

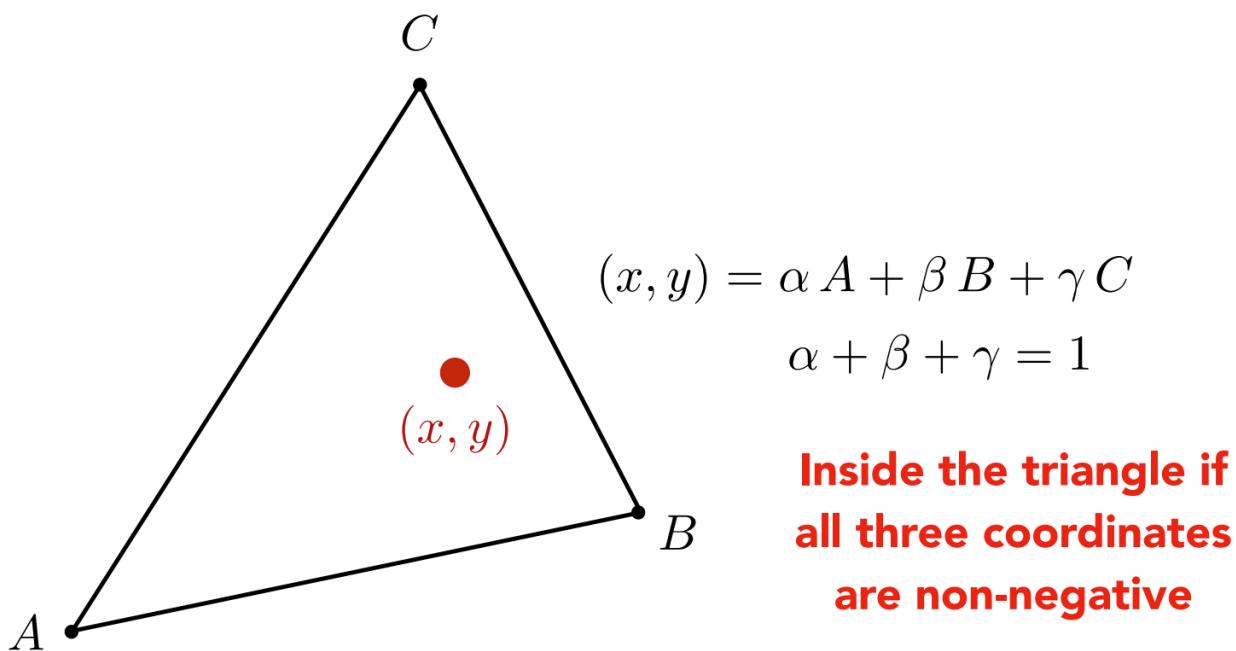
Lecture 09

1. Interpolation Across Triangles: Barycentric Coordinates 重心坐标

插值的目的

- 指定任意一个顶点的值
- 平滑地获取三角形内部点的值
- 方式
- 重心坐标系

1. Barycentric Coordinates 重心坐标

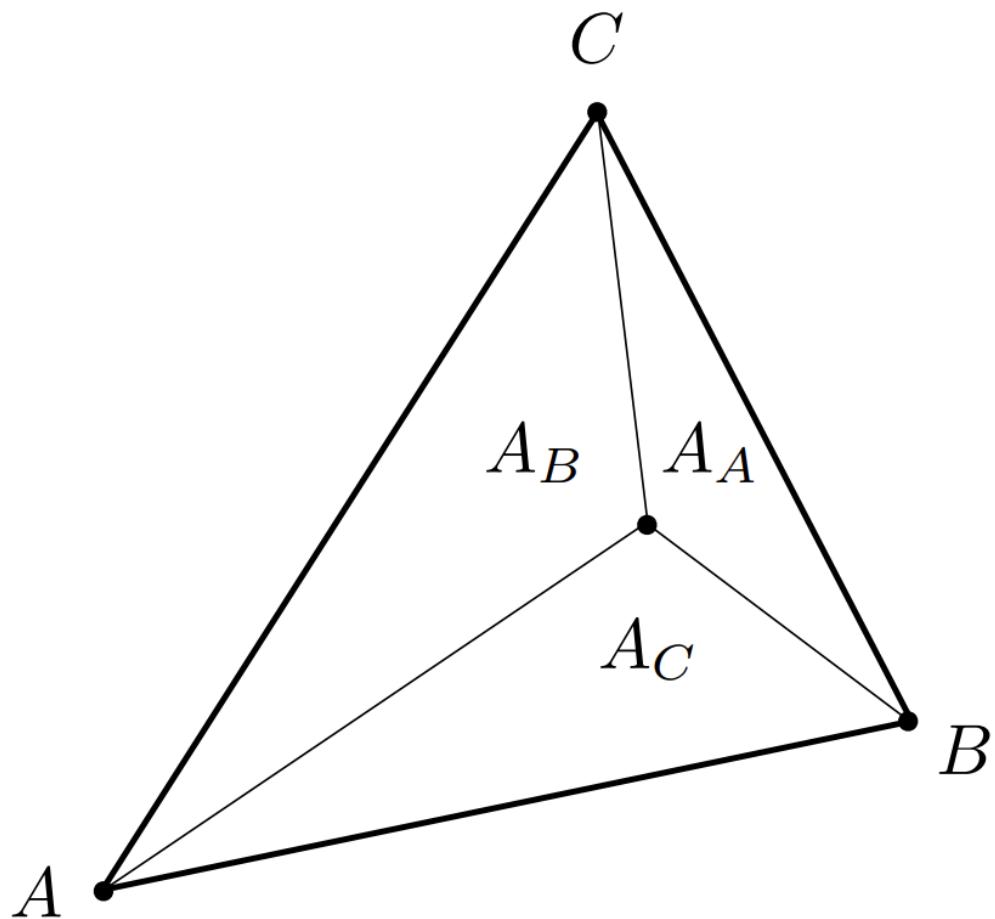


重心坐标系

- 根据三角形的三个顶点，能表达出三角形内任意一个点
- 如果表达的点在三角形内，**所有的系数都为非负值** ($\alpha, \beta, \gamma > 0$)，**三角形三点系数和为1** (三点在同一平面)
- **用顶点到任意点的长度 系数计算**

$$\alpha = \frac{A_A}{A_A + A_B + A_C}$$
$$\beta = \frac{A_B}{A_A + A_B + A_C}$$
$$\gamma = \frac{A_C}{A_A + A_B + A_C}$$

◦



◦ **用坐标 系数计算**

$$\text{有 } \alpha x_A + \beta x_B + \gamma x_C = x$$

$$\alpha y_A + \beta y_B + \gamma y_C = y$$

于是由 $\alpha + \beta + \gamma = 1$ 消元

$$\alpha = \frac{-(x - x_B)(y_C - y_B) + (y - y_B)(x_C - x_B)}{-(x_A - x_B)(y_C - y_B) + (y_A - y_B)(x_C - x_B)}$$

$$\beta = \frac{-(x - x_C)(y_A - y_C) + (y - y_C)(x_A - x_C)}{-(x_B - x_C)(y_A - y_C) + (y_B - y_C)(x_A - x_C)}$$

$$\gamma = 1 - \alpha - \beta$$

2. 重心坐标的运用

- 重心坐标可以运用在位置、纹理、坐标、颜色、深度、等等
- 但是不能在投影后的物体上做重心插值**, 其原因是: 比如将 3D 物体投影到 2D 屏幕后, 点的重心坐标会发生变化, 即插值计算的结果发生了变化, 在 3D 空间和 2D 空间中插值的结果并不相同。解决方法 是在 3D 空间中插值计算, 再将该信息填到该点投影到 2D 平面上的点中。

3. 纹理插值

简单的纹理插值伪代码如下:

```

    Usually a pixel's center
    ↑
for each rasterized screen sample (x,y):
    (u,v) = evaluate texture coordinate at (x,y)
    texcolor = texture.sample(u,v);
    set sample's color to texcolor;
    ↑
    Usually the diffuse albedo Kd
    (recall the Blinn-Phong reflectance model)
    ←
    Using barycentric
    coordinates!

```

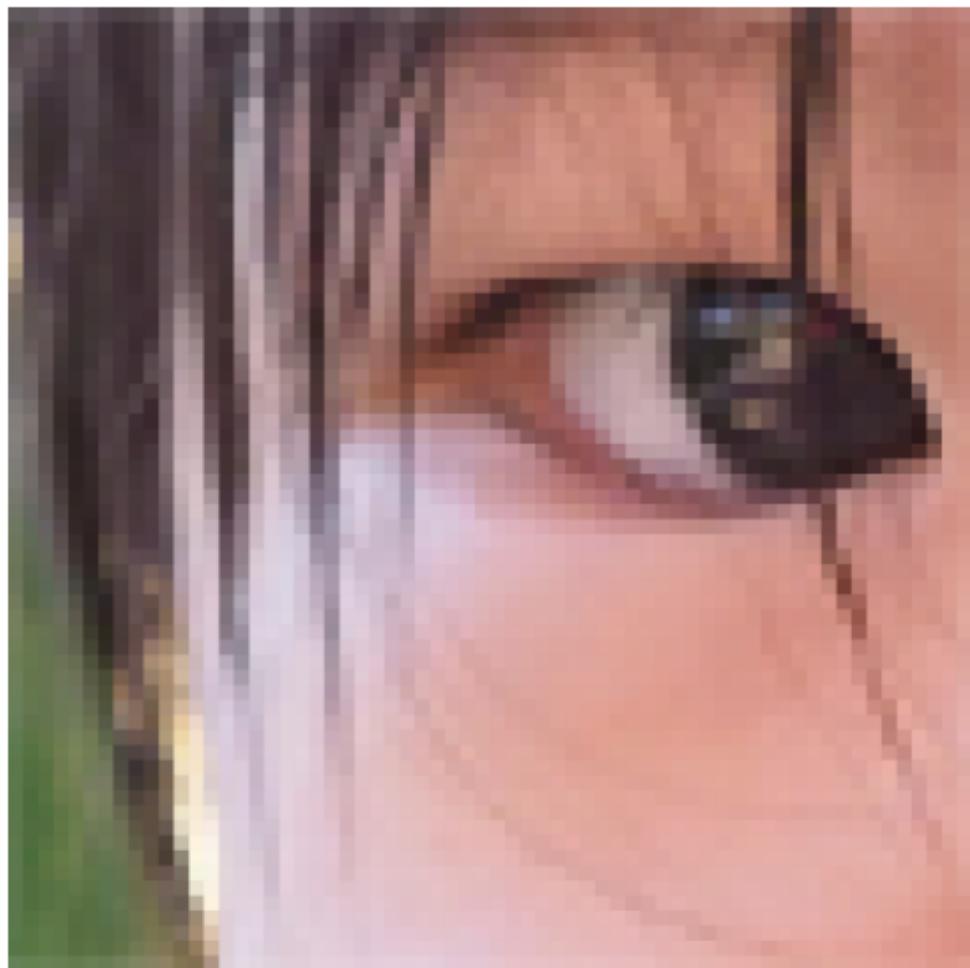
- 插值计算某点纹理坐标系的值

- 应用函数计算颜色系数
- 设置该点的某些性质，比如 k_d [Lecture 08 > 3.Blinn-Phong Reflection Model 完整公式](#)
- 纹理上的一个像素称之为 texel / 纹素

2.纹理渲染的一些问题

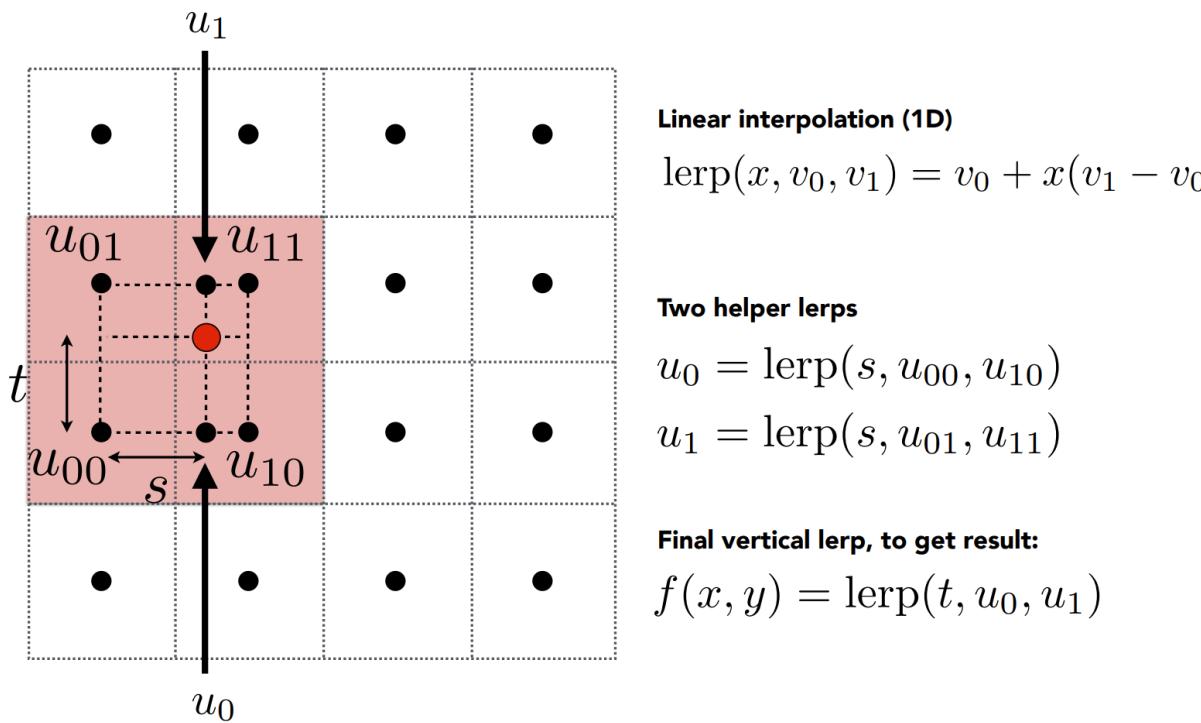
1. Texture Magnification (纹理放大)

- 当插值的点并不在某个像素的中心，即图像分辨率大于纹理分辨率，此时仅仅按最近的点计算会导致 **方块** 的渲染结果



Nearest

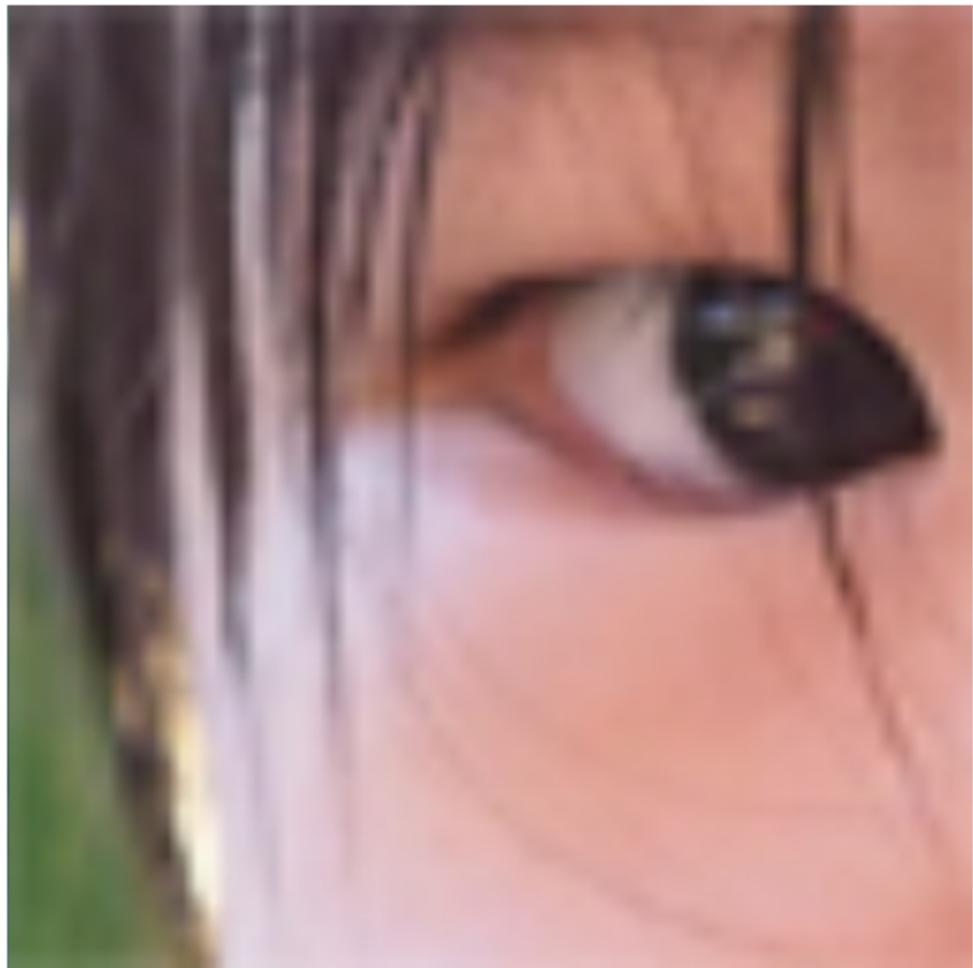
- 使用最近的四个点 做 双线性插值(Bilinear)



- 使用线性插值

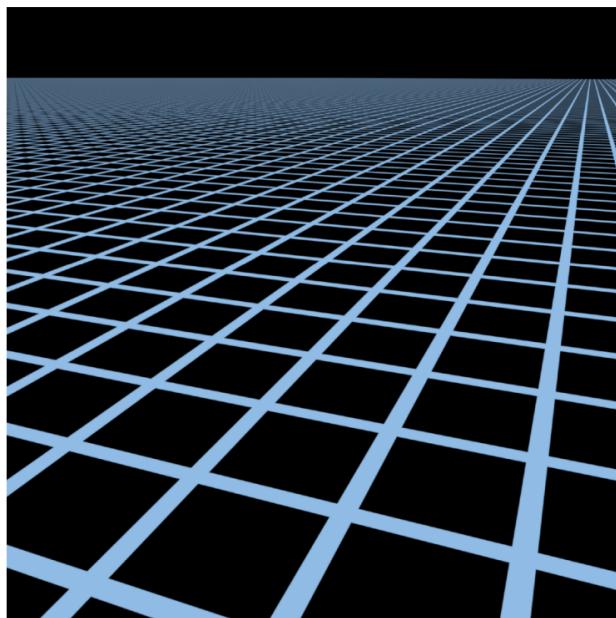
$$\text{lerp}(x, v_0, v_1) = v_0 + x(v_1 - v_0)$$

- 在水平方向上使用两次插值，再于垂直方向上进行一次插值（水平和垂直上均可任选一个做两次插值），效果会较好

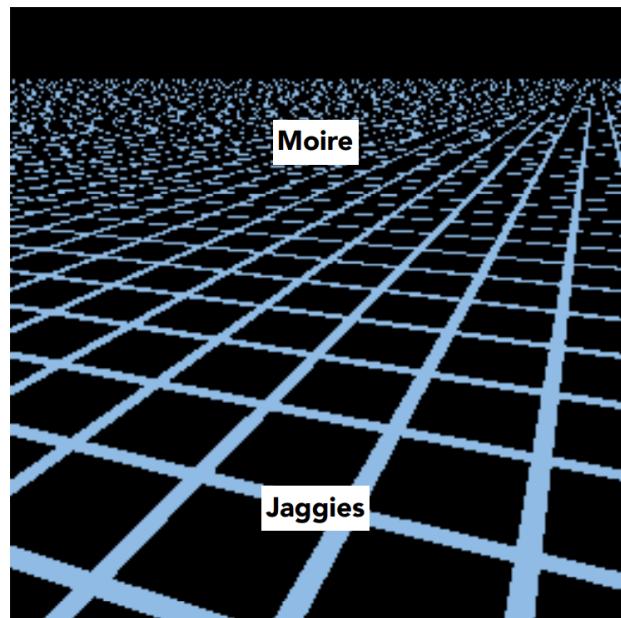


Bilinear

2. exture Minification (纹理缩小)

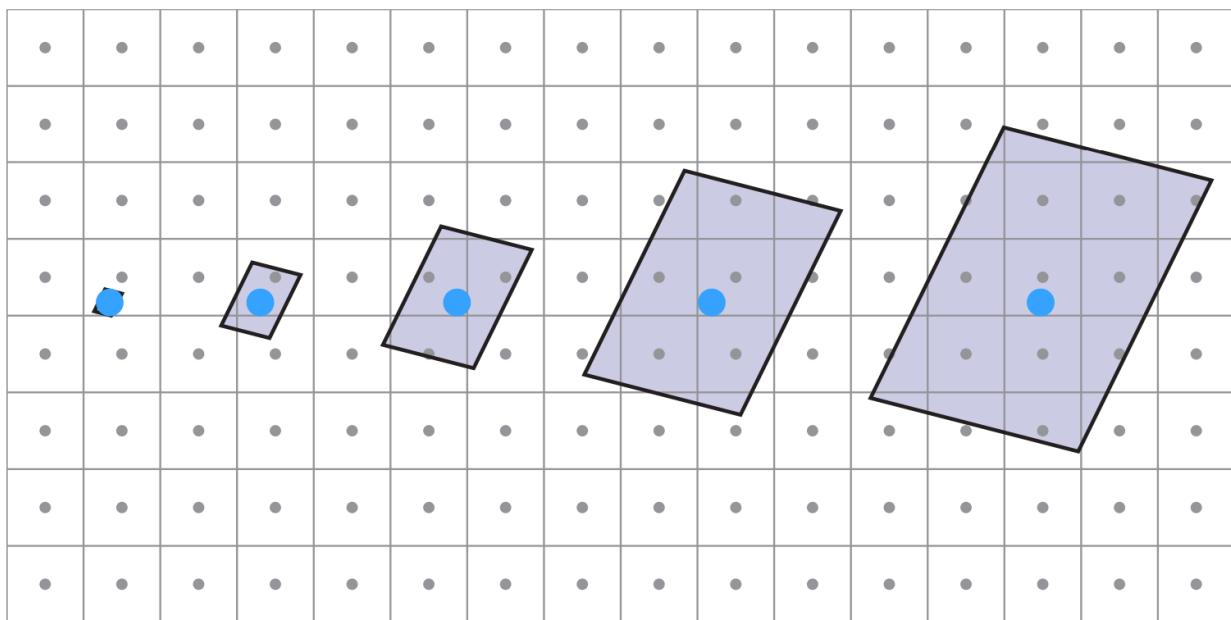


Reference



Point sampled

由于 **近大远小** 的关系，随着距离的变大，单个像素覆盖到的纹素会变多，而此时对像素内包含所有的纹素取均值是错误的方法，将纹素类比为样本数，像素类比为采样频率，像素大于纹素会导致失真/Aliasing。而这里也一样会导致失真。如下图



解决问题有两个思路

- 增大采样率，使用超采样。性能代价大

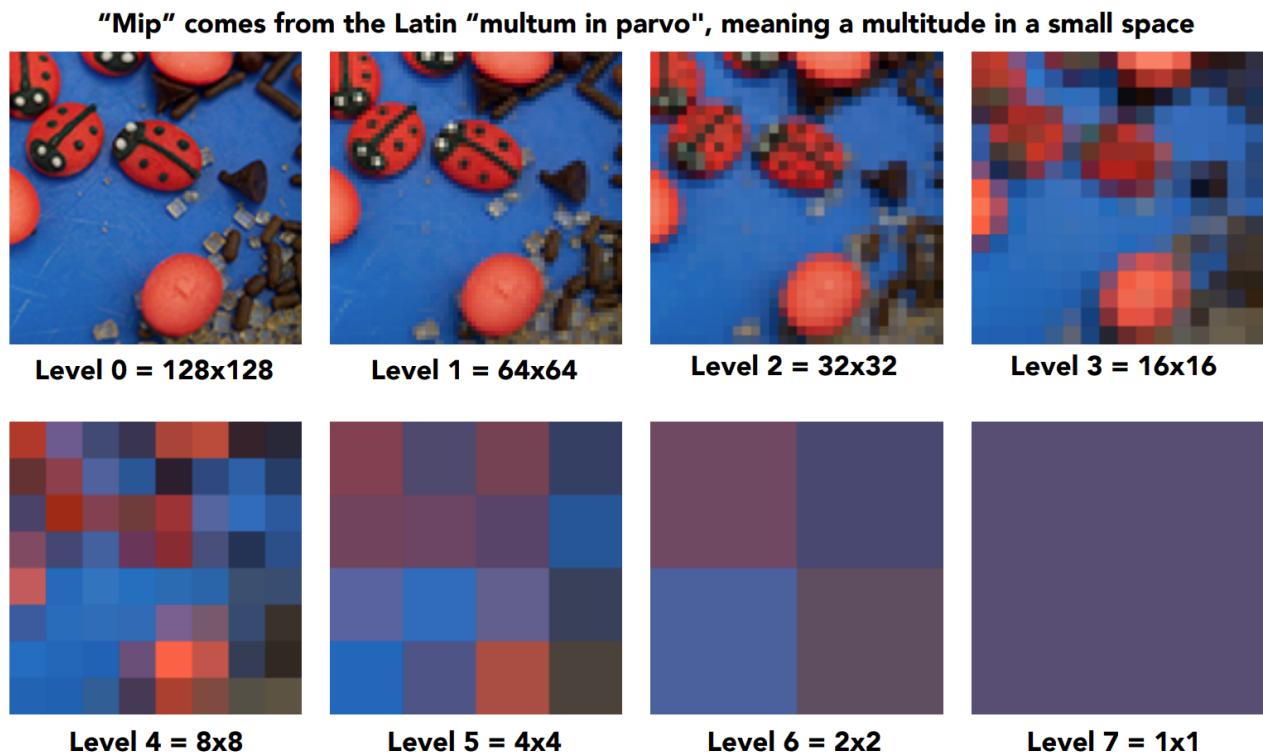
- 求出每个像素覆盖的纹素平均值，在离摄像机不同远近的距离，取固定范围内纹素的均值作为采样结果。**空间换时间**

3. Mipmap

- 允许**快速，近似，方形**的范围查询
- 可以从一张图生成多张图片，逐次降低分辨率（长宽逐次减半）
- Mipmap 计算，将屏幕空间和纹理空间两者中相同的边予以近似，或者是映射的形状予以近似。

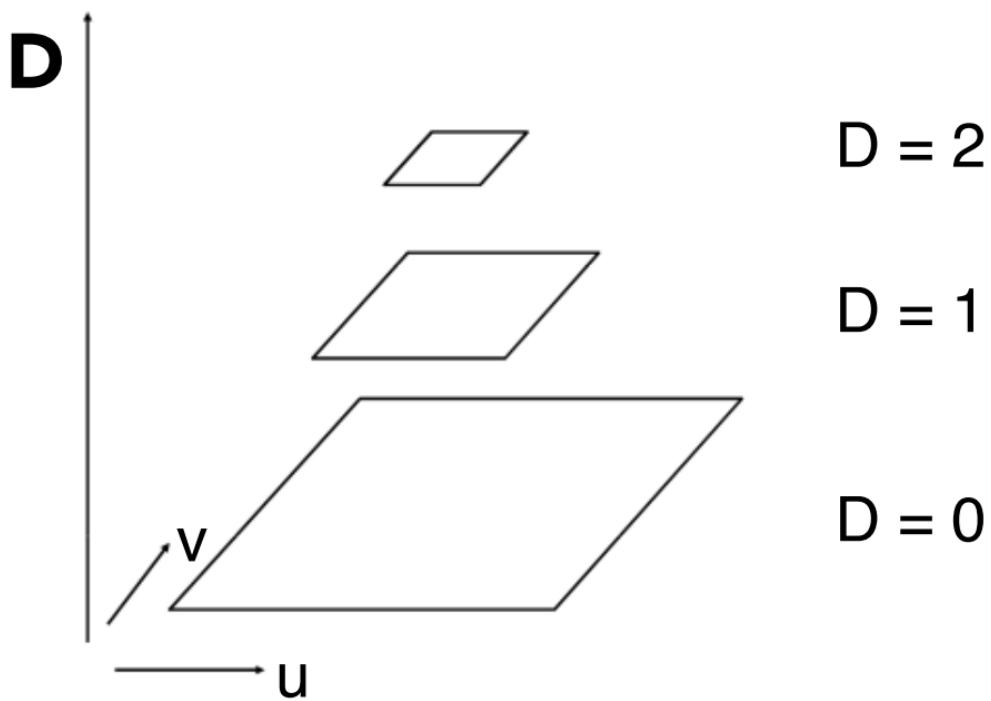
Mipmap 可以快速对一个正方形内部进行纹理近似查询，**并且只能对正方形范围**进行纹理插值。

Mipmap 将原始的纹理图不断进行下采样，每次分辨率大小变为 $1/4$ ，最后只剩下 1×1 的分辨率



通过上面的操作，可以获得不同层次的纹理图，其所占消耗的空间是原图像的 $1/3$ 大小

$$Y = 1/4 + (1/4)^2 + (1/4)^3 + \dots \approx 1/3$$

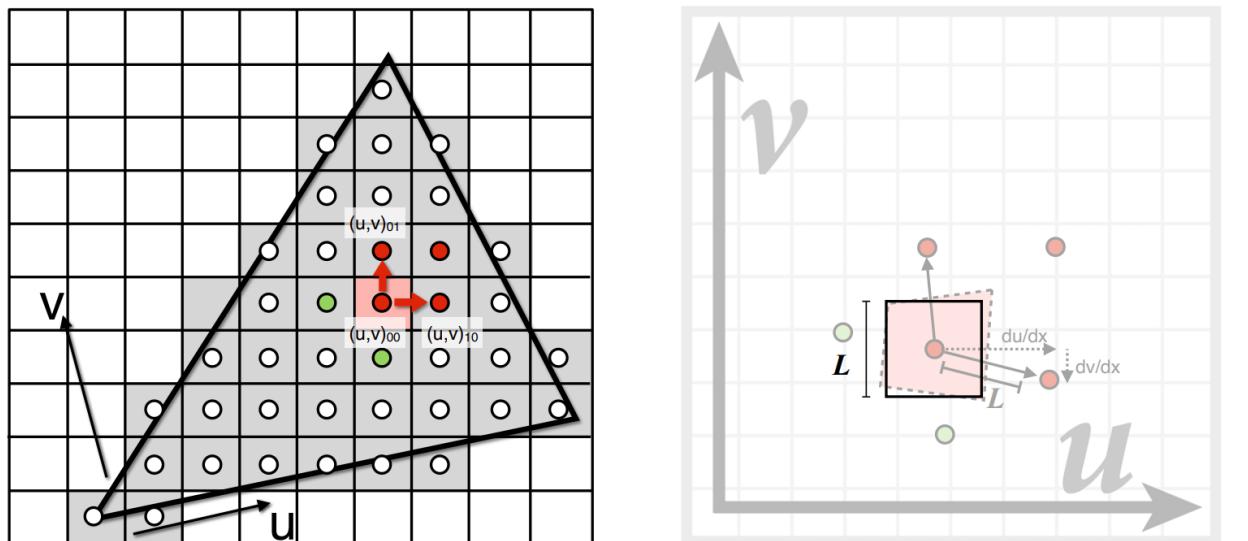


“Mip hierarchy”
level = D

在选择像素点对应 Mipmap 的层数时，需要计算它与相邻像素点（此时取2个/4个没有影响）的距离 L ，即在纹理图中覆盖的像素个数

$$L = \max\left(\sqrt{\left(\frac{du}{dx}\right)^2 + \left(\frac{dv}{dx}\right)^2}, \sqrt{\left(\frac{du}{dy}\right)^2 + \left(\frac{dv}{dy}\right)^2}\right)$$

此时得出了该点覆盖纹素的个数 L ，而对应 Mipmap 的层数也可以根据 $D = \log_2 L$ 求出



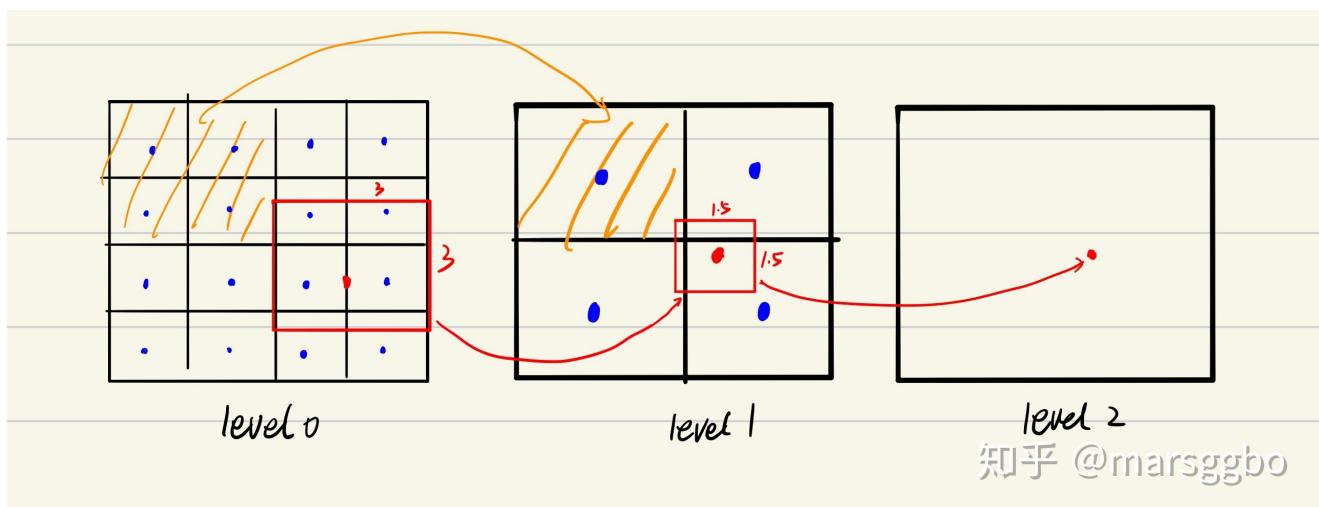
$$D = \log_2 L \quad L = \max \left(\sqrt{\left(\frac{du}{dx} \right)^2 + \left(\frac{dv}{dx} \right)^2}, \sqrt{\left(\frac{du}{dy} \right)^2 + \left(\frac{dv}{dy} \right)^2} \right)$$

对于非整数的 L ，一般有两种方法

- 将非整数近似成整数
- 将邻近的两层做线性插值

4.三线性插值算法 (Trilinear interpolation)

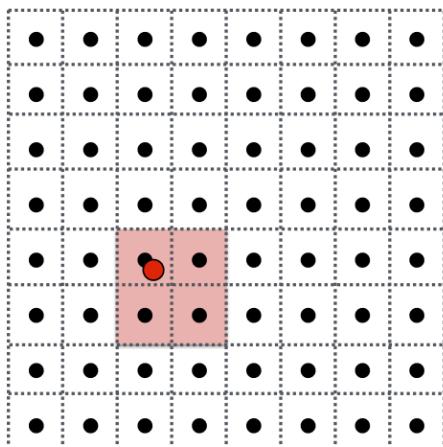
如图， $L=3$ 时， $D=1.58$ ，非整数



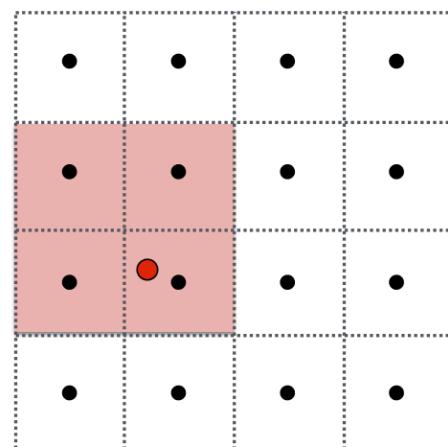
而 D 值位于 level 1 与 level 2 之间，其像素覆盖的大小大于 level 1 层单个纹素，但又小于 level 2 层的单个纹素，这时方法为

- 在两个 level 上做双线性插值求出红点的值
- 在层与层之间做插值

- 层与层之间的插值，是根据 (u, v) 坐标而做的，每一层的纹理图都会归一化到 $(0, 1)$ ，所以层与层之间的插值将两个 (u, v) 坐标进行插值即可。



Mipmap Level D



Mipmap Level D+1

Bilinear result

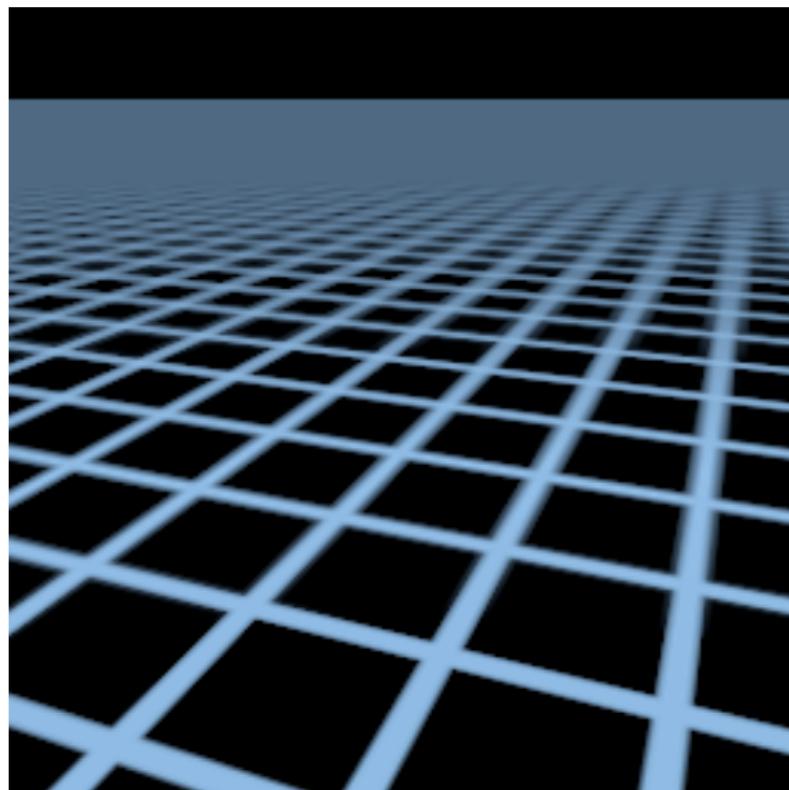
Bilinear result

Linear interpolation based on continuous D value

5. Anisotropic Filtering 各向异性过滤

Overblur

Why?



Mipmap trilinear sampling

Mipmap 在近处的插值效果较好，但到了远处，因为其方格不再近似正方形，Mipmap 的正方形格近似就会产生很大的误差，这时可以使用 **各向异性过滤** 来解决问题。

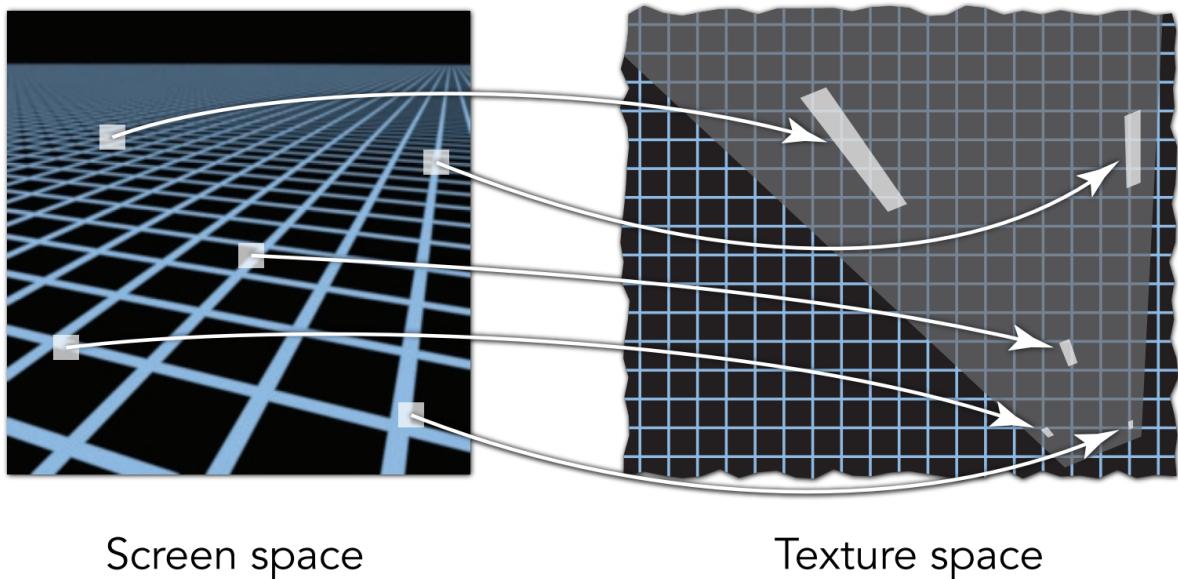
- 各向异性过滤会在每一行/每一列上压缩长/宽，达到能够符合该像素覆盖范围的几何形状，即下图

-



Wikipedia

- 再去近似像素包含纹素的范围，效果会变好



- 而使用各向异性过滤的方法又称为 **Ripmap**，能解决矩形问题
- EWA Filtering** 使用椭圆形近似不规则的形状。