# 一、Canonical Cube to Screen(规范立方体投影到屏幕)

## 1.前情提要What’s after MVP?

Model transformation (摆物体)

View transformation (摆相机)

Projection transformation

​ Orthographic projection (长方体转换为规范正方体)

​ Perspective projection (视锥转换为规范正方体)

## **2.本次概要**

Canonical Cube to Screen，即为光栅化Rasterization

## 3.基本概念

### 3.1Screen

​ – An array of pixels（像素组成的数组）

​ -Size of the array: resolution （数组的大小:分辨率）

– A typical kind of raster display （一种典型的光栅显示）

– Rasterize == drawing onto the screen （光栅化是绘制到屏幕上）

### **3.2Pixel** (FYI, short for “picture element”)

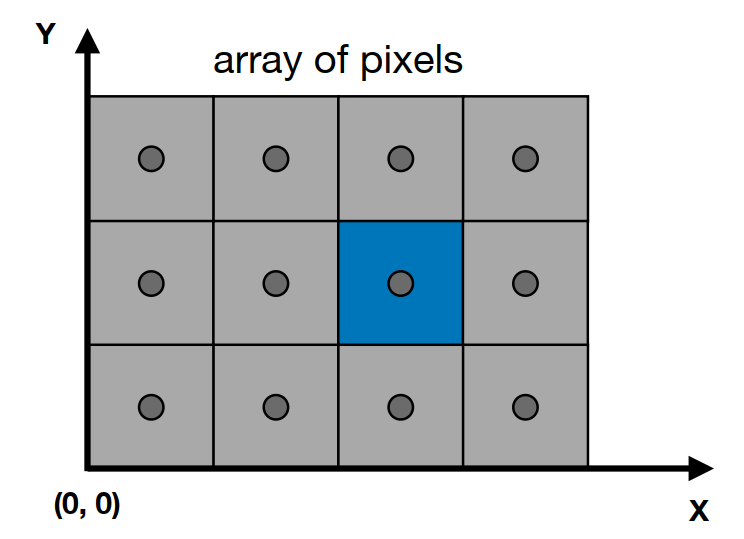
–For now: A pixel is a little square with uniform color （一个像素是一个具有统一颜色的小正方形）

–Color is a mixture of (red, green, blue) （一个像素是由红绿蓝三种颜色混合而成）

#### 3.2.1Screen space

Pixels’ indices are in the form of (x, y), where both x and y are integers(像素的索引采用 (x, y) 的形式，其中 x 和 y 都是整数)

Pixels’ indices are from (0, 0) to (width-1, height-1)( 像素的索引从 (0, 0) 到 (width-1, height-1))

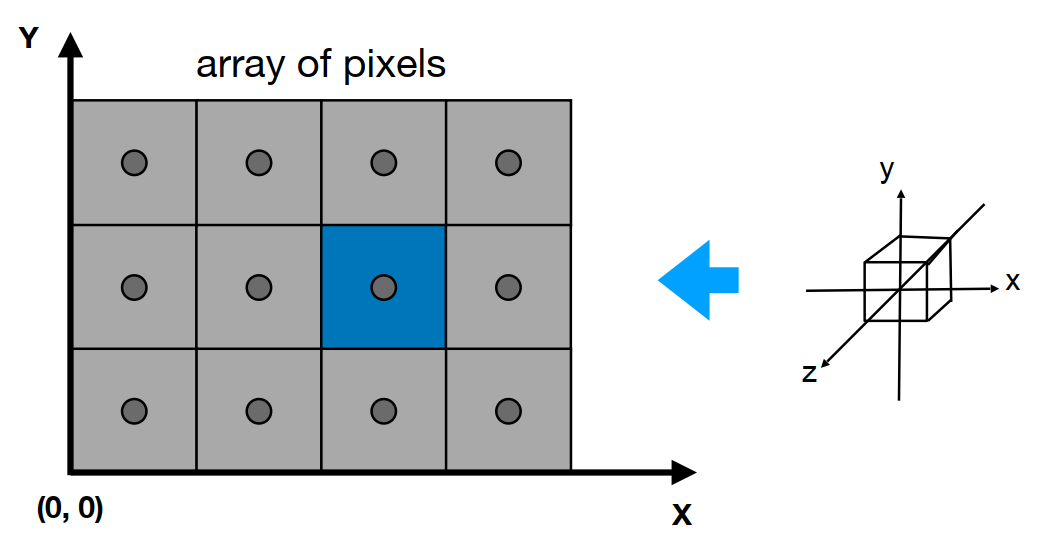


我们定义每一个像素位置都以该像素左下角的点的坐标来表示，定义屏幕左下角的点（原点）坐标为（0,0），那么上图中蓝色像素的坐标应为（2,1）。如果整个屏幕的分辨率是width×height，那么整个屏幕中的像素位置从（0,0）到（width-1, height-1）。

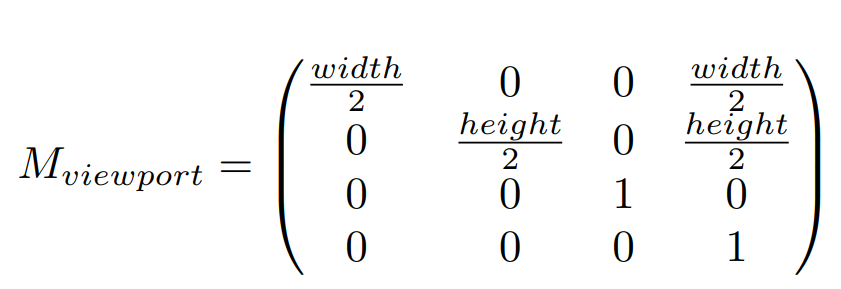
Pixel (x, y) is centered at (x + 0.5, y + 0.5) The screen covers range (0, 0) to (width, height)

虽然我们用整数的坐标来描述坐标的位置，但是每个像素实际中心准确来讲在（x+0.5，y+0.5）上。一个像素会覆盖1的宽度，整个屏幕覆盖的像素应该从（0,0）到（width, height）

#### 3.2.2Canonical Cube to Screen



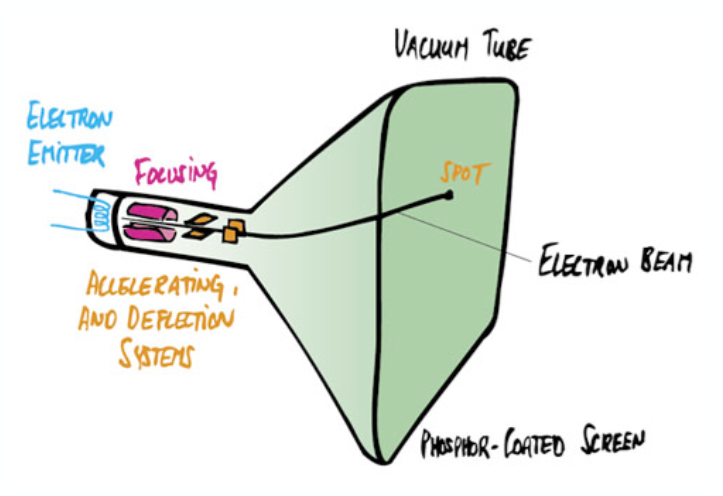
整个的过程暂时与z无关，那么就相当于将一个x从-1到1、y从-1到1的平面拉伸到x从0到width、y从0到height上，我们可以用一个变换矩阵Mviewport来写出这个变换过程。(视口变换)



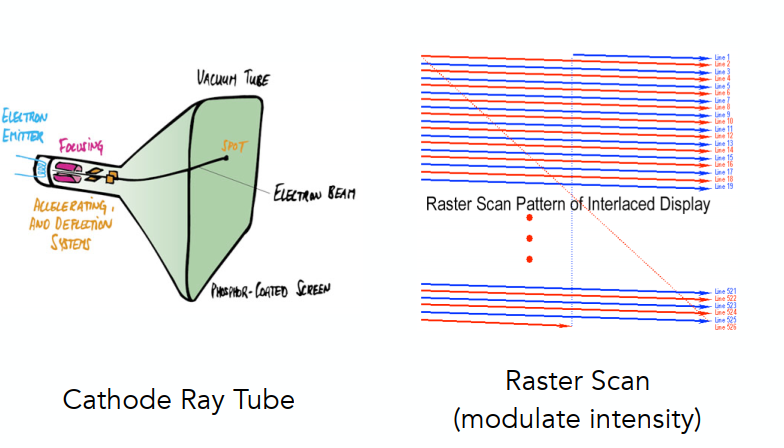
# 二、Drawing Machines(成像设备)

Different Raster Displays

## 1、Television - Raster Display CRT（Cathode Ray Tube阴极射线管）

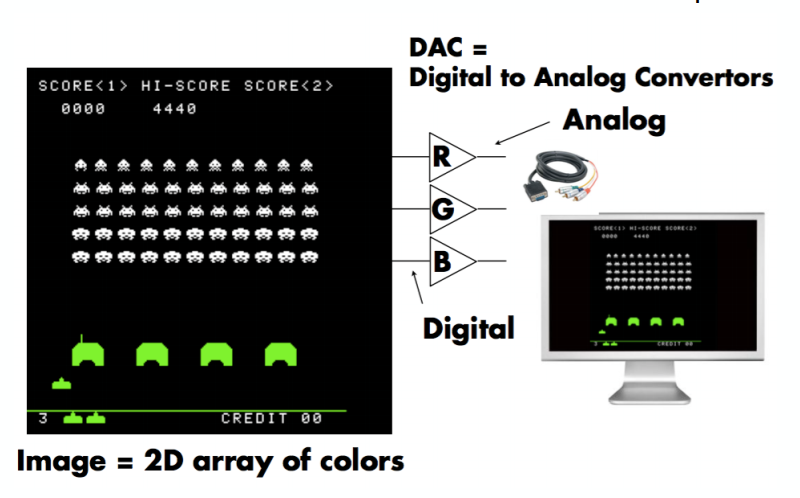


一条条画水平的线，当线足够密集的时候，就覆盖了整个屏幕。由于人眼有视觉暂留现象，而画面的切换速度又很快，因此为了让成像更快一点，可以隔一行画一条线，比如在当前时刻，只画奇数行的线，在下一时刻（或者下一张图）的时候，只画偶数行的线。这种技术就被称为隔行扫描技术。隔行扫描现在在一些视频压缩技术里依然存在。但是当画面高速运动的时候，隔行扫描可能会造成画面撕裂。

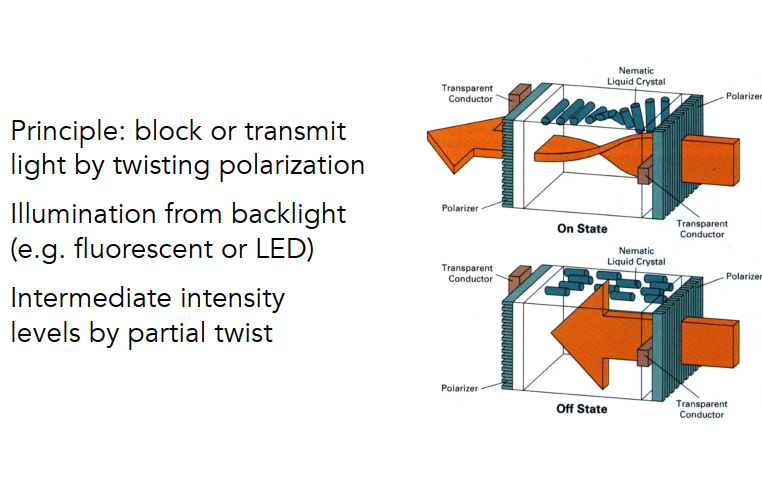


## 2、Frame Buffer: Memory for a Raster Display(帧缓冲区：光栅显示的内存)

显示的图像就是内存中的一块区域



## 3、LCD (Liquid Crystal Display) Pixel（液晶显示器）



液晶通过自己的不同排布影响光的极化（光的偏振方向），通过液晶的扭曲，将光的振动方向渐渐调整过来

# 三、Rasterization: Drawing to Raster Displays(绘制到光栅显示)

## 1、Triangles - Fundamental Shape Primitives（基本图元）

Why triangles?

Most basic polygon （三角形是最基本的多边形）

Break up other polygons （其他多边形都可以分解为三角形）

Unique properties （三角形有其独特的属性）

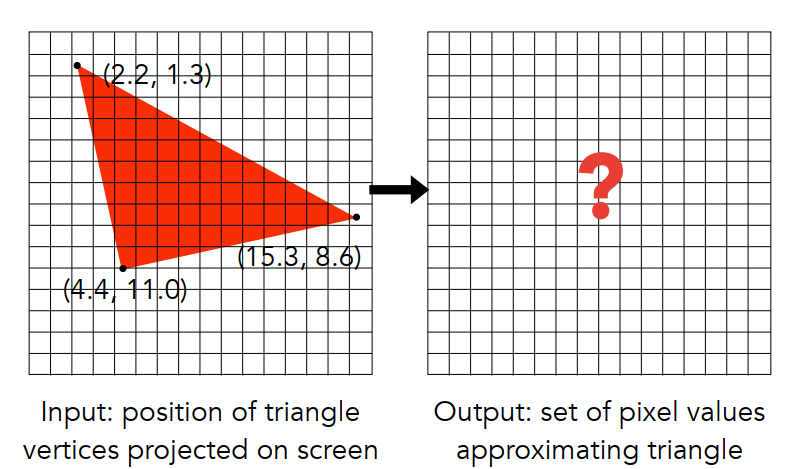
Guaranteed to be planar （确保是平面，三点确定一个唯一的平面）

Well-defined interior （三角形的内部外部定义的非常清楚）

Well-defined method for interpolating values at vertices over triangle (barycentric interpolation) （定义良好的三角形顶点插值方法(重心插值)，定义了三角形三个顶点的属性，其内部任意一个点都可以根据三个顶点的属性而定义出来）

（二）、Sampling

1、What Pixel Values Approximate a Triangle?



最简单的方法就是通过采样来实现光栅化。

### Sampling a Function

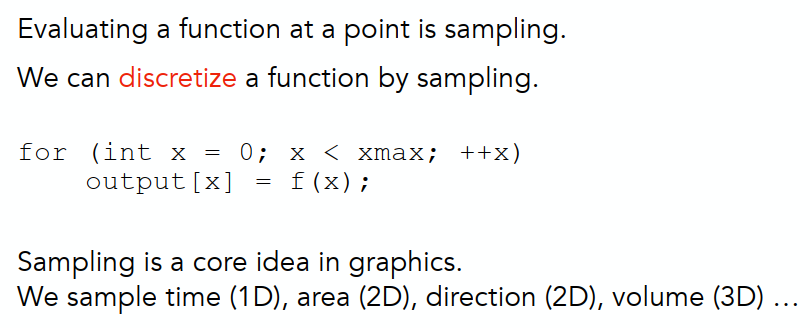
Evaluating a function at a point is sampling.

We can discretize a function by sampling.

什么是采样？通俗的解释是，给你一个连续函数，在不同的地方（x），我去问：这个函数的值（y）是多少。

采样即为把一个函数离散化的过程

用代码表示：



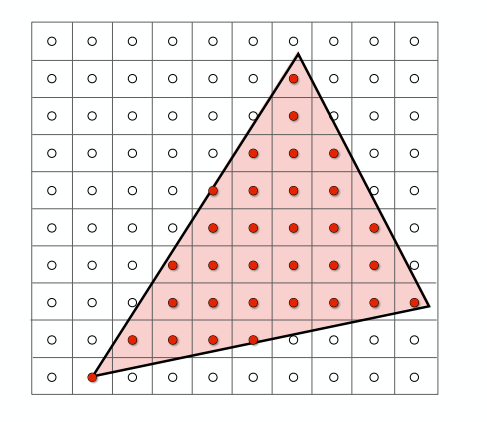
Sampling is a core idea in graphics. We sample time (1D), area (2D), direction (2D), volume (3D)

采样是图形的核心思想，应用在各个领域、各个维度。

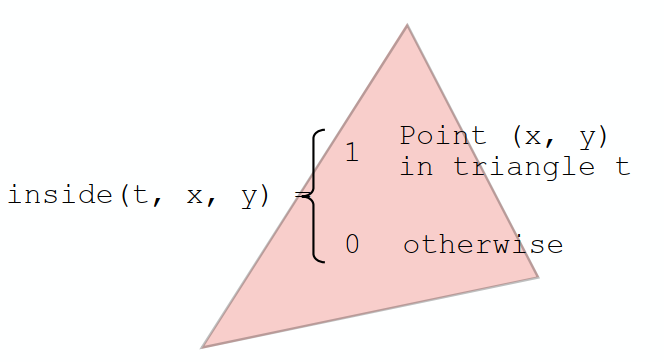
## 2、The Process of Sampling the Pixels in aTriangle

Sample If Each Pixel Center Is Inside Triangle

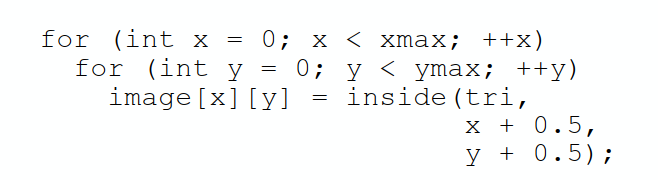
如果一个像素中心在三角形内，则进行采样。



Define Binary Function: inside(tri, x, y)

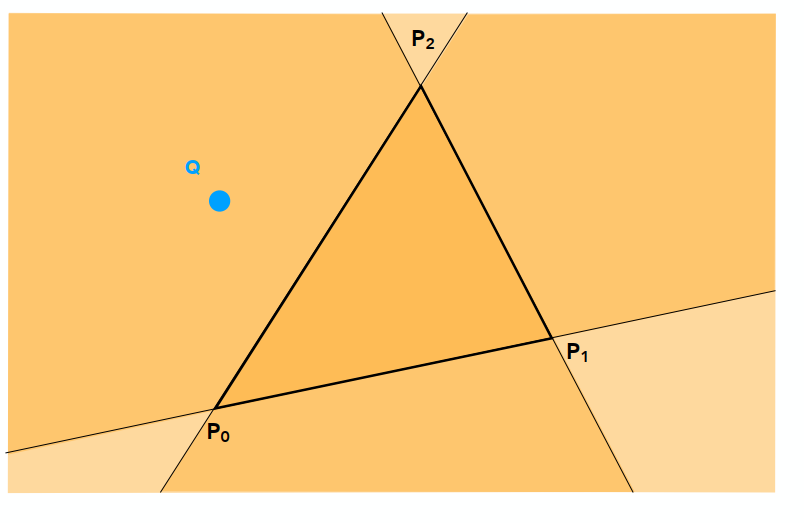


用代码表示这个过程：



通过遍历整个屏幕空间中的所有像素，来判断哪些像素的中心点在三角形的内部。

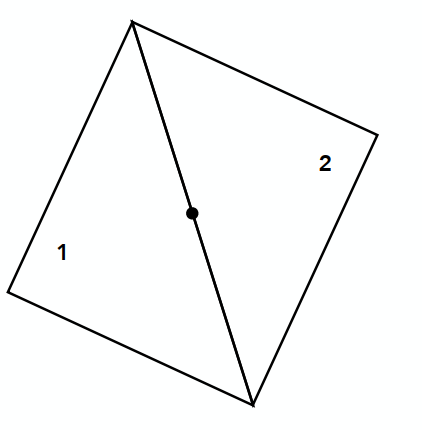
How to determine if a point is inside a triangle？Three Cross Products!(如何判断一个点是否在三角形内？三叉积！)



我们可以利用第二节课里讲的三个叉乘来判断一个点是否在三角形的内部。

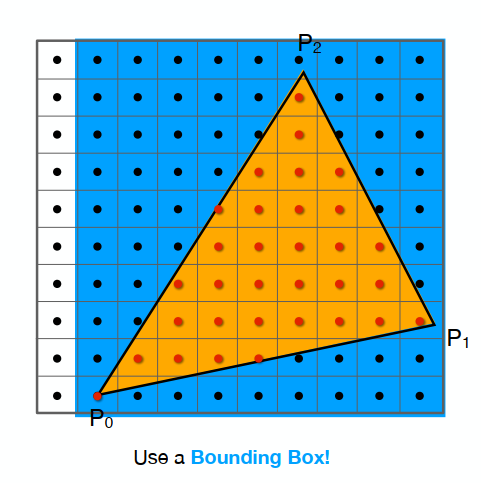
### Edge Cases

如果一个点在两个三角形的边界上，只要自己定义一个标准可以说通就好。（在OpenGL和DirextX上，规定左边和上边算，右边和下边不算）

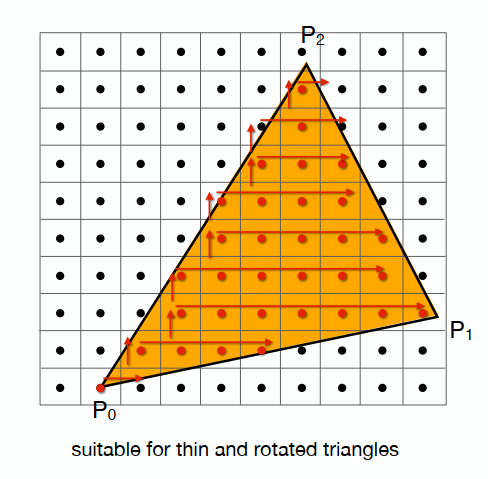


## 3、Optimization of check

Checking All Pixels on the Screen?



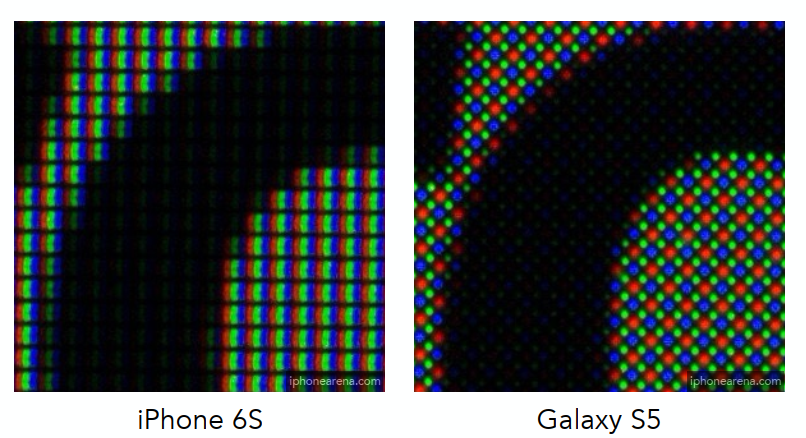
最左侧的一列显然不会有任何可以判断的点，此时我们可以采用一个包围盒AABB（蓝色区域），让所有的x和y都可以分别取最小和最大(像素遍历范围从0,0-width-1,height-1变为xmin,ymin-xmax,ymax)，将三角形用一个矩形的区域包起来，只有在这个区域内的像素才需要做刚刚的遍历，以加速光栅化的过程。



甚至可以让每一行都找其对应的最左和最右，这样一个像素都不会多考虑，相当于每一行都有一个包围盒。（包围盒的方法也不是适用于所有三角形，细长的三角形，又作了一定的旋转，占据了左下和右上，那么此时的包围盒要想把这个三角形包住，就会特别大。这时候就比较适合用这个方法）

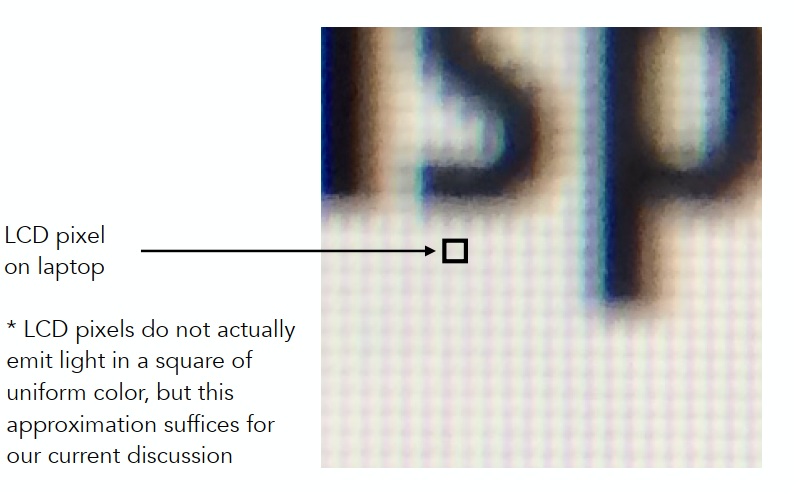
## 4、Rasterization on Real Displays(光栅化实际展示)

Real LCD Screen Pixels



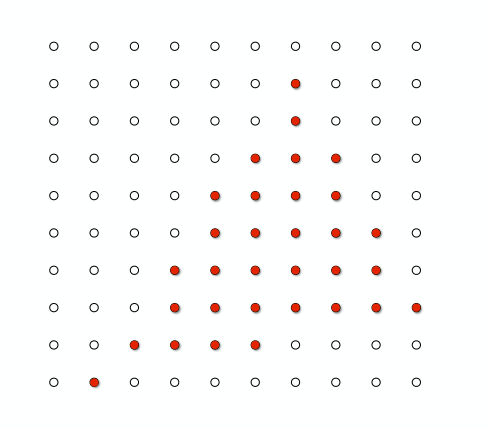
上方右侧这幅图可以看到，绿色的点明显多于其他颜色的点，因为人眼对绿色相对最敏感

Assume Display Pixels Emit Square of Light(假设显示像素发出方形光)

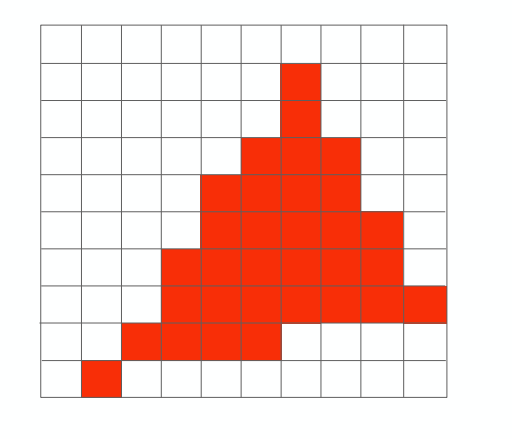


LCD 像素实际上并不以均匀颜色的正方形发光，但这个近似值足以满足我们当前的讨论

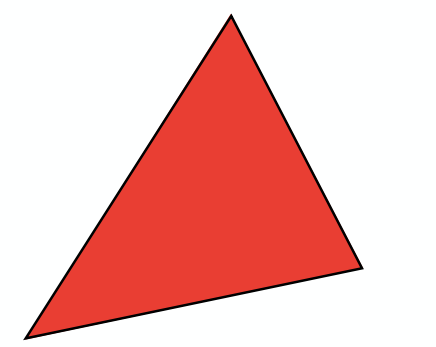
以下是采样点



每个采样点进行一个正方形扩充，可以看到锯齿化很严重



但是我们需要的是这样



所以我们要抗锯齿/反走样，参考下一讲