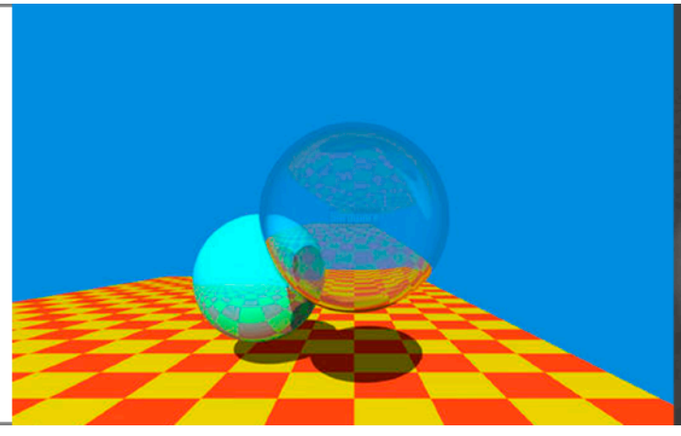
Introduction to Computer Graphics

GAMES101, Lingqi Yan, UC Santa Barbara

Lecture 9:
Shading 3 (Texture Mapping cont.)






http://www.cs.ucsb.edu/~lingqi/teaching/games101.html

Announcements

- About homework
  - Homework 1 is being graded
  - Homework 2
    - 271 submissions so far
  - Homework 3 will be released soon

Last Lectures

- Shading 1 & 2
  - Blinn-Phong reflectance model
  - Shading models / frequencies
  - Graphics Pipeline
  - Texture mapping

Today

- Shading 3
  - Barycentric coordinates
  - Texture queries
  - Applications of textures
- Shadow mapping

Interpolation Across Triangles: Barycentric Coordinates

(重心坐标)

Interpolation Across Triangles

Why do we want to interpolate?

- Specify values **at vertices**
- Obtain smoothly varying values **across triangles**

What do we want to interpolate?

- Texture coordinates, colors, normal vectors, …

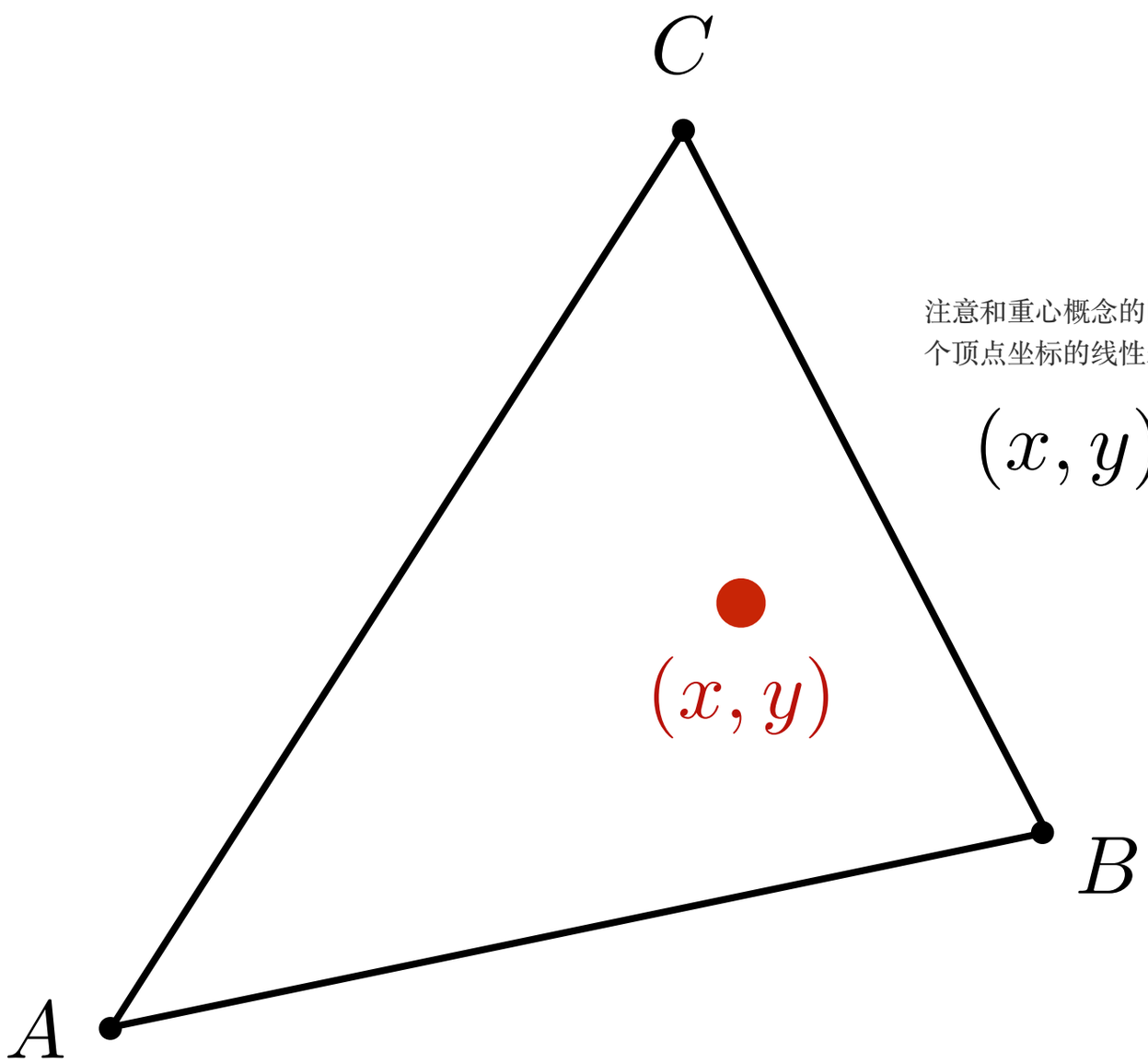
How do we interpolate?

- **Barycentric coordinates**

Barycentric Coordinates

该点的重心坐标，必须满足大于 0 才能保证在三角形内

A coordinate system for triangles (α, β, γ)



注意和重心概念的区别！对于三角形所在平面上的任意一点的坐标，都可以用三角形的三个顶点坐标的线性表达式表示

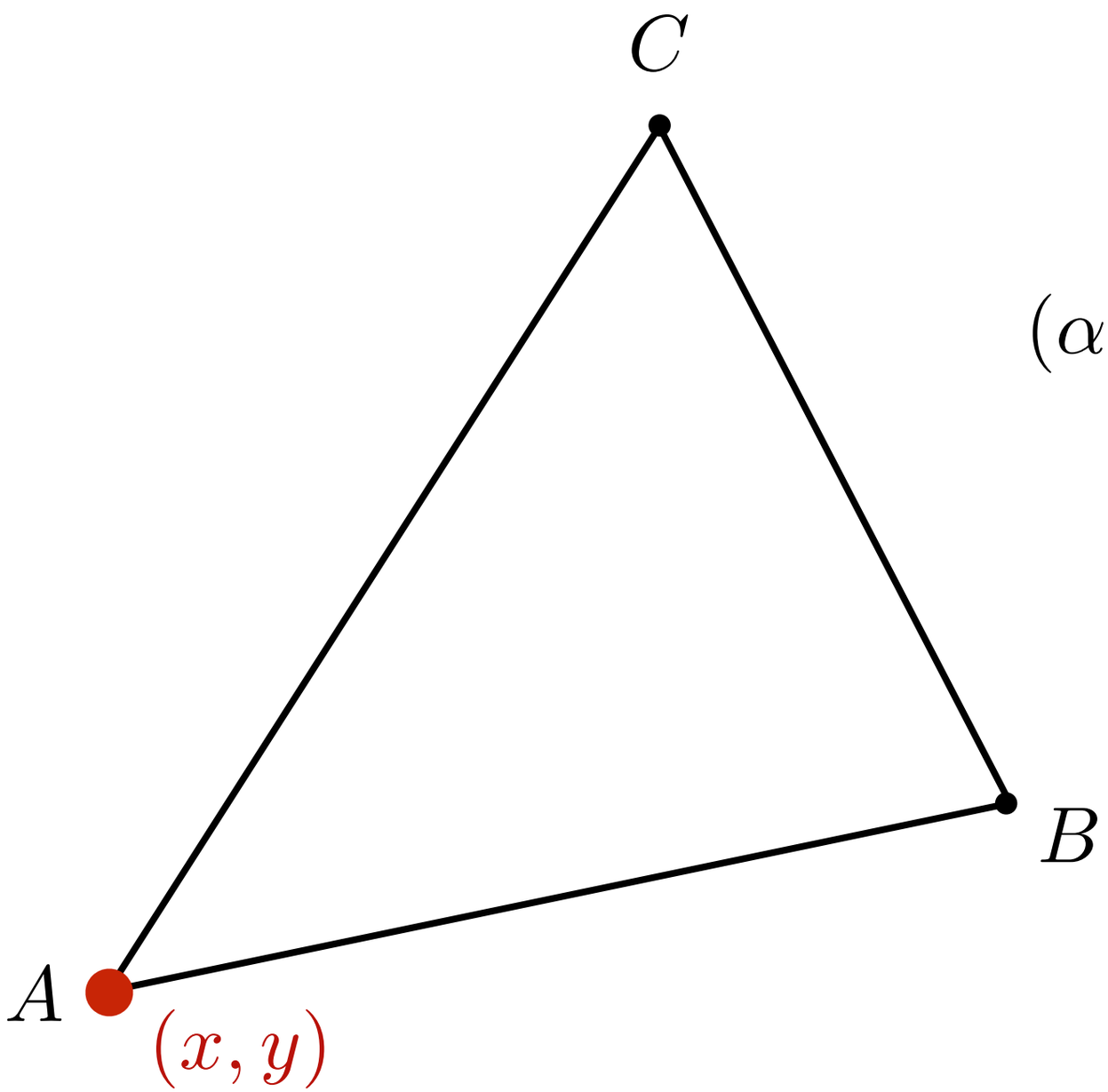
$$(x, y) = \alpha A + \beta B + \gamma C$$

$$\alpha + \beta + \gamma = 1$$

Inside the triangle if all three coordinates are non-negative

Barycentric Coordinates

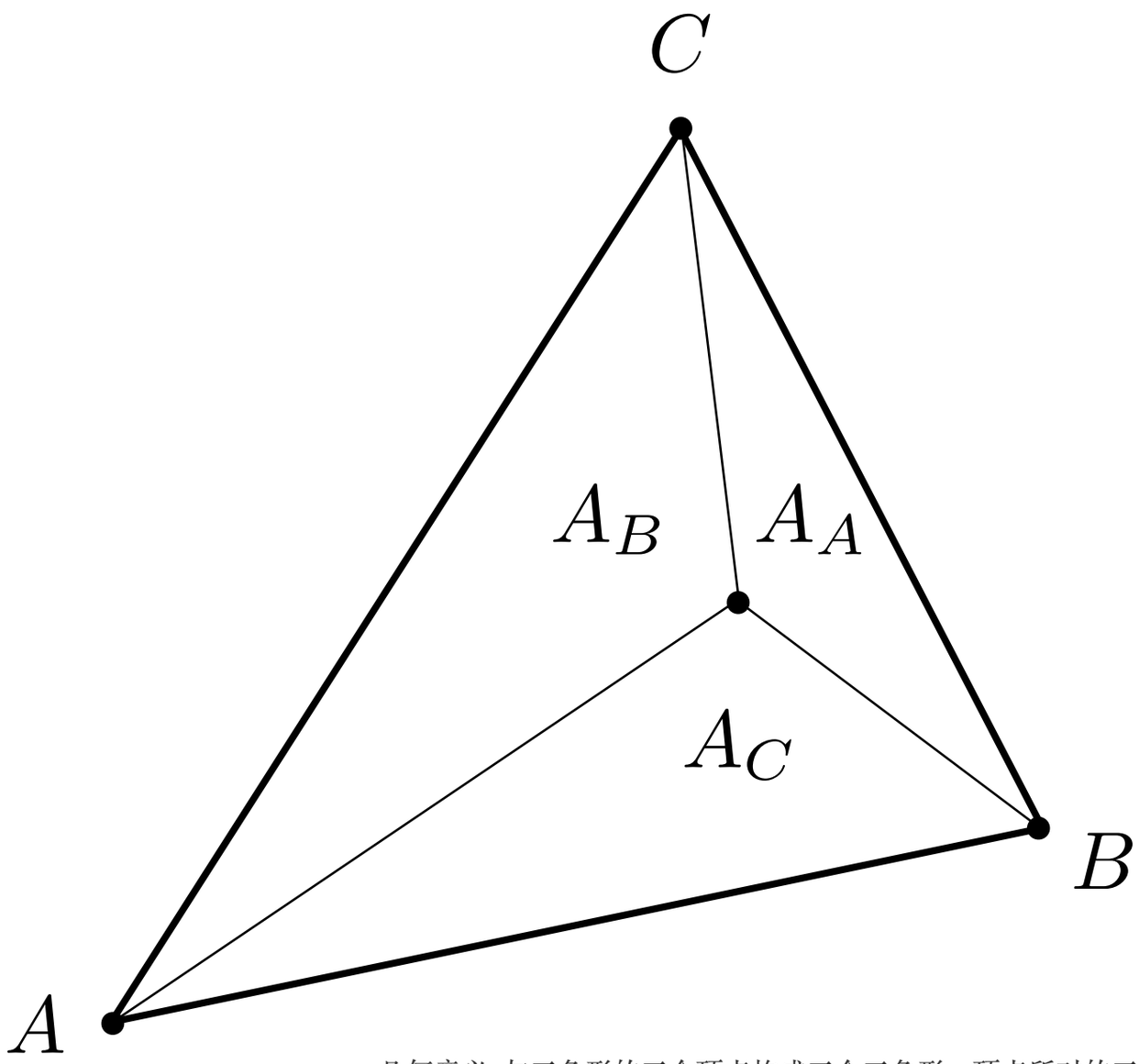
What's the barycentric coordinate of A?



$$(\alpha, \beta, \gamma) = (1, 0, 0)$$
$$(x, y) = \alpha\, A + \beta\, B + \gamma\, C$$
$$= A$$

Barycentric Coordinates

Geometric viewpoint — proportional areas



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

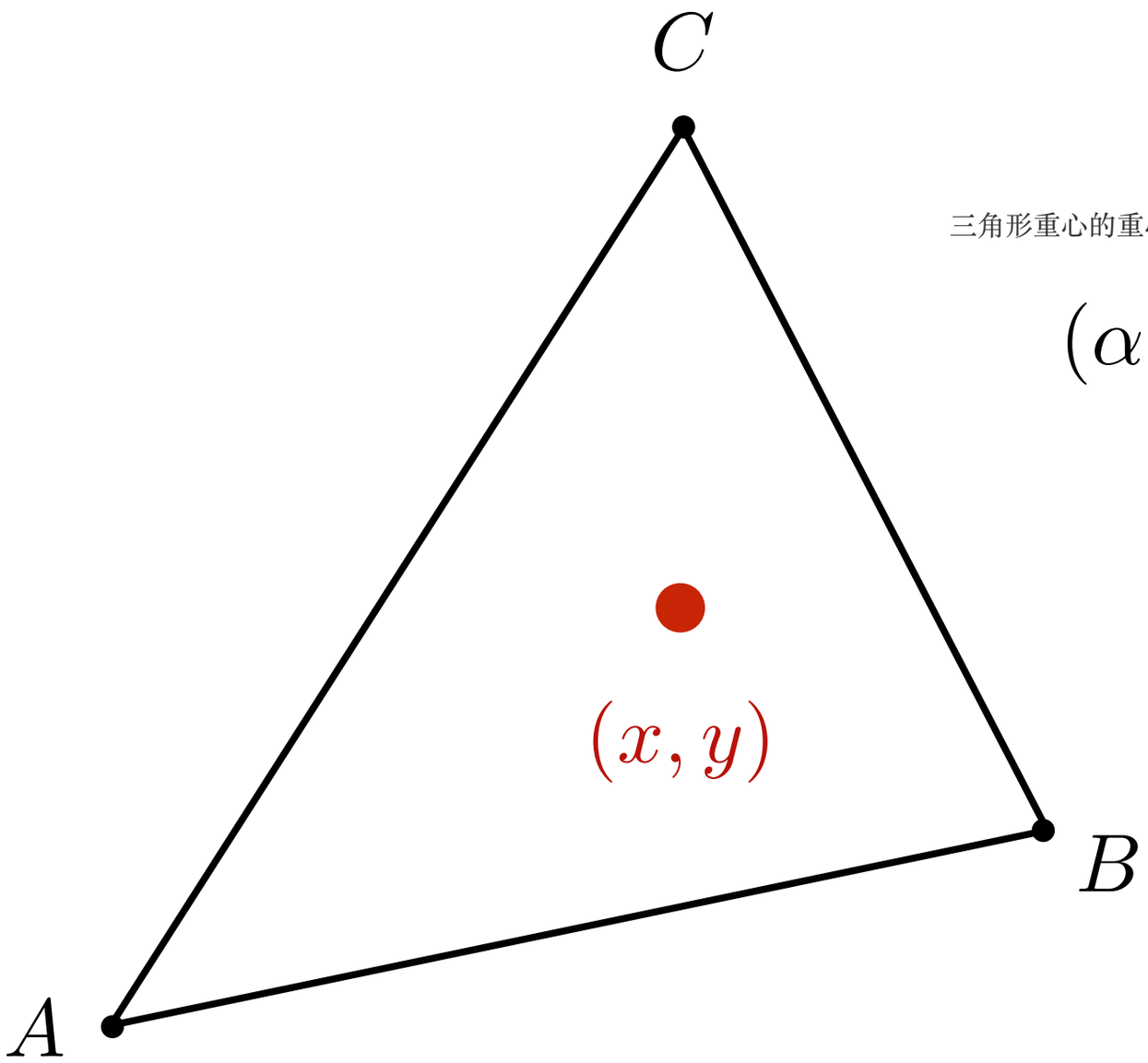
$$\beta = \frac{A_B}{A_A + A_B + A_C}$$

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

几何意义: 与三角形的三个顶点构成三个三角形，顶点所对的三角形的面积与三角形总面积的比值，即为对应的重心坐标值

Barycentric Coordinates

What's the barycentric coordinate of the centroid?



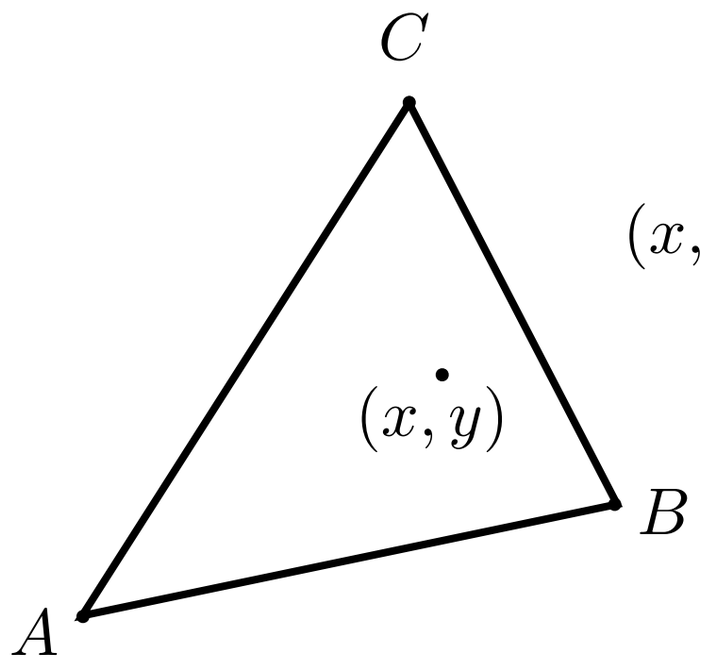
三角形重心的重心坐标

$$(\alpha, \beta, \gamma) = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$$

$$(x, y) = \frac{1}{3}A + \frac{1}{3}B + \frac{1}{3}C$$

Barycentric Coordinates: Formulas

重心坐标计算公式



$$(x, y) = \alpha\, A + \beta\, B + \gamma\, C$$

$$\alpha + \beta + \gamma = 1$$

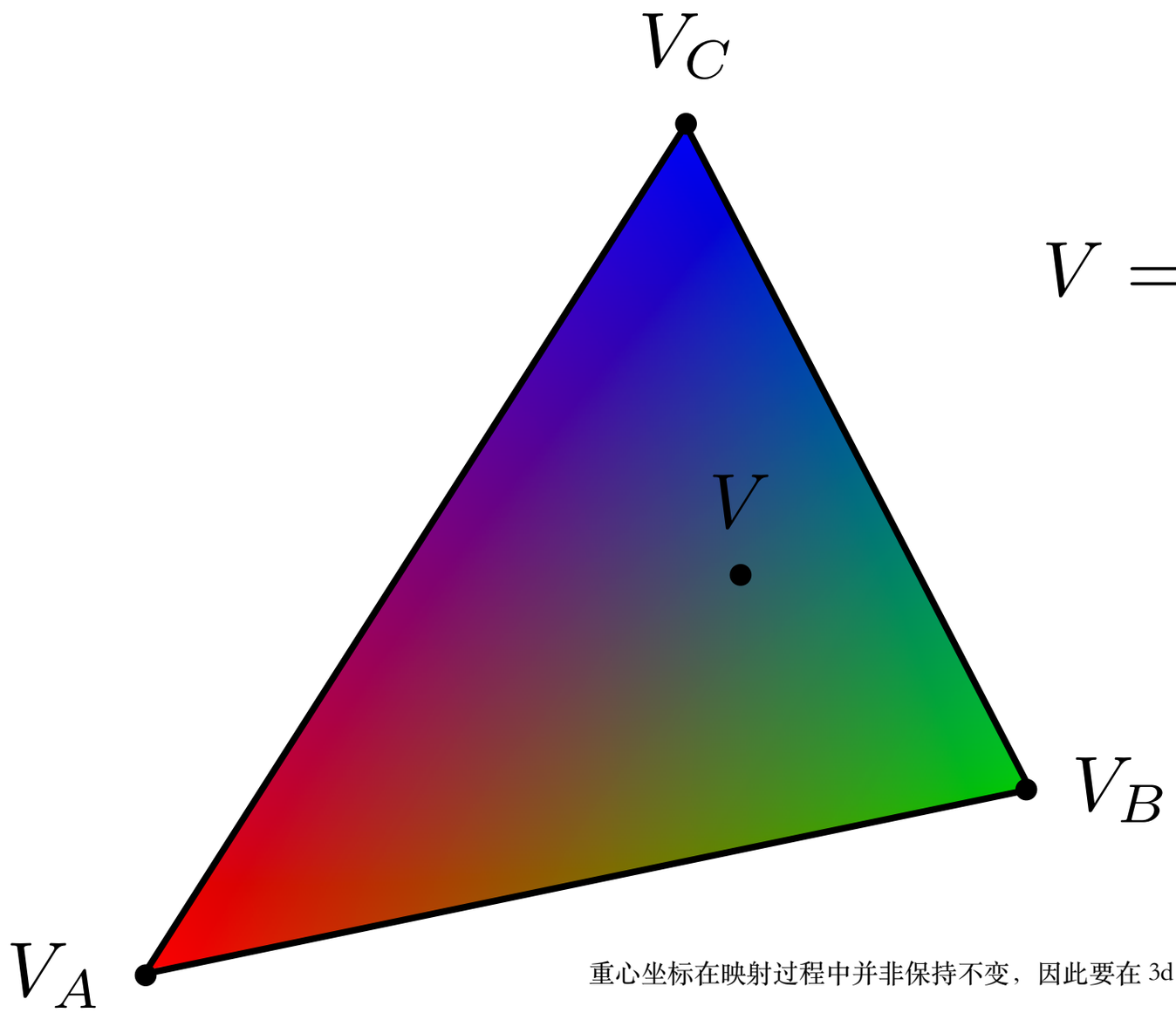
$$\alpha = \frac{-(x - x_B)(y_C - y_B) + (y - y_B)(x_C - x_B)}{-(x_A - x_B)(y_C - y_B) + (y_A - y_B)(x_C - x_B)}$$

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

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

Using Barycentric Coordinates

Linearly interpolate values at vertices



$$V = \alpha V_A + \beta V_B + \gamma V_C$$

V_A, V_B, V_C can be positions, texture coordinates, color, normal, depth, material attributes…

重心坐标在映射过程中并非保持不变，因此要在 3d 空间内插值

However, barycentric coordinates are not invariant under projection!

Applying Textures

Simple Texture Mapping: Diffuse Color

纹理映射过程

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;

Using barycentric coordinates!

Usually the diffuse albedo Kd
(recall the Blinn-Phong reflectance model)

Texture Magnification

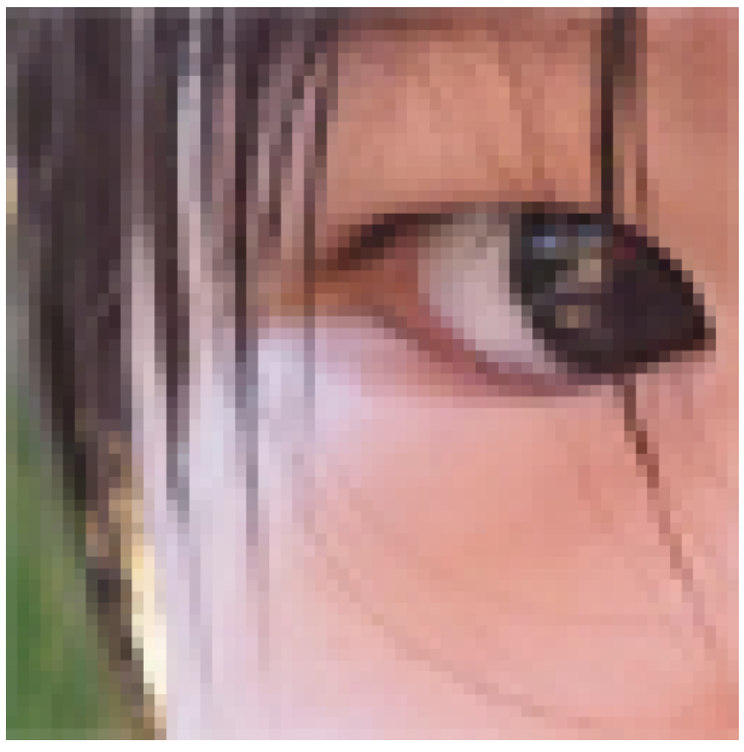
(What if the texture is too small?)

Texture Magnification - Easy Case

纹理太小： 可以理解为多个pixel映射到了同一个纹理

Generally don’t want this — insufficient texture resolution

A pixel on a texture — a **texel** (纹理元素、纹素)

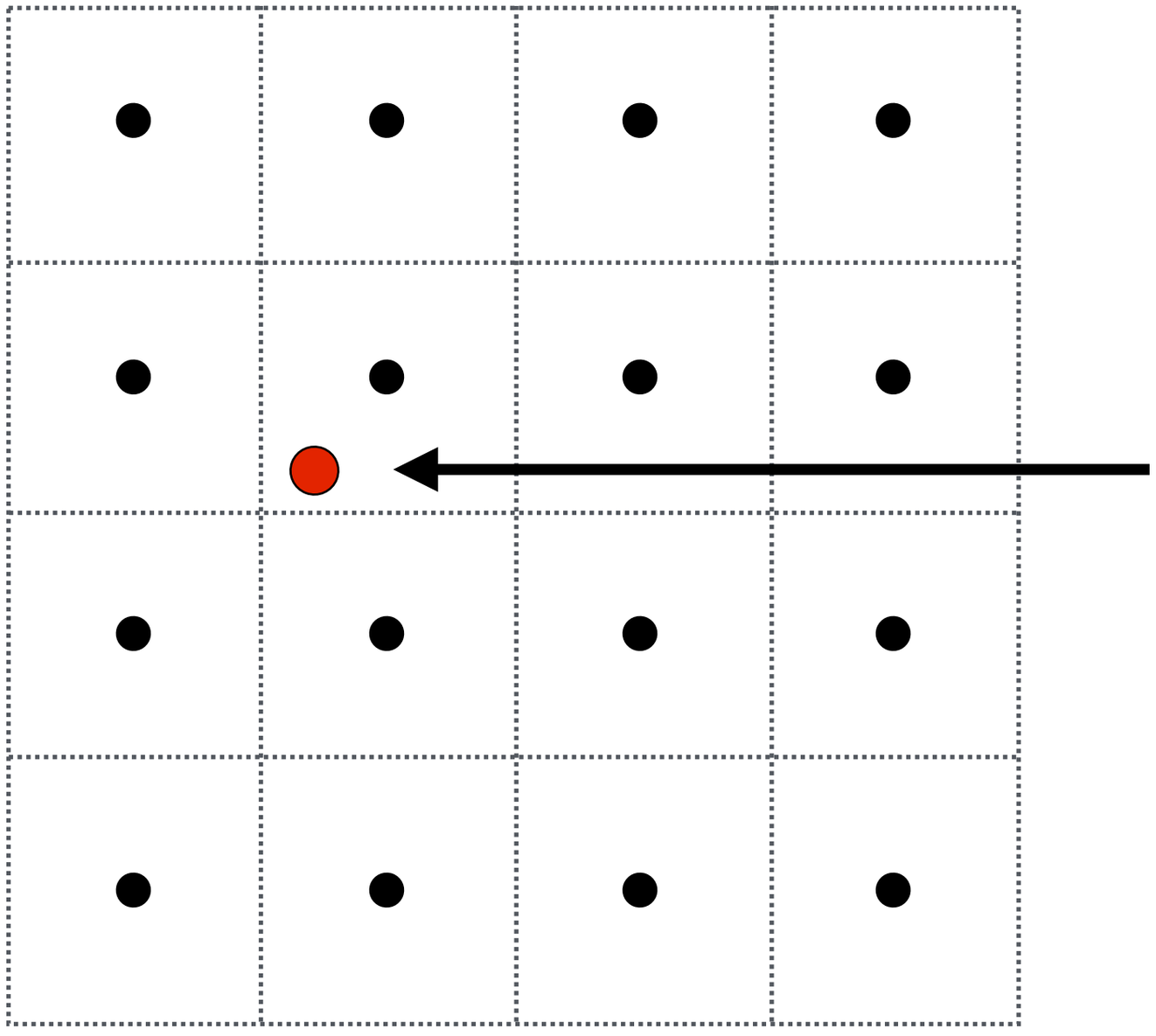


Nearest

Bilinear

Bicubic

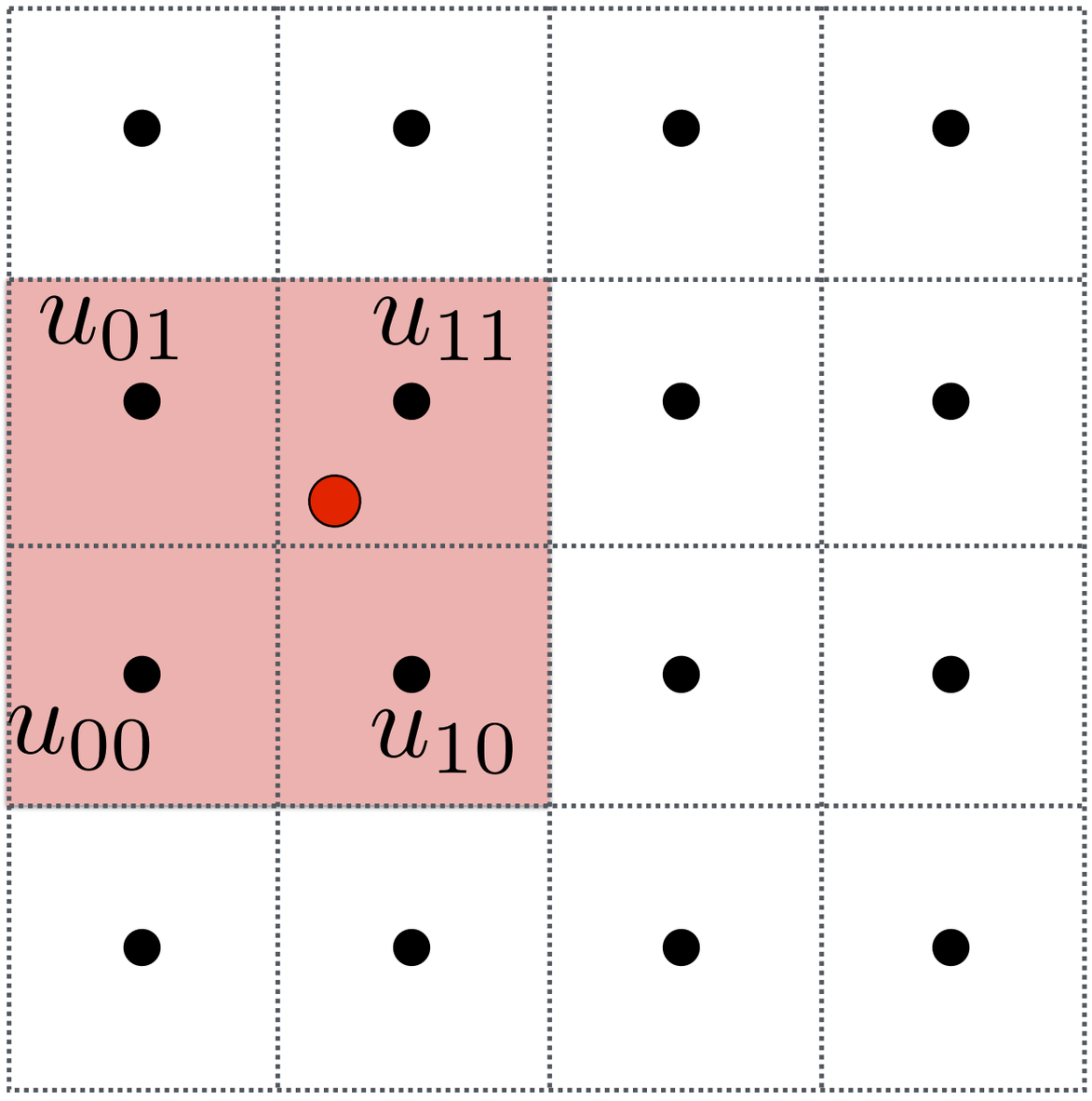
Bilinear Interpolation



Want to sample texture value f(x,y) at red point

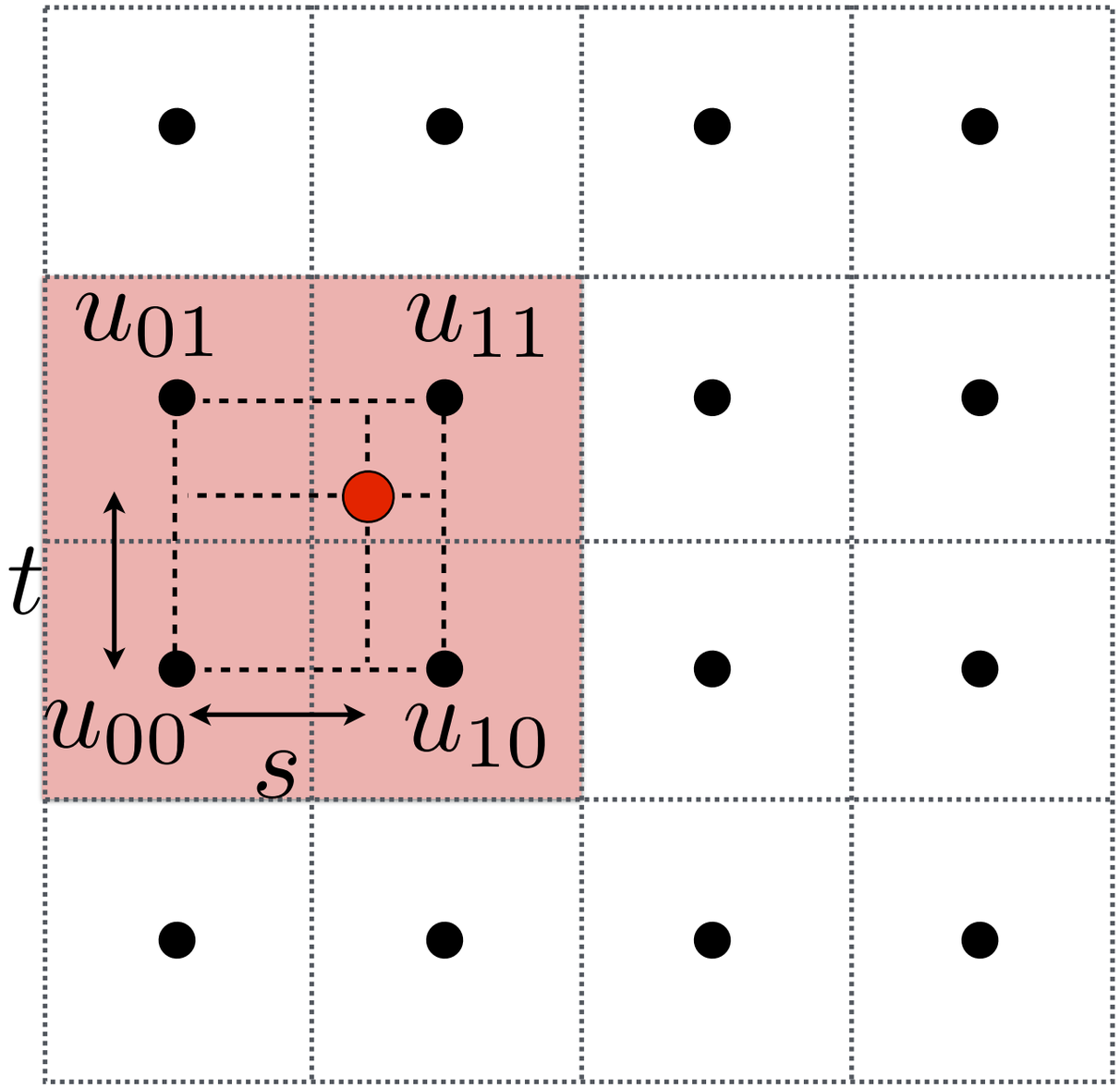
Black points indicate texture sample locations

Bilinear Interpolation



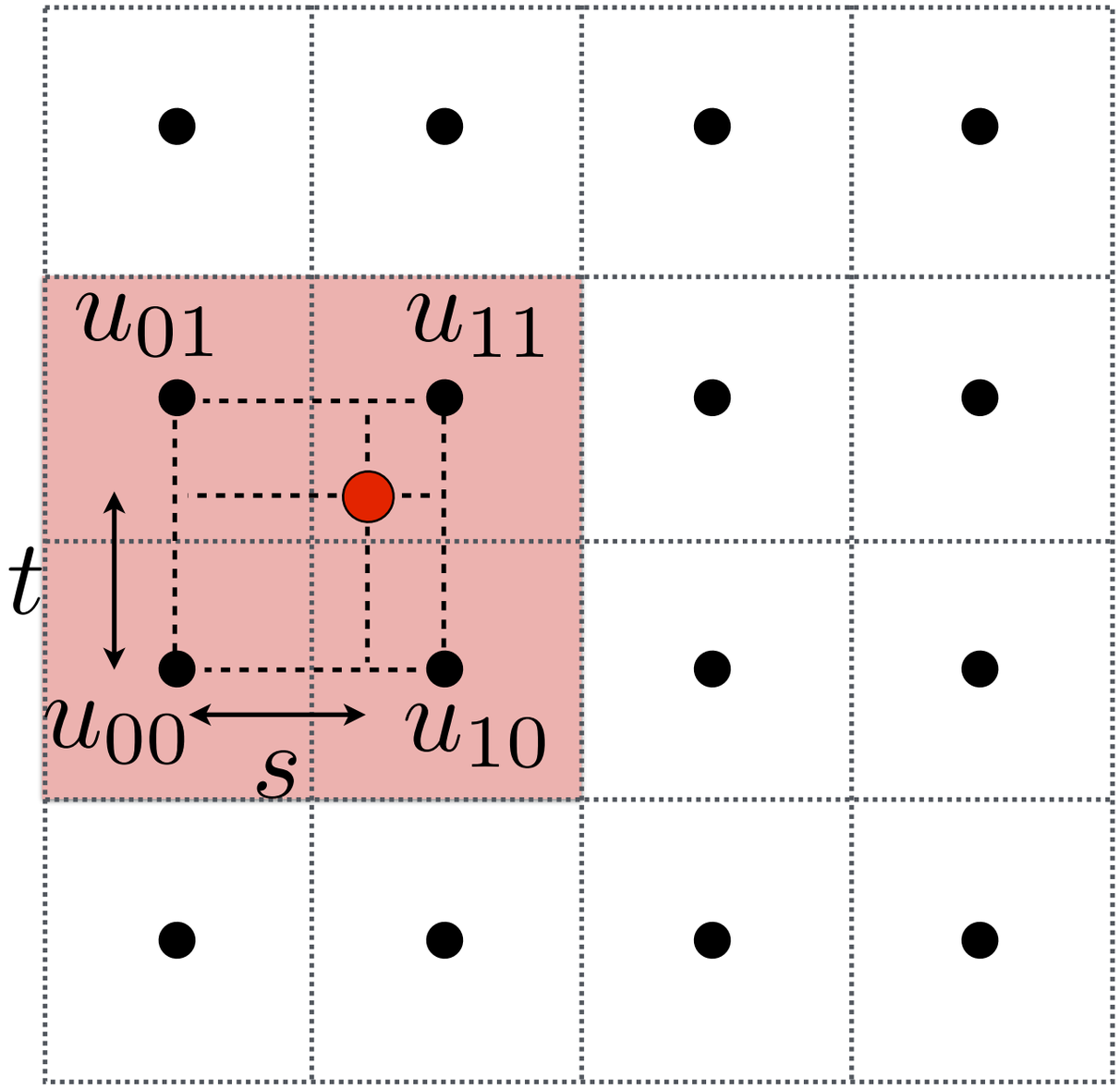
Take 4 nearest sample locations, with texture values as labeled.

Bilinear Interpolation



And fractional offsets, (s,t) as shown

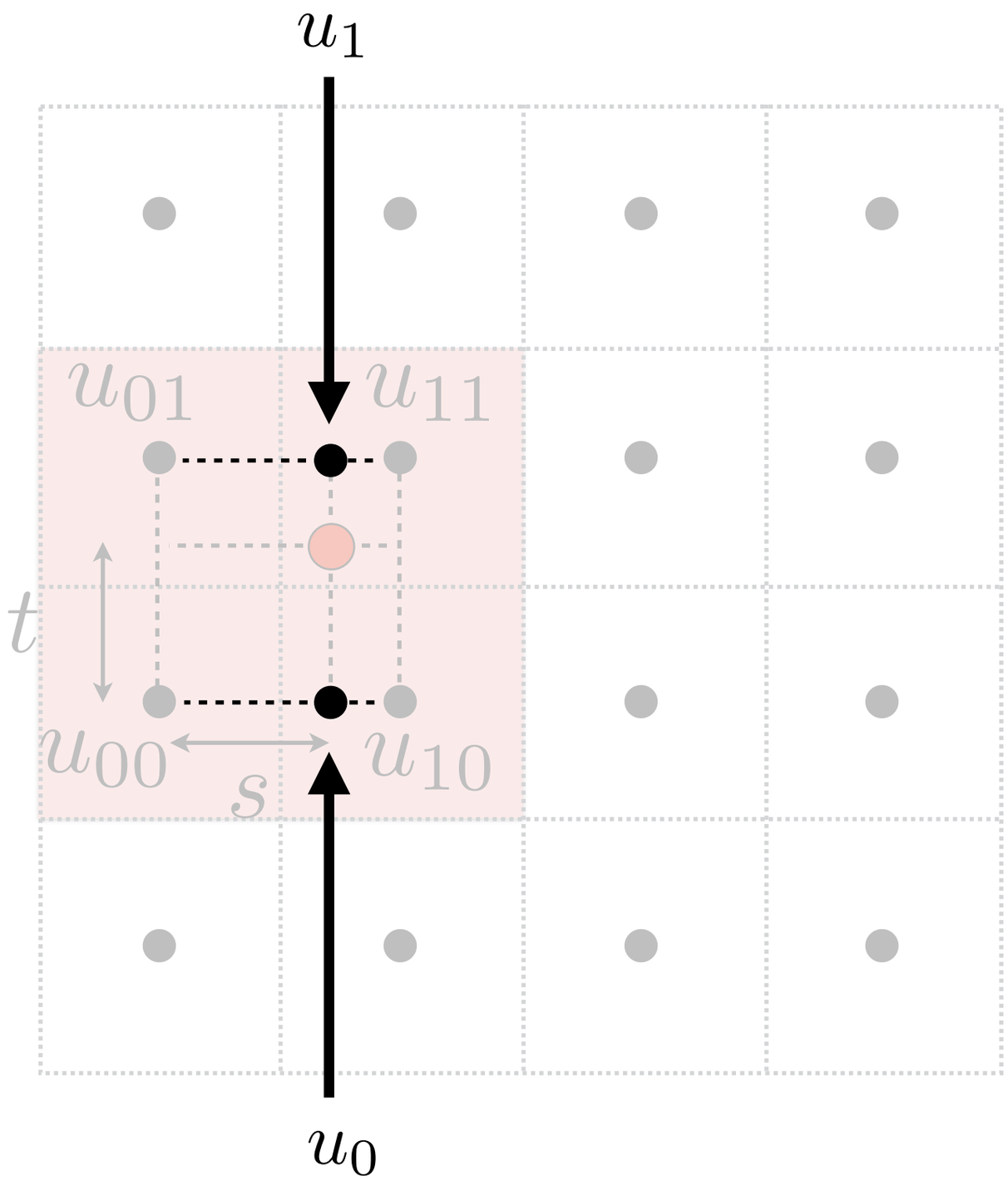
Bilinear Interpolation



Linear interpolation (1D)

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

Bilinear Interpolation



Linear interpolation (1D)

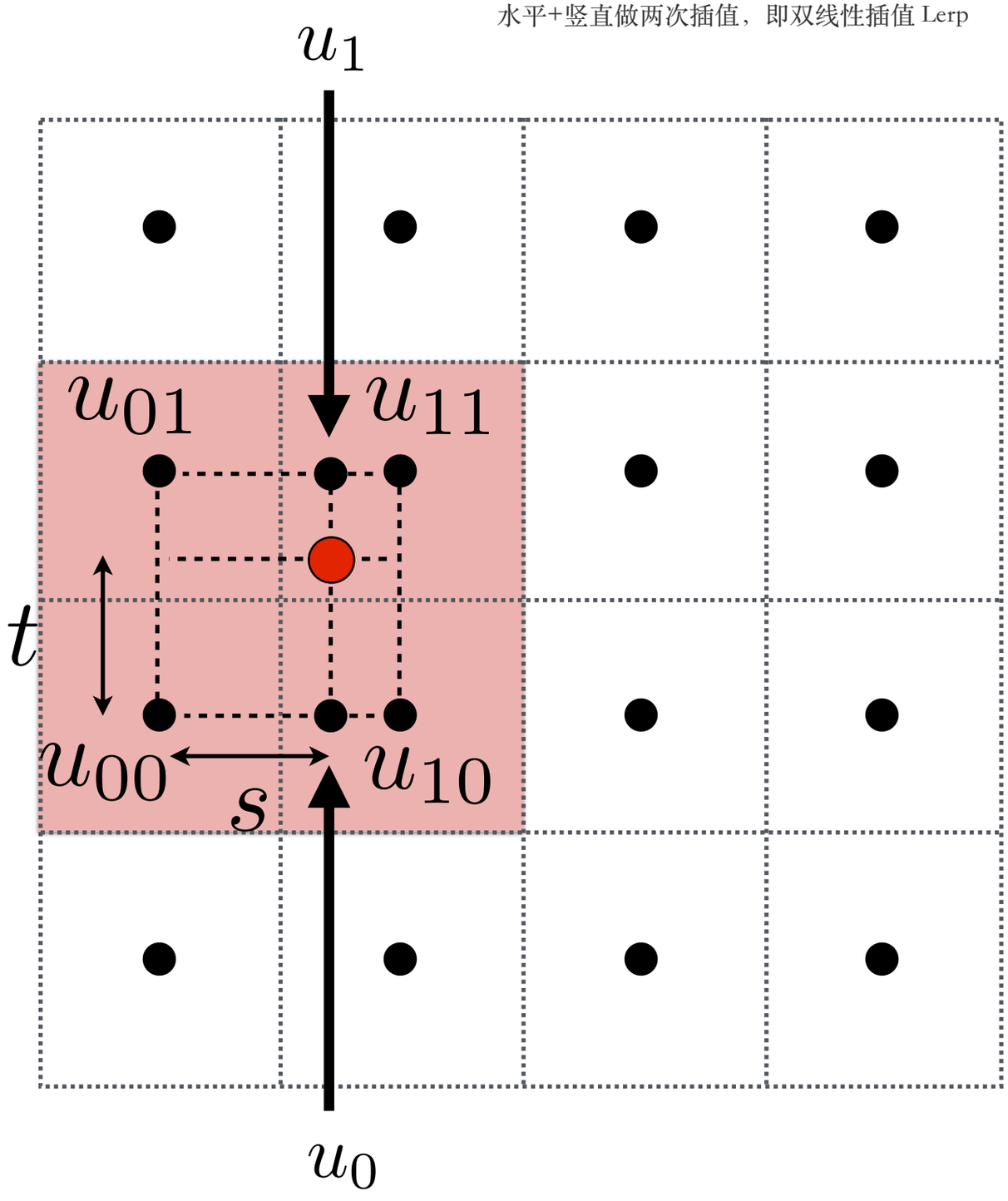
$$\text{lerp}(x, v_0, v_1) = v_0 + x(v_1 - v_0)$$

Two helper lerps (horizontal)

$$u_0 = \text{lerp}(s, u_{00}, u_{10})$$
$$u_1 = \text{lerp}(s, u_{01}, u_{11})$$

Bilinear Interpolation

水平+竖直做两次插值，即双线性插值 Lerp



Linear interpolation (1D)

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

Two helper lerps

$$u_0 = \text{lerp}(s, u_{00}, u_{10})$$

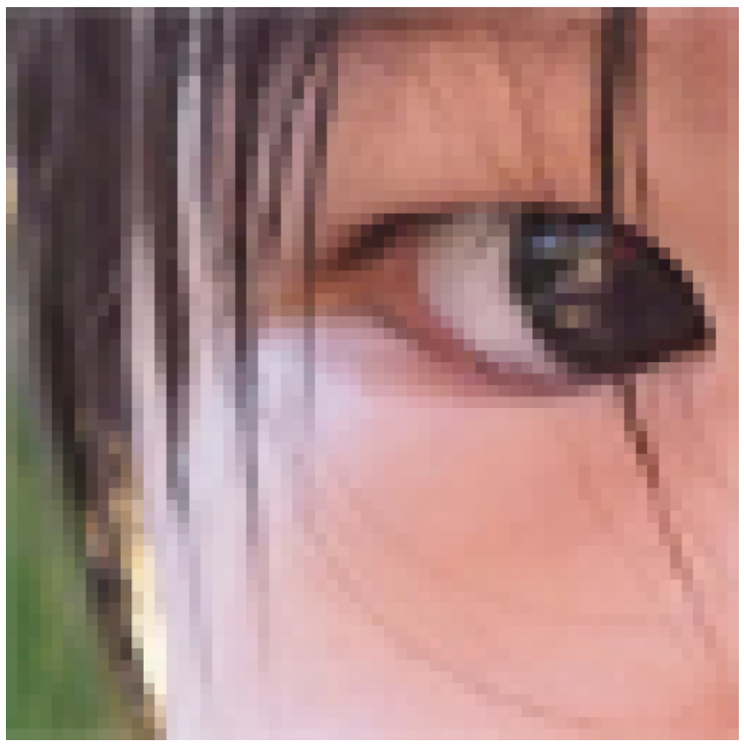
$$u_1 = \text{lerp}(s, u_{01}, u_{11})$$

Final vertical lerp, to get result:

$$f(x, y) = \text{lerp}(t, u_0, u_1)$$

Texture Magnification - Easy Case

Bilinear interpolation usually gives pretty good results at reasonable costs



Nearest

Bilinear

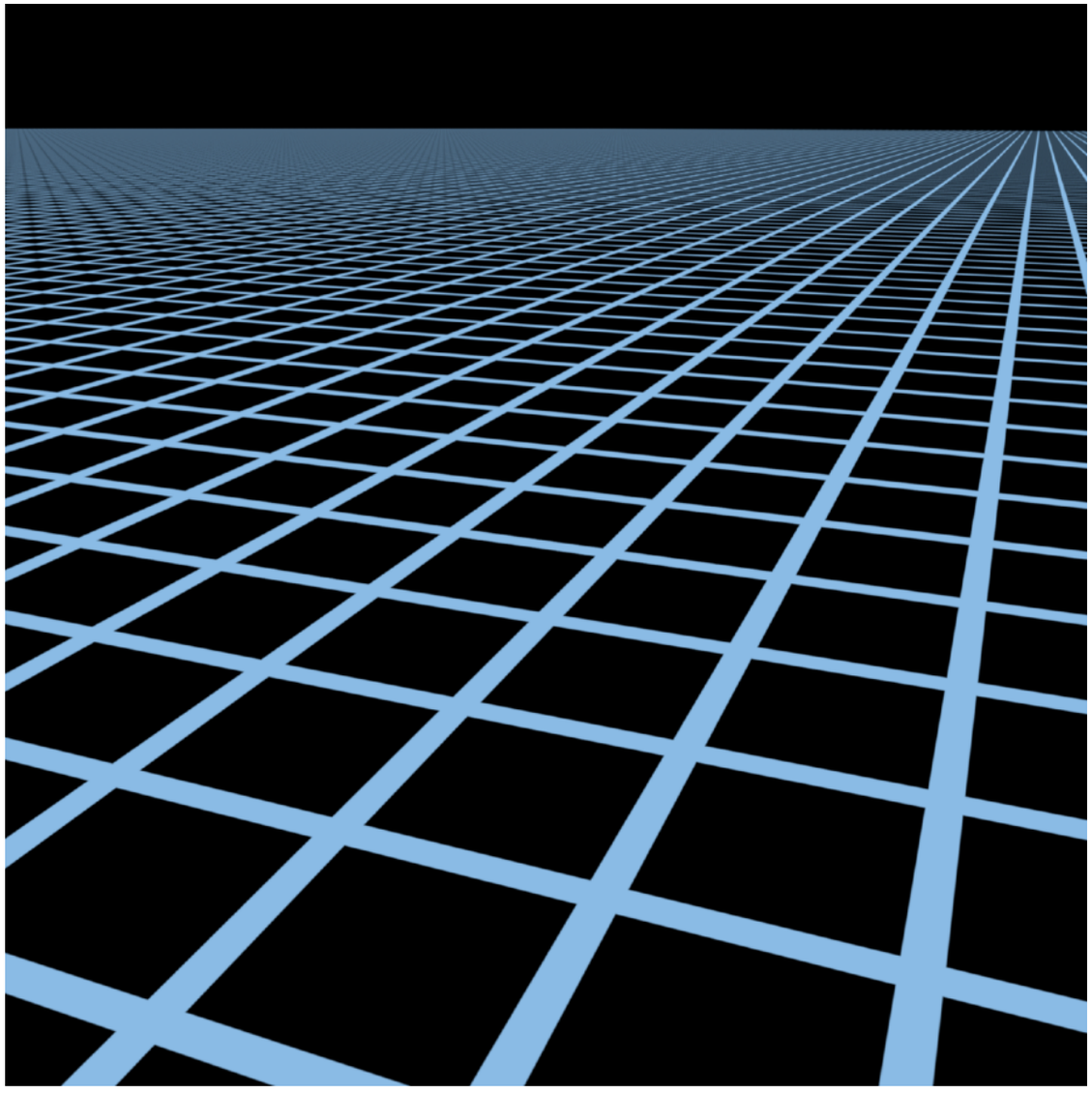
Bicubic

Texture Magnification **(hard case)**

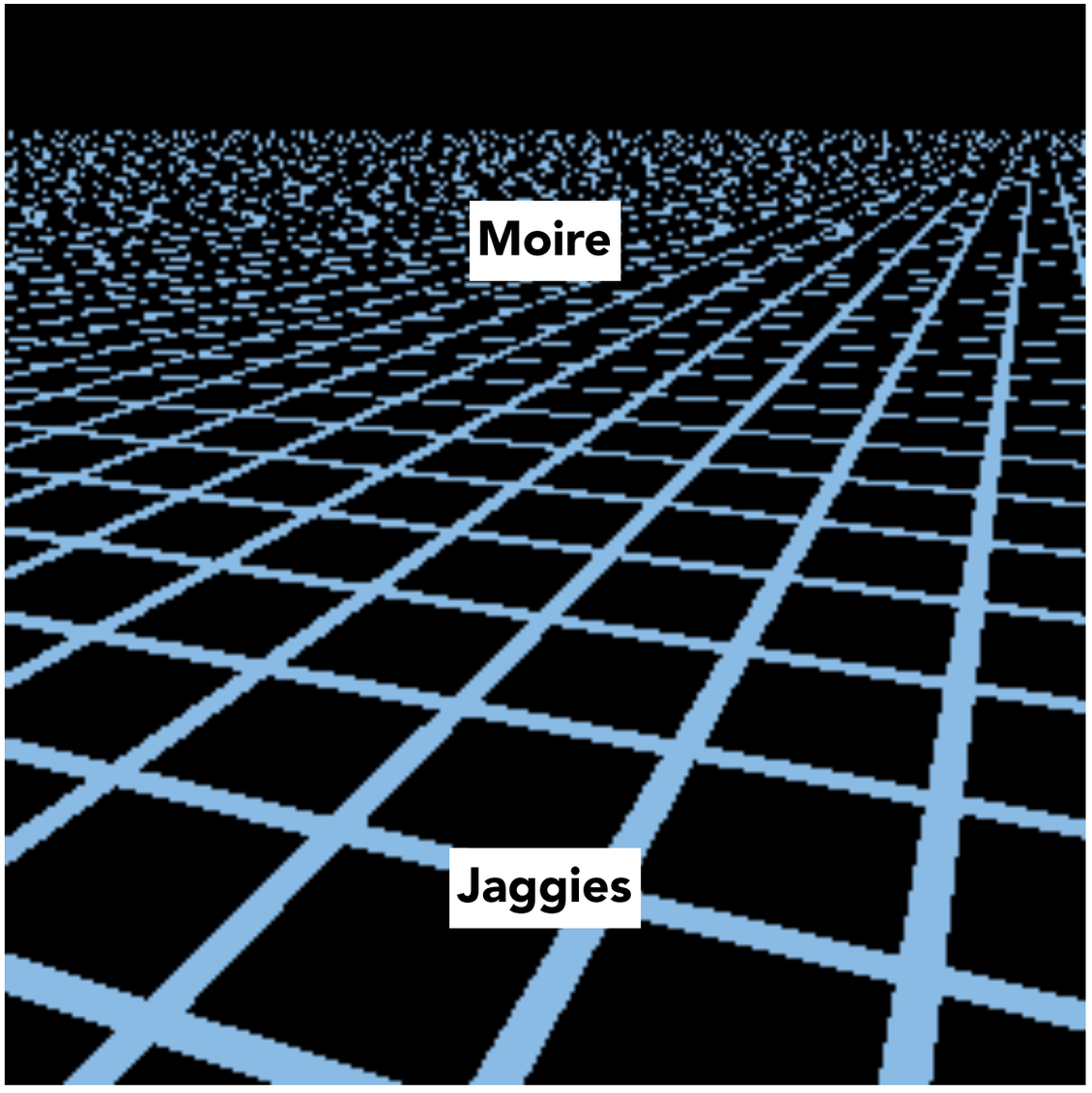
(What if the texture is too large?)

Point Sampling Textures — Problem

纹理太大： 可以理解为一个pixel对应了多个纹素，因采样频率不足而导致摩尔纹+锯齿

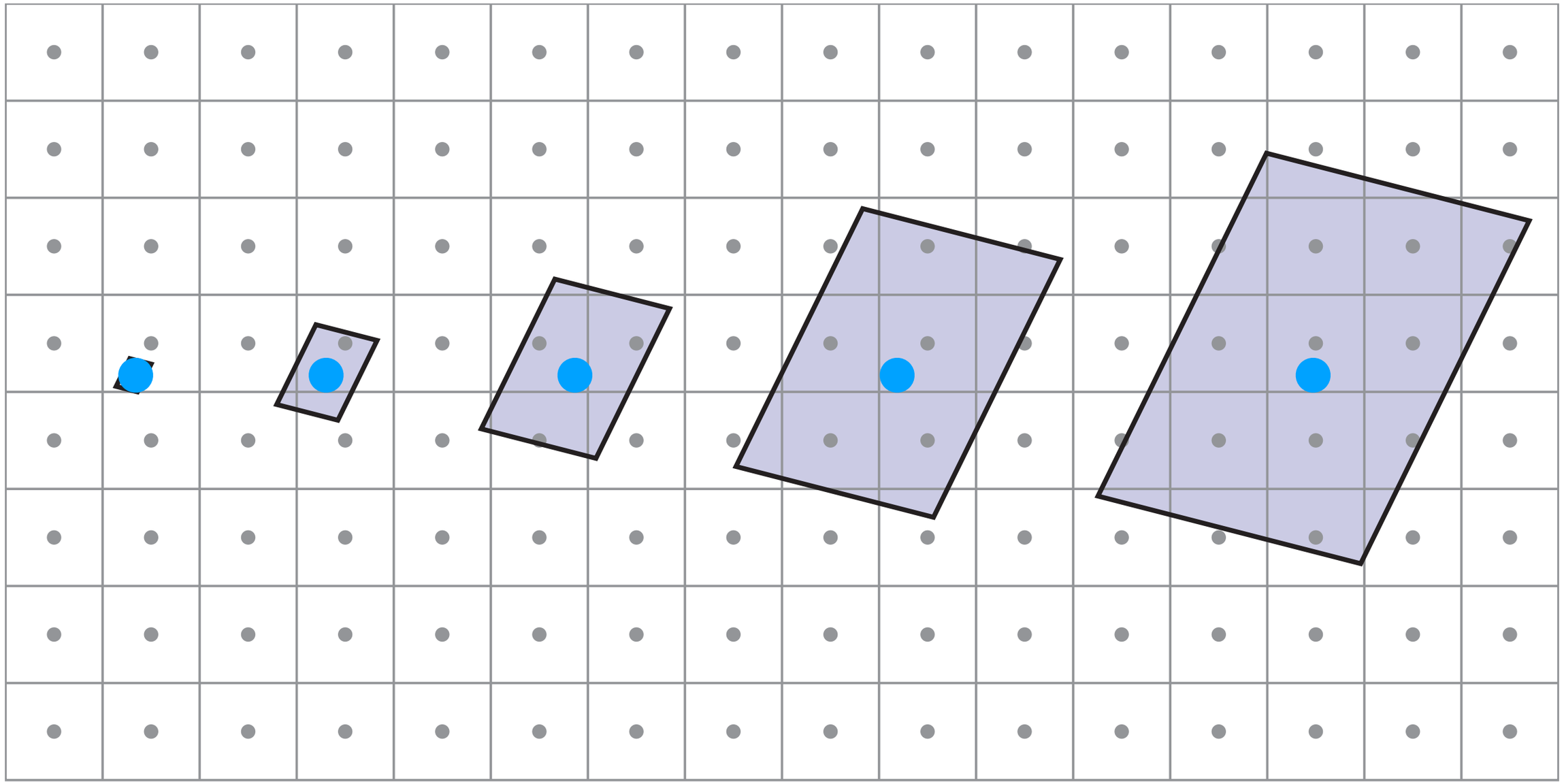


Reference



Point sampled

Screen Pixel "Footprint" in Texture

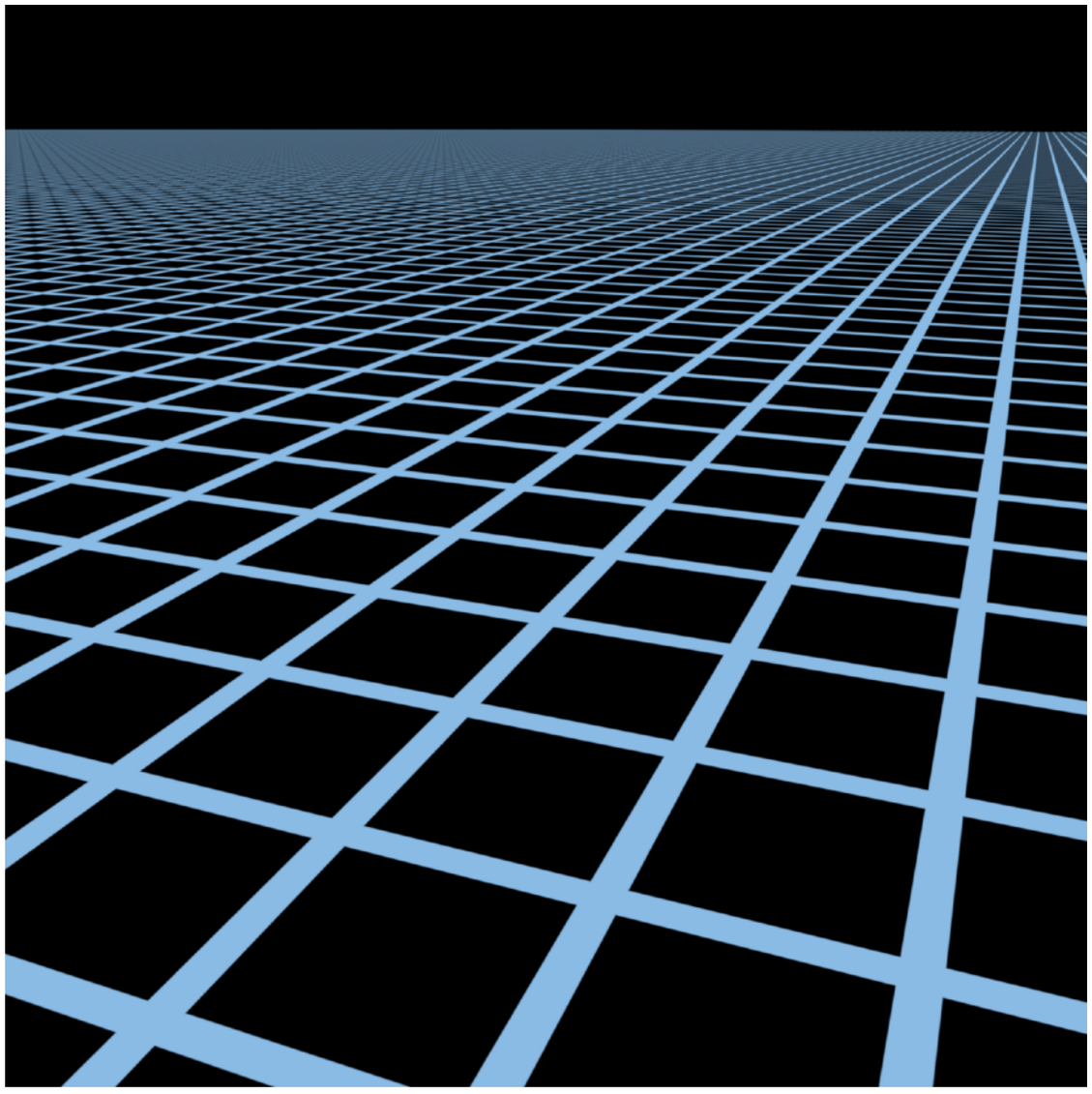


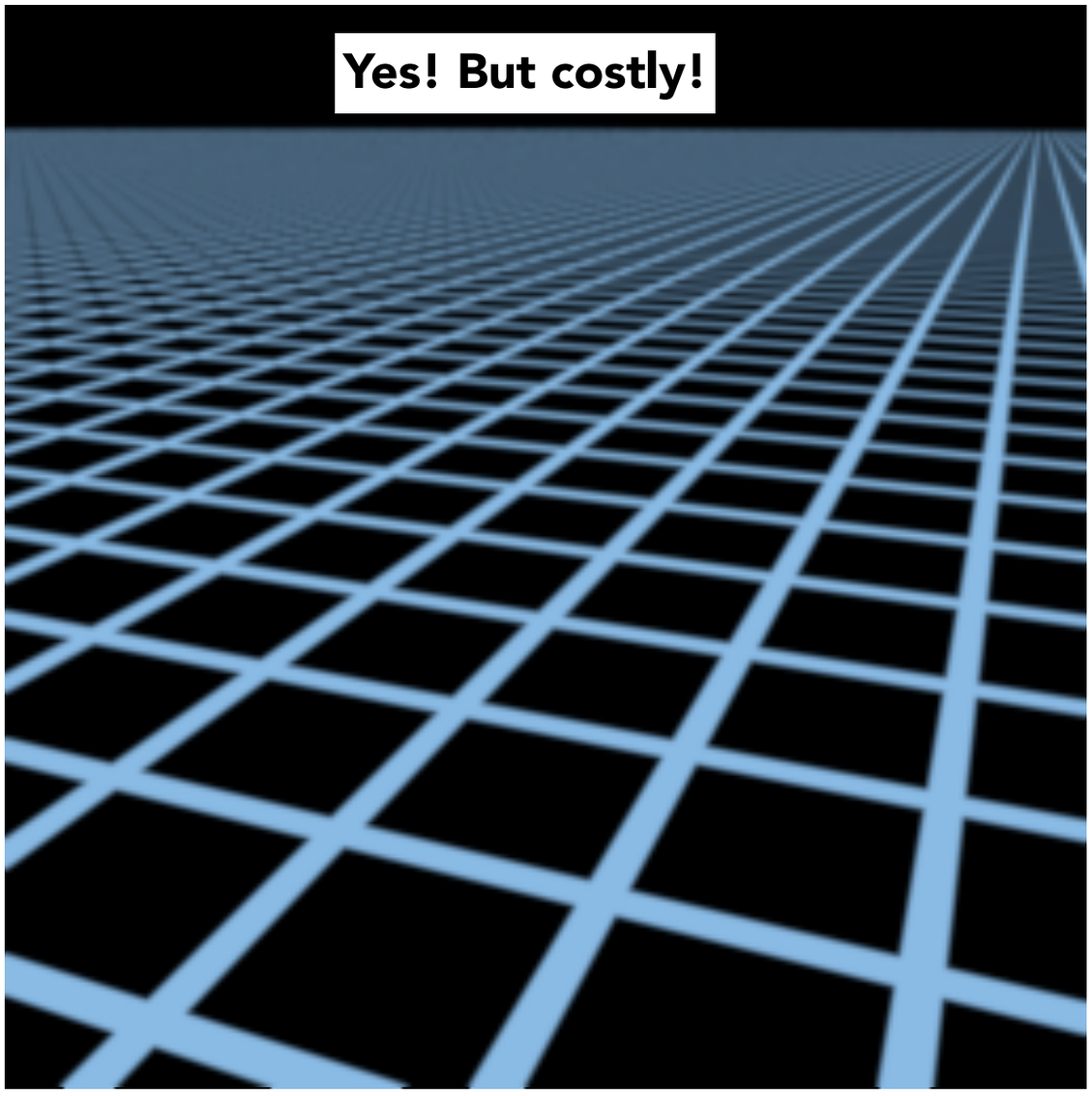
Upsampling (Magnification)

Downsampling (Minification)

Will Supersampling Do Antialiasing?

Supersampling多重采样，性能开销过大





512x supersampling

Antialiasing — Supersampling?

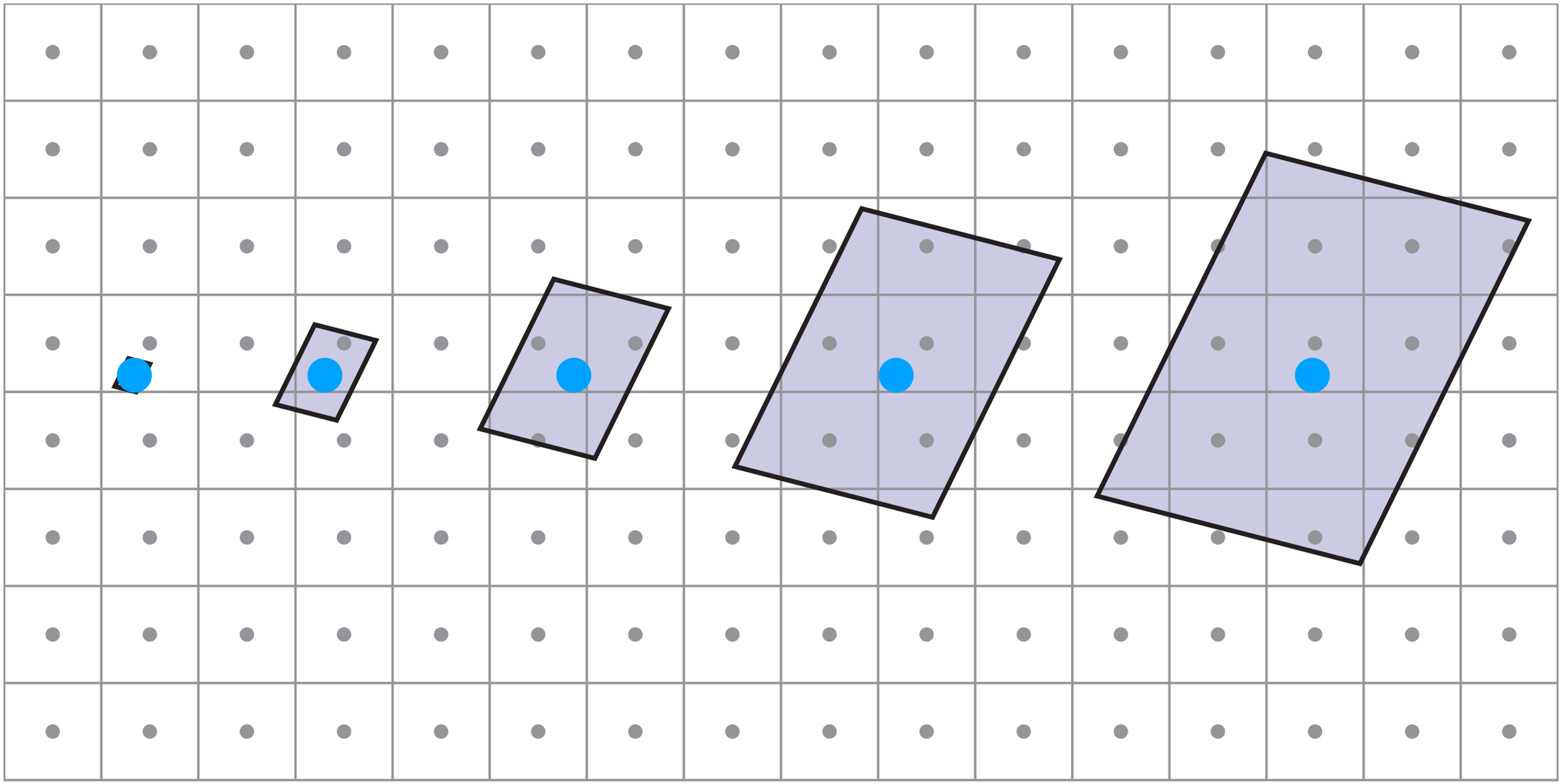
Will supersampling work?

- Yes, high quality, but costly
- When highly minified, many texels in pixel footprint
- Signal frequency too large in a pixel
- Need even higher sampling frequency

Let’s understand this problem in another way

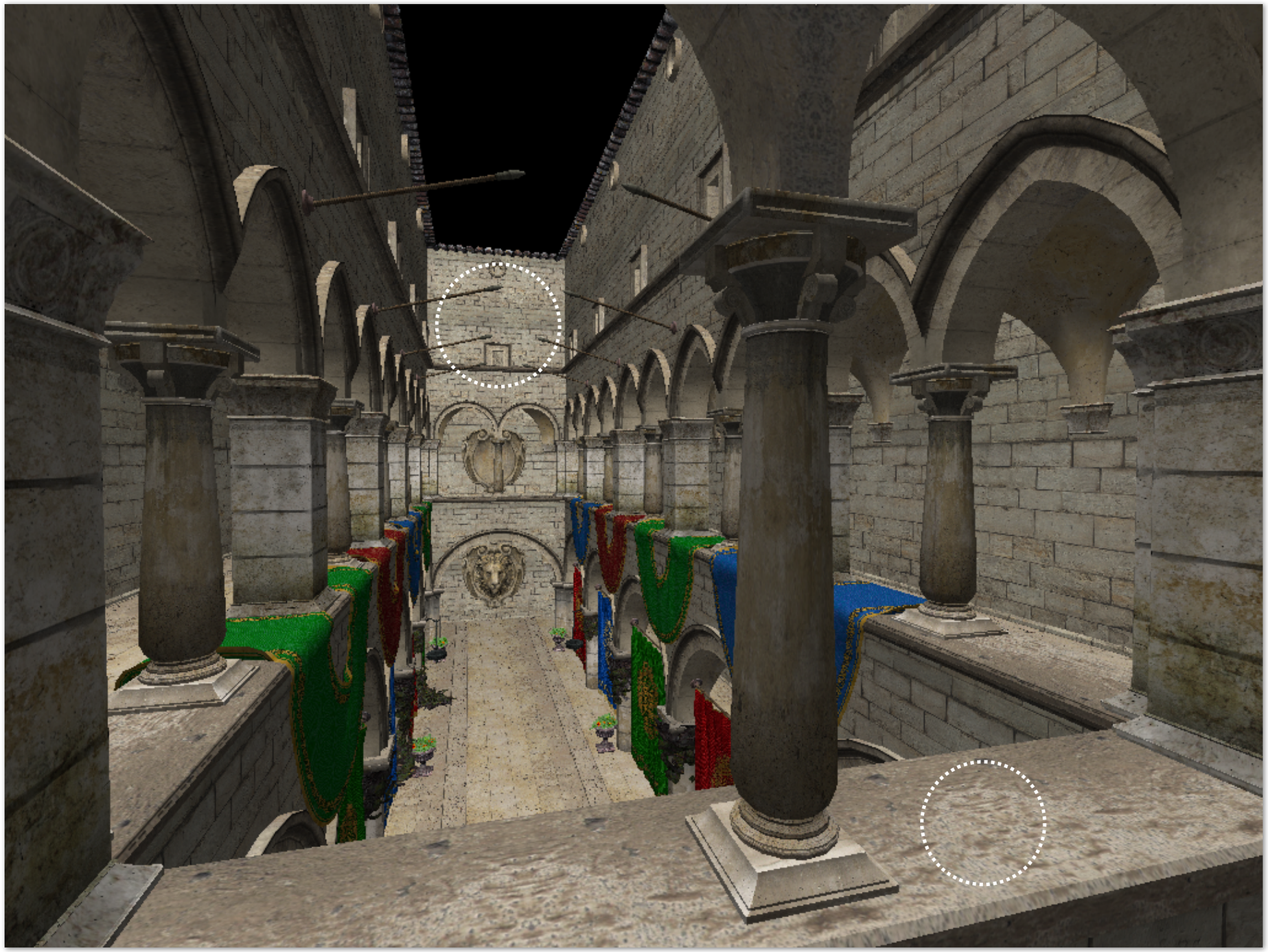
- What if we don’t sample?
- Just need to **get the average value within a range**!

Point Query vs. (Avg.) Range Query



点查询： 已知一个点，快速查出其结果是多少。对应纹理上来说就是给定像素点，查得最近邻，双线性，双三次插值的结果
范围查询：不知道哪个点，但是给定了你一个区域，可以直接得到对应的值（平均值之类的），而mipmap就是基于范围查询的

Different Pixels -> Different-Sized Footprints



Mipmap

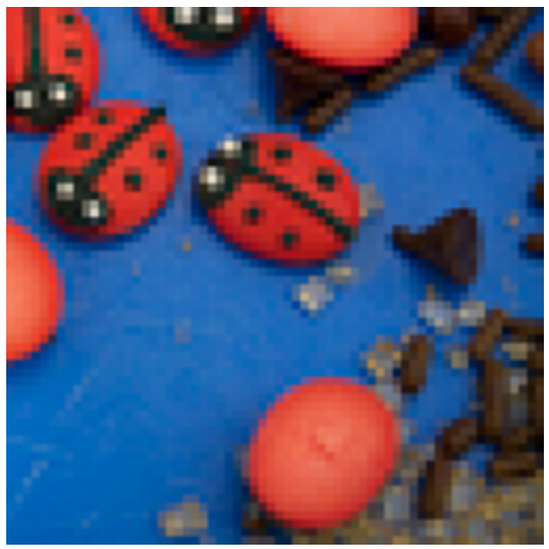
Allowing (fast, approx., square) range queries

一种快速，近似，仅限于正方形范围的范围查询

Mipmap (L. Williams 83)

“Mip” comes from the Latin “multum in parvo", meaning a multitude in a small space

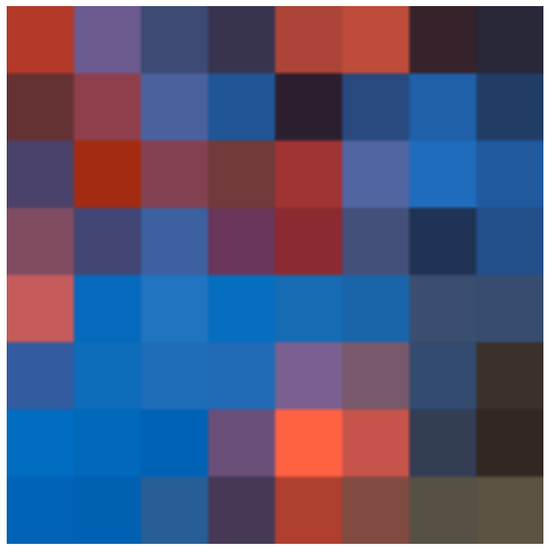
Level 0 = 128x128



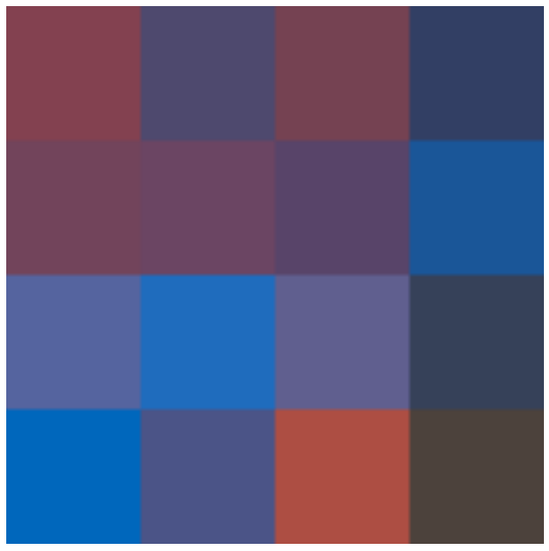
Level 1 = 64x64

Level 2 = 32x32

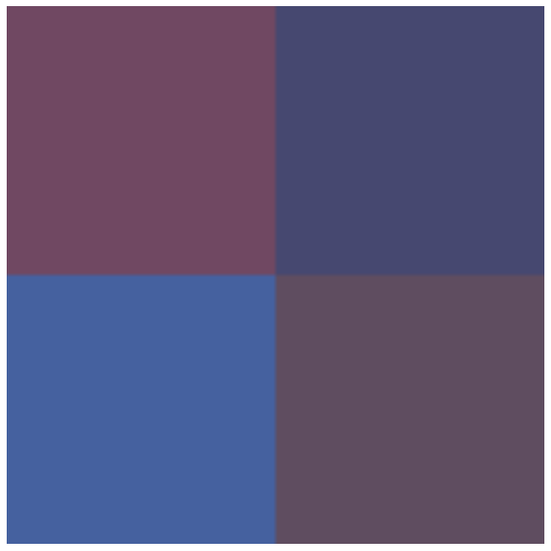
Level 3 = 16x16



Level 4 = 8x8



Level 5 = 4x4

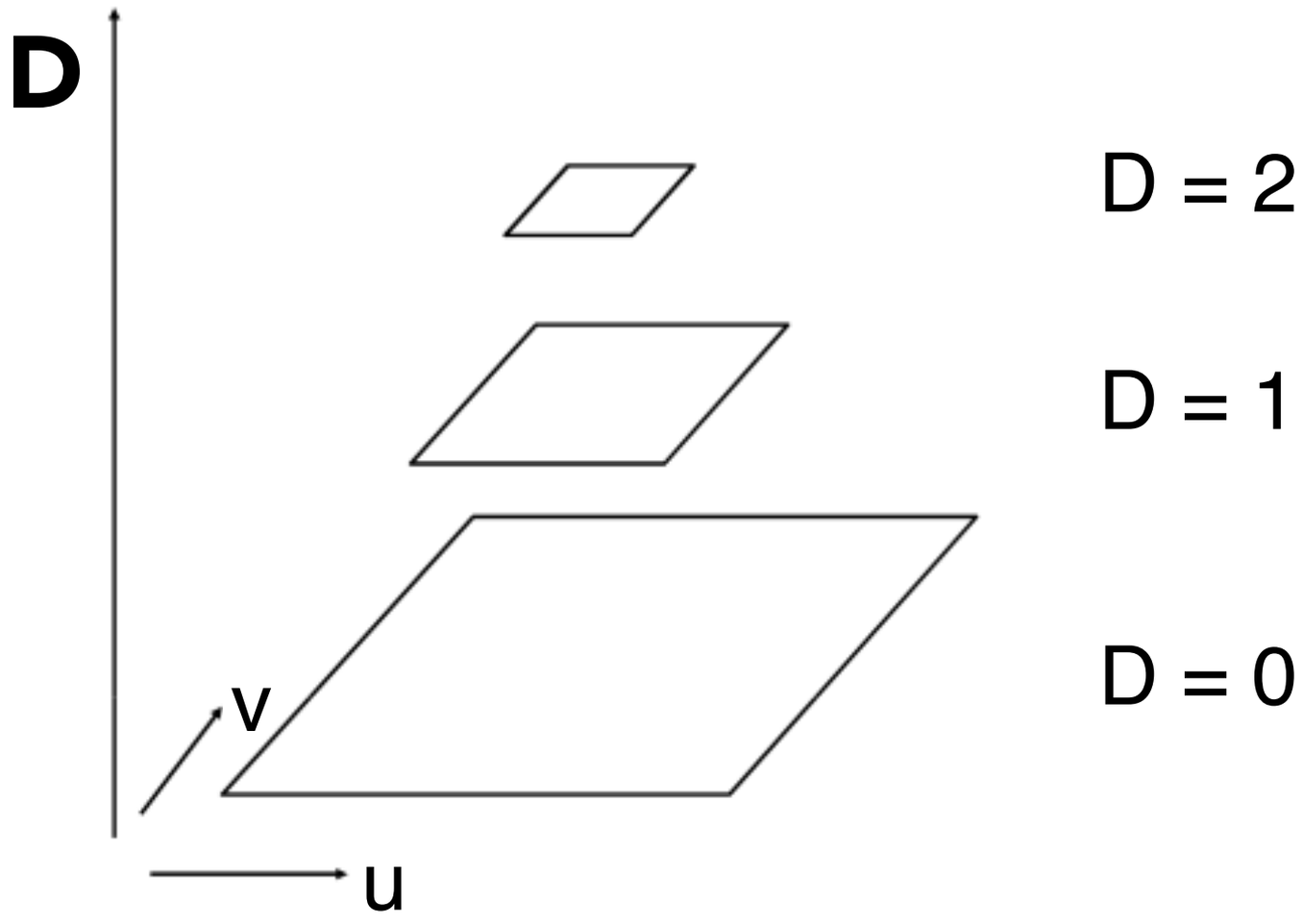


Level 6 = 2x2

Level 7 = 1x1

通过一张纹理，生成许多高层（分辨率每次缩小到一半）的纹理

Mipmap (L. Williams 83)



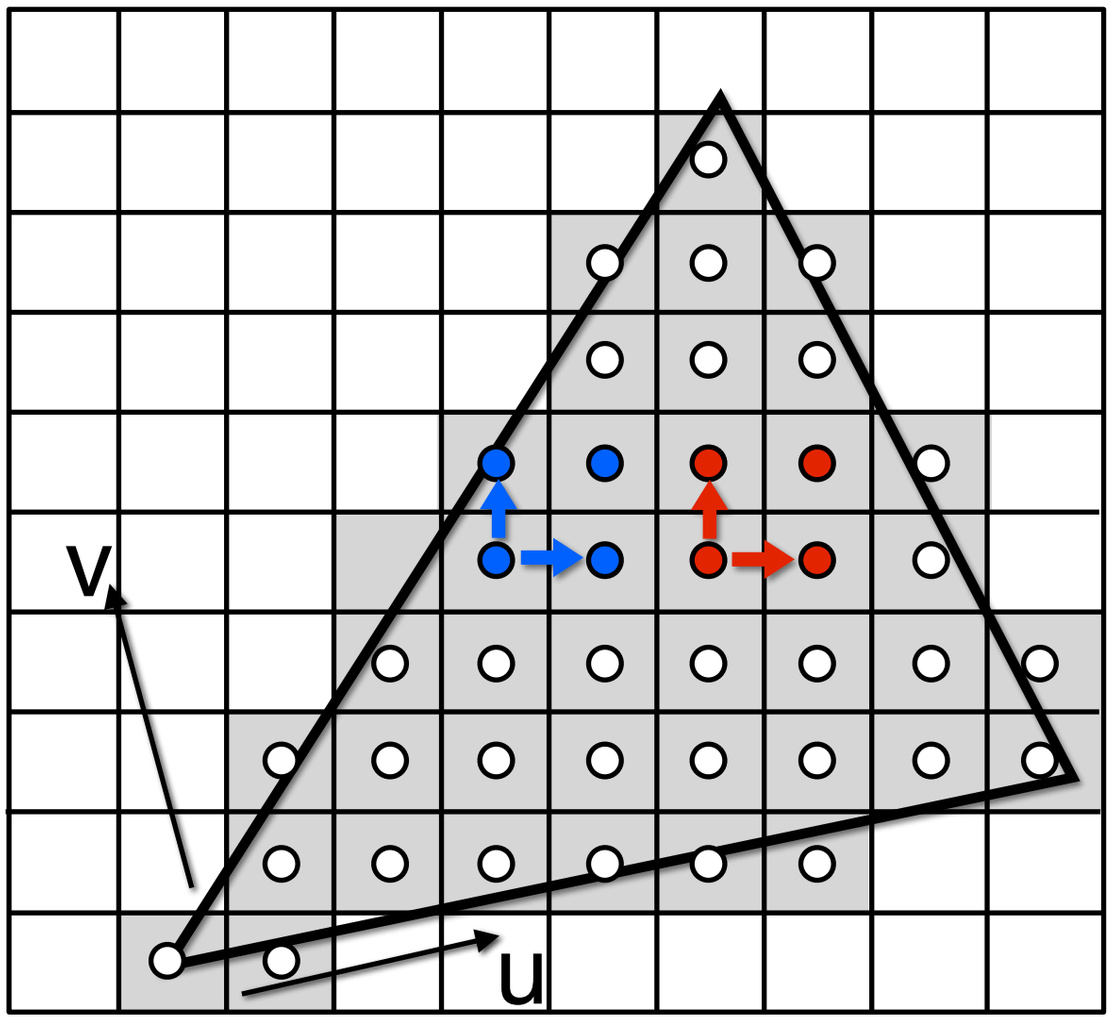
“Mip hierarchy”

level = D

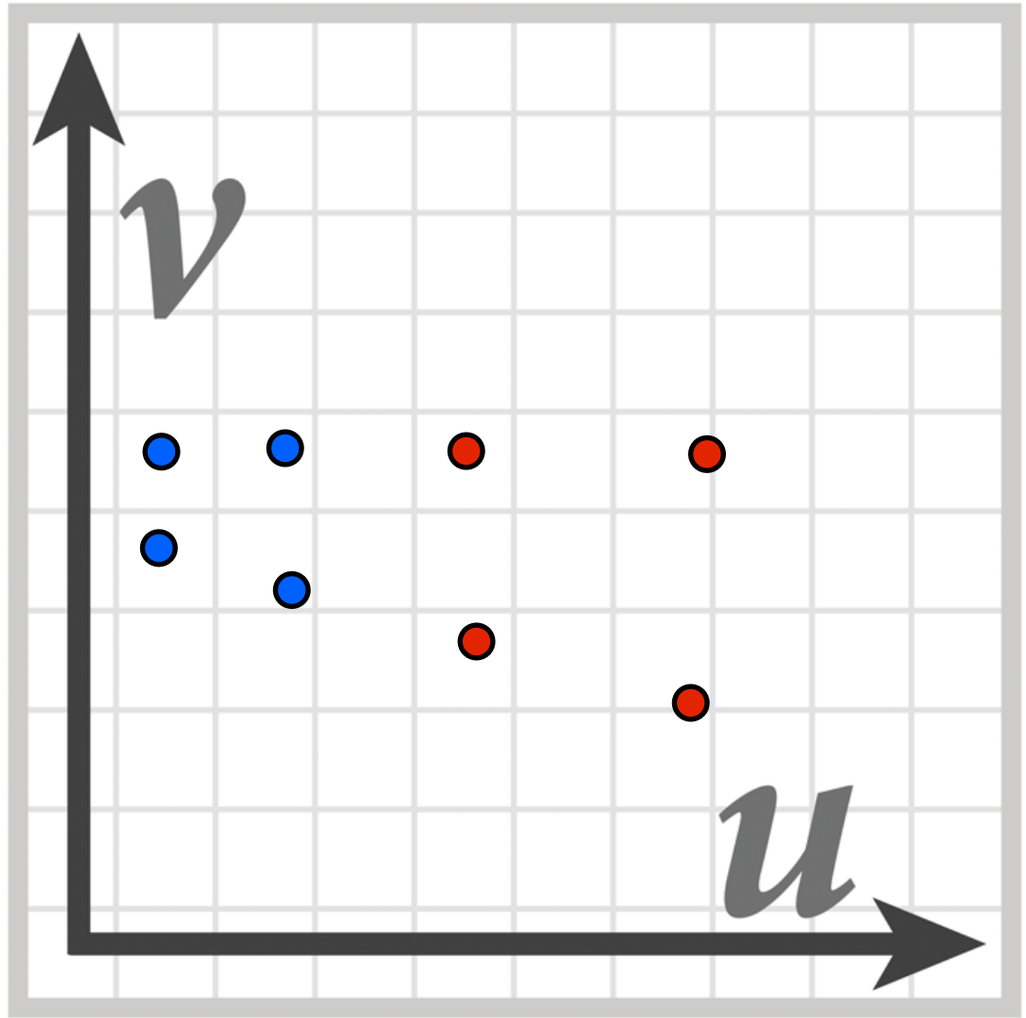
What is the storage overhead of a mipmap?

确定某个像素 P 对应的 Mipmap 层级

Computing Mipmap Level D



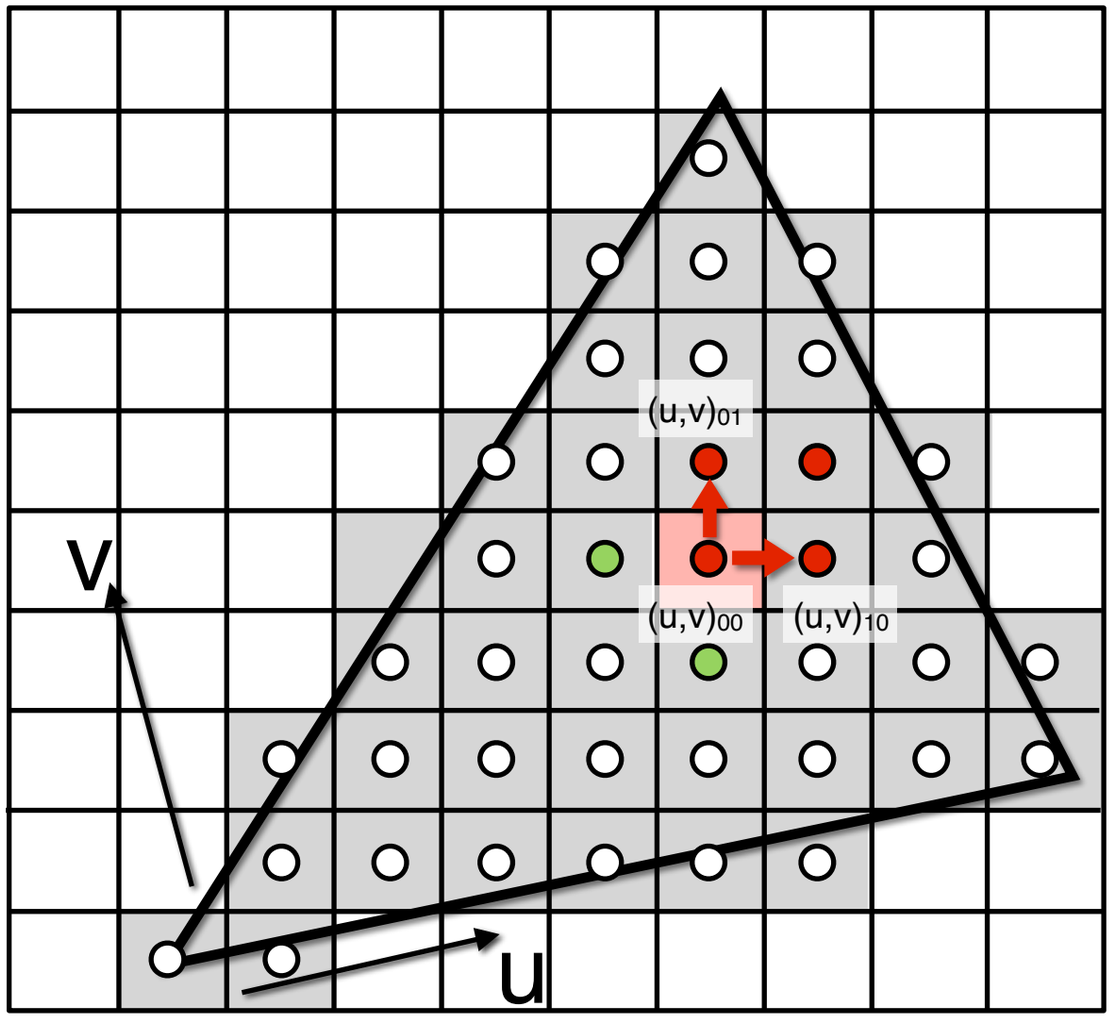
Screen space (x,y)

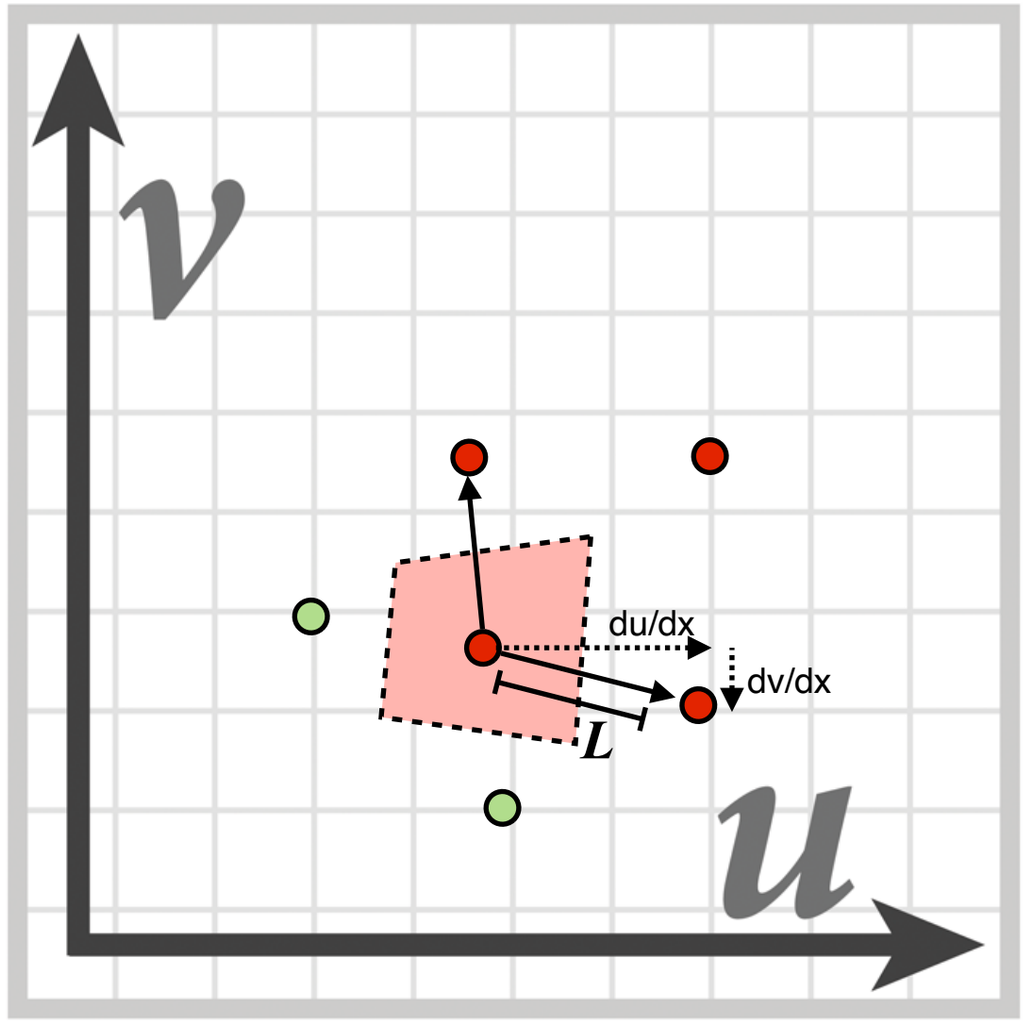


Texture space (u,v)

Estimate texture footprint using texture coordinates of neighboring screen samples

Computing Mipmap Level D

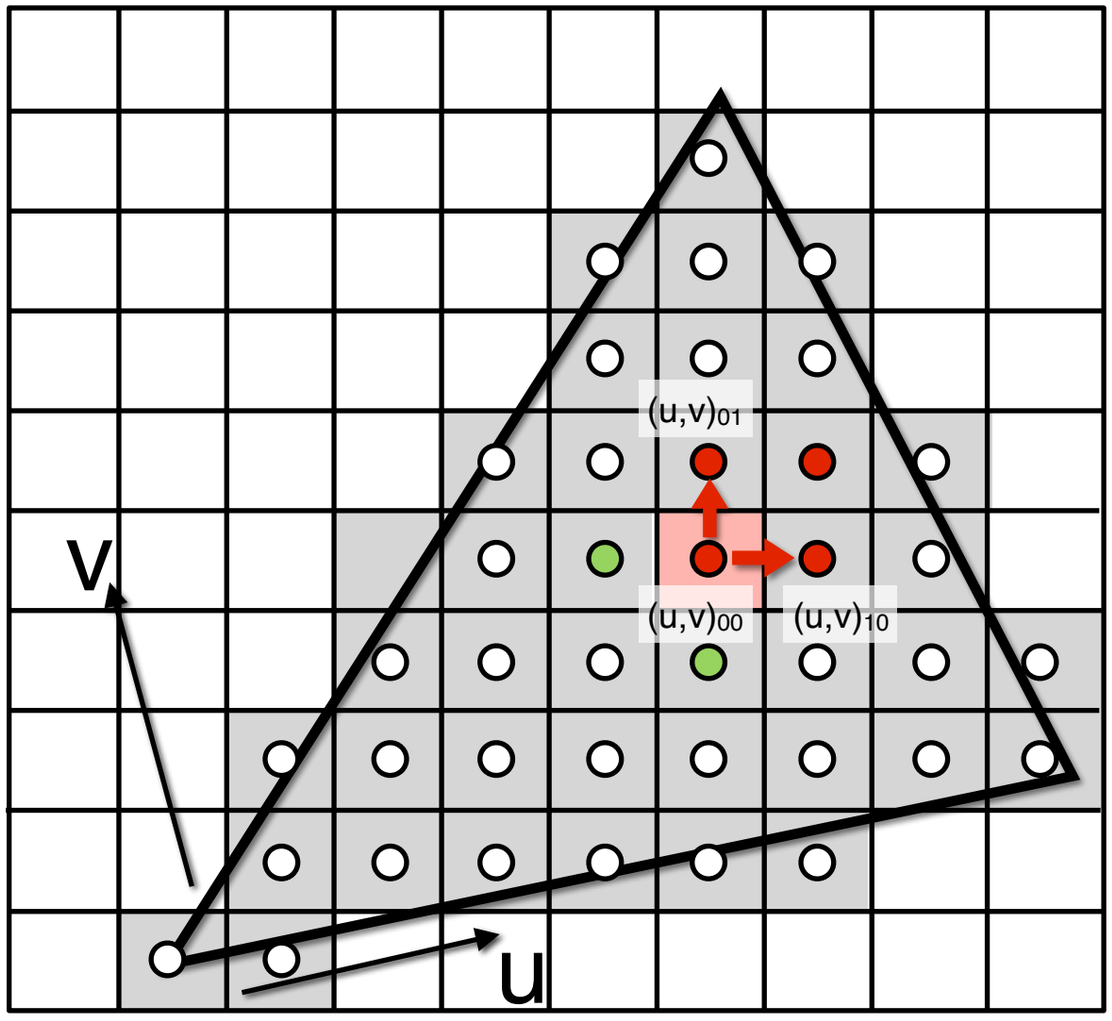


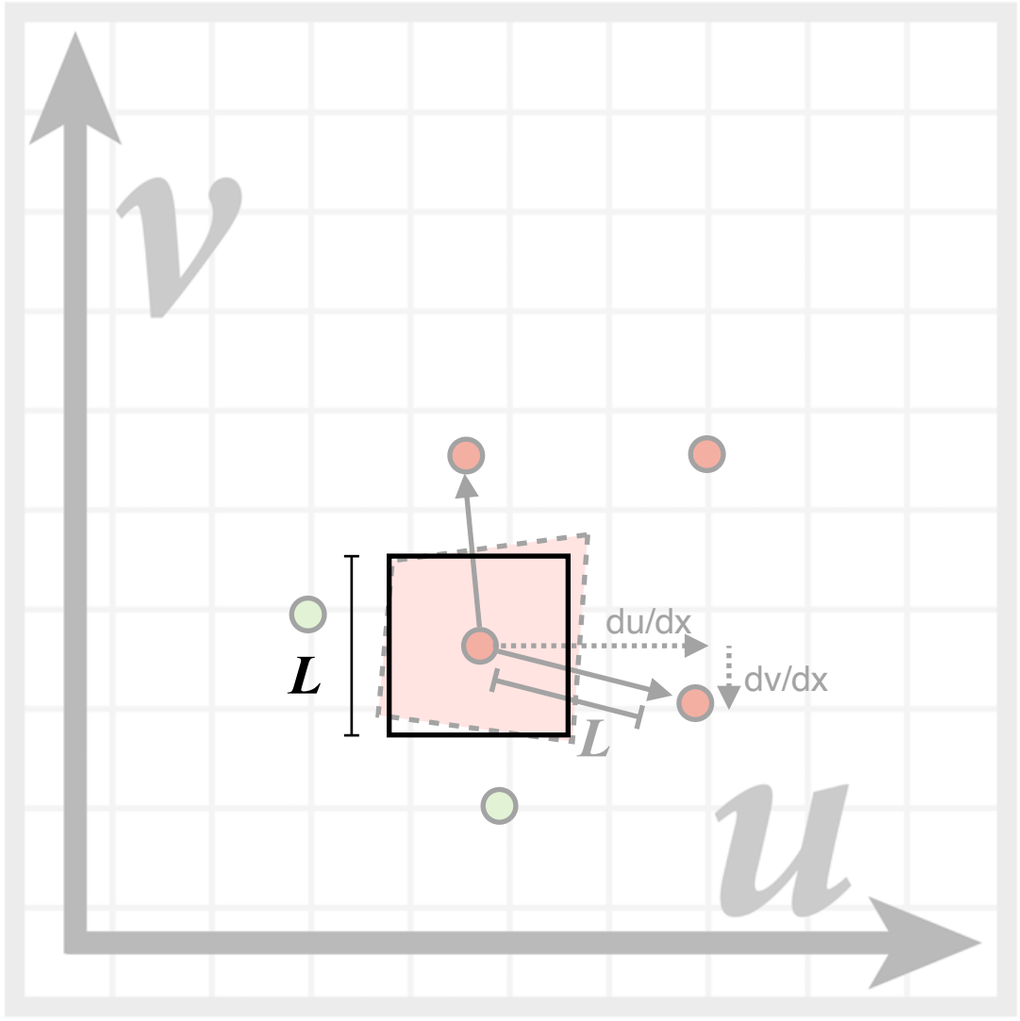


$$D = \log_2 L \qquad L = \max\left(\sqrt{\left(\frac{du}{dx}\right)^2 + \left(\frac{dv}{dx}\right)^2}, \sqrt{\left(\frac{du}{dy}\right)^2 + \left(\frac{dv}{dy}\right)^2}\right)$$

Computing Mipmap Level D

首先找到该点和相邻像素点在纹理上的映射点，计算边长L，所以层级可以由 D = log2L 算出
（因为随着层数增加，纹理分辨率是以4倍的速度减小，所以L是以2倍的速度增加的）

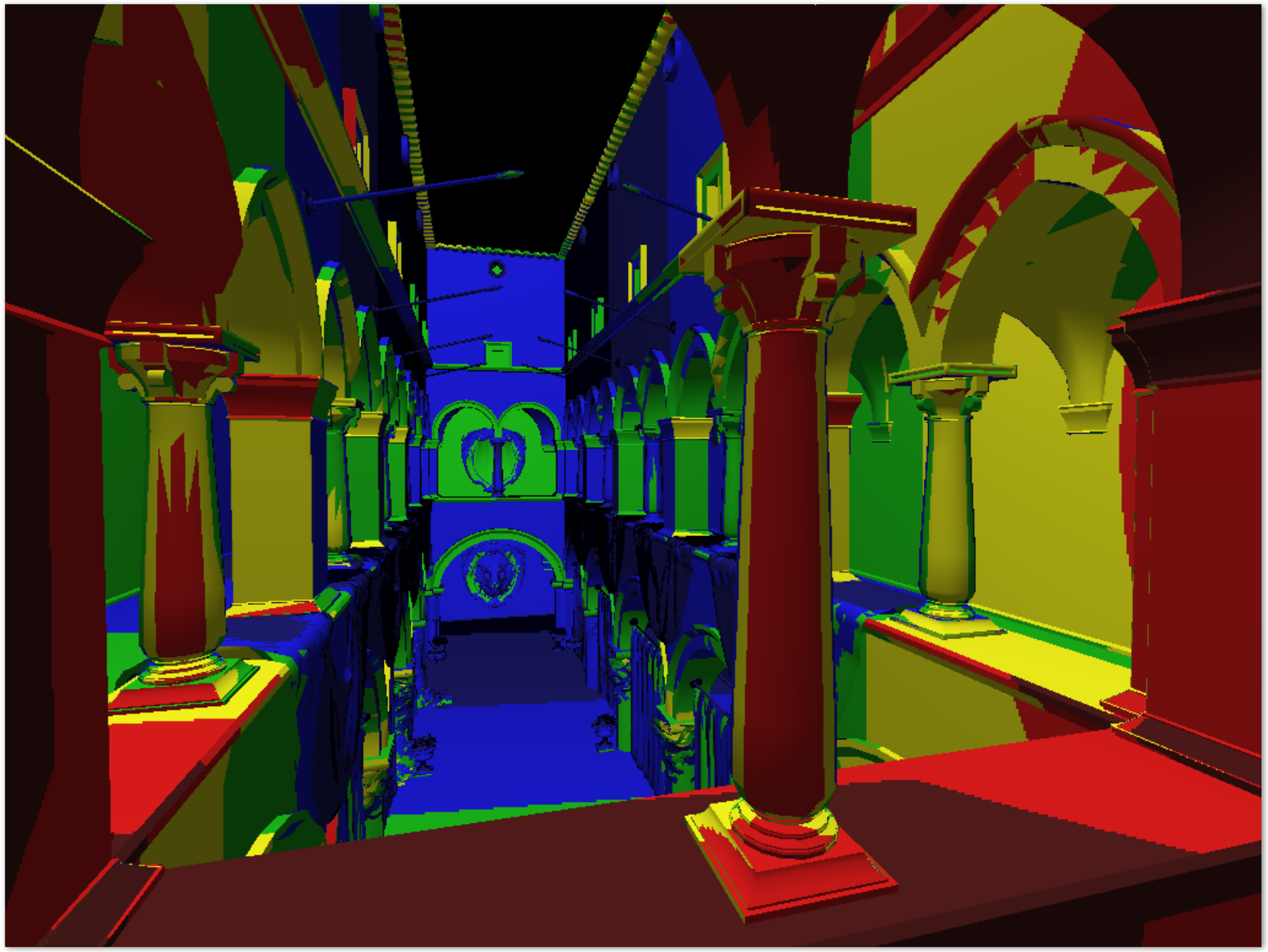




$$D = \log_2 L \qquad L = \max\left(\sqrt{\left(\frac{du}{dx}\right)^2 + \left(\frac{dv}{dx}\right)^2}, \sqrt{\left(\frac{du}{dy}\right)^2 + \left(\frac{dv}{dy}\right)^2}\right)$$

Visualization of Mipmap Level

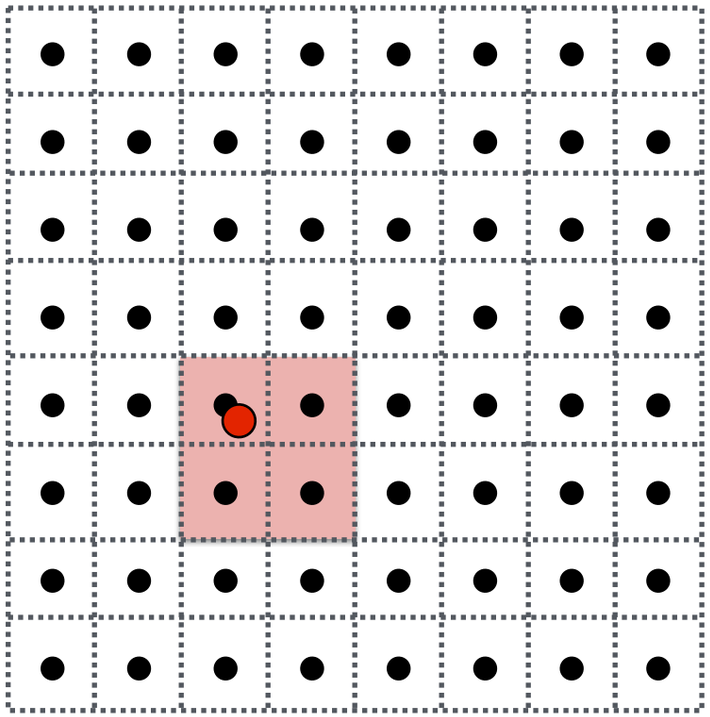
当我们得到了层级后，在这一层上进行纹理插值即可。但是我们发现，层级之间是离散的，也就是说，一些像素在0层插值，一些像素在1层插值，这种变化也会使得纹理“割裂”：



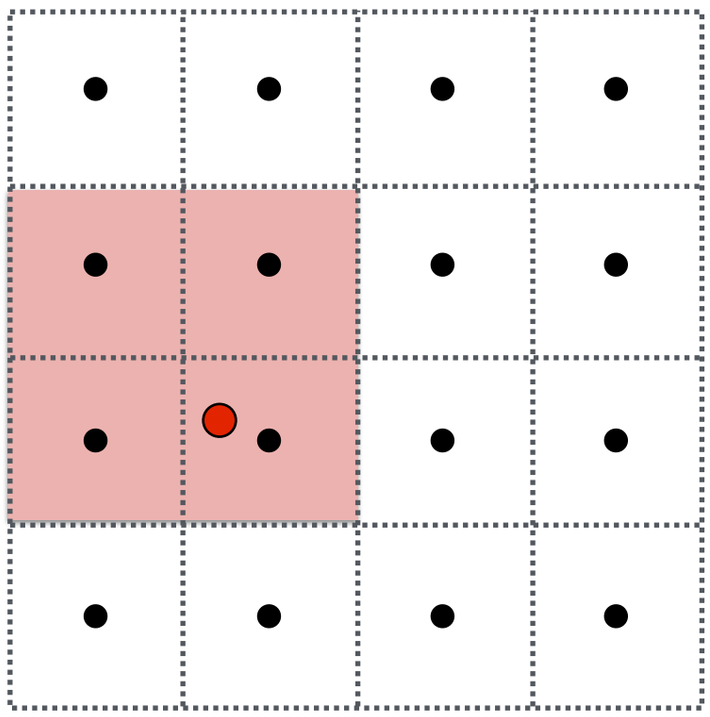
D rounded to nearest integer level

Trilinear Interpolation

解决方法就是，每次插值，现在D层插值纹理（可以用双线性之类的方法），再在D+1层插值纹理，把两层结果再进行插值，这样的结果就是平滑的了：



Mipmap Level D



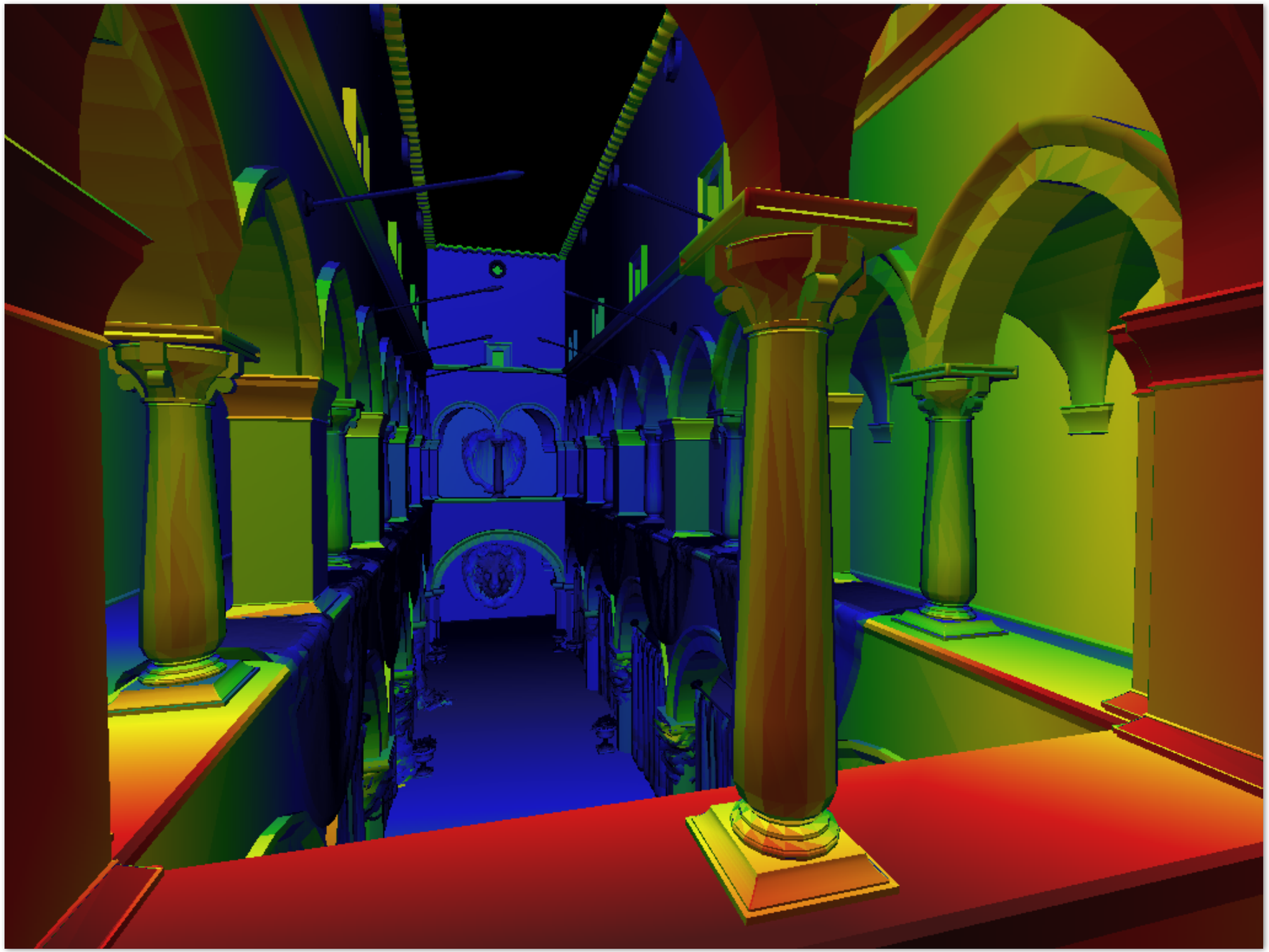
Mipmap Level D+1

Bilinear result

Bilinear result

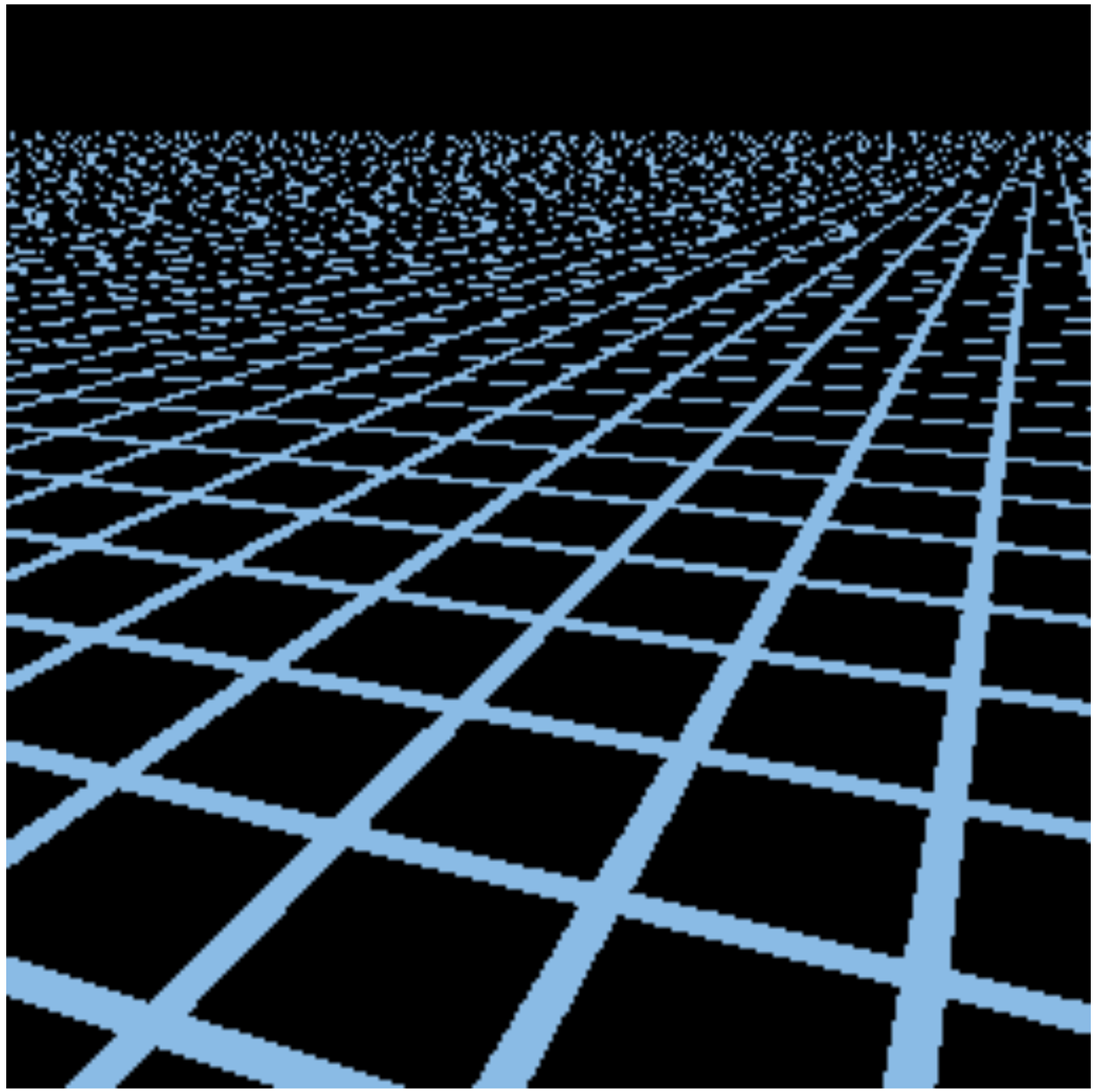
Linear interpolation based on continuous D value

Visualization of Mipmap Level



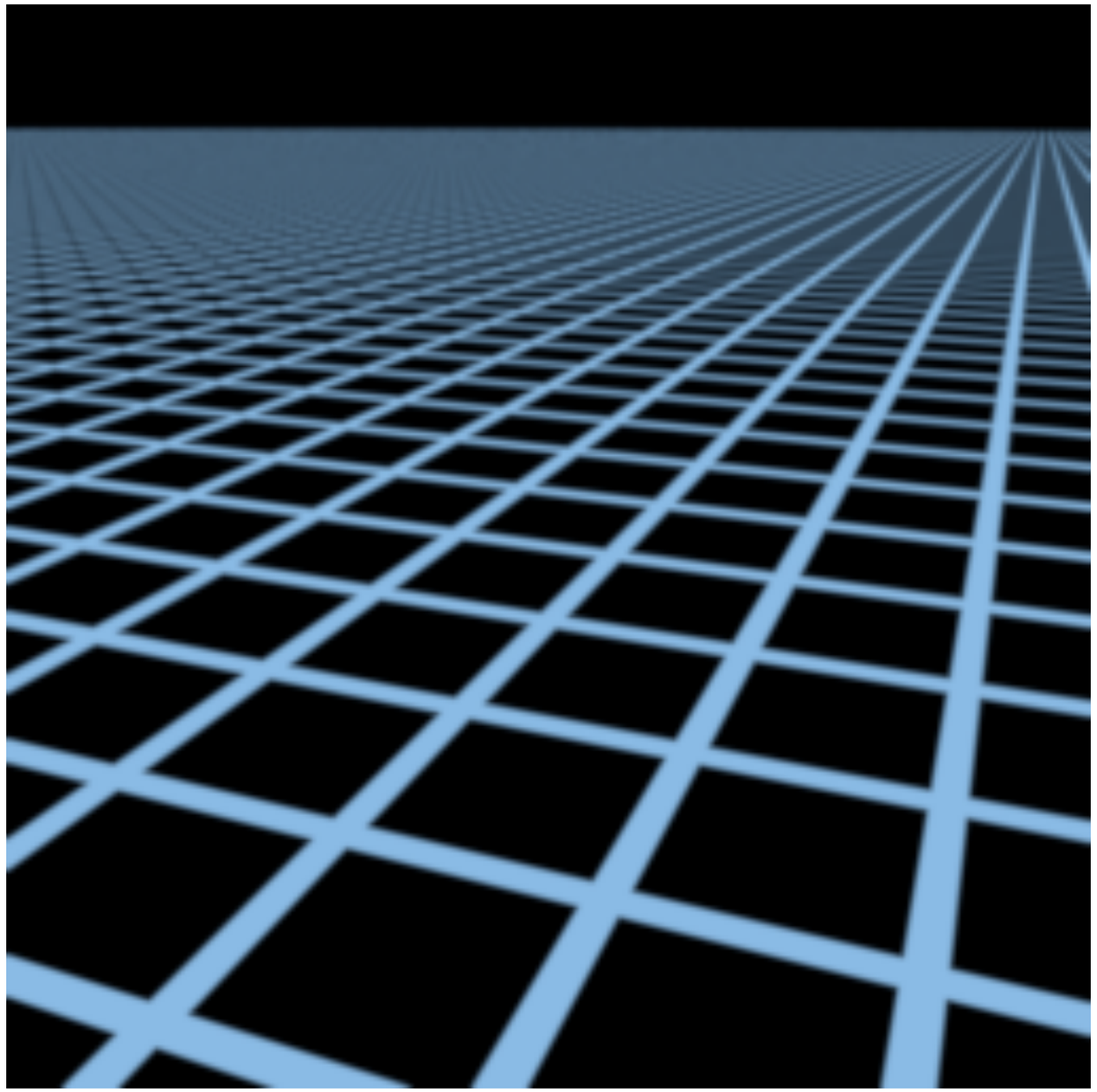
Trilinear filtering: visualization of continuous D

Mipmap Limitations



Point sampling

Mipmap Limitations



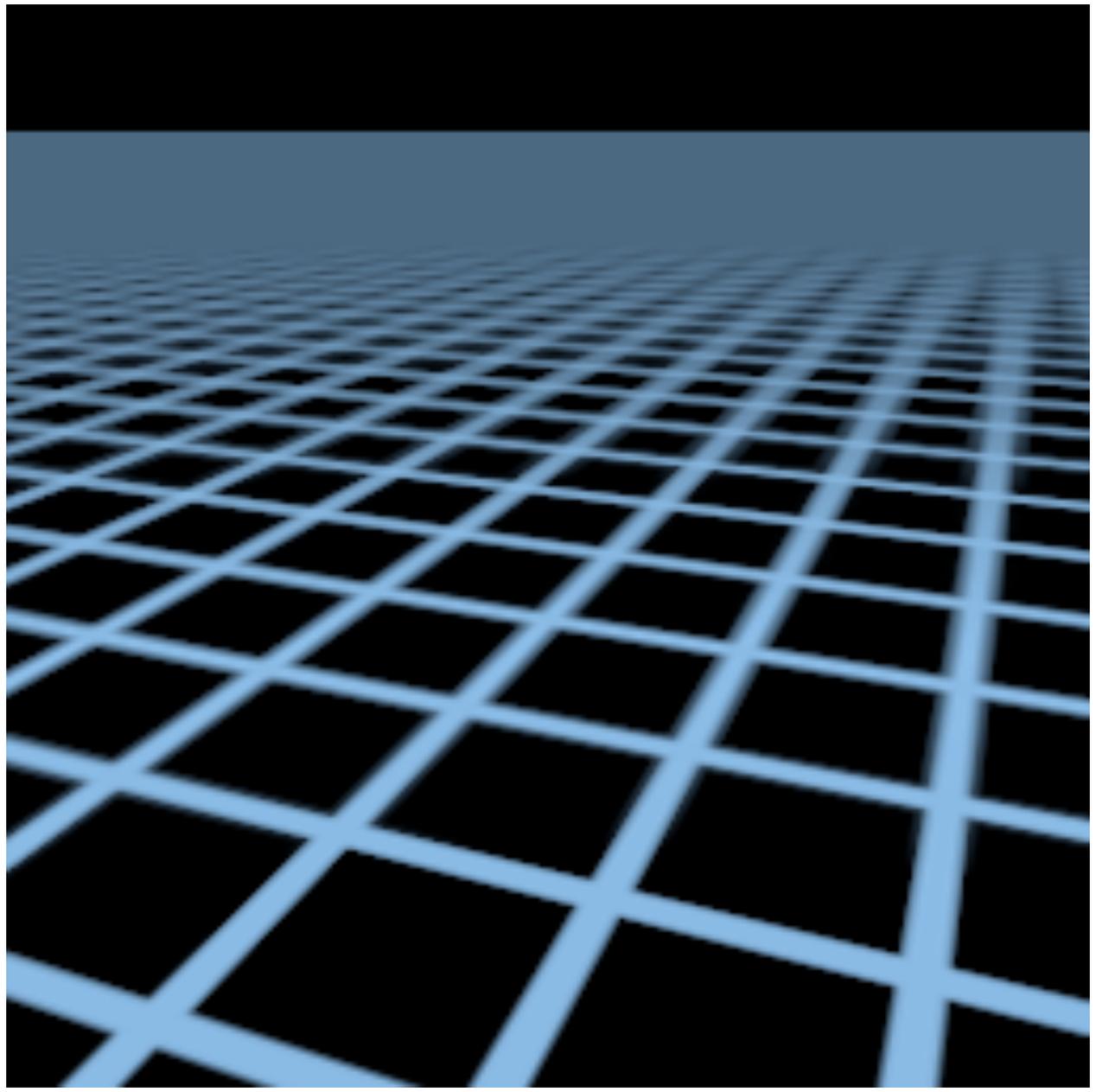
Supersampling 512x (assume this is correct)

Mipmap Limitations

只能在u-v坐标系下做方块查询，有时候会造成过度模糊的情况

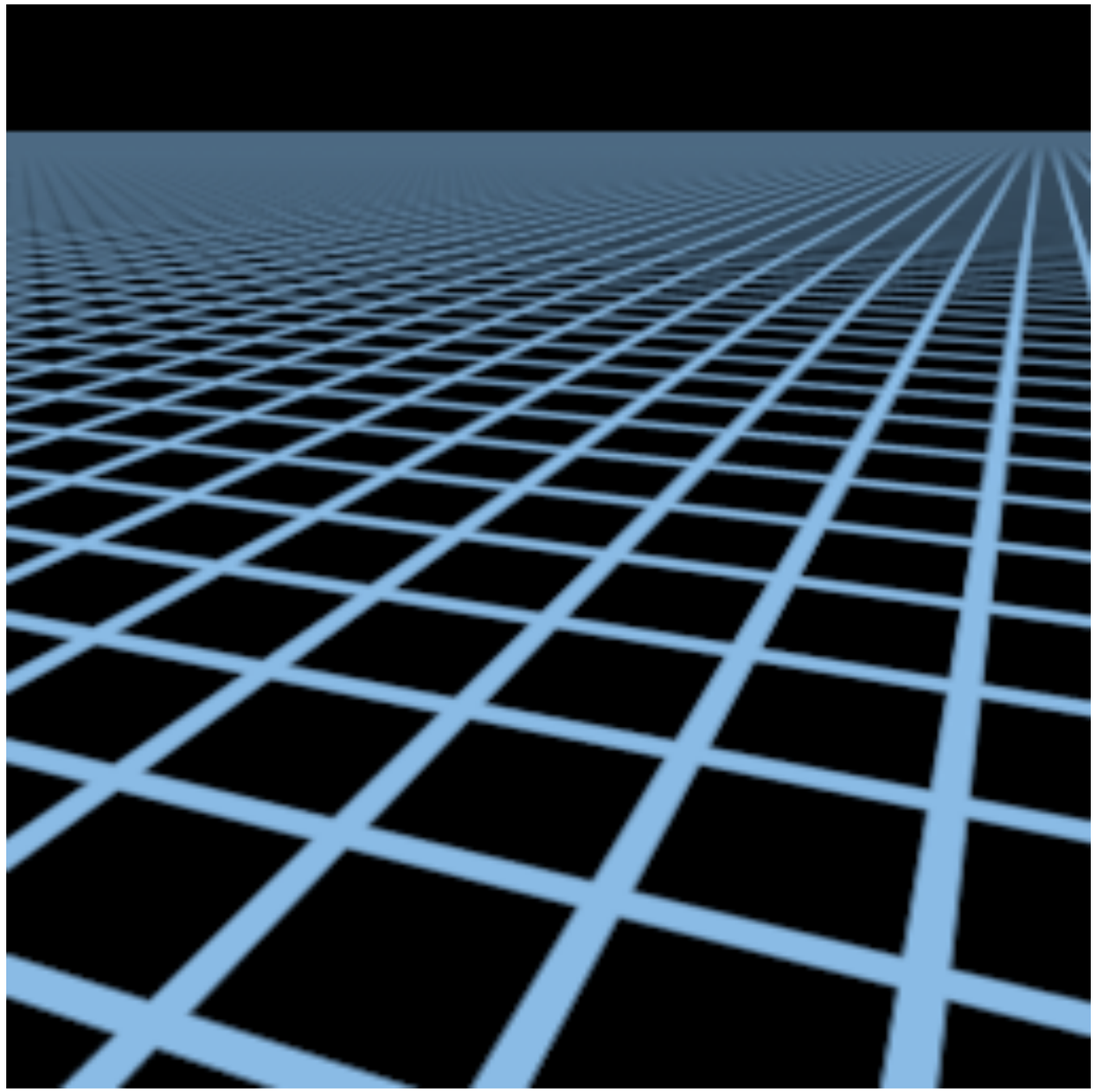
Overblur

Why?



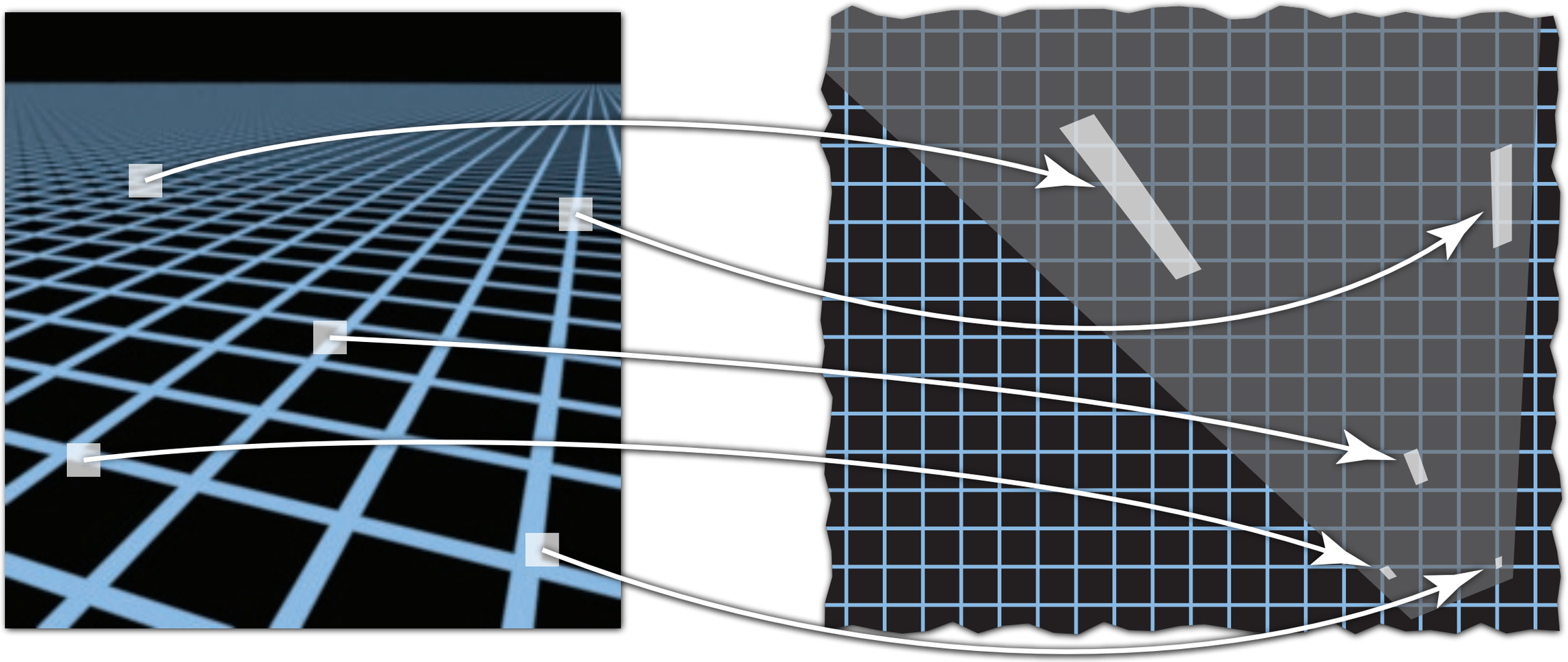
Mipmap trilinear sampling

Anisotropic Filtering



Better than Mipmap!

Irregular Pixel Footprint in Texture



Screen space

Texture space

这是由于 Mipmap 擅长解决正方形的采样，但不擅长非正方形的采样

Anisotropic Filtering

Ripmaps and summed area tables

- Can look up **axis-aligned rectangular zones**
- Diagonal footprints still a problem

Wikipedia

为了避免这种情况，引入各向异性过滤，在准备不同级别的纹理贴图时，不再是简简单单横纵纹素各减小一半进行分 级，而是长减半宽不变 or 宽减半长不变 or 长和宽各减半三种情况各进行一次分级，显存消耗为原来的三倍，但性能 方面并没有多少影响，这种方法就可以实现在u-v坐标系下进行矩形查询。

Anisotropic Filtering

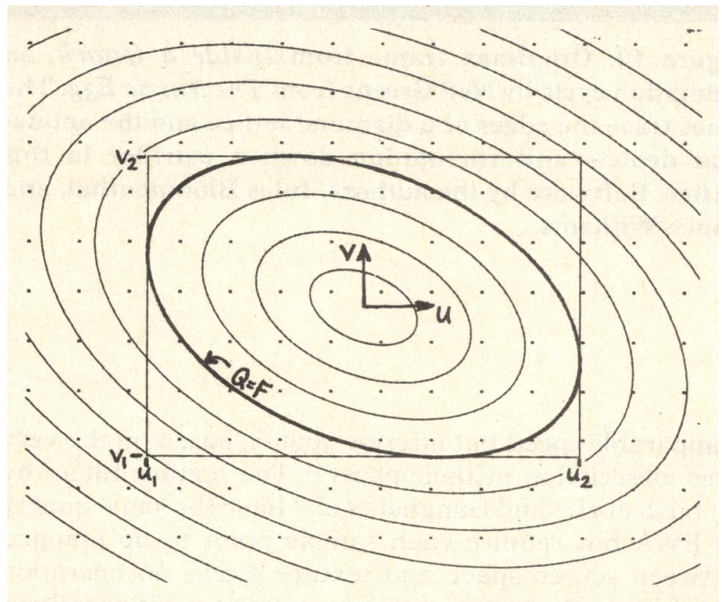
Ripmaps and summed area tables

- Can look up axis-aligned rectangular zones
- Diagonal footprints still a problem

Wikipedia

EWA filtering

- Use multiple lookups
- Weighted average
- Mipmap hierarchy still helps
- Can handle irregular footprints



Greene & Heckbert ‘86

比各向异性更进一步的过滤，如EWA filtering 椭圆取样，则利用多次查询求平均值的方法来处理不规则区域，相应的 性能开销就会比较大了

Thank you!

(And thank Prof. Ravi Ramamoorthi and Prof. Ren Ng for many of the slides!)