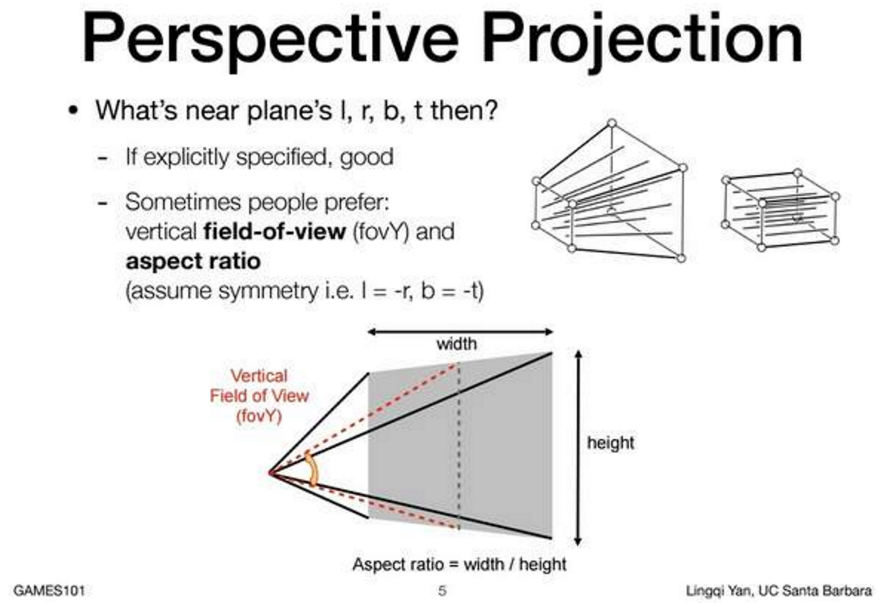
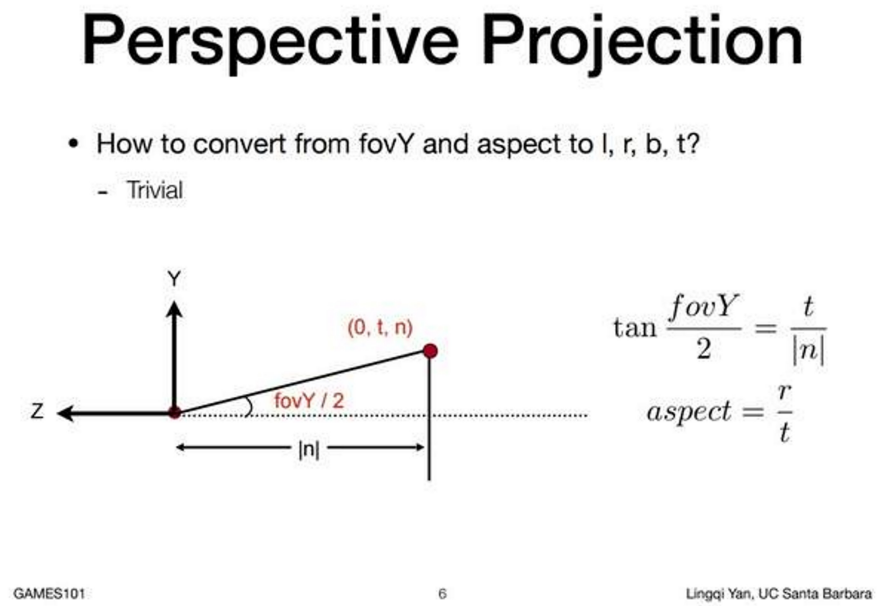


上节课把透视投影转化成正交投影

这里引入另外一个概念 Field of View，表示你能看到的角度的范围
注意看上图中红色线的夹角，就是垂直可视角度，他越大，可视角度越大
同理还有水平可视角度



现在假设从侧面看第一张示意图
右边那条线是近平面，到相机距离是n绝对值，上半部分角度是$\frac{fov\ Y}{2}$，红点是屏幕上侧（最高点）

What's after MVP?

- **M**odel transformation (placing objects)
- **V**iew transformation (placing camera)
- **P**rojection transformation
  - Orthographic projection (cuboid to "canonical" cube $[-1, 1]^3$)
  - Perspective projection (frustum to "canonical" cube)
- Canonical cube to ?

MVP这三个变换之后，所有东西都会停留在一个1，1，1的位于原点的标准立方体中
下一步就要把这立方体画在屏幕上

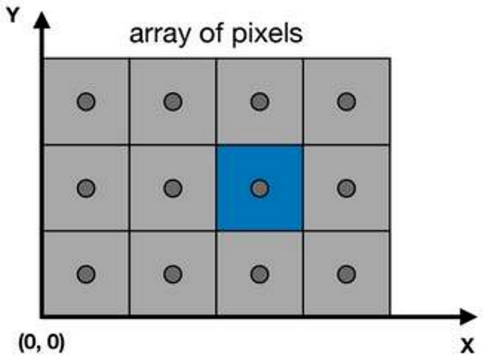
Canonical Cube to Screen

- 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)

屏幕本质上是二维数组，一组像素
Raster就是德语屏幕，rasterization就是把东西画在屏幕上
像素就是最小的屏幕单位，每个像素有不同的颜色

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)

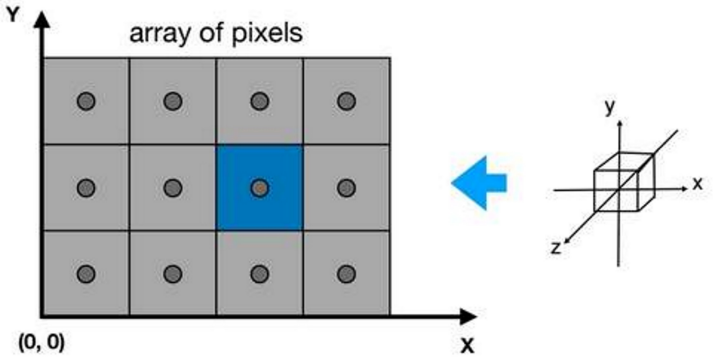
屏幕空间：就是给屏幕定义一个坐标系

比如，可以定义左下角是原点

实际上像素的中心是$(x+0.5, y+0.5)$上

Canonical Cube to Screen

- Irrelevant to z
- Transform in xy plane: [-1, 1]² to [0, width] x [0, height]



Canonical Cube to Screen

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

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

我们要做的就是把标准立方体空间映射到屏幕这个二维世界中去

z暂时不管

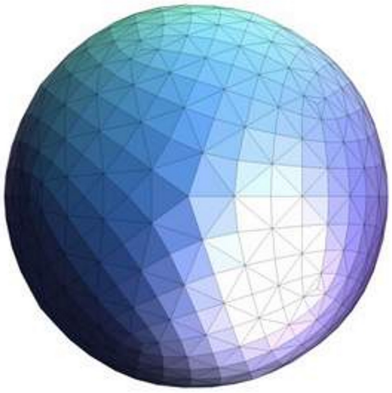
其他两个坐标是$[-1,1]^2$转换到 $[0, width] * [0, height]$

使用上面这个矩阵做变换

Next: Rasterizing Triangles into Pixels

经过三维转化为二维后，我们还需要把二维图片打散成像素！

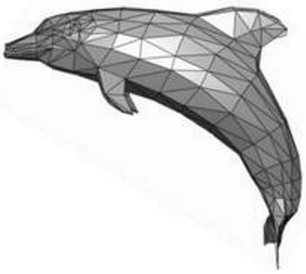
Triangle Meshes



Triangles - Fundamental Shape Primitives

Why triangles?

- Most basic polygon
  - Break up other polygons
- Unique properties
  - Guaranteed to be planar
  - Well-defined interior
  - Well-defined method for interpolating values at vertices over triangle (barycentric interpolation)

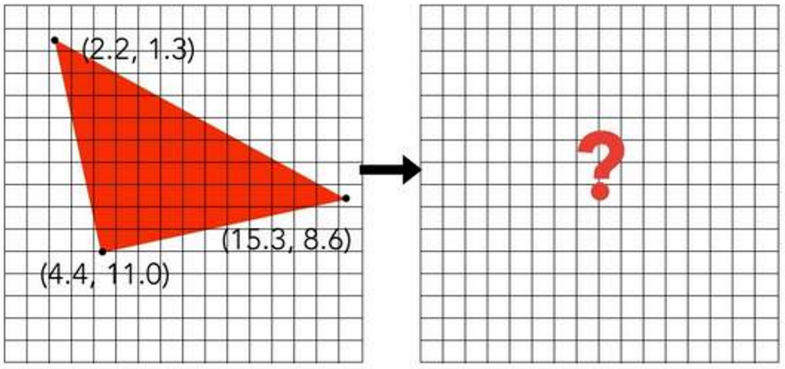


三角形可以拼接在三维空间中的面，或者二维空间中复杂的图形

三角形内部一定是平面的

给三角形顶点定义不同属性，可以在三角形内部进行插值

What Pixel Values Approximate a Triangle?



Input: position of triangle vertices projected on screen

Output: set of pixel values approximating triangle

现在就是要把三角形变成像素

Sampling a Function

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);
```

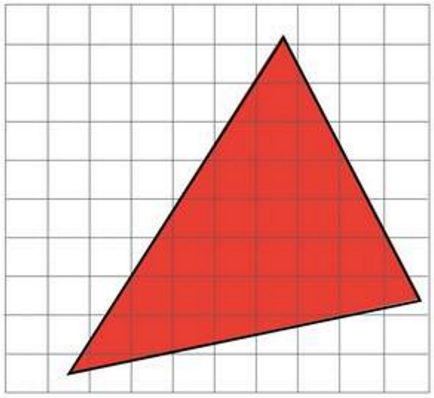
Sampling is a core idea in graphics.
We sample time (1D), area (2D), direction (2D), volume (3D) …

怎么做呢？通过采样的方法

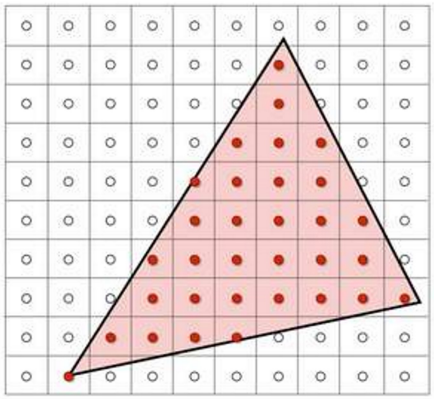
采样就是把函数给离散化的过程

可以对时间，面积，方向，体积……进行采样

Rasterization As 2D Sampling

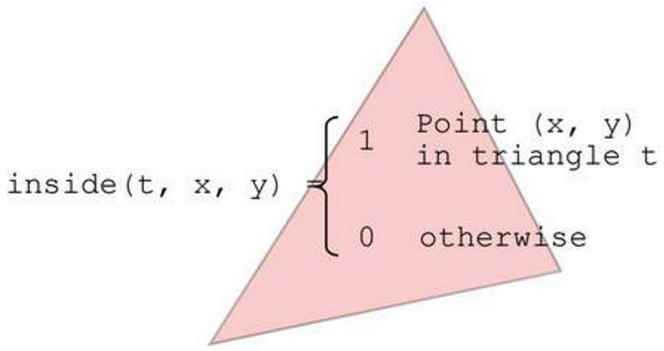


Sample If Each Pixel Center Is Inside Triangle



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

x, y: not necessarily integers



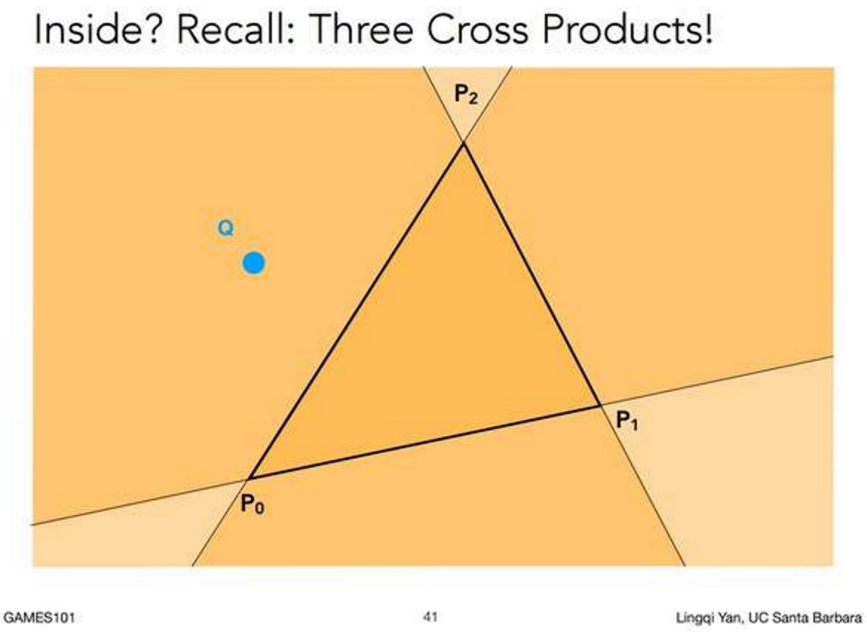
这里我们要做的就是给定一个三角形，判断像素中心是否在三角形内部。

Rasterization = Sampling A 2D Indicator Function

```
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);
```


如何去做这个采样？用两个for循环，逐个考虑每个像素中心

注意像素中心是在$(x+0.5, y+0.5)$



那么，如何判断一个点是否在三角形内？用叉乘！！

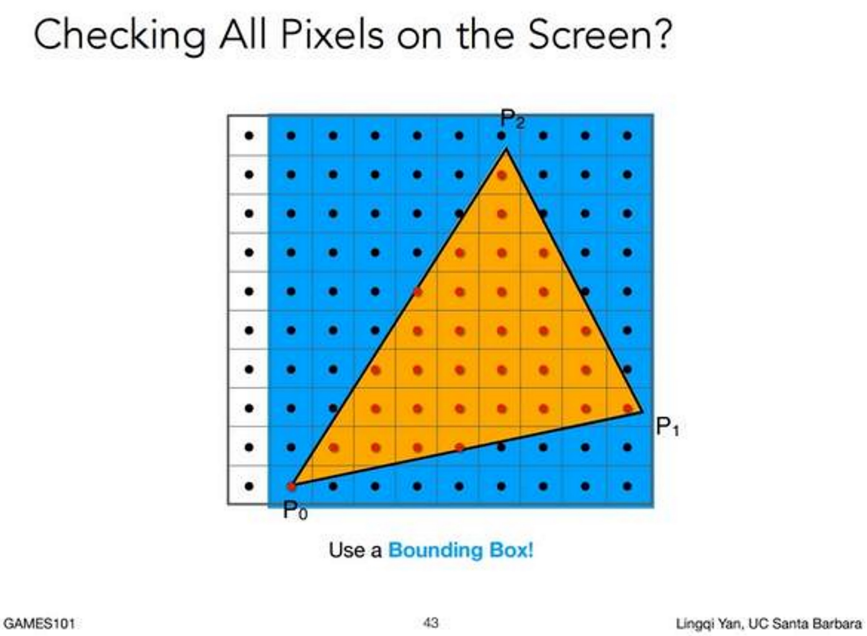
比如对上图，判断Q是否在三角形内部

首先$P1P2\ X\ P1Q$，将会得到一个z为正数的向量，也就是结果向量朝向屏幕外的，利用右手定则，可以得知Q在$P1P2$的左侧（因为如果在右侧，那么结果将会是向量z为负数，那么结果向量就朝向屏幕内部）

类似的$P2P0\ X\ P2Q$，得到Q在右侧，不对劲！

$P0P1\ X\ P0Q$，得到Q在左侧

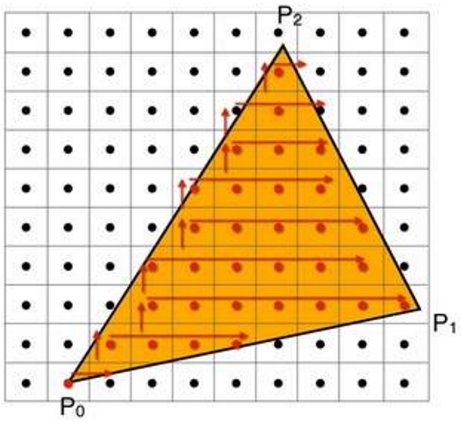
注意，向量按照一定的顺序去判断，比如我们上面是按照P1, P2, P0去判断的



检查屏幕所有的像素太花时间！

可以只检查蓝色的包围盒（Bounding box）部分。

Incremental Triangle Traversal (Faster?)



suitable for thin and rotated triangles

也可以每一行设置一个包围盒，进一步减小包围盒

很适用于那种三角形很小，但是包围盒很大的（窄长三角形）