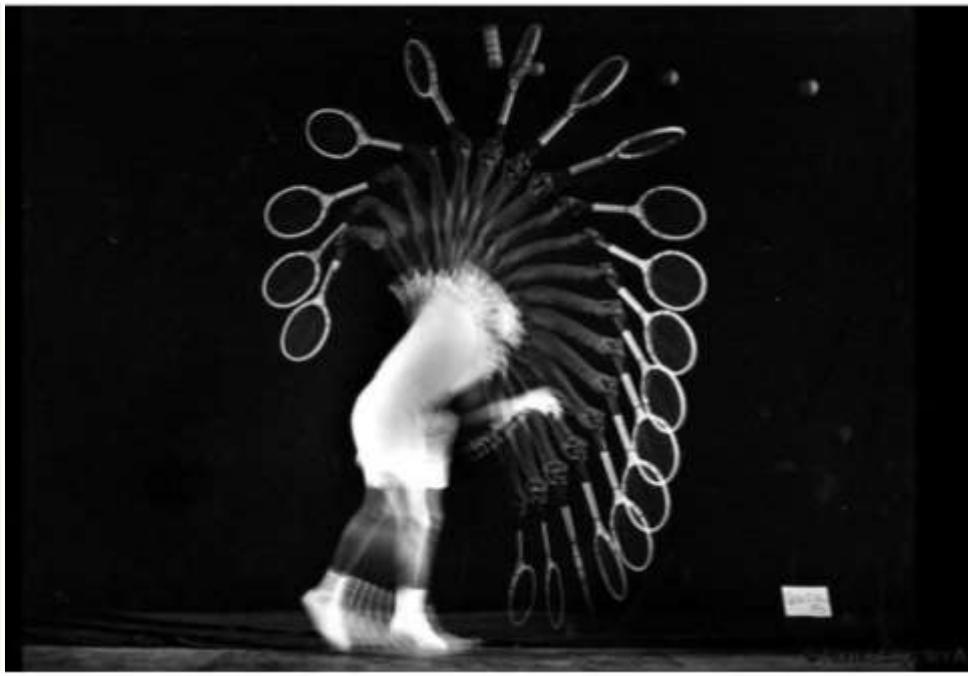
➢ 给定一个连续的函数，在某点计算函数的值（离散化一个函数的过程）

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

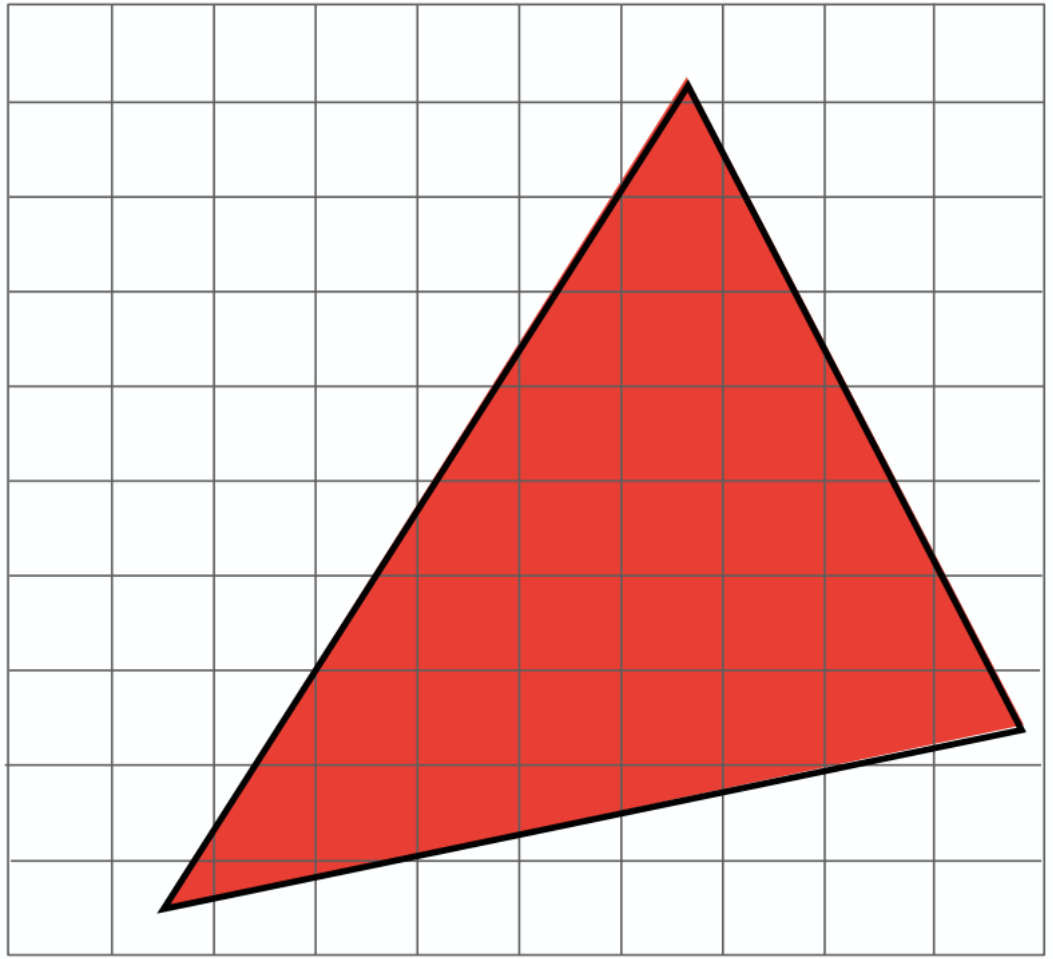
采样

- 采样是图形学中常用的方法

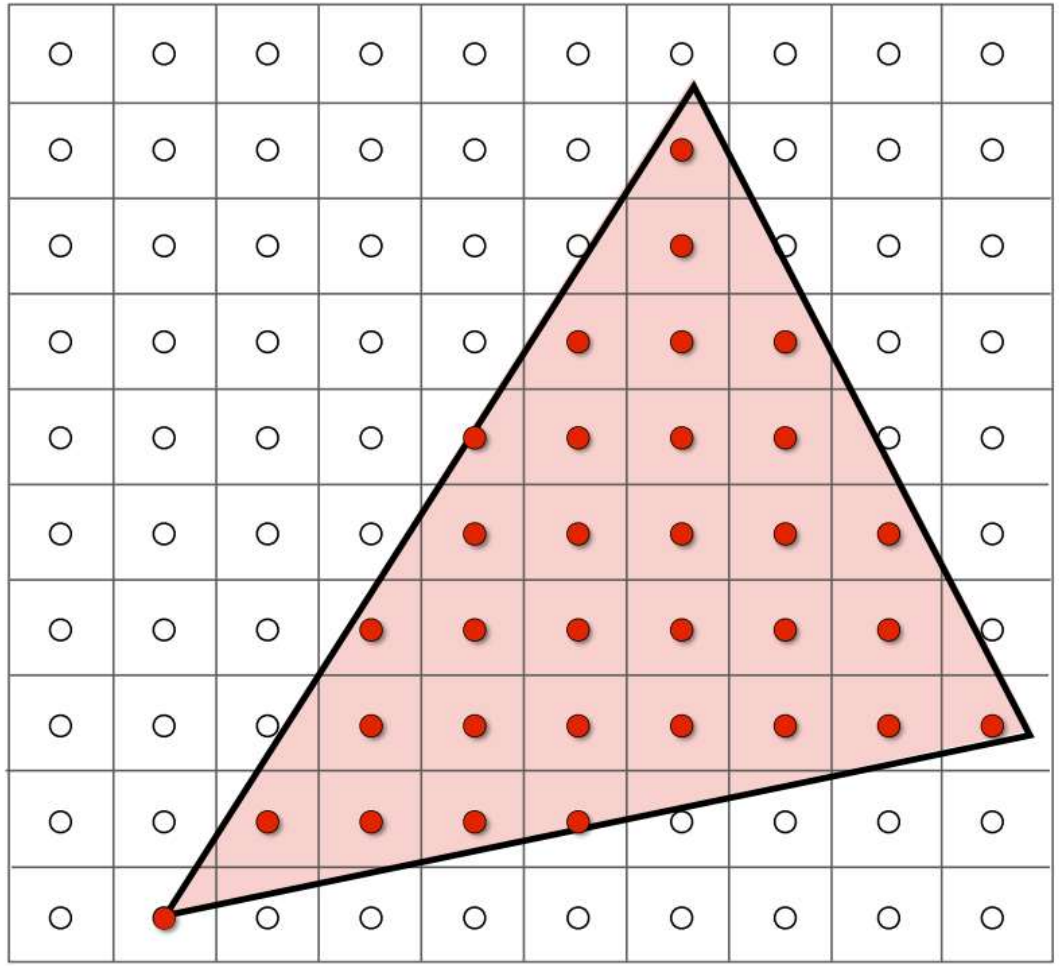
➢ 对时间（一维）、面积（二维）、方向（二维）、体积（三维）采样



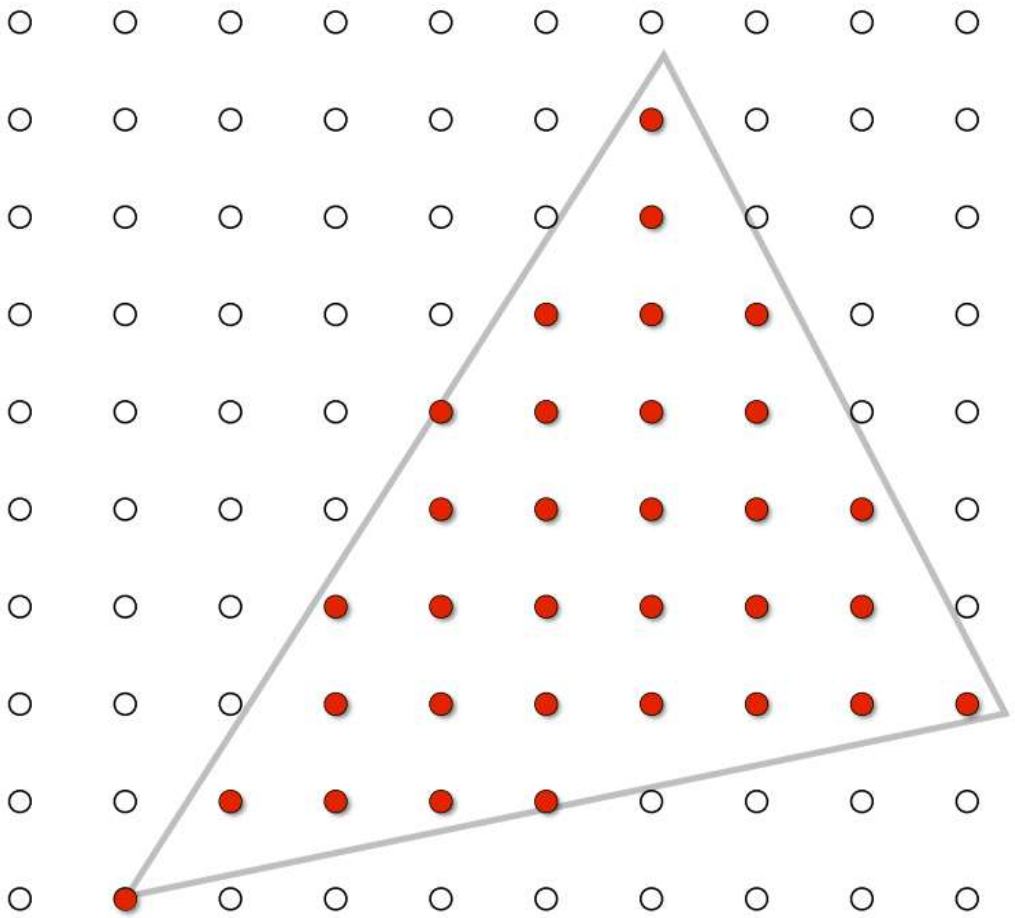
光栅化：二维采样



采样判断每一个像素中心是否在三角形内



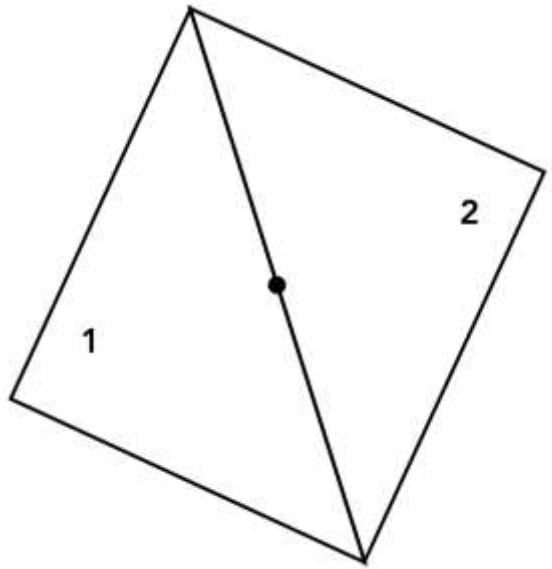
采样判断每一个像素中心是否在三角形内



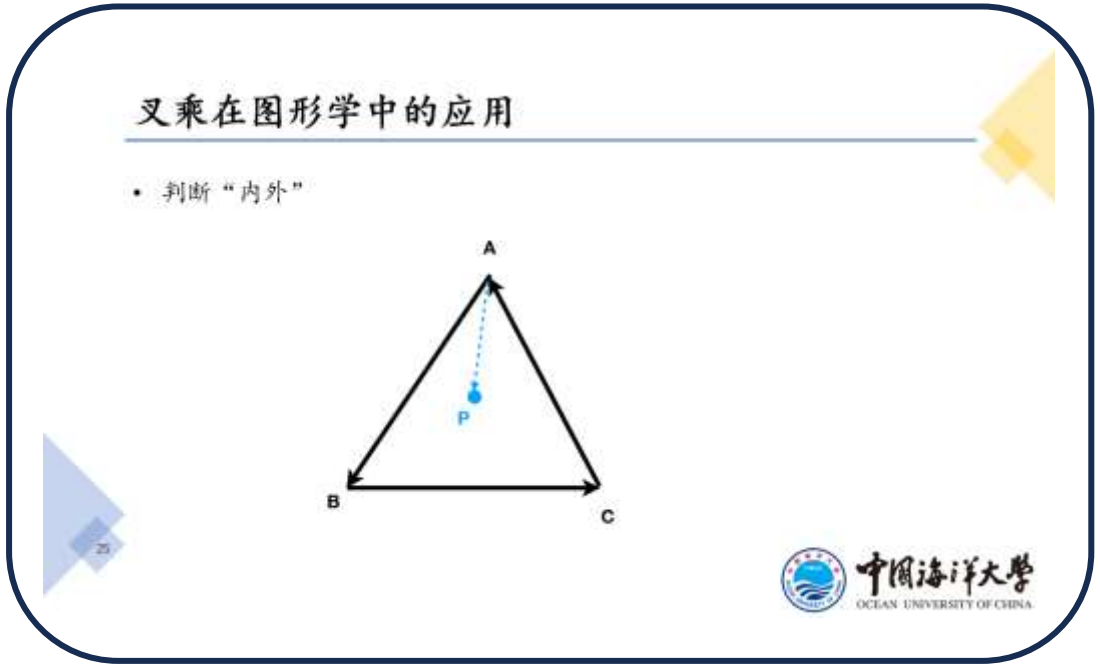
定义二值函数inside(tri,x,y)

$$inside(t,x,y)=\begin{cases}1 & point(x,y)\ in\ triangle\ t\\ 0 & otherwise\end{cases}$$

- x,y不需要是整数
- 边界情况自行定义

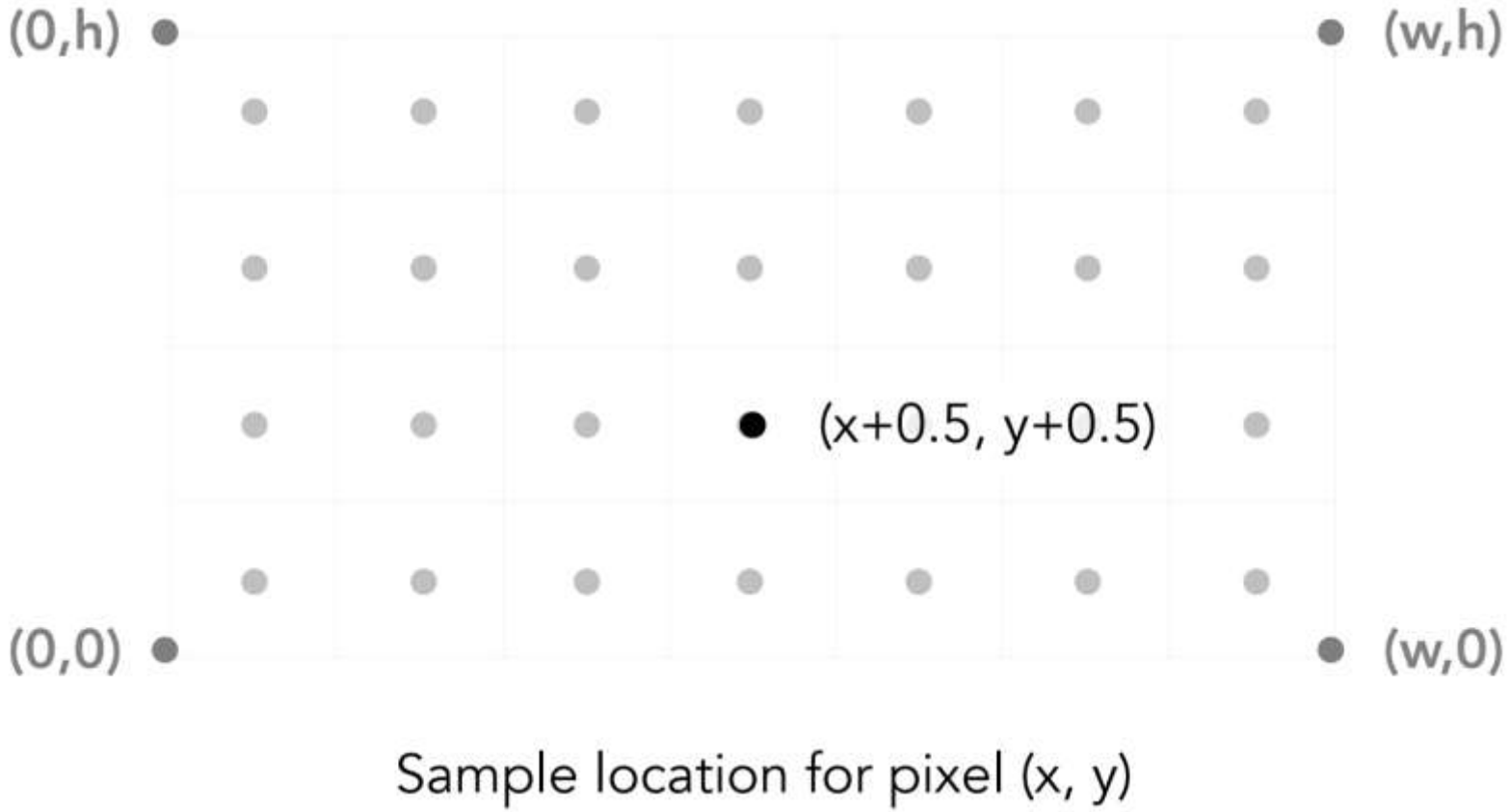


求解二值函数inside(tri,x,y)



$$inside(t,x,y)=\begin{cases}1 & point(x,y)\ in\ triangle\ t\\ 0 & otherwise\end{cases}$$

像素中心

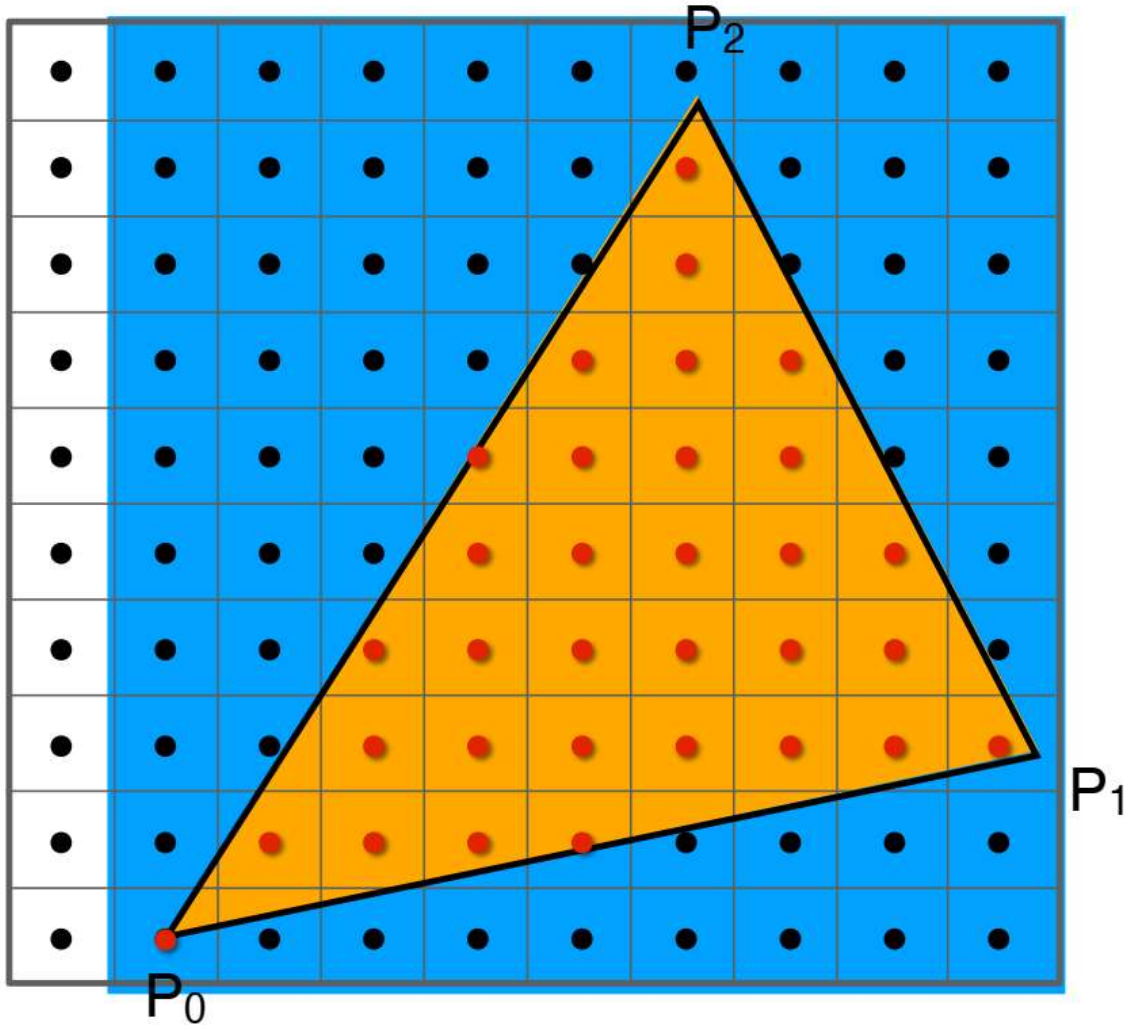


Sample location for pixel (x, y)

采样判断每一个像素中心是否在三角形内

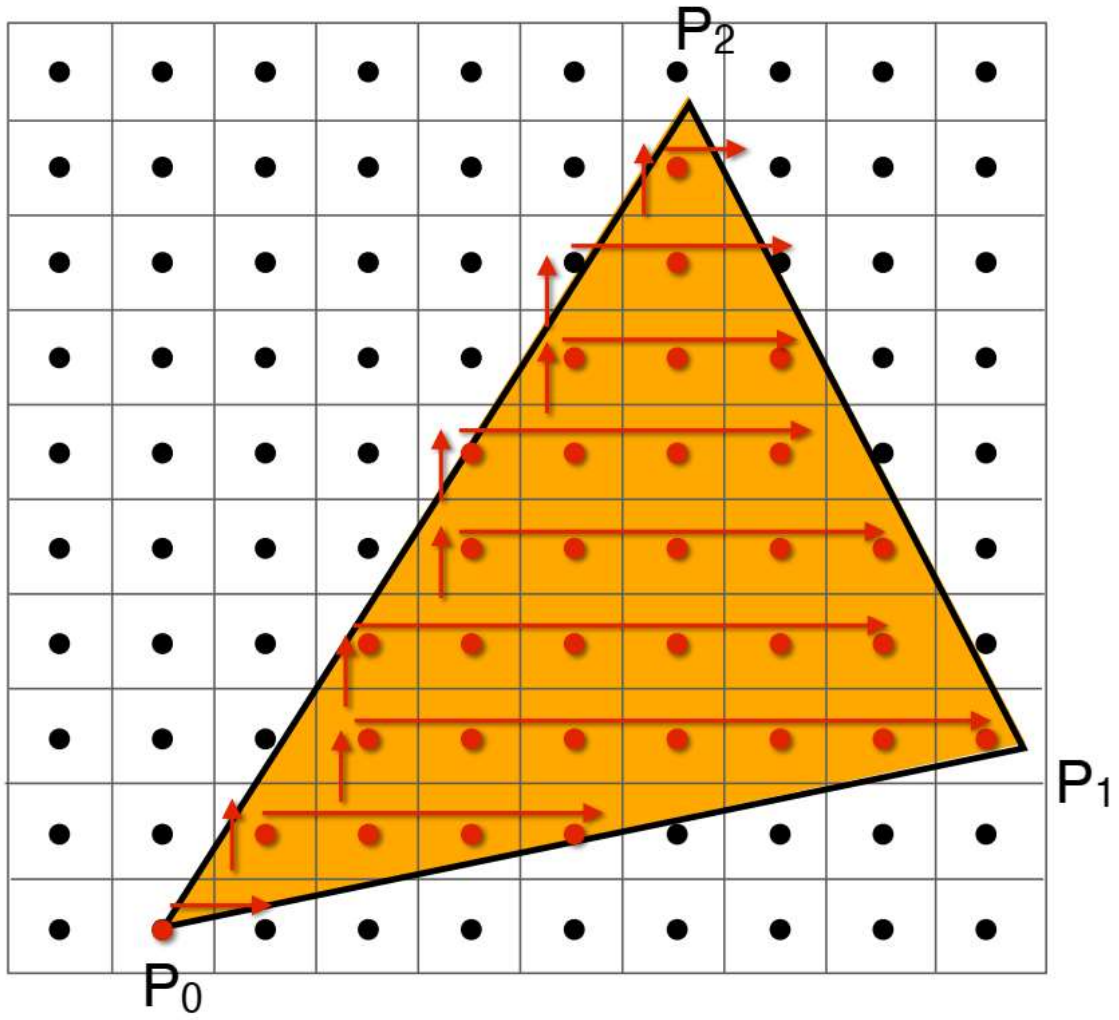
```
for (int x = 0; x < xmax; ++x)
  for (int y = 0; y < ymax; ++y)
    image[x][y] = inside(tri,
                         x + 0.5,
                         y + 0.5);
```

采样判断每一个像素中心是否在三角形内？



Use a **Bounding Box!**

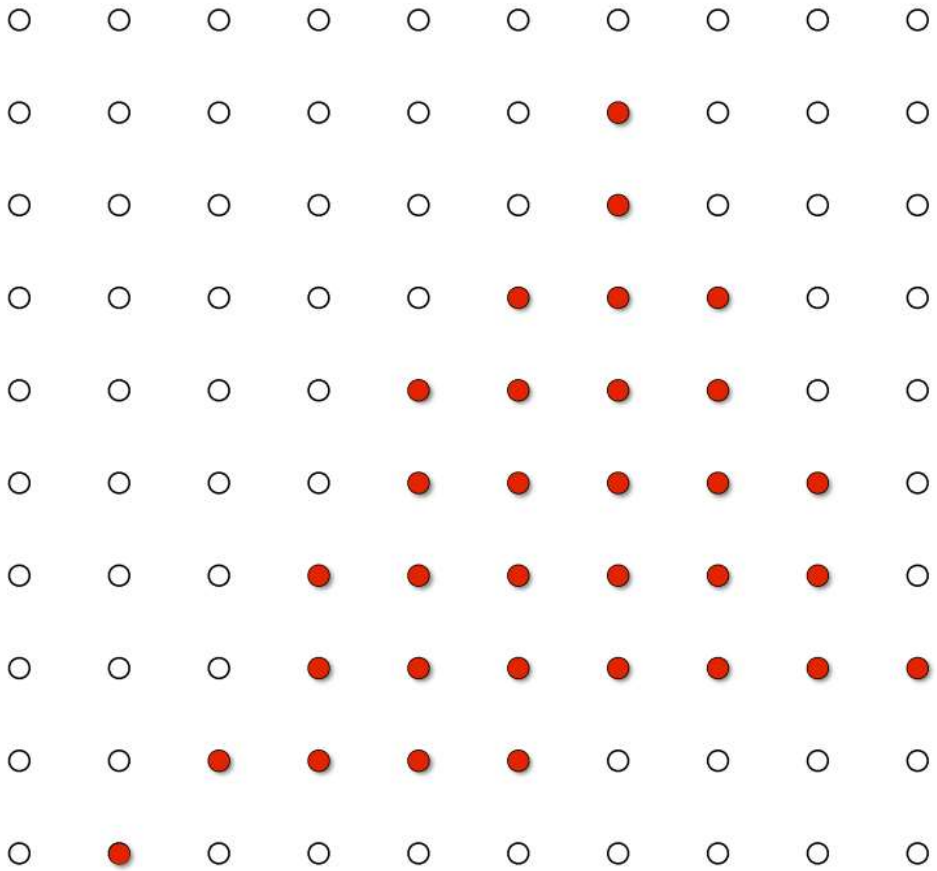
采样判断每一个像素中心是否在三角形内？



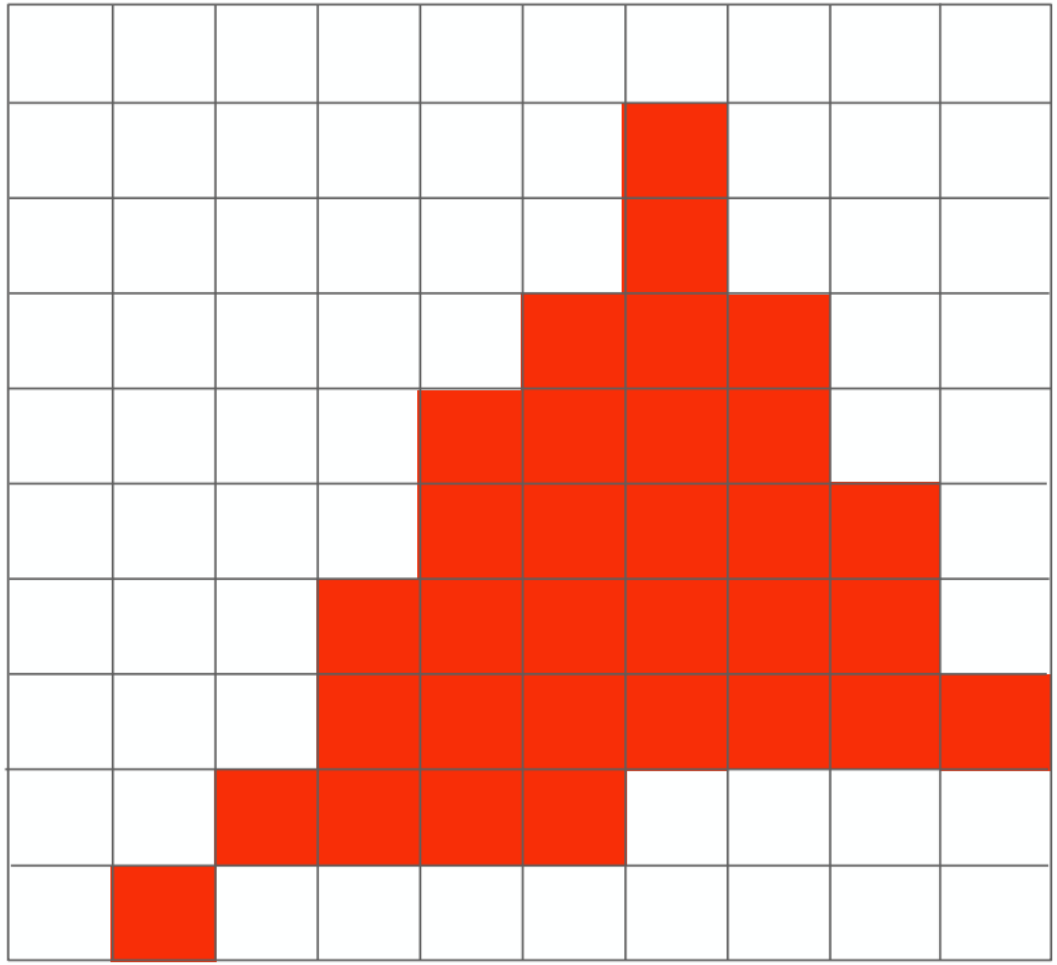
suitable for thin and rotated triangles

Q&A

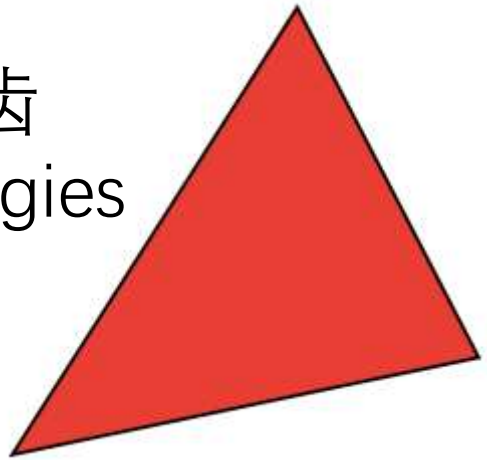
采样结果



光栅化结果

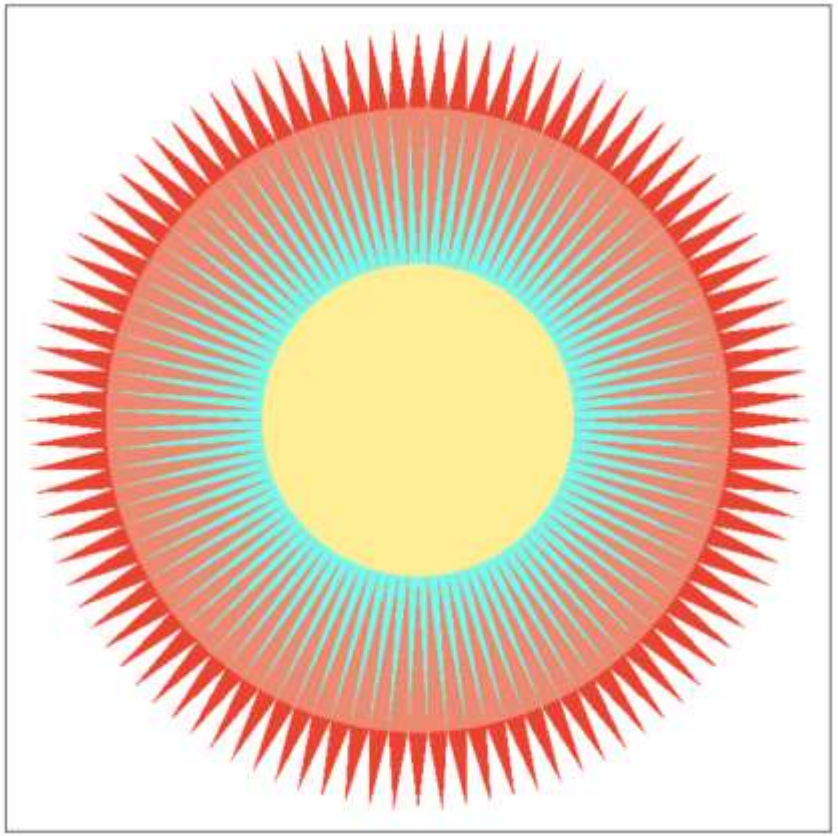


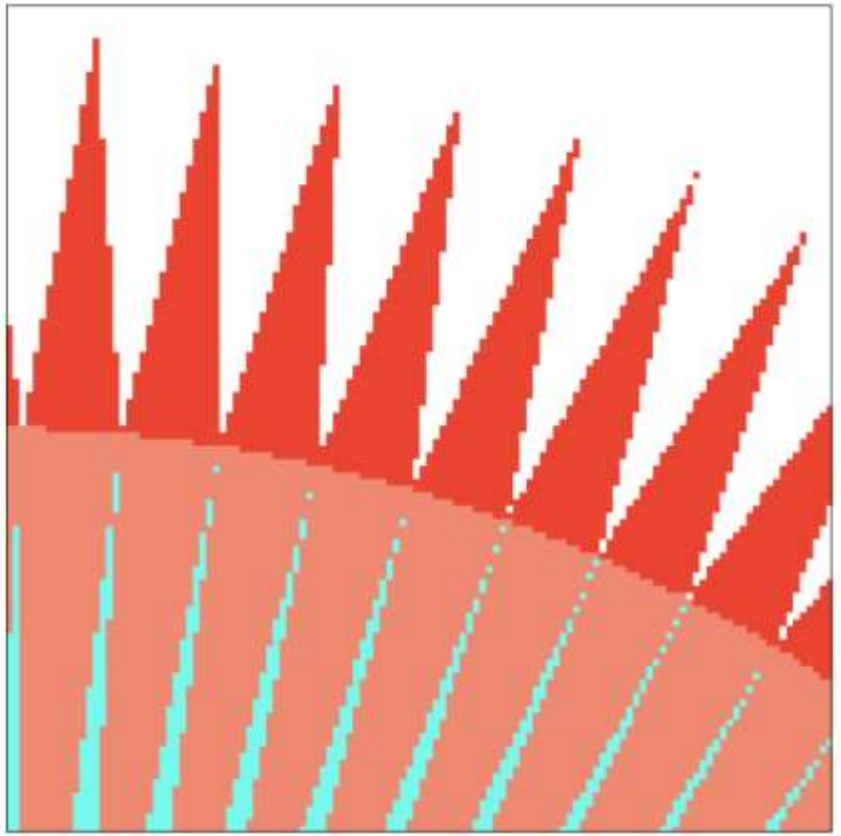
锯齿
Jaggies



图形学中常见的采样伪迹（Artifacts）

- 锯齿





图形学中常见的采样伪迹

- 摩尔纹

Skip odd rows and columns

lystit.com

图形学中常见的采样伪迹

- 车轮错觉

https://www.youtube.com/watch?v=QOwzkND_ooU

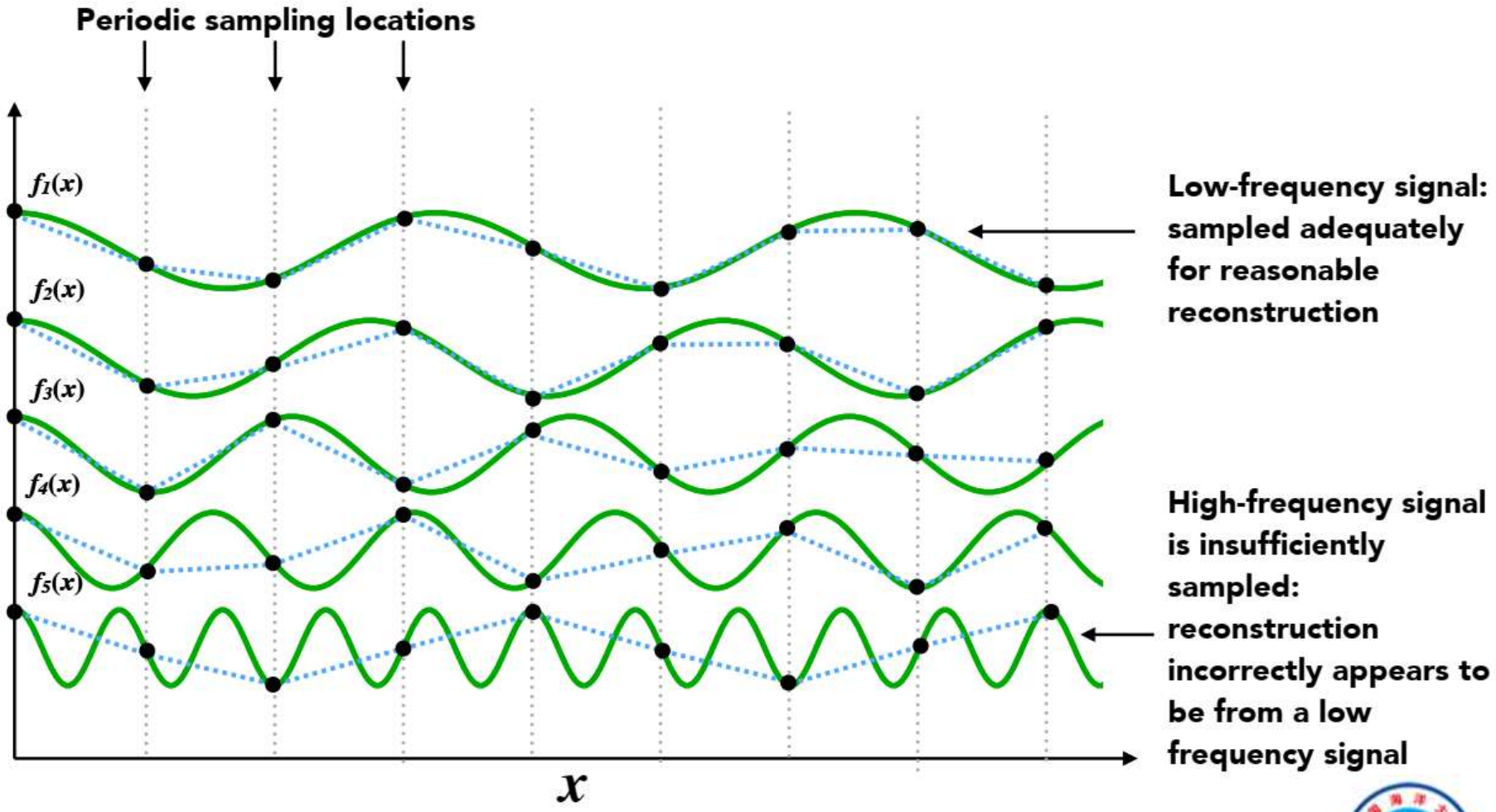
图形学中常见的采样伪迹

- 采样伪迹：走样（Aliasing）
  - 锯齿：空间采样
  - 摩尔纹：欠采样的图像
  - 车轮错觉：时间采样

Q：出现走样的原因是什么？

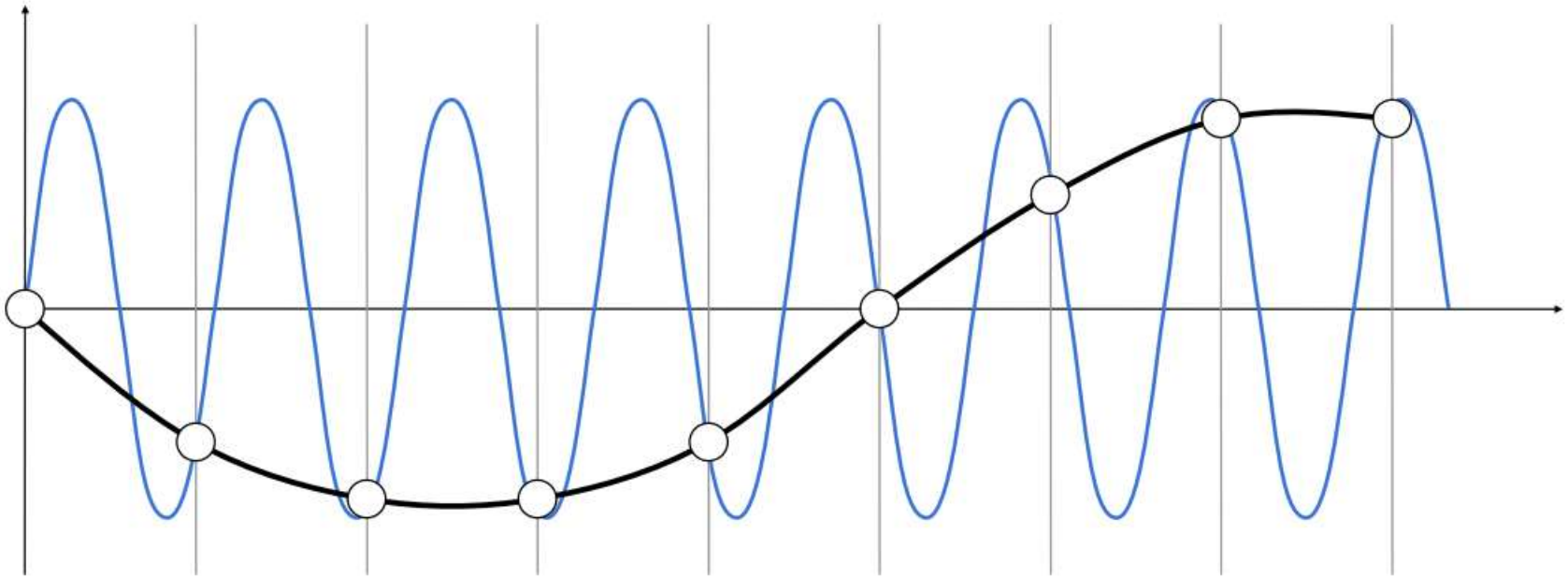
- 信号变换的太快（高频），而采样频率太慢！

越高频的信号需要越高频的采样



走样

- 错误识别信号频率



如何改善走样问题？

- 提高采样率
  - 代价问题
  - 采样率需要多高才能改善？

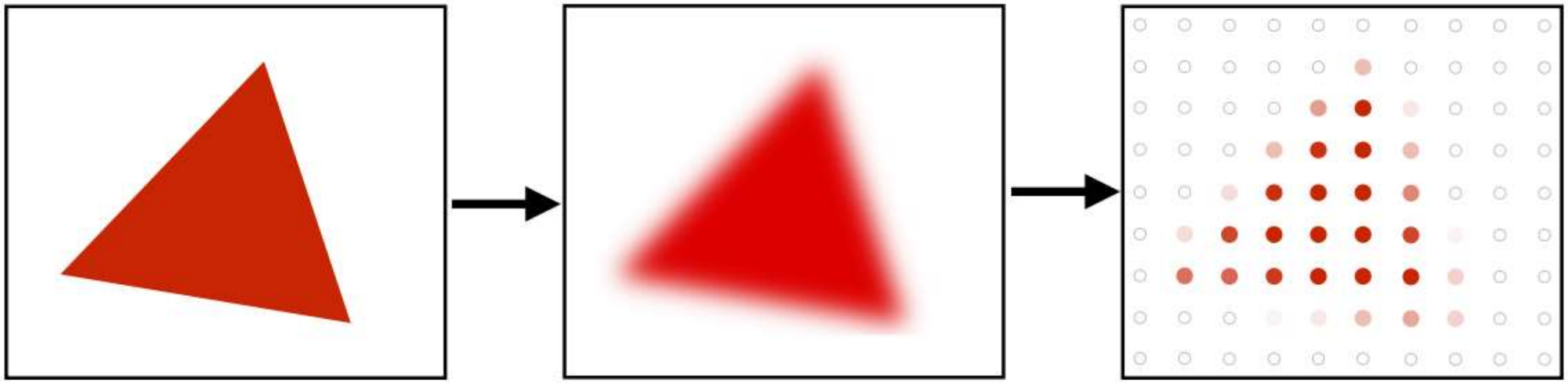
- 反走样（Antialiasing）
  - 在采样前先过滤掉高频信号*

反走样

- 一般的采样过程：光栅化三角形出现锯齿的地方，像素值为纯红或纯白

反走样

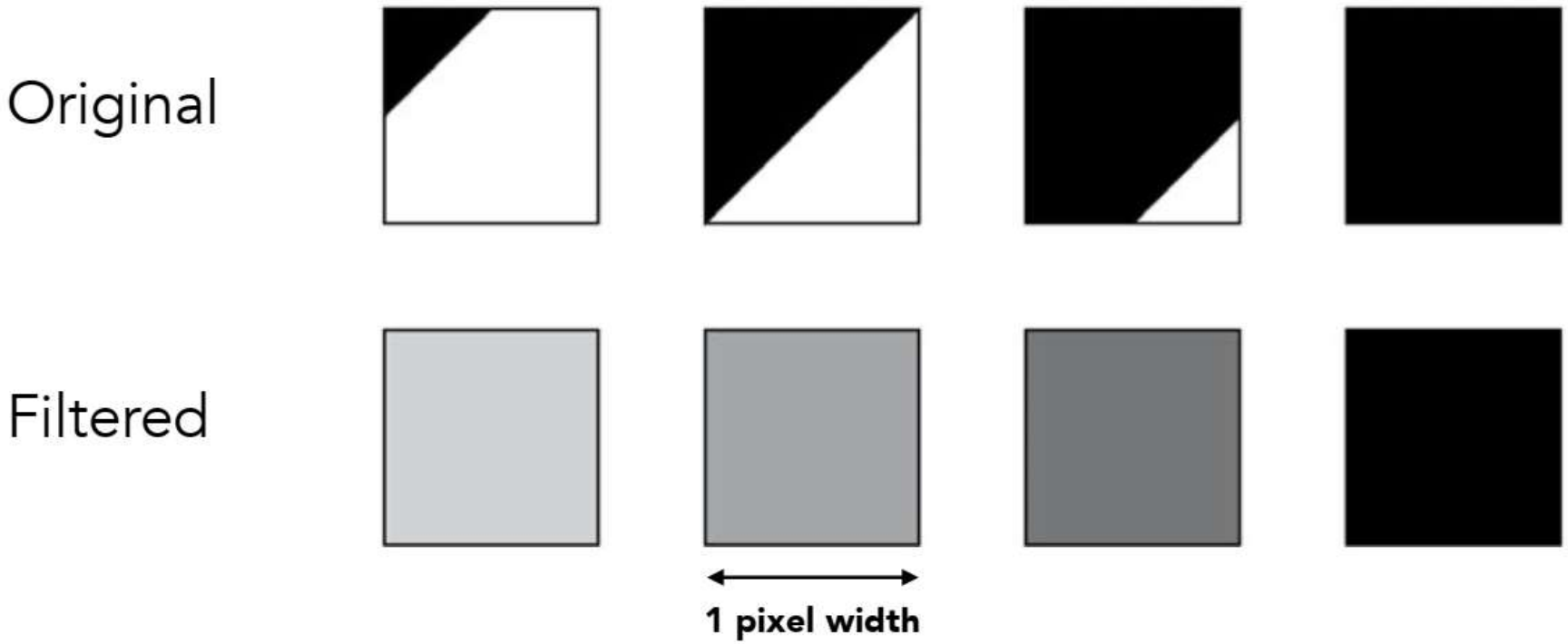
- 反走样过程：光栅化三角形反走样的边界，像素值为中间值



平均像素值

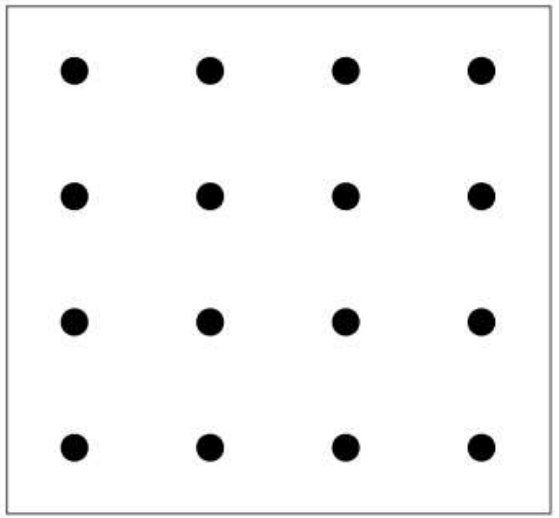
- 光栅化一个三角形时：平均像素值通过三角形覆盖像素面积来计算

Q：覆盖面积如何计算？



超级采样（Supersampling）

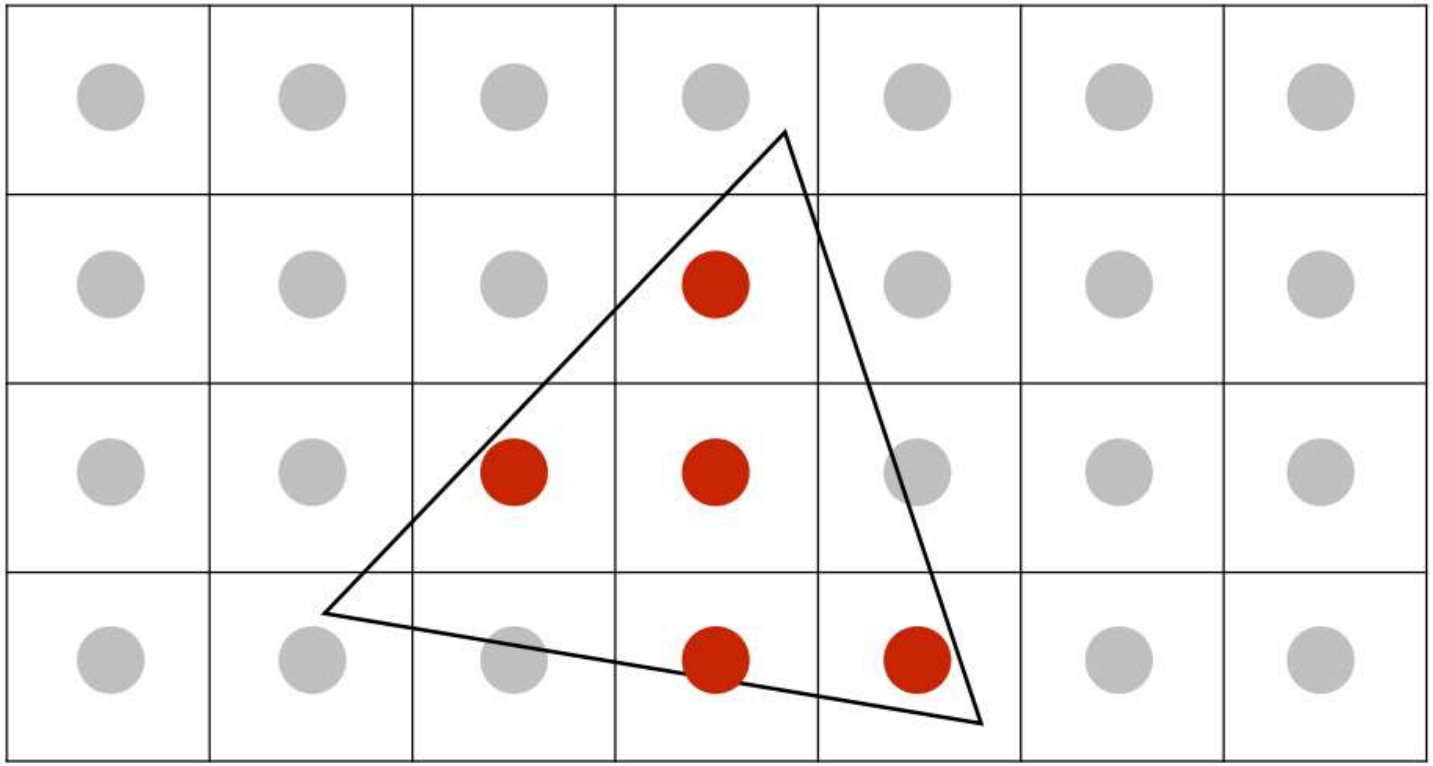
- 通过在一个像素中多次采样，计算他们的平均值作为平均像素值



4x4 supersampling

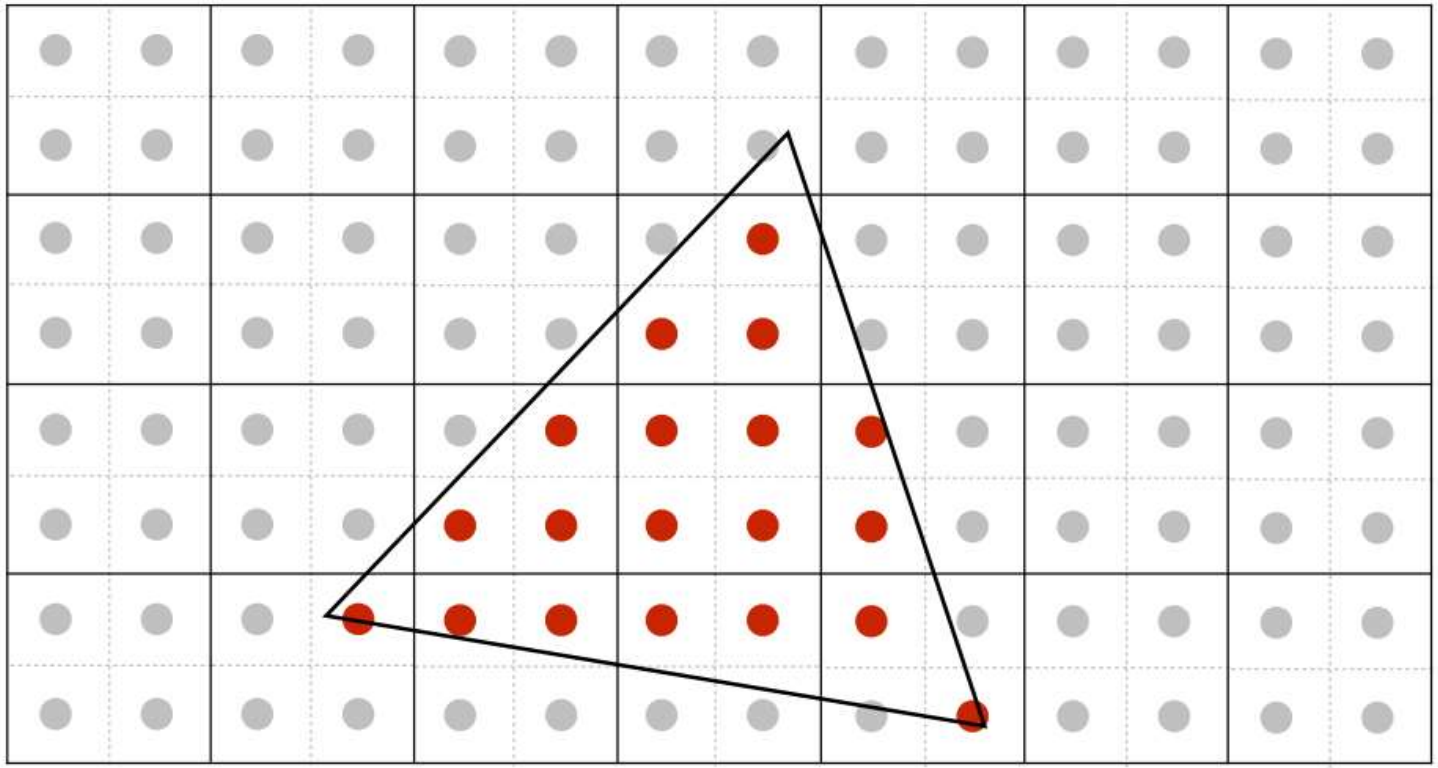
点采样

- 一个像素中一个采样点



超级采样

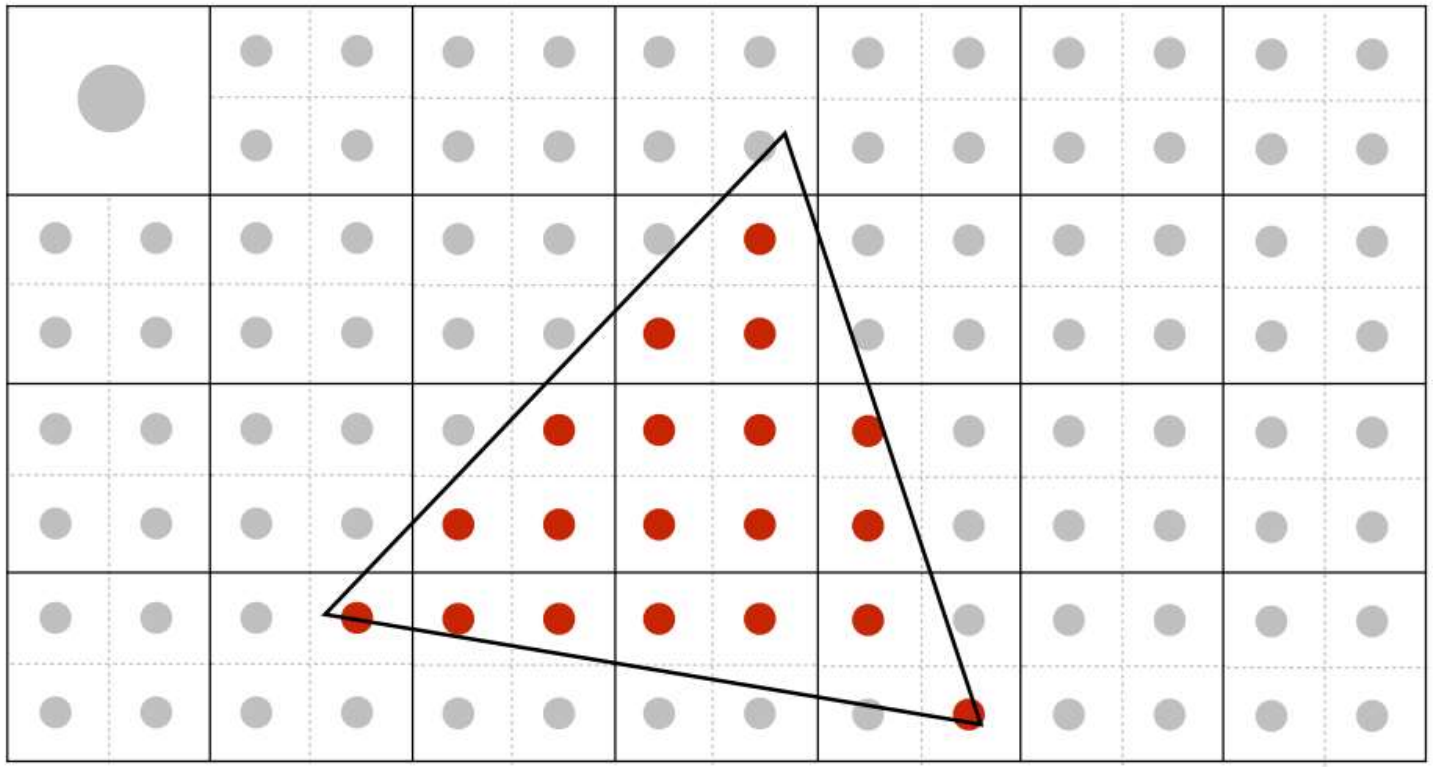
1. 一个像素中N*N个采样点



2x2 supersampling

超级采样

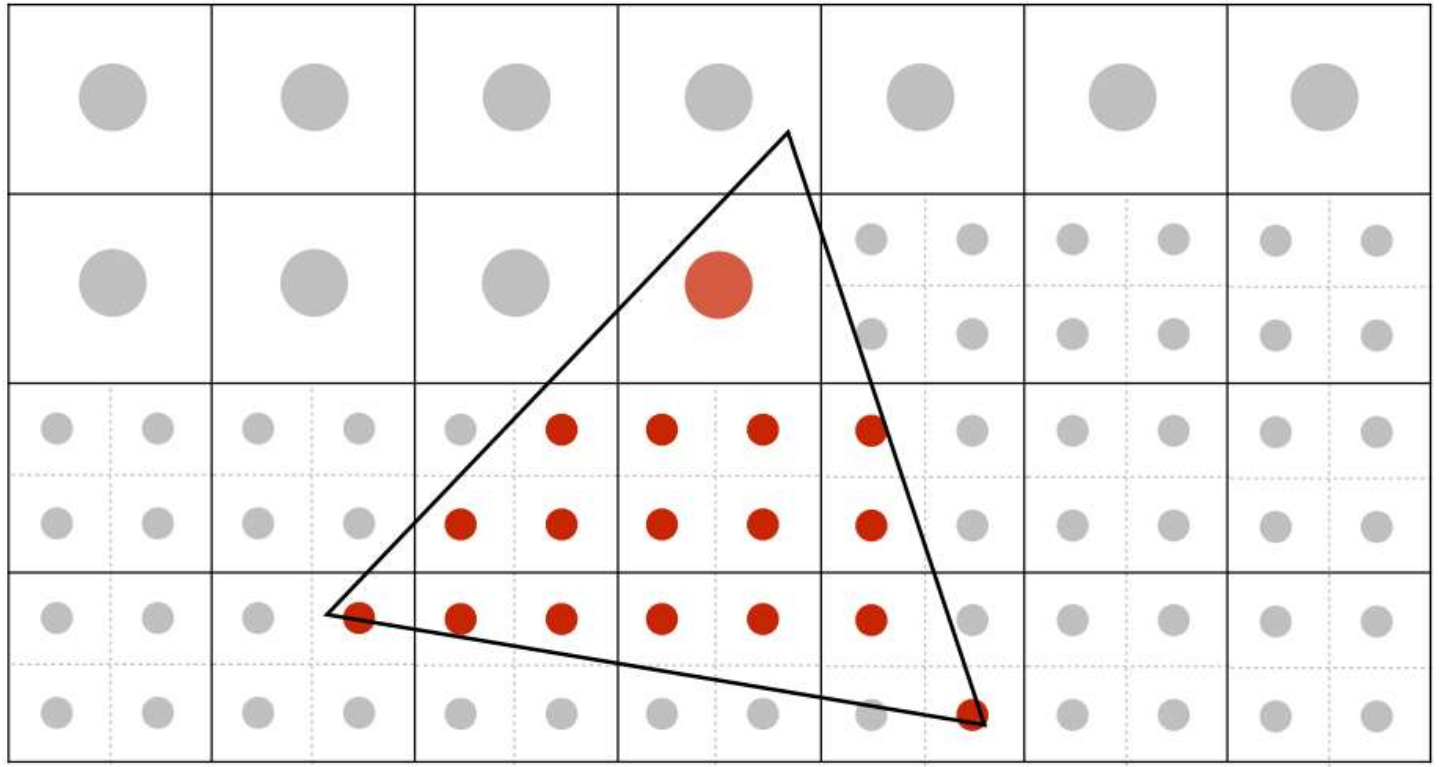
2. 计算每个像素中N*N个采样点的平均值



Averaging down

超级采样

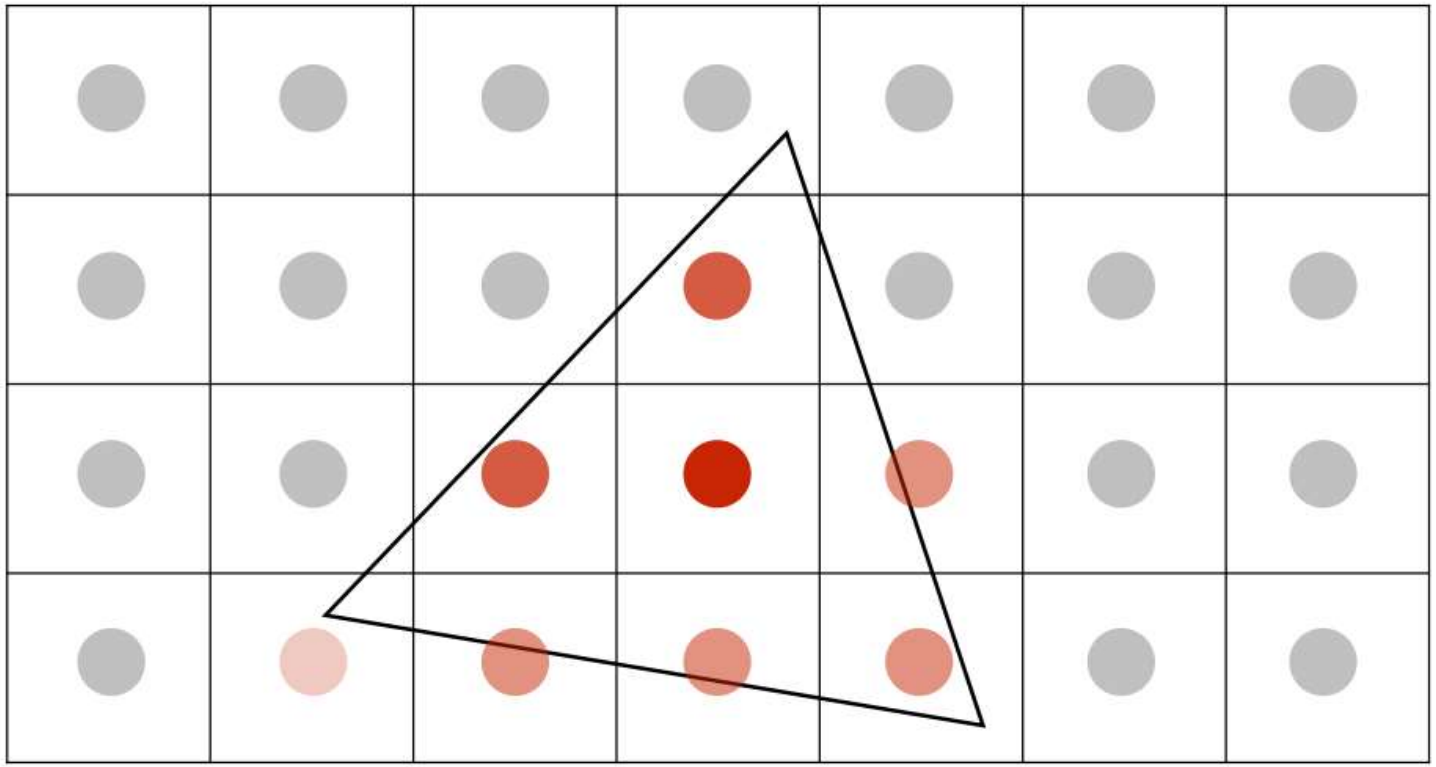
2. 计算每个像素中N*N个采样点的平均值



Averaging down

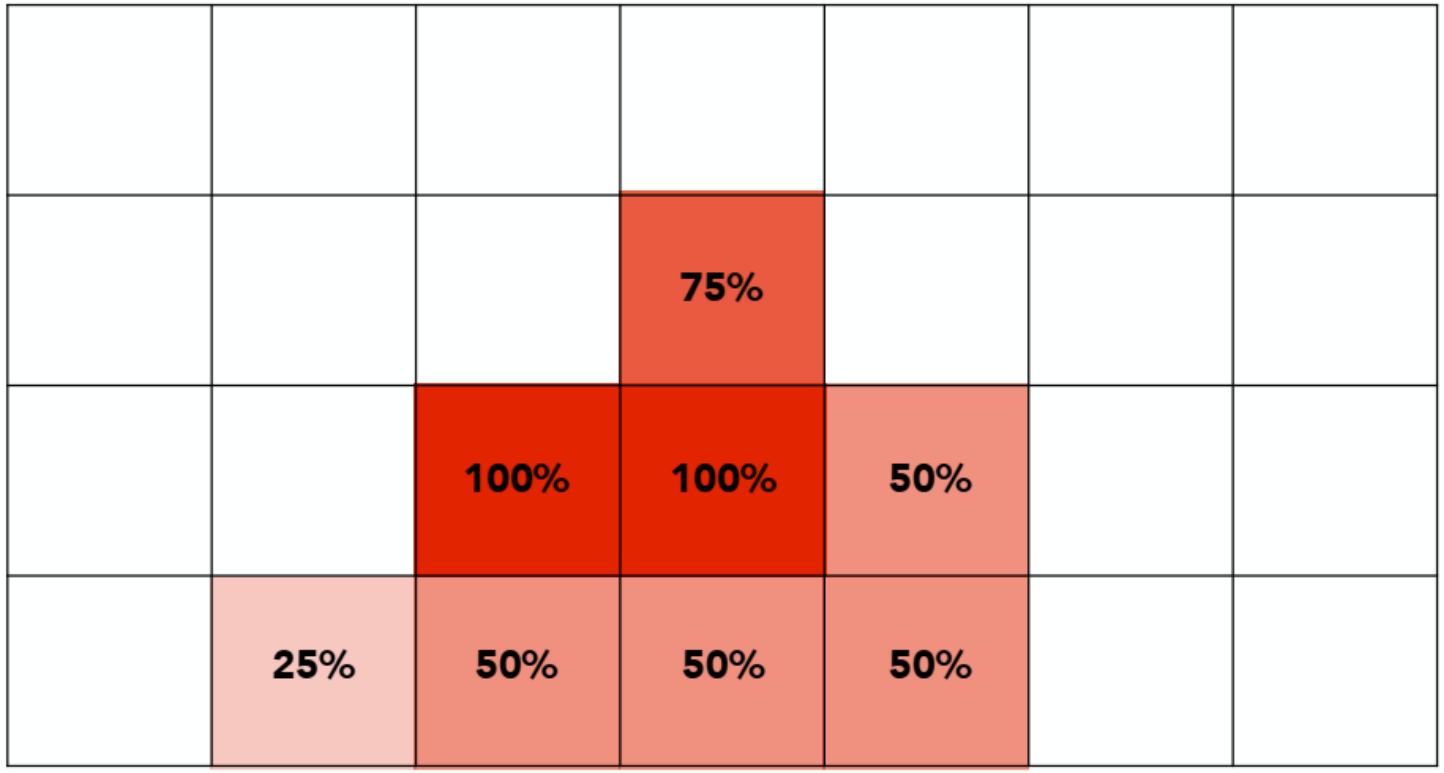
超级采样

2. 计算每个像素中N*N个采样点的平均值

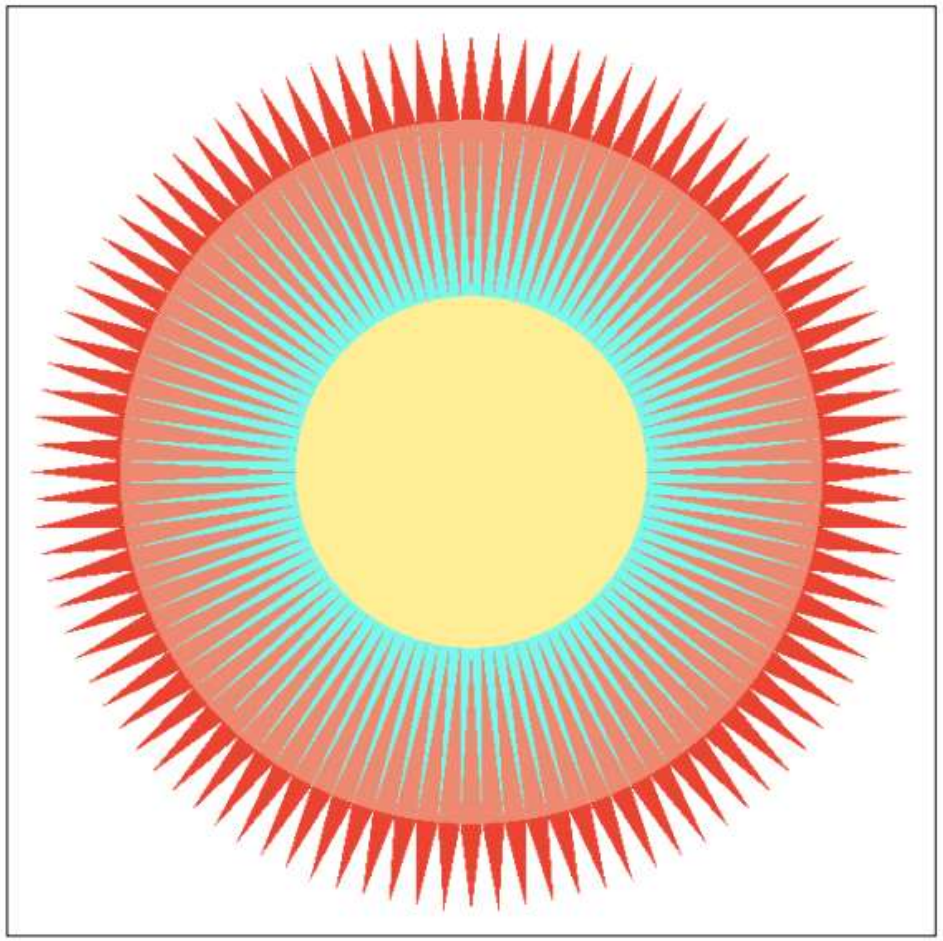


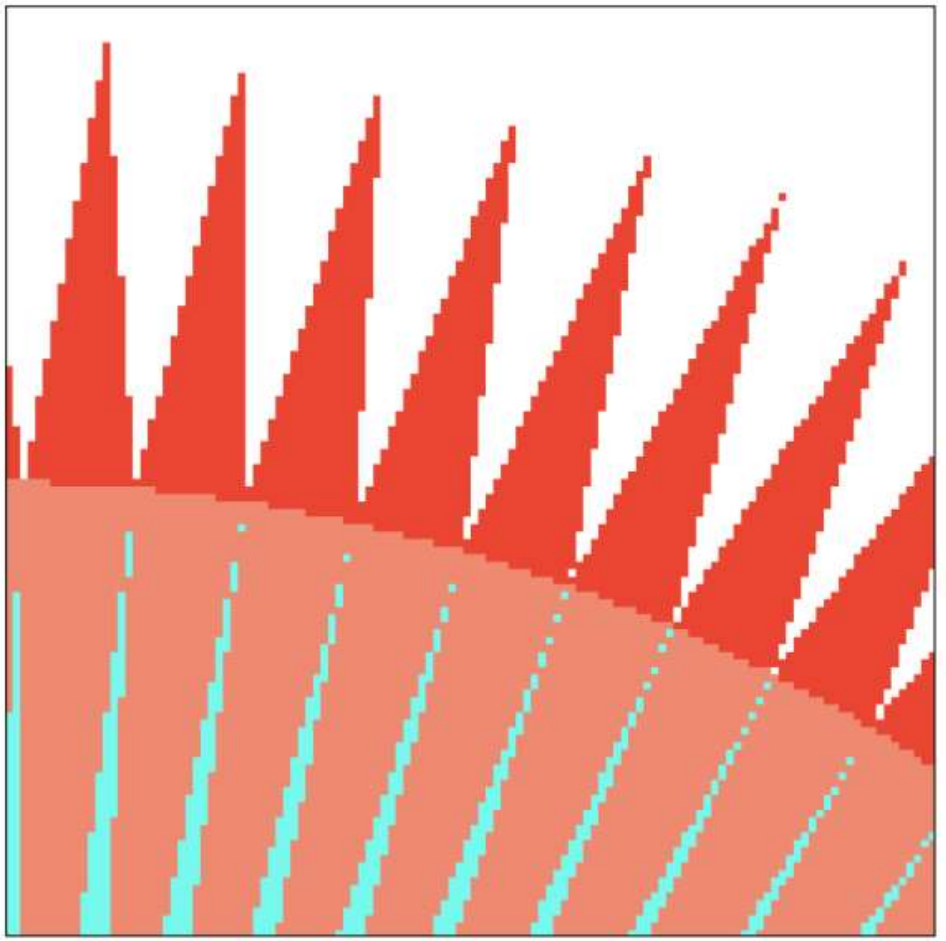
超级采样

- 显示结果

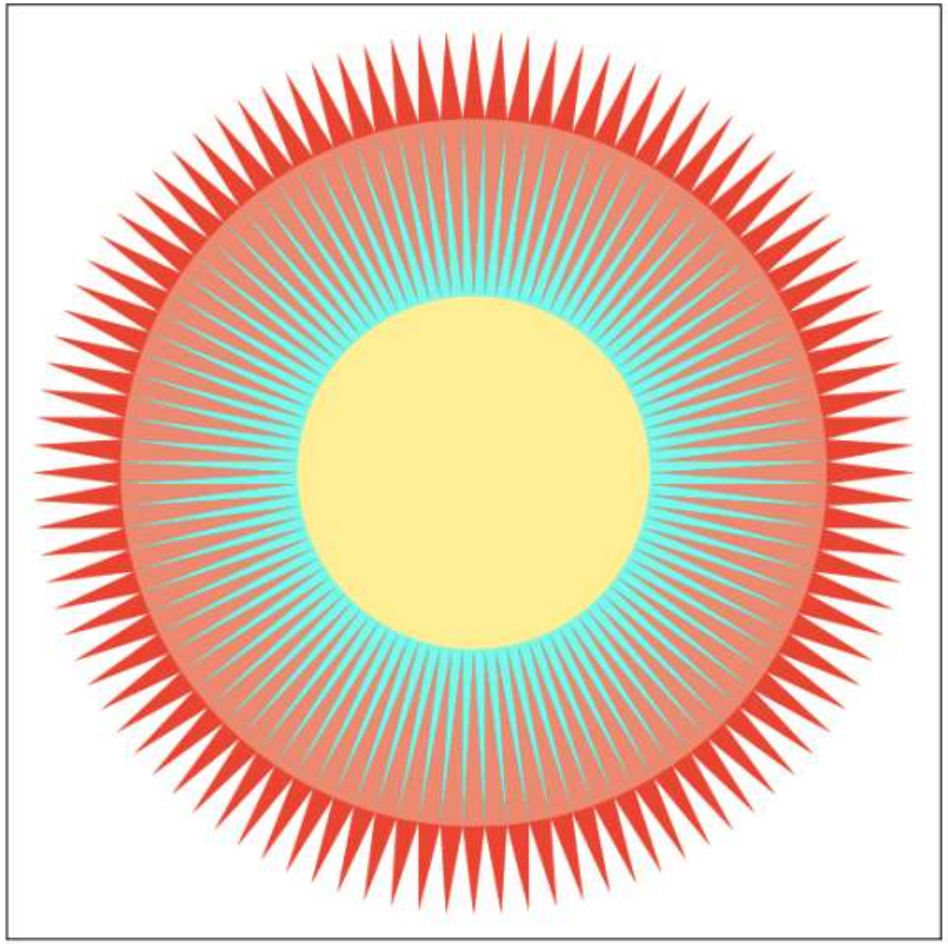


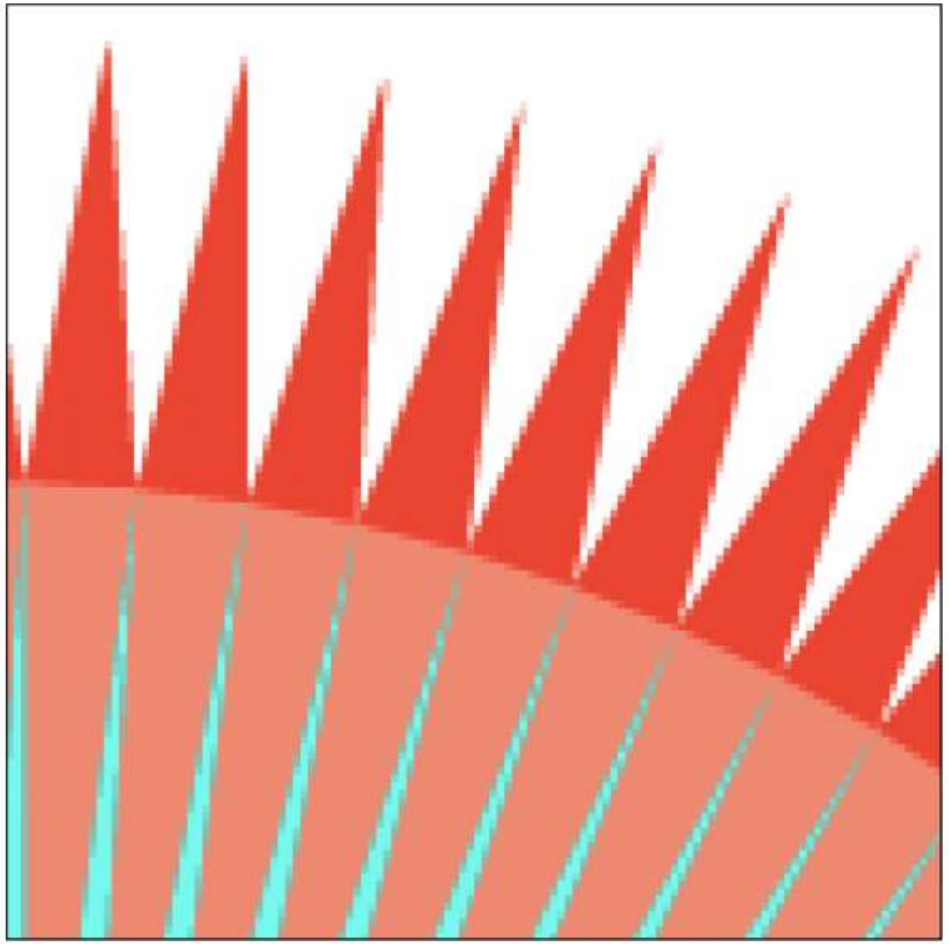
点采样





4*4超级采样





Q&A

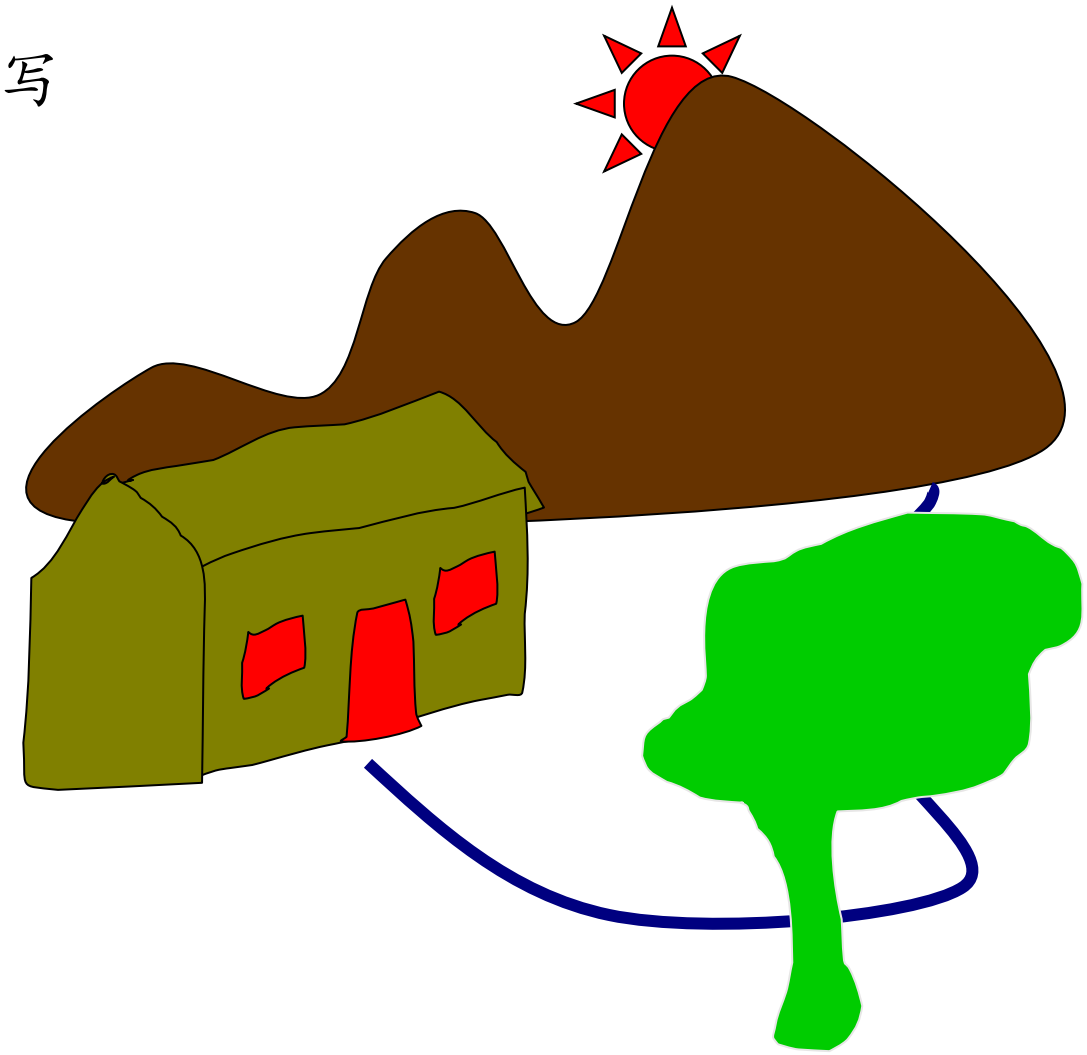
遮挡/可见性

- 光栅化一个三角形

Q:如果是多个三角形呢？会存在什么别的问题吗？

画家算法

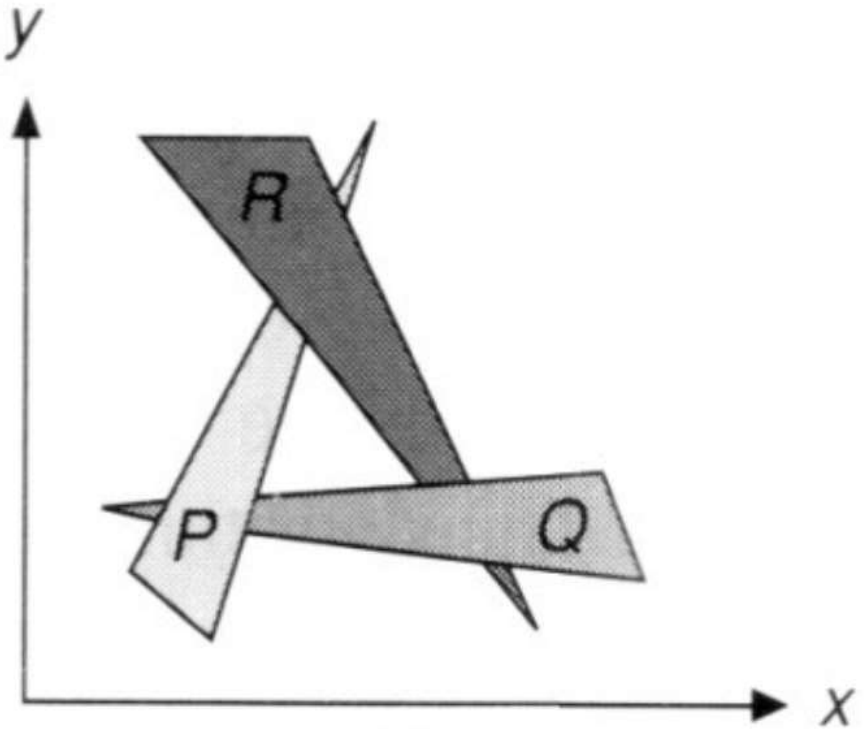
- 受画家由远至近作画方式的启发，近处物体覆盖遮挡远处物体
- 在帧缓冲器中重写



画家算法

- 对场景中的多边形按深度进行排序（n个三角形，复杂度就是O($nlogn$)）
- 存在无解的深度顺序

Q：如何解决？



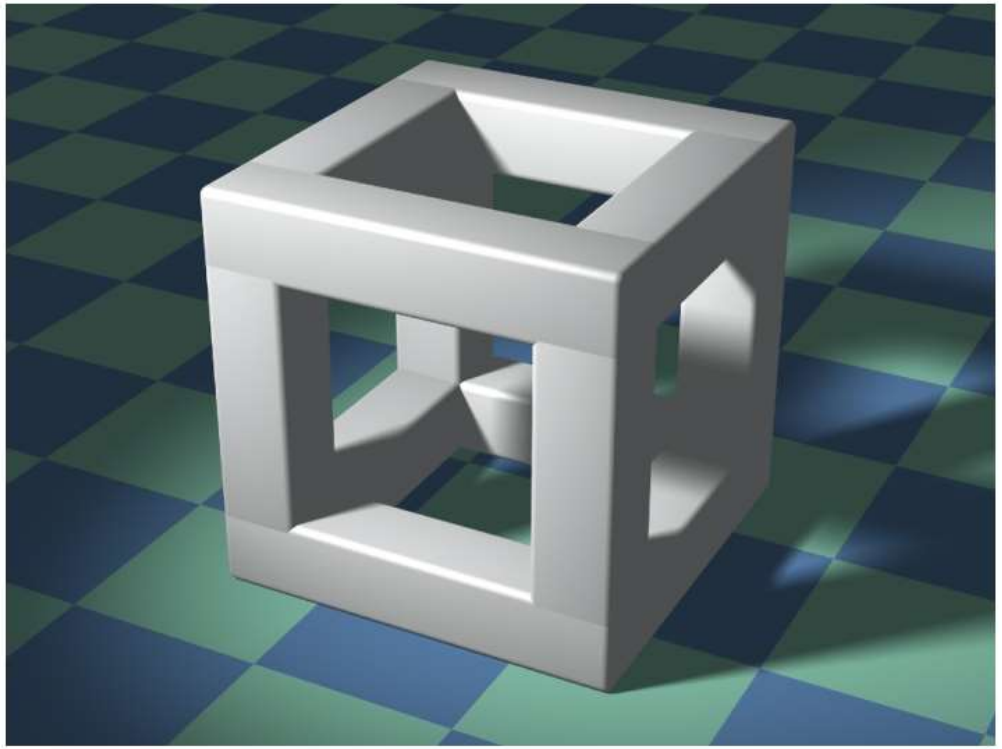
[Foley et al.]

深度缓冲器（ Z-buffer ）算法

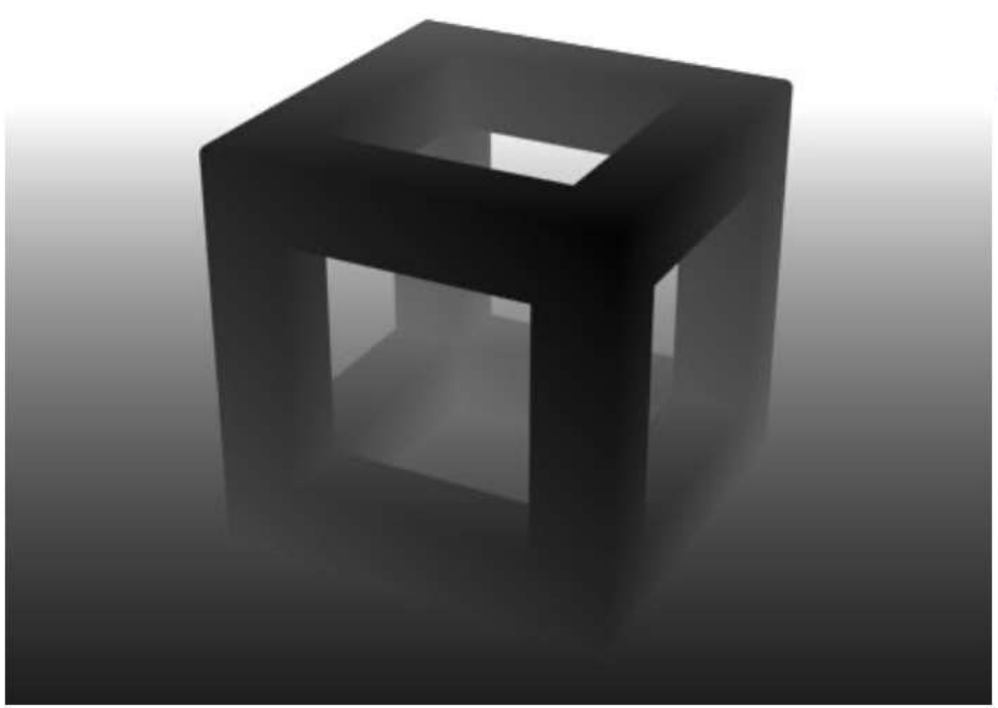
- 思想：对每个采样点（像素）记录当前的最小Z值
- 注意：我们假设Z永远是正的，离着相机越近Z越小，越远Z越大

- 需要一个额外的缓冲器来记录深度值
  - 帧缓冲器（frame buffer）记录颜色值
  - 深度缓冲器（depth buffer/z-buffer）记录深度值

深度缓冲器（ Z-buffer ）算法



Rendering



Depth / Z buffer

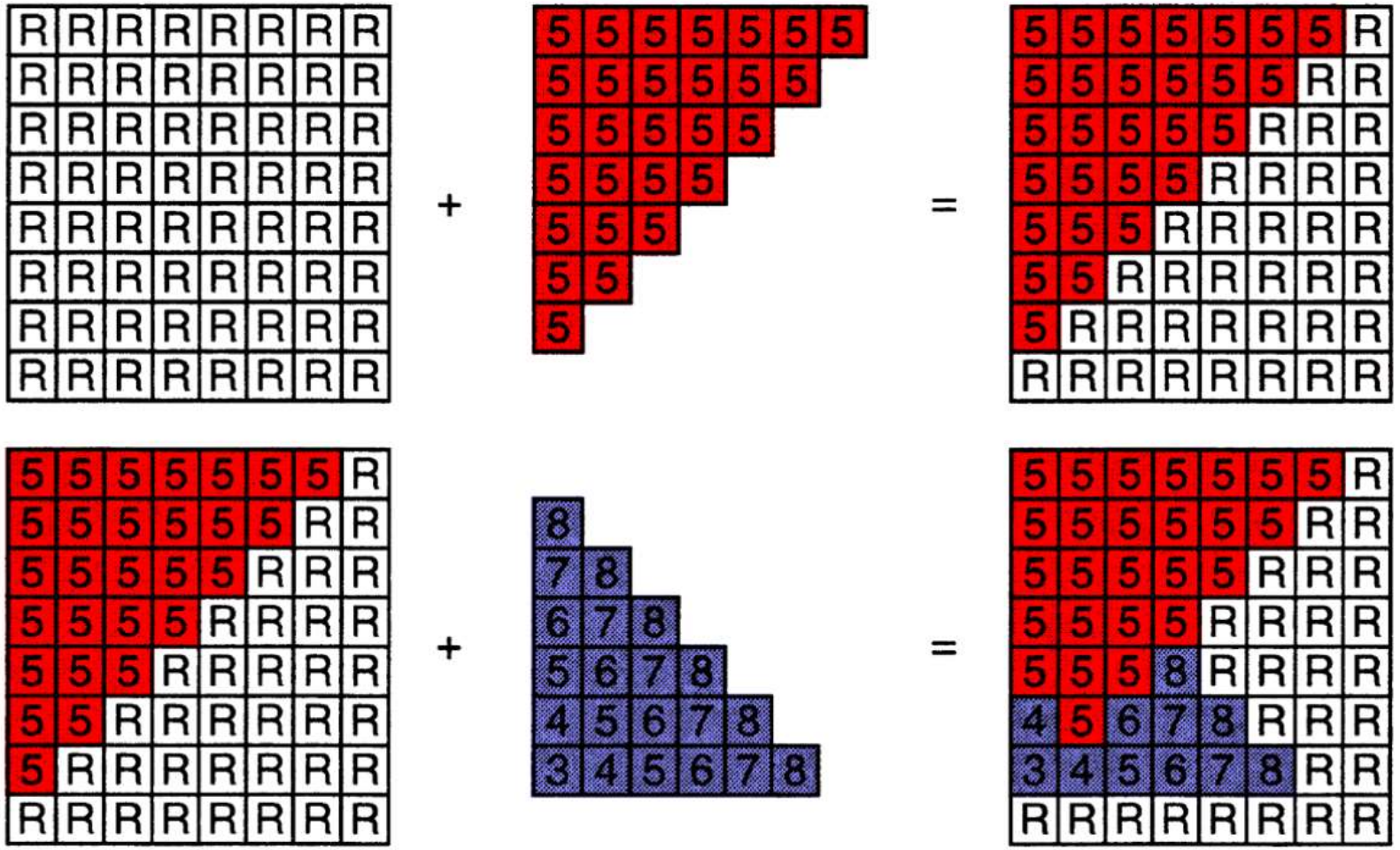
Image source: Dominic Alves, flickr.

深度缓冲器（ Z-buffer ）算法

- 初始化深度缓冲器的所有值为∞
- 光栅化过程中：

```
for (each triangle T)
    for (each sample (x,y,z) in T)
        if (z < zbuffer[x,y])                  // closest sample so far
            framebuffer[x,y] = rgb;            // update color
            zbuffer[x,y] = z;                  // update depth
        else
            ;                                  // do nothing, this sample is occluded
```

深度缓冲器（ Z-buffer ）算法



深度缓冲器（ Z-buffer ）算法

- 复杂度：假设每个三角形覆盖常数个像素，n个三角形复杂度为$O(n)$

Q:在线性时间内完成n个三角形的排序？

Q:如果以不同的顺序光栅化场景中的三角形，对结果是否有影响？

- 最重要的可见性算法，易于硬件实现

Q&A