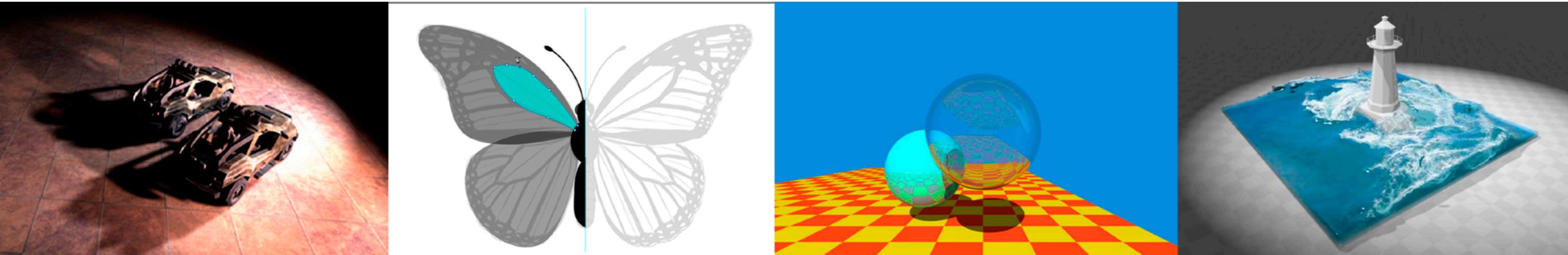


# Introduction to Computer Graphics

AMES101, Lingqi Yan, UC Santa Barbara

## Lecture 9: Shading 3 (Texture Mapping cont.)



# 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?

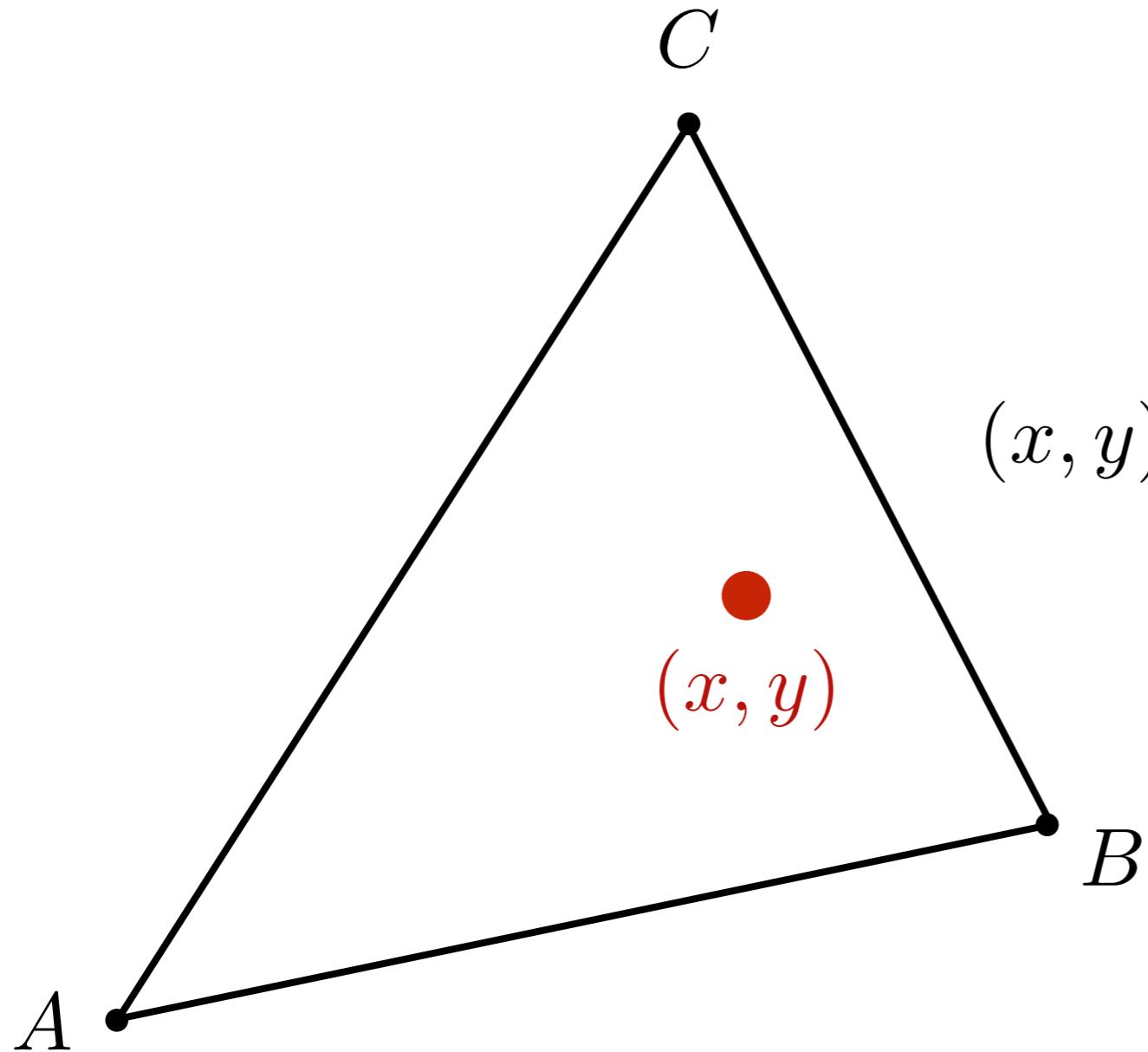
- Texture coordinates, colors, normal vectors, ...

How do we interpolate?

- **Barycentric coordinates**

# Barycentric Coordinates

A coordinate system for triangles  $(\alpha, \beta, \gamma)$



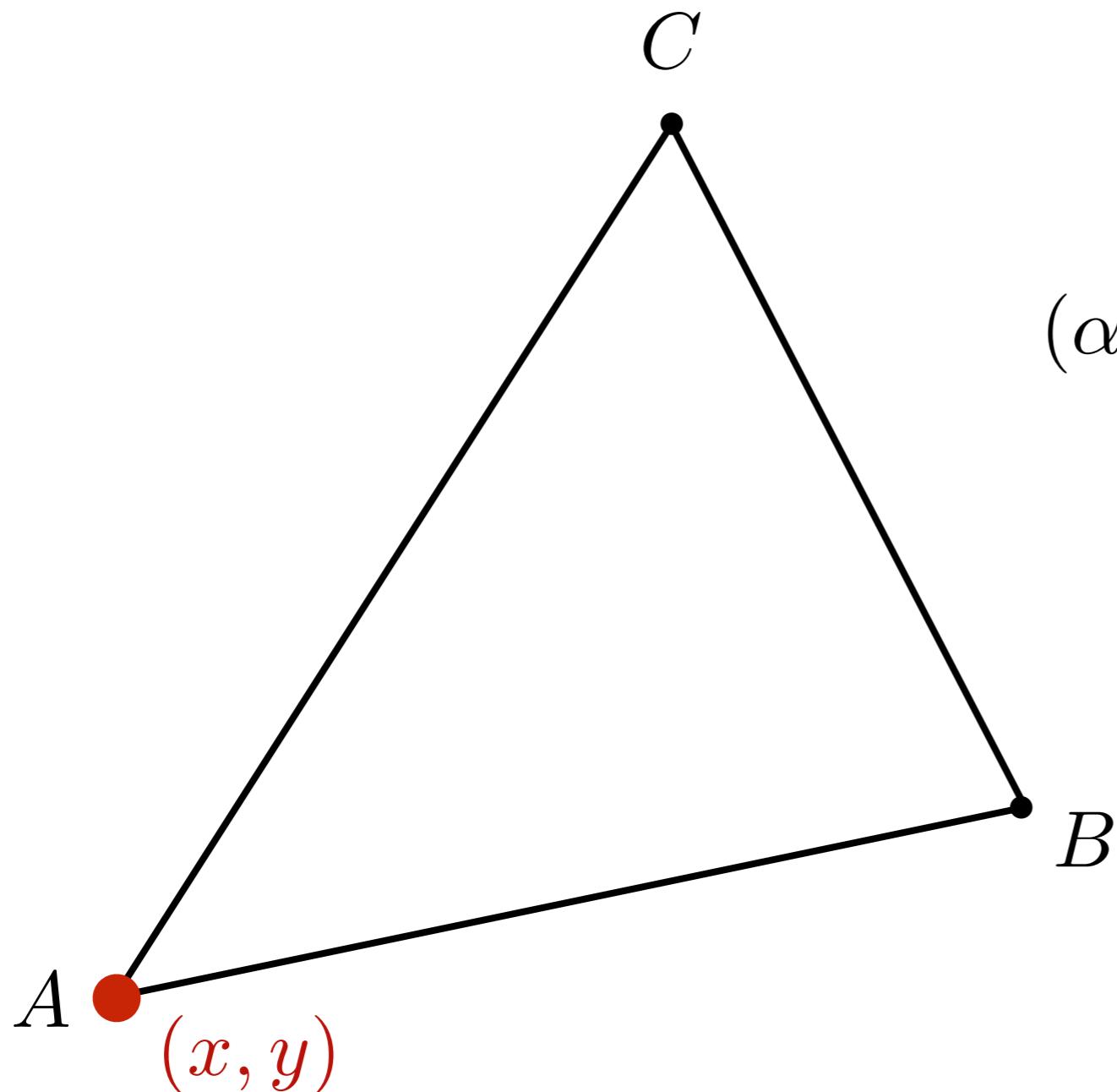
$$(x, y) = \alpha A + \beta B + \gamma C$$

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

**Inside the triangle if  
all three coordinates  
are non-negative**

# Barycentric Coordinates

What's the barycentric coordinate of A?

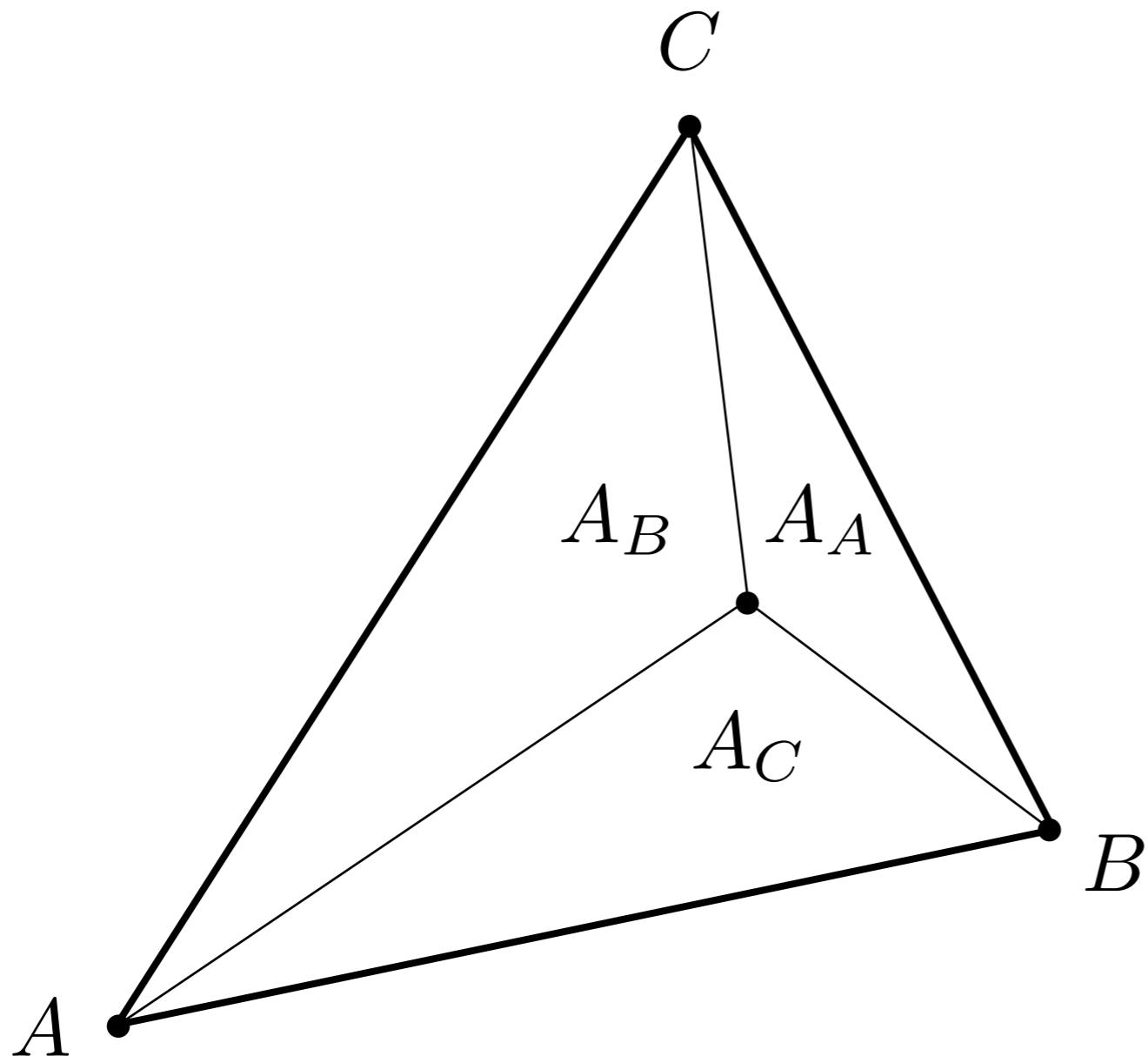


$$(\alpha, \beta, \gamma) = (1, 0, 0)$$

$$\begin{aligned}(x, y) &= \alpha A + \beta B + \gamma C \\ &= A\end{aligned}$$

# Barycentric Coordinates

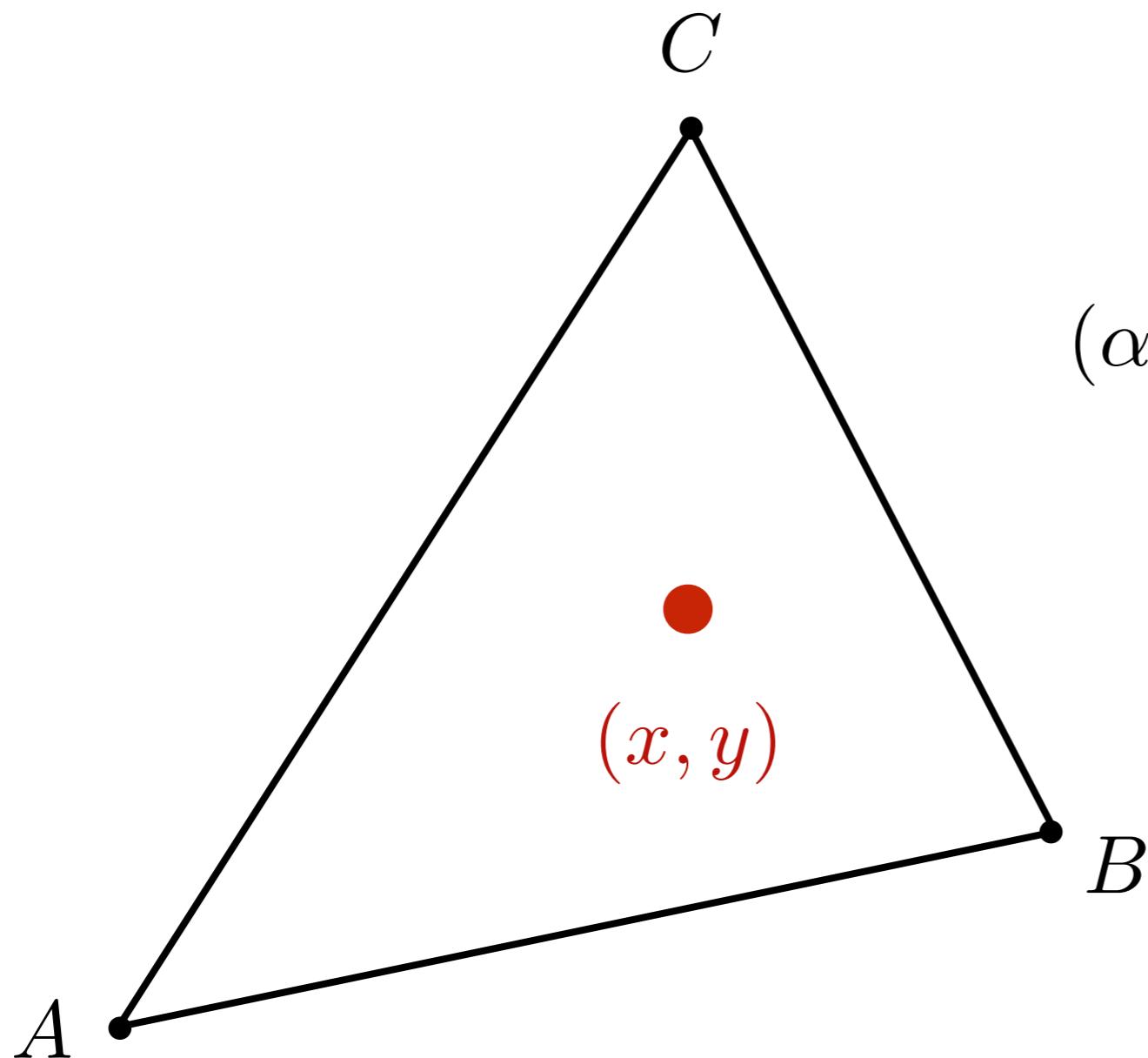
Geometric viewpoint — proportional areas



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

# Barycentric Coordinates

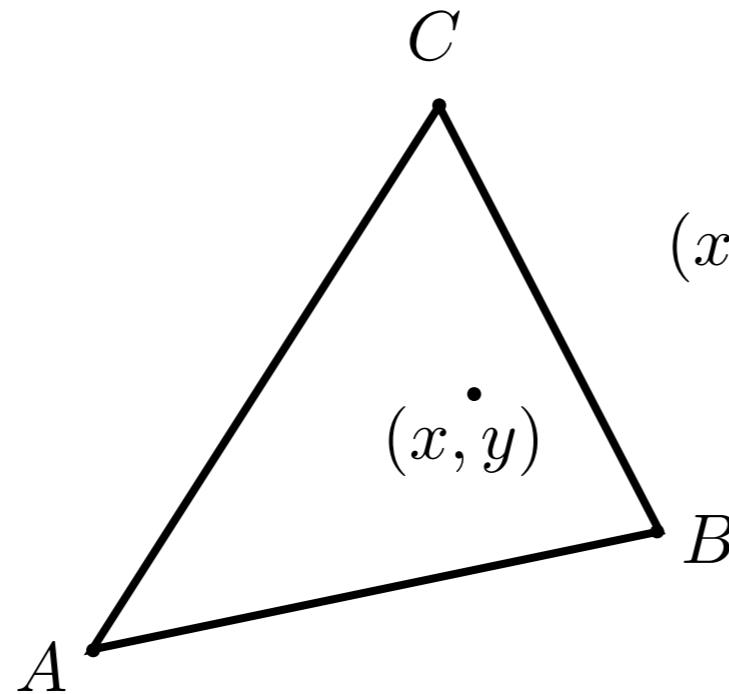
What's the barycentric coordinate of the centroid?



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

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

# Barycentric Coordinates: Formulas



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

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

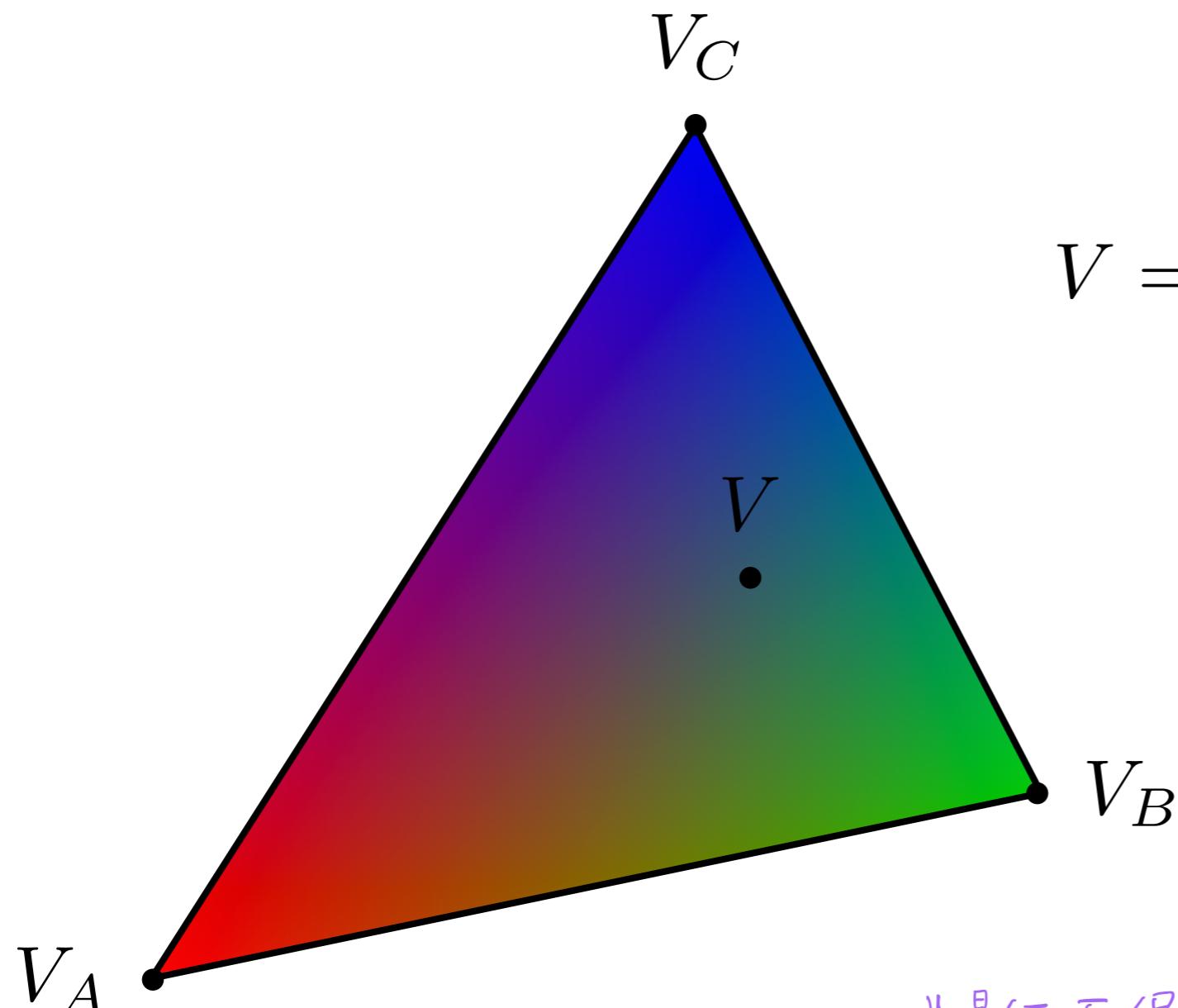
$$\alpha = \frac{-(x - x_B)(y_C - y_B) + (y - y_B)(x_C - x_B)}{-(x_A - x_B)(y_C - y_B) + (y_A - y_B)(x_C - x_B)}$$

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

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

# 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...

**However, barycentric coordinates are not invariant under projection!**

投影下不保证重心坐标不变  
深度要在投影前三维坐标上计算

# Applying Textures

# Simple Texture Mapping: Diffuse Color

for each rasterized screen sample  $(x, y)$ :

$(u, v) = \text{evaluate texture coordinate at } (x, y)$

`texcolor = texture.sample(u, v);`

set sample's color to texcolor;

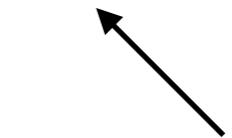


Usually the diffuse albedo  $K_d$   
(recall the Blinn-Phong reflectance model)

Usually a pixel's center



Using barycentric  
coordinates!



纹理放大

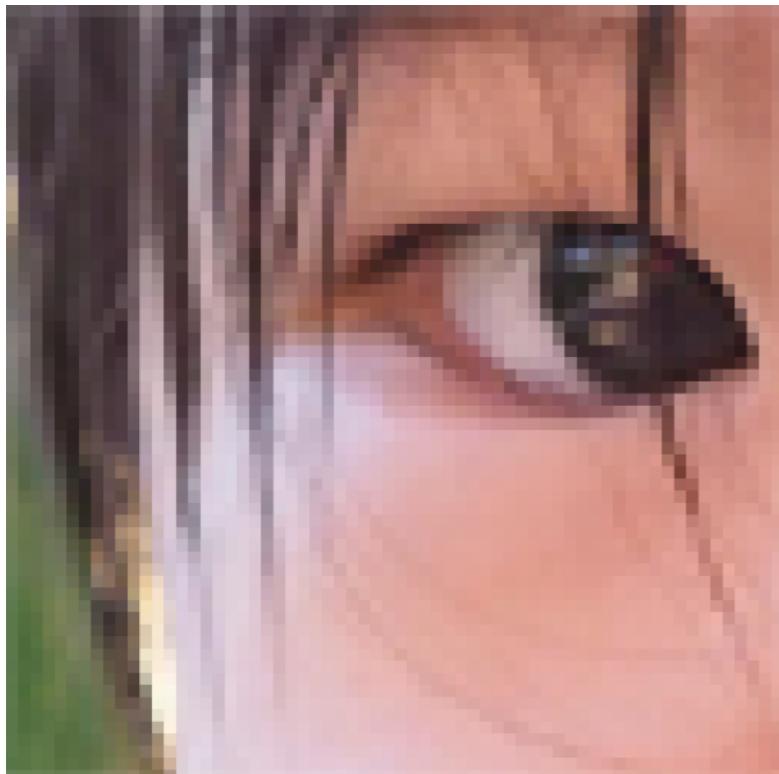
# Texture Magnification

(What if the texture is too small?)

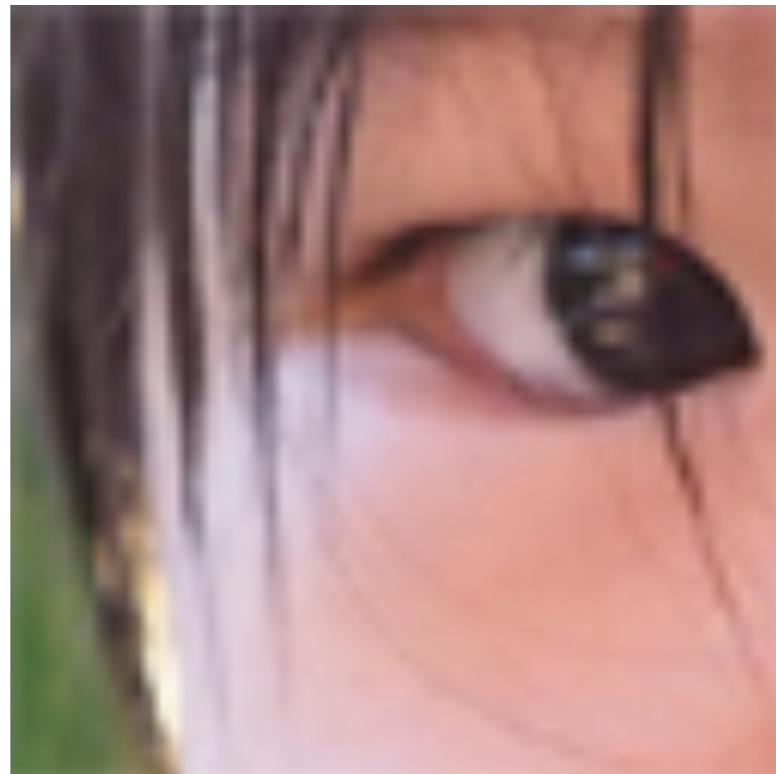
# Texture Magnification - Easy Case

Generally don't want this — insufficient texture resolution

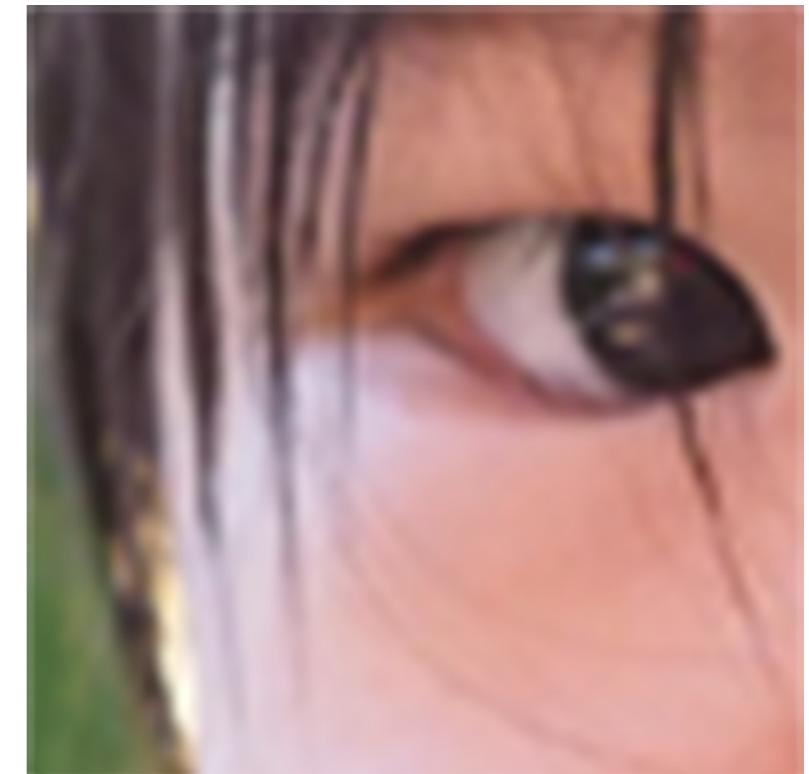
A pixel on a texture — a **texel** (纹理元素、纹素)



**Nearest**

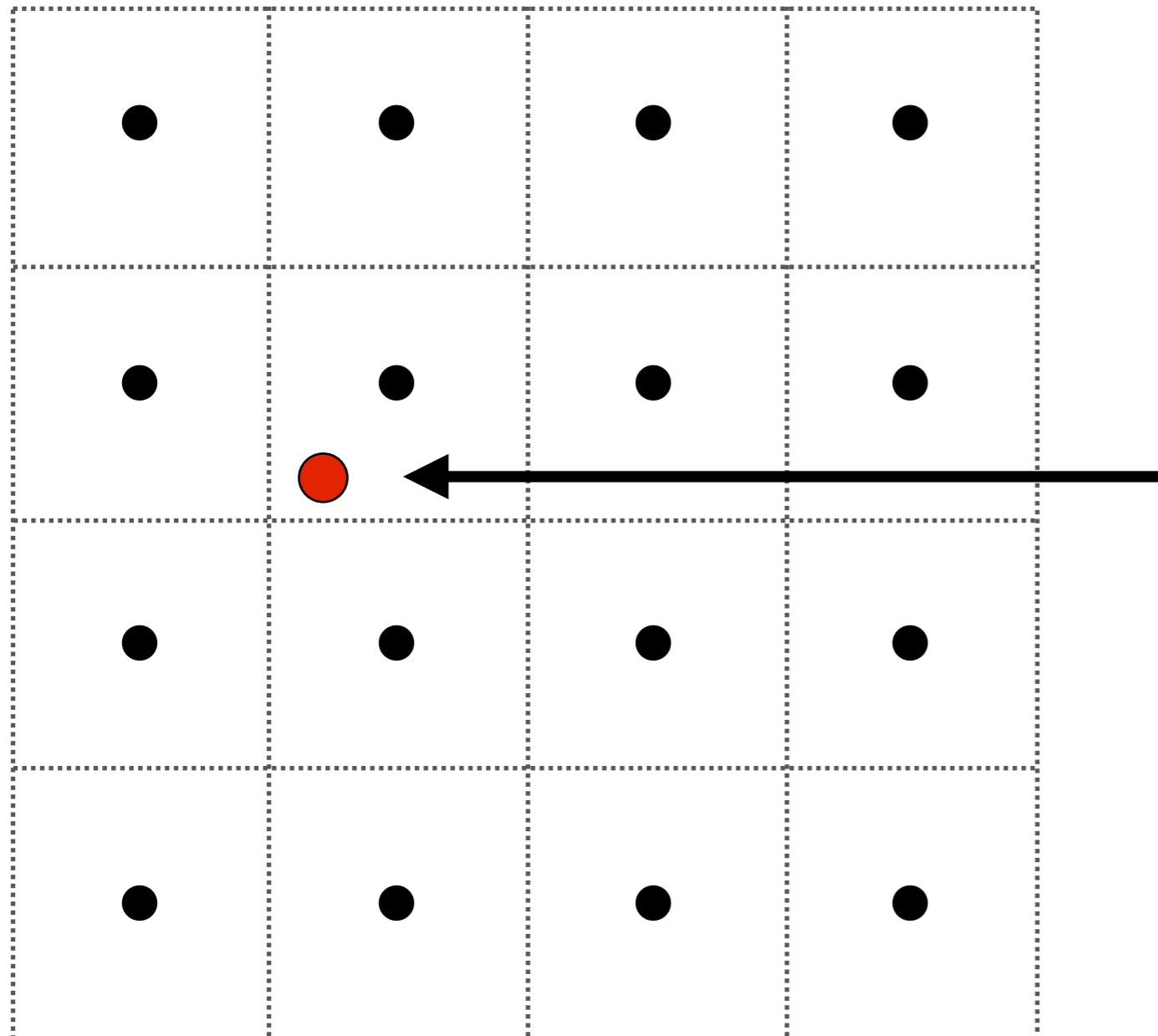


**Bilinear**



**Bicubic**

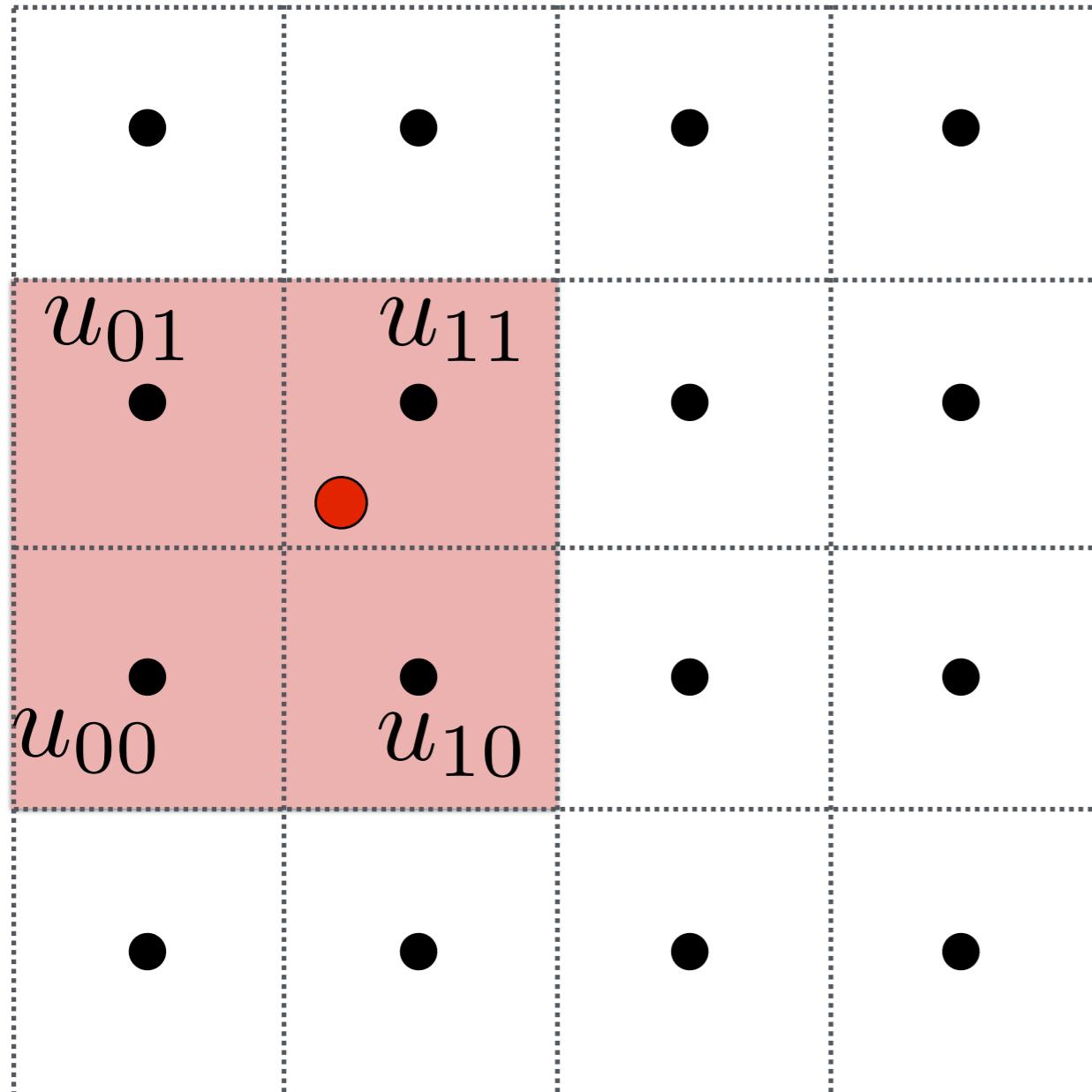
# Bilinear Interpolation



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

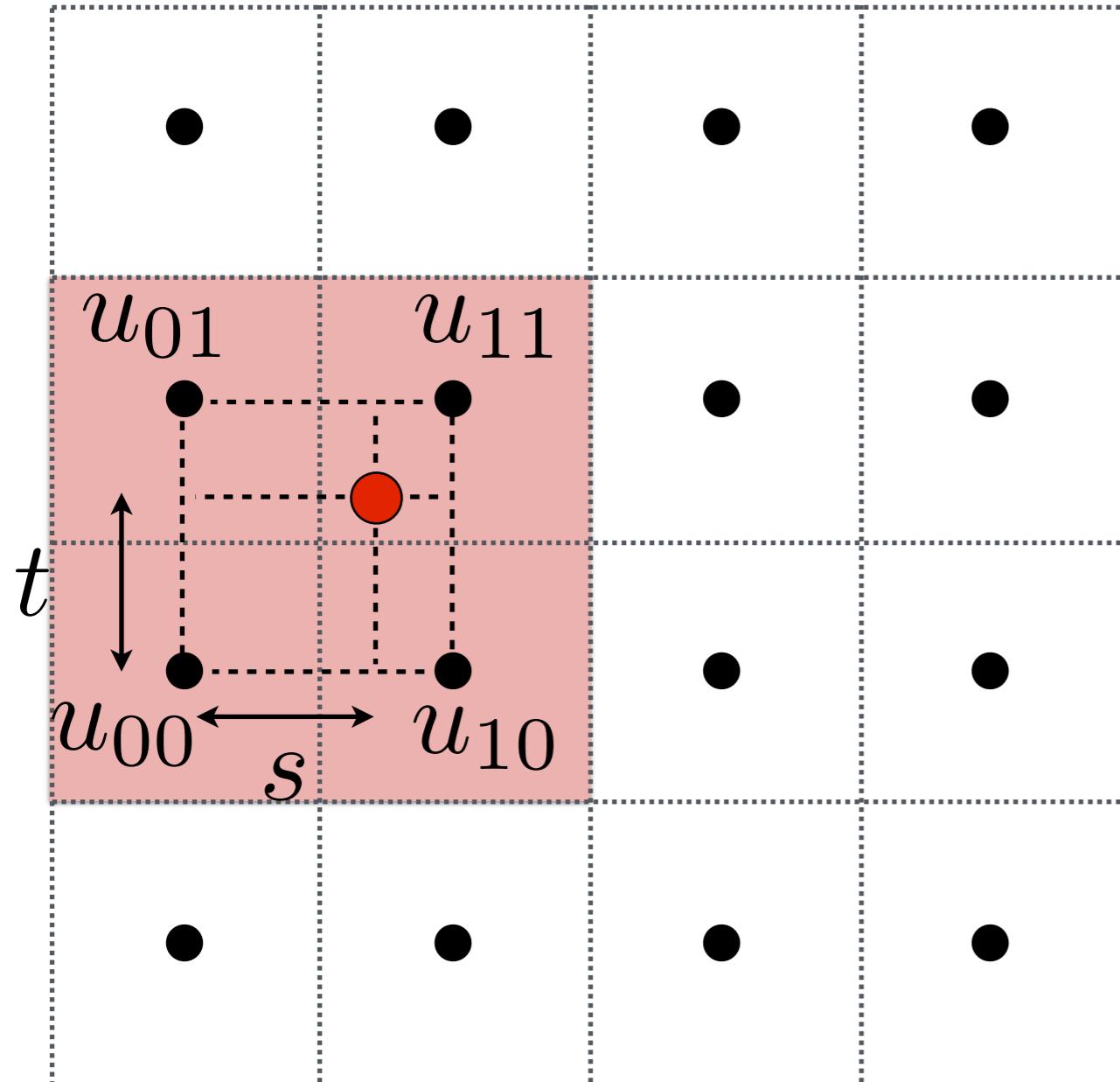
Black points indicate  
texture sample  
locations

# Bilinear Interpolation



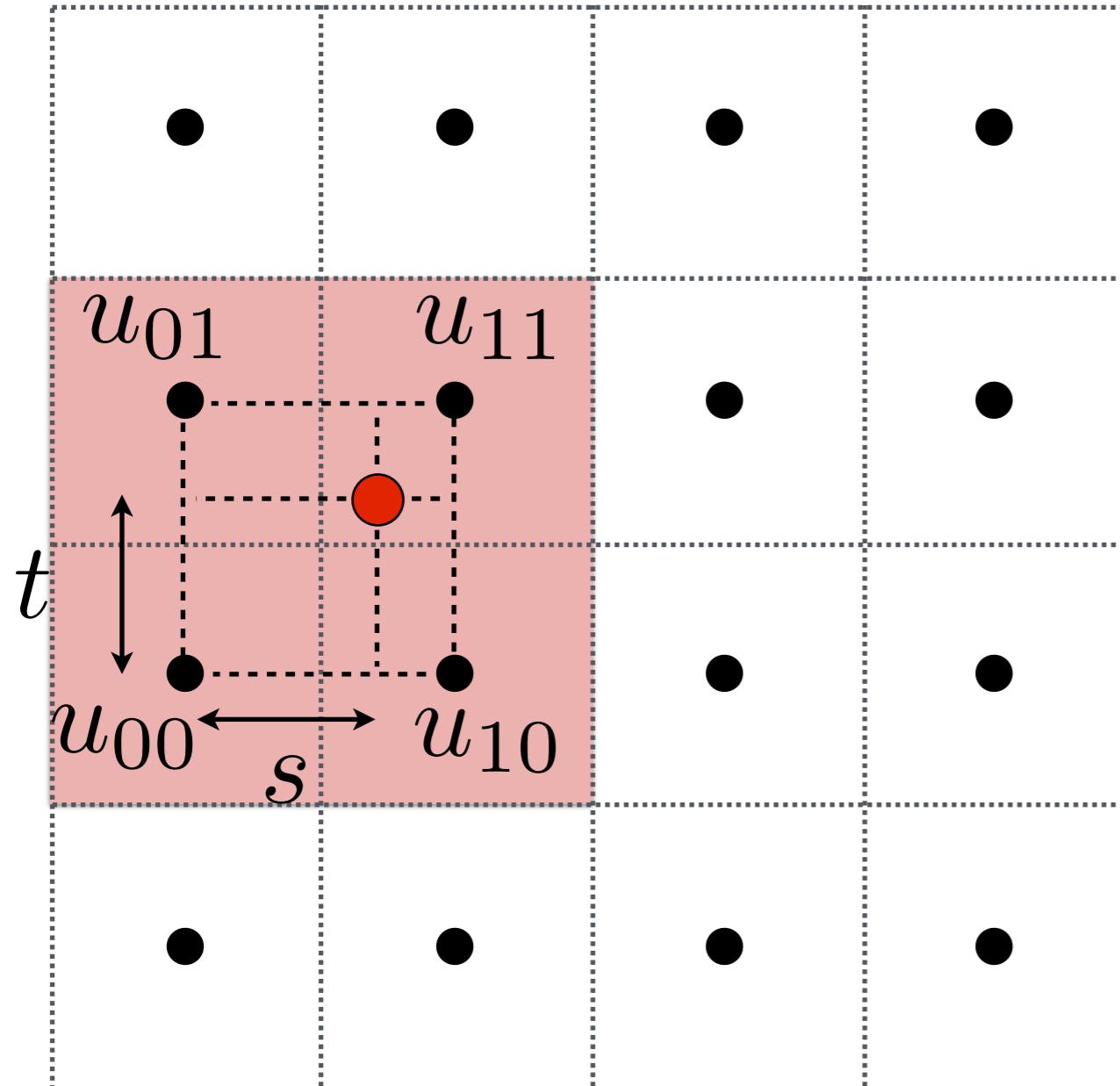
**Take 4 nearest sample locations, with texture values as labeled.**

# Bilinear Interpolation



And fractional offsets,  
( $s, t$ ) as shown

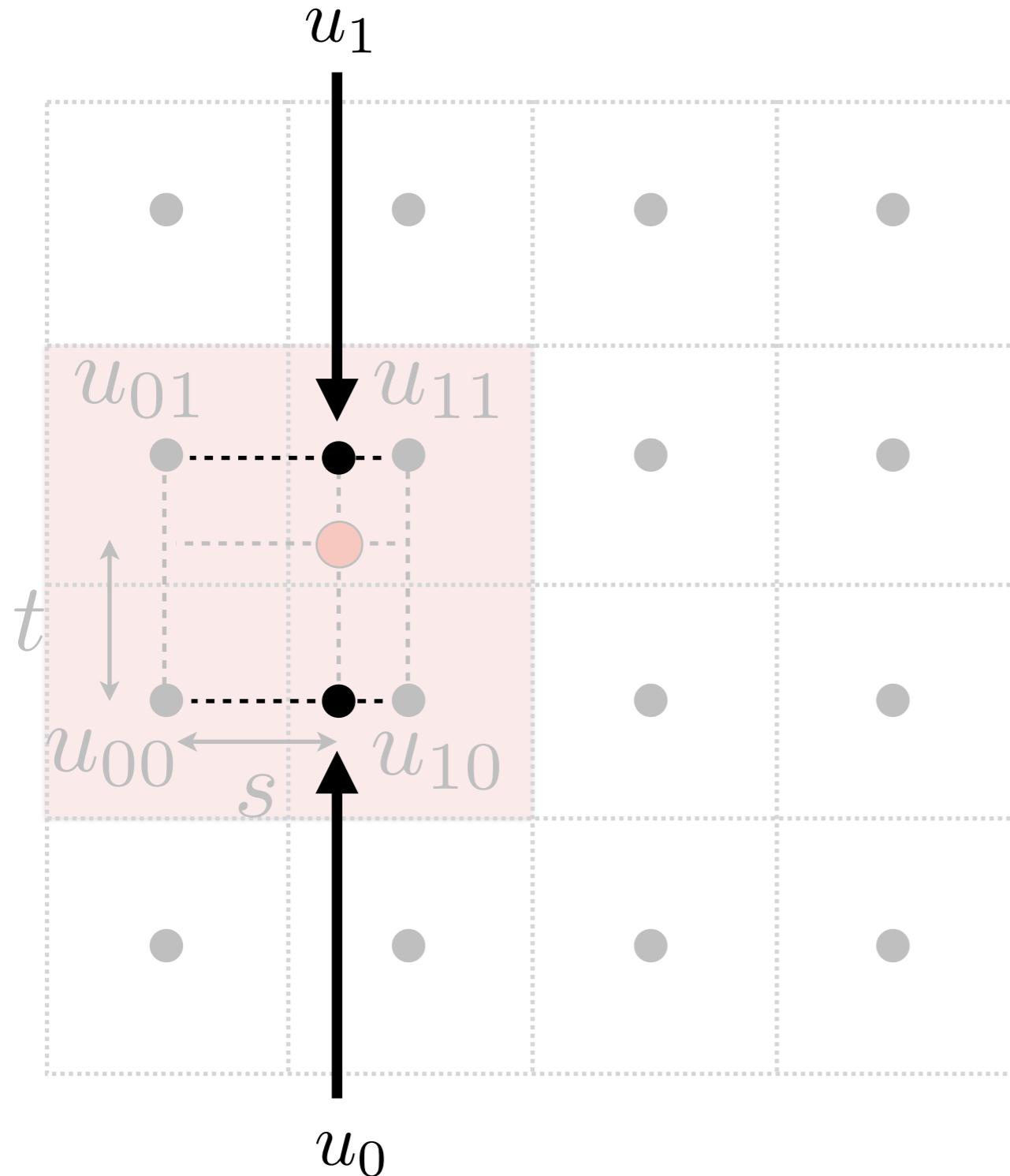
# Bilinear Interpolation



**Linear interpolation (1D)**

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

# Bilinear Interpolation



**Linear interpolation (1D)**

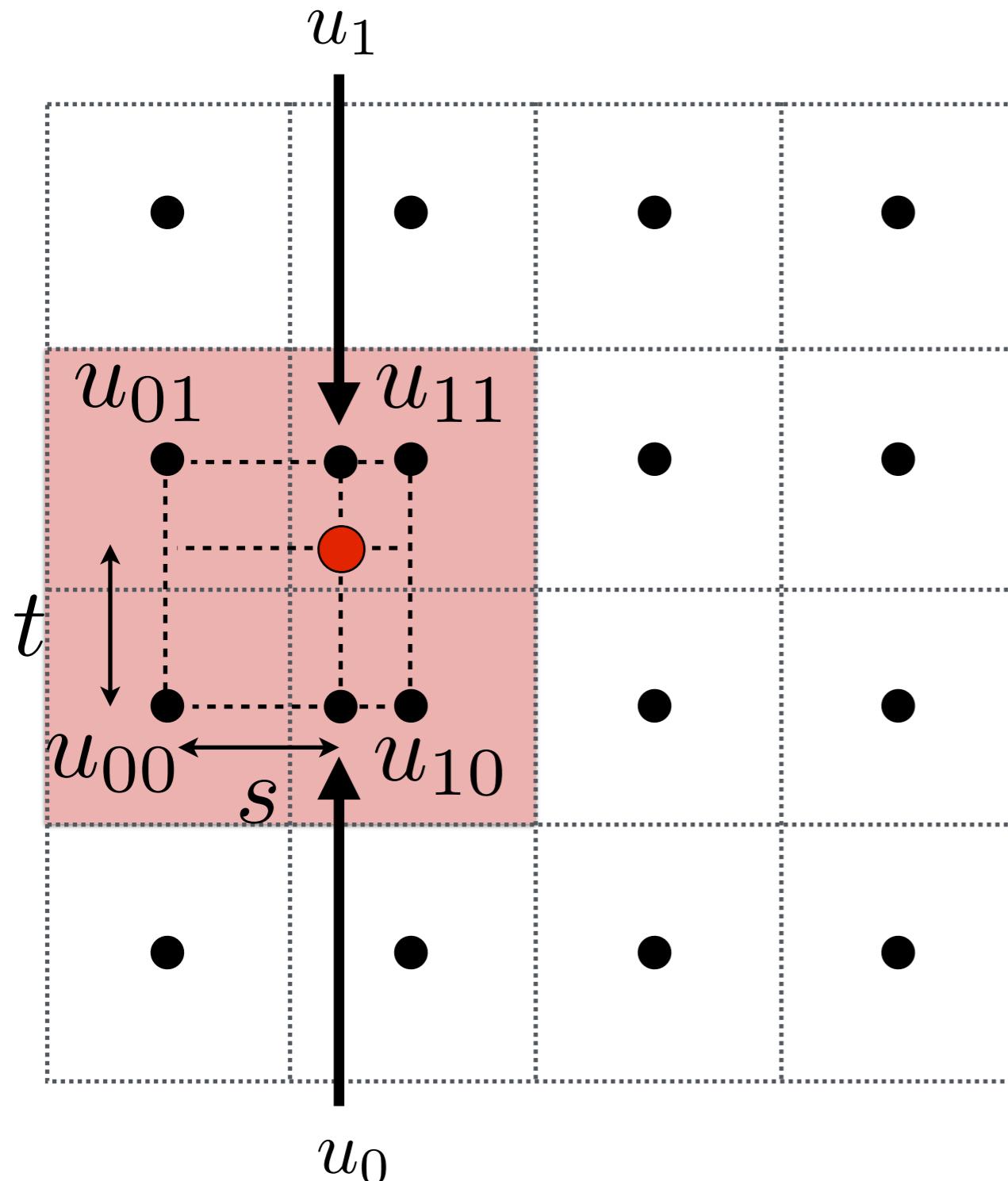
$$\text{lerp}(x, v_0, v_1) = v_0 + x(v_1 - v_0)$$

**Two helper lerps (horizontal)**

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

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

# Bilinear Interpolation



**Linear interpolation (1D)**

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

**Two helper lerps**

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

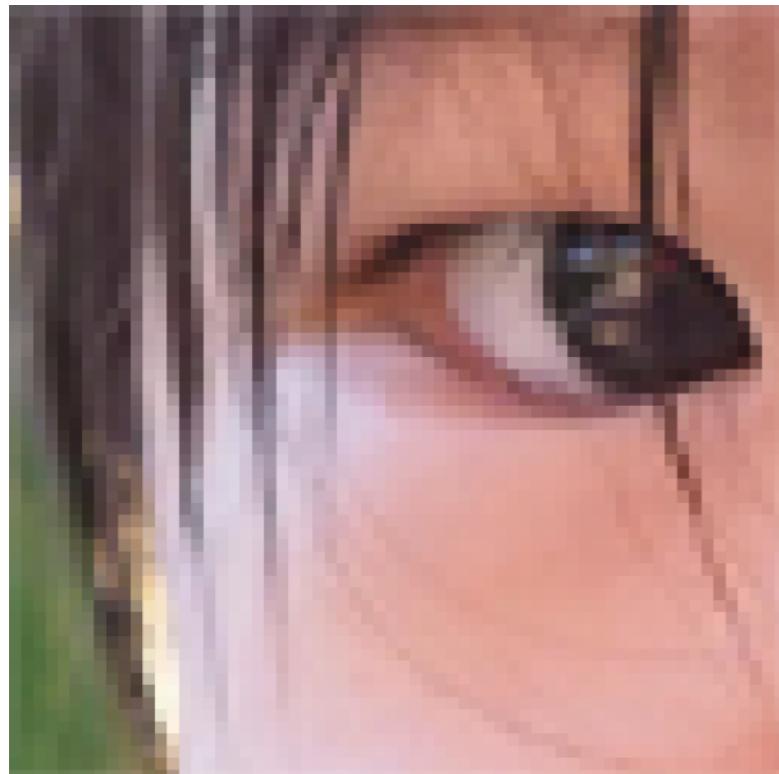
$$u_1 = \text{lerp}(s, u_{01}, u_{11})$$

**Final vertical lerp, to get result:**

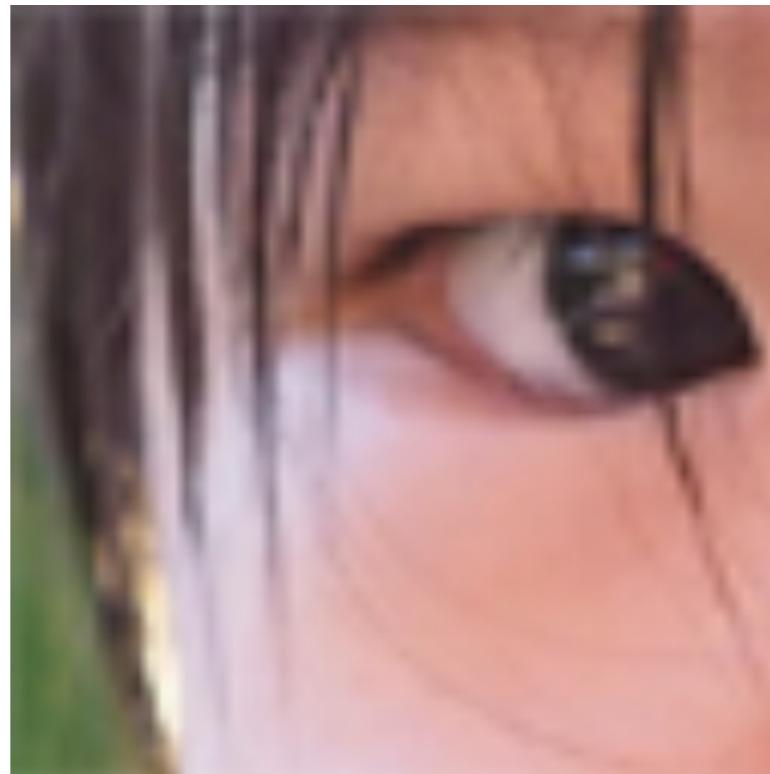
$$f(x, y) = \text{lerp}(t, u_0, u_1)$$

# Texture Magnification - Easy Case

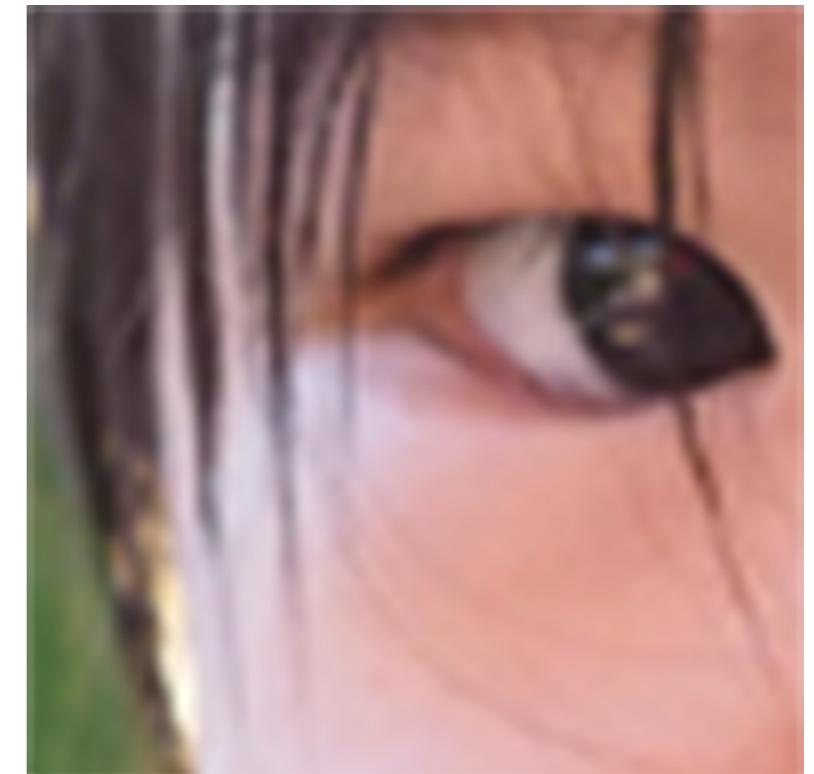
Bilinear interpolation usually gives pretty good results at reasonable costs



**Nearest**



**Bilinear**

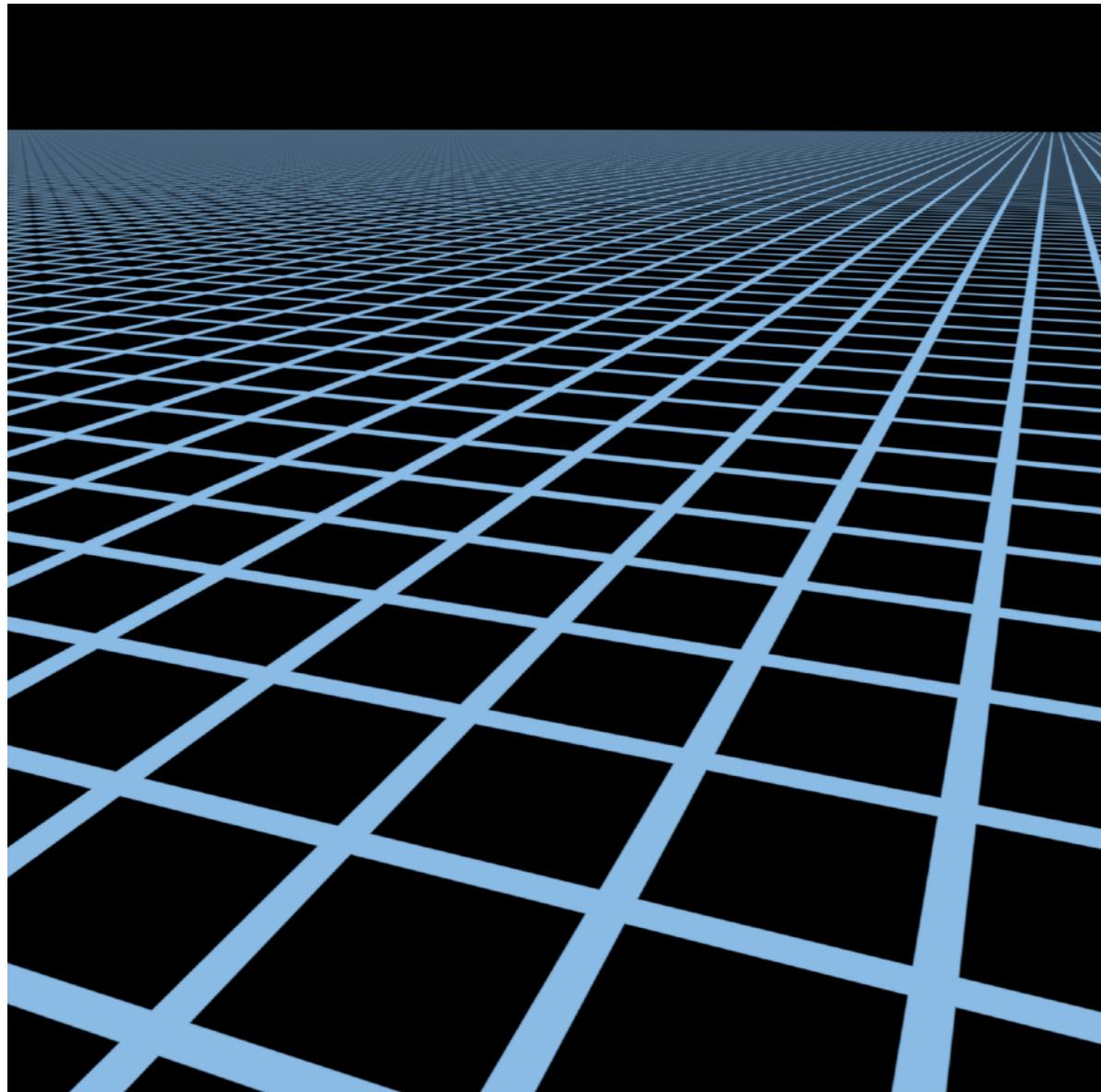


**Bicubic**

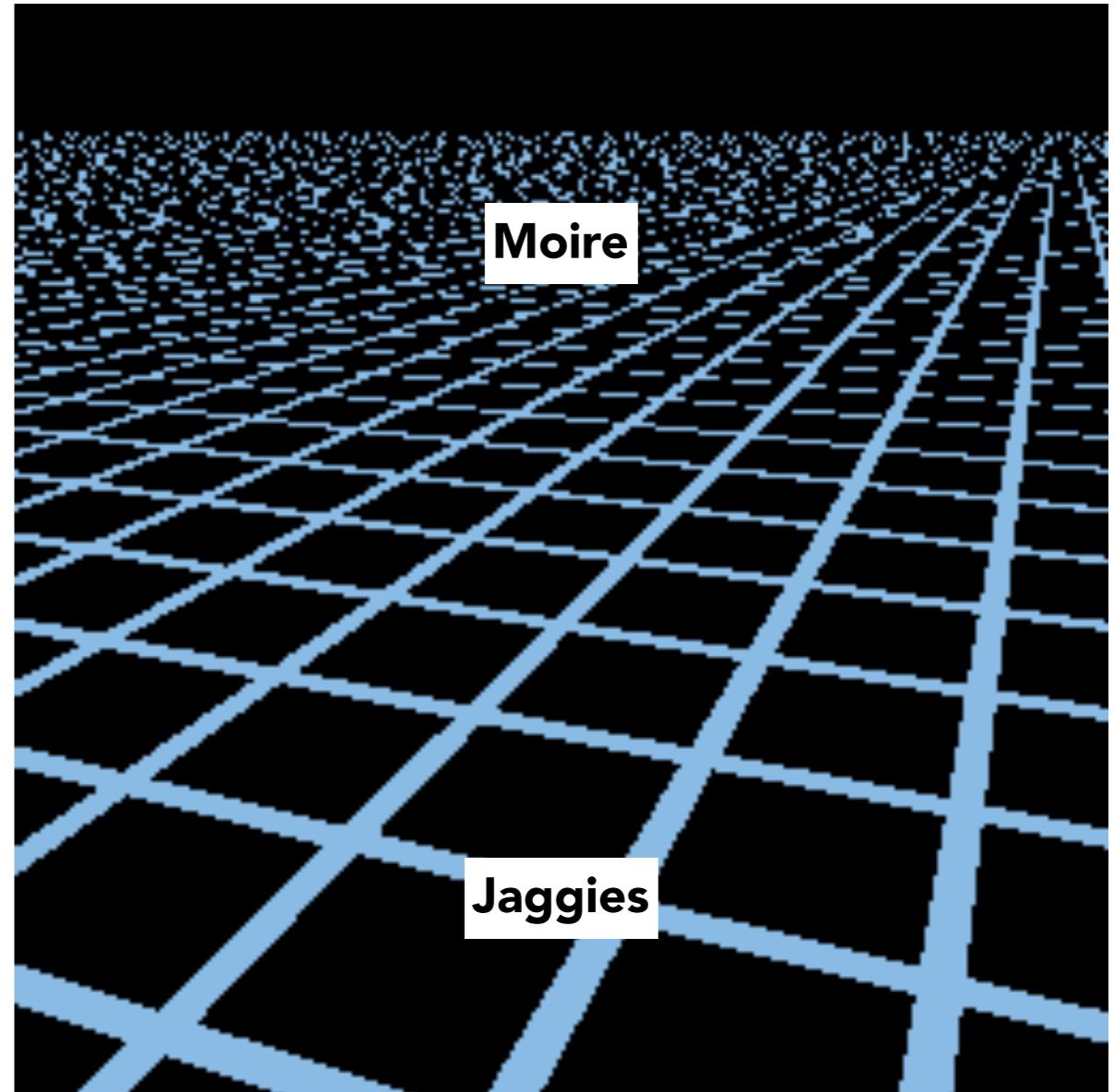
# Texture Magnification (**hard case**)

(What if the texture is too large?)

# Point Sampling Textures — Problem

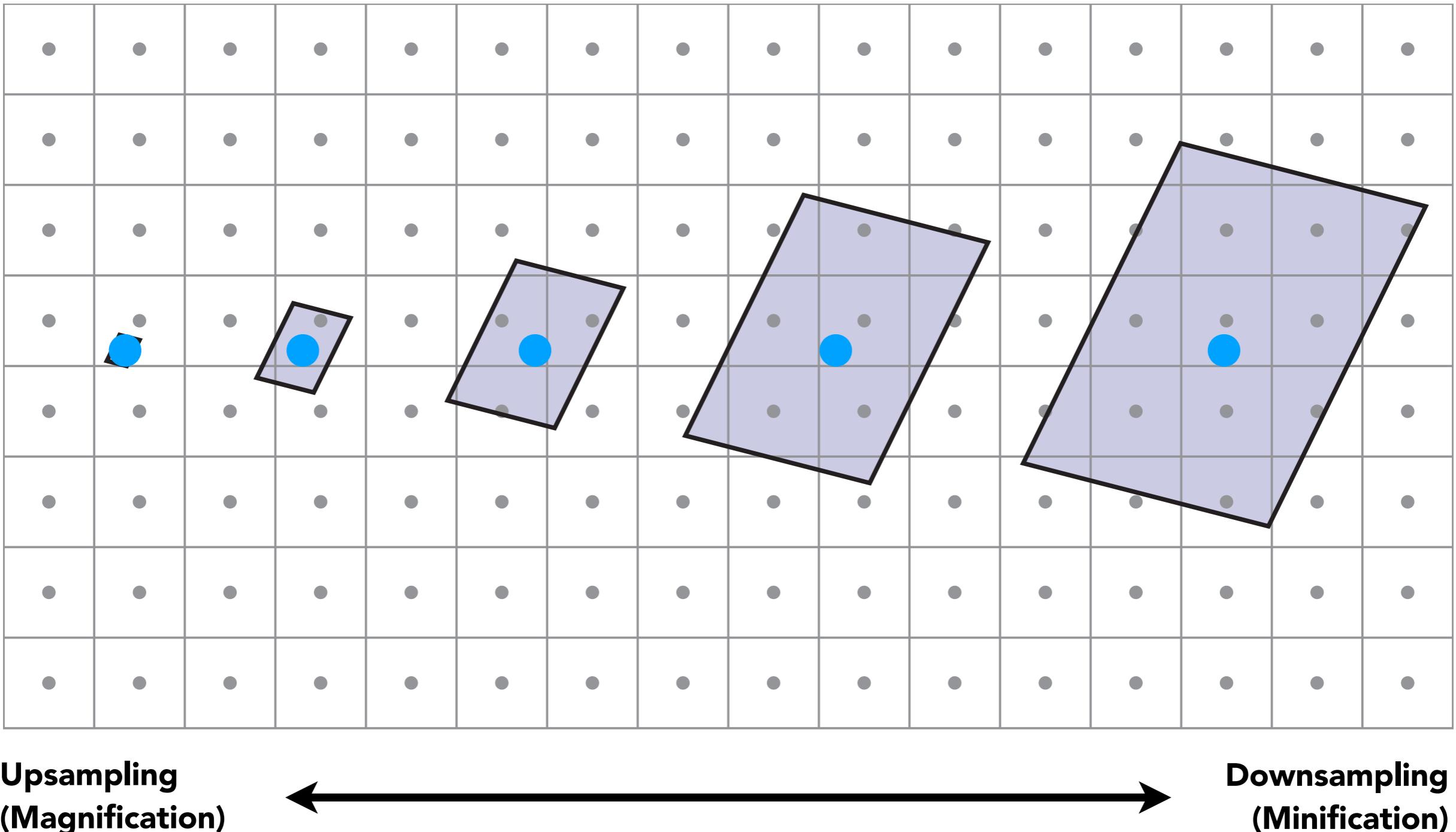


**Reference**

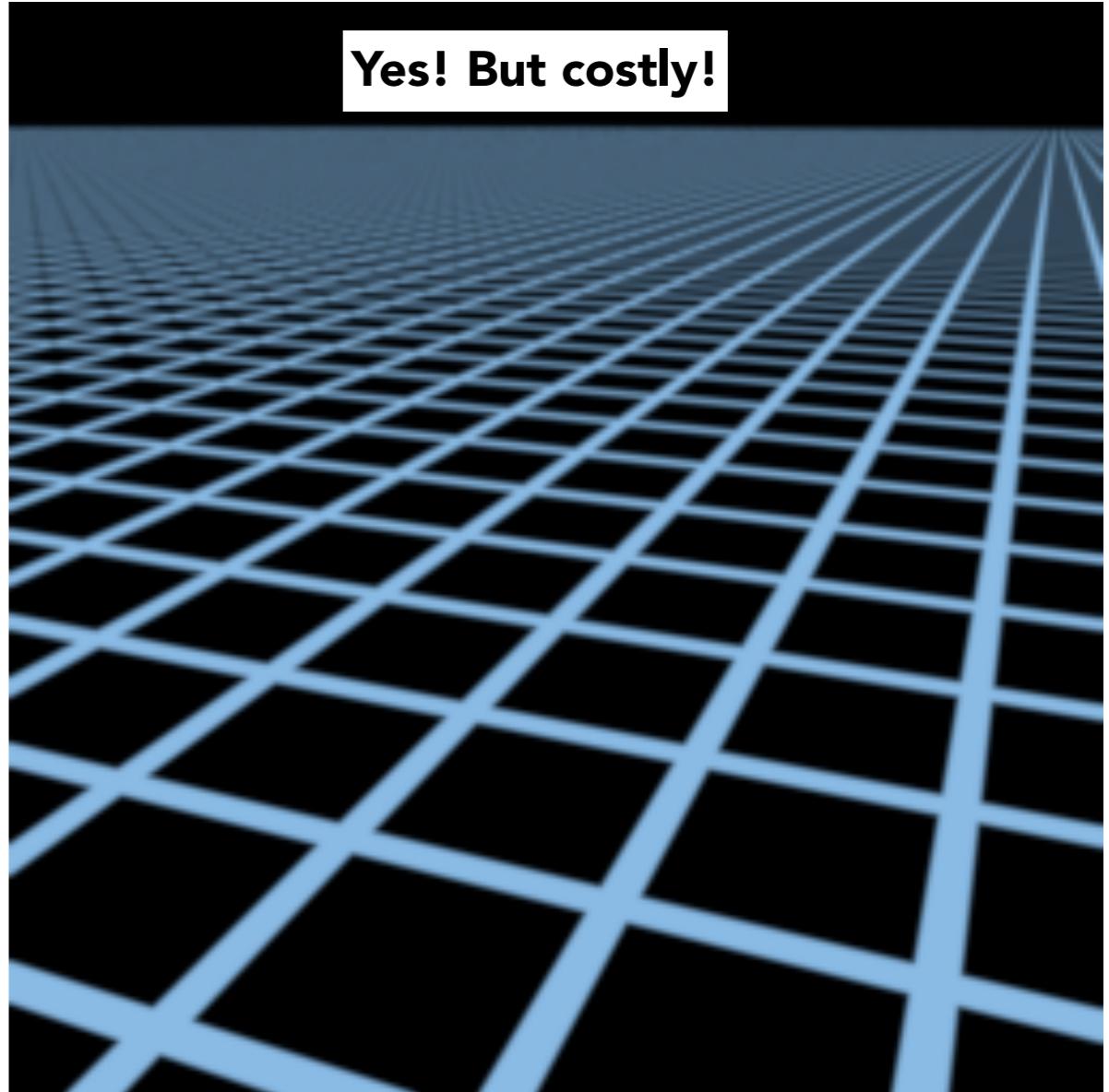
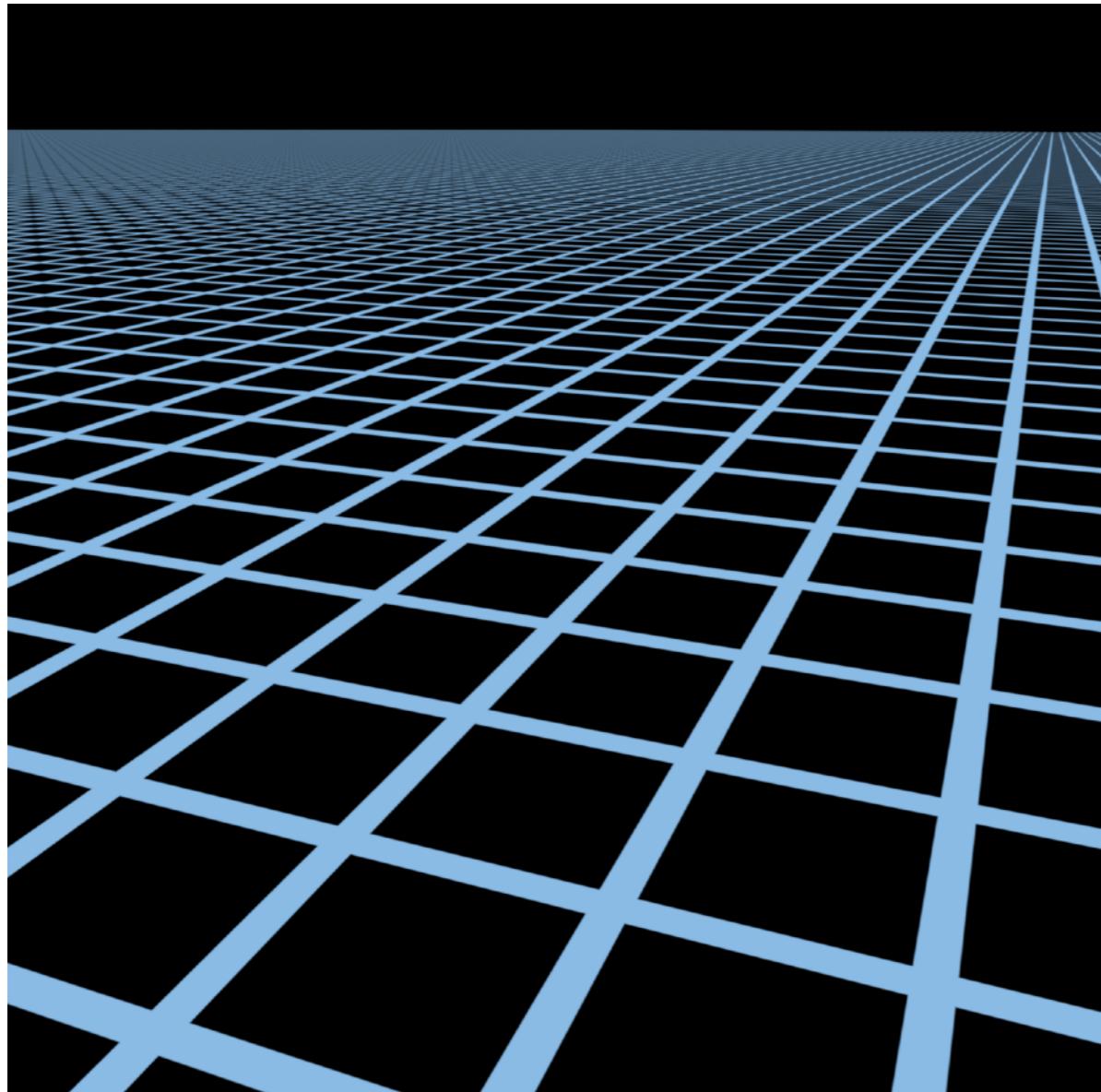


**Point sampled**

# Screen Pixel “Footprint” in Texture



# Will Supersampling Do Antialiasing?



512x supersampling

# Antialiasing — Supersampling?

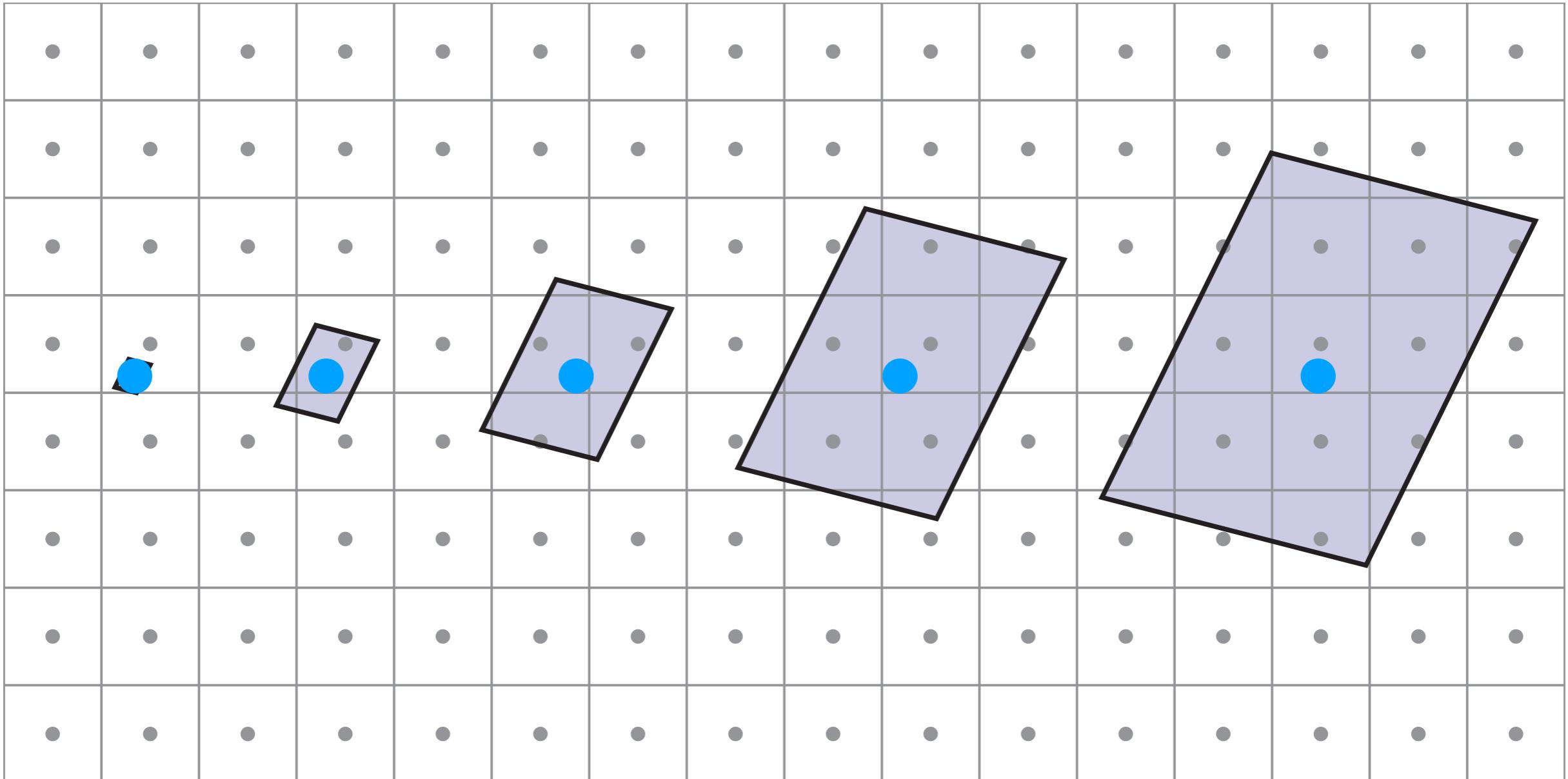
Will supersampling work?

- 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

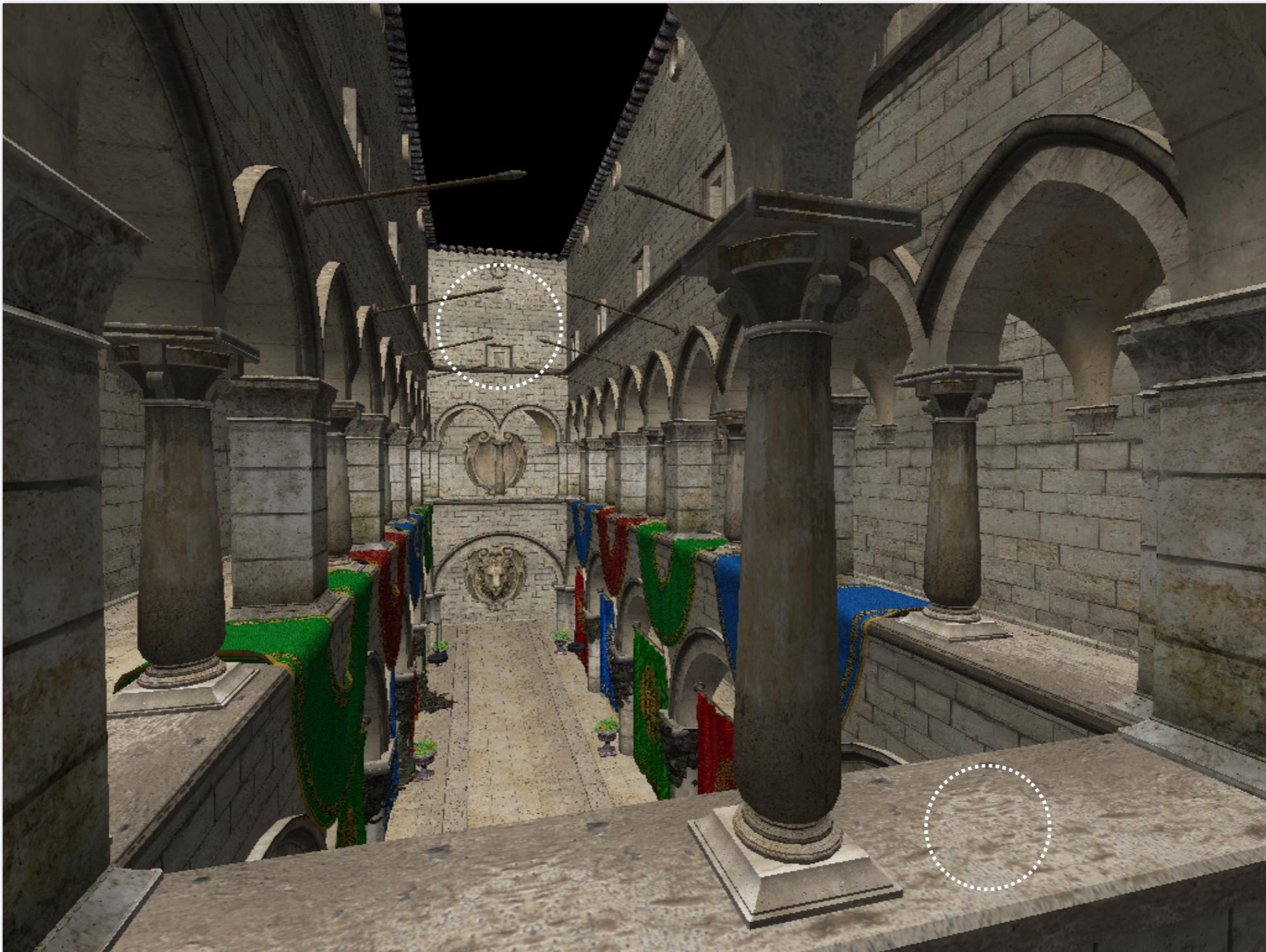
Let's understand this problem in another way

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

# Point Query vs. (Avg.) Range Query



# Different Pixels -> Different-Sized Footprints



# Mipmap

Allowing (fast, approx., square) range queries

近似

正方形

# Mipmap (L. Williams 83)

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



**Level 0 = 128x128**



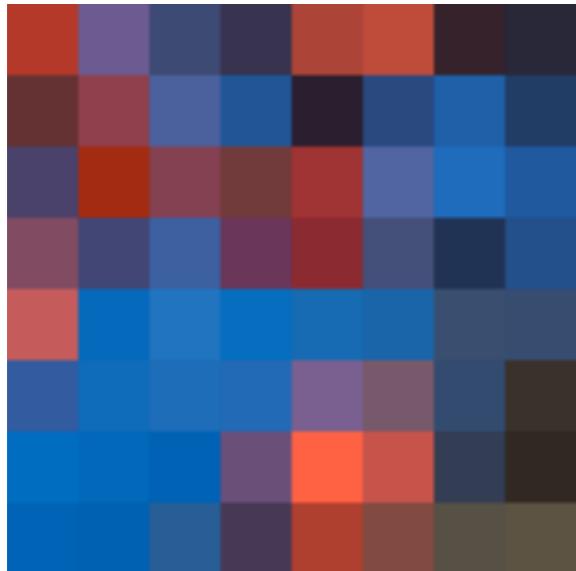
**Level 1 = 64x64**



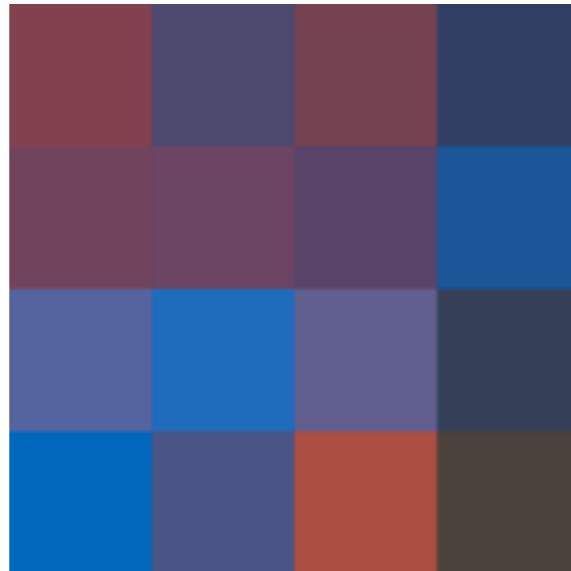
**Level 2 = 32x32**



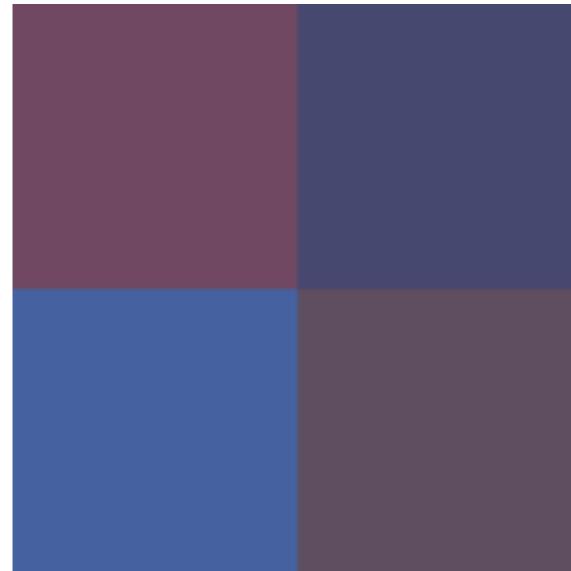
**Level 3 = 16x16**



**Level 4 = 8x8**



**Level 5 = 4x4**



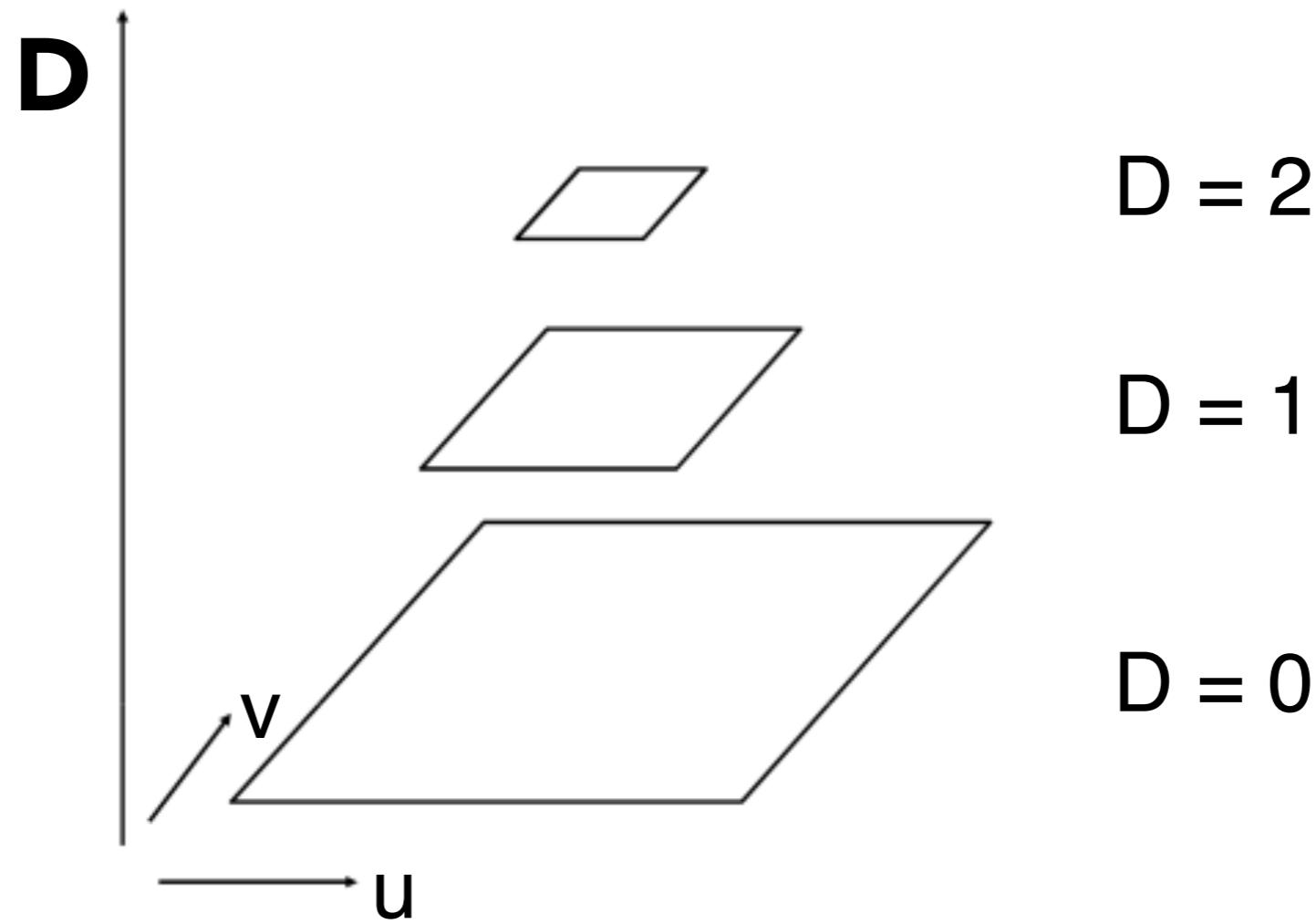
**Level 6 = 2x2**



**Level 7 = 1x1**

# Mipmap (L. Williams 83)

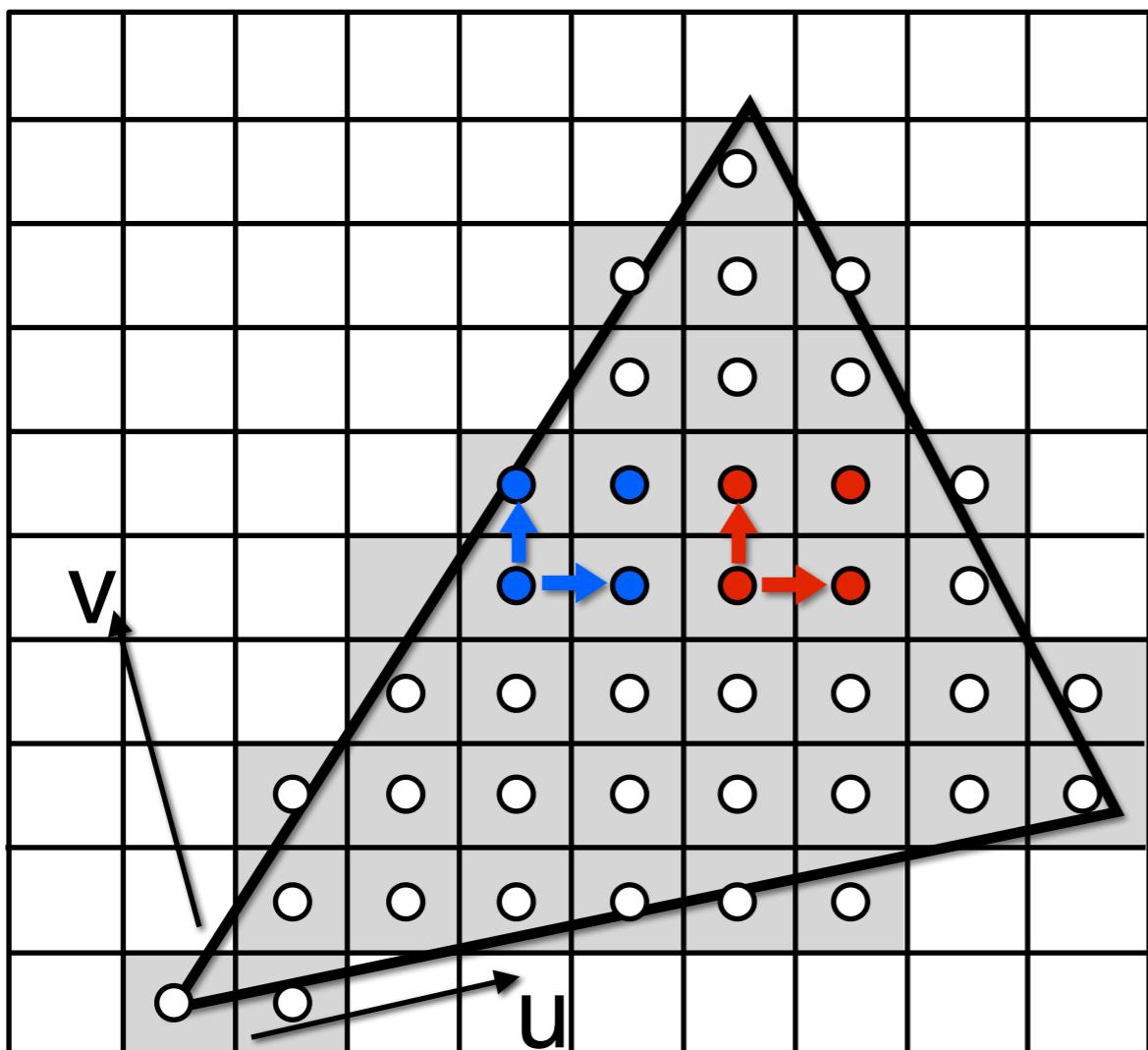
的  
存儲



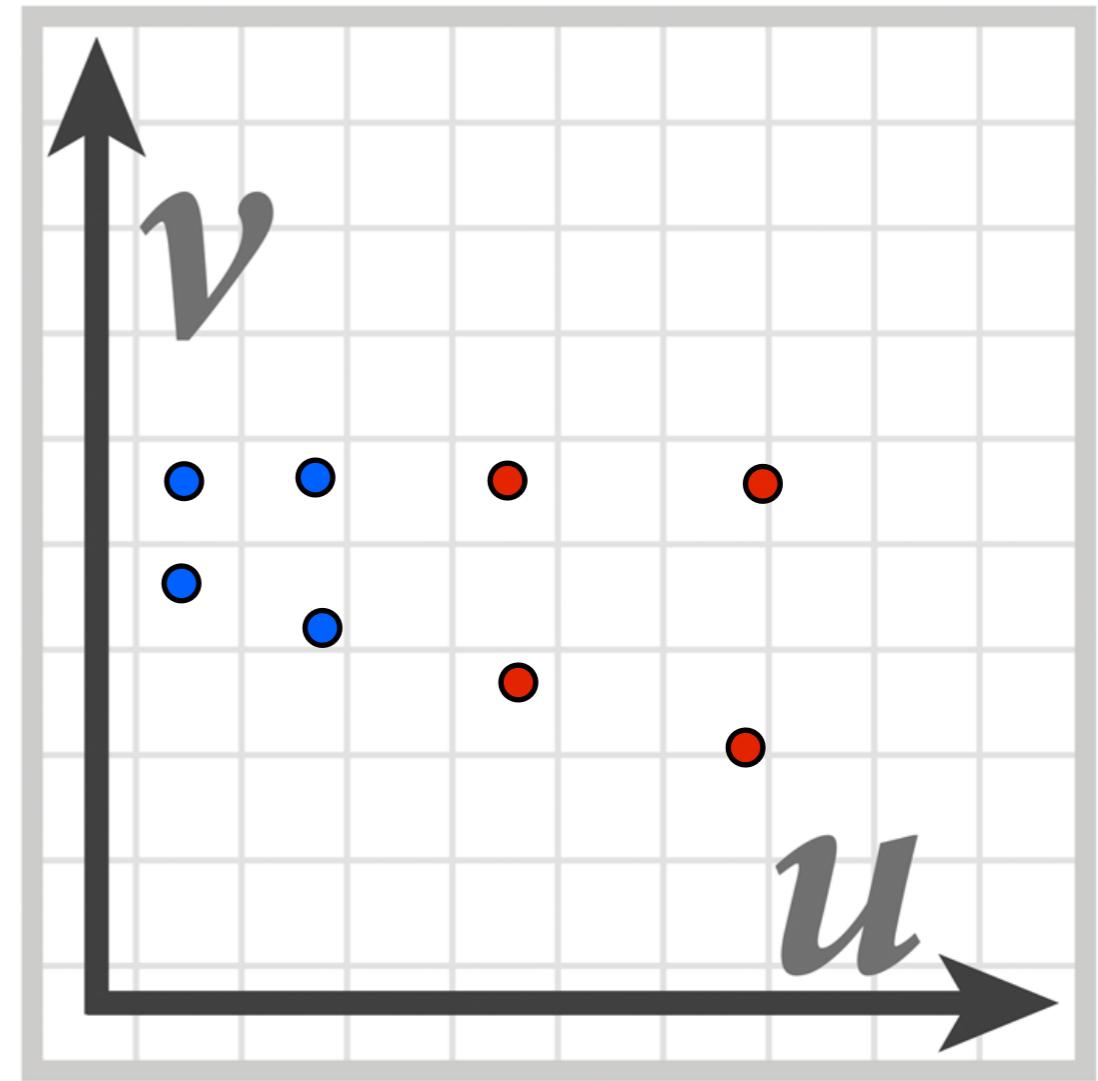
“Mip hierarchy”  
level = D

What is the storage overhead of a mipmap?

# Computing Mipmap Level D



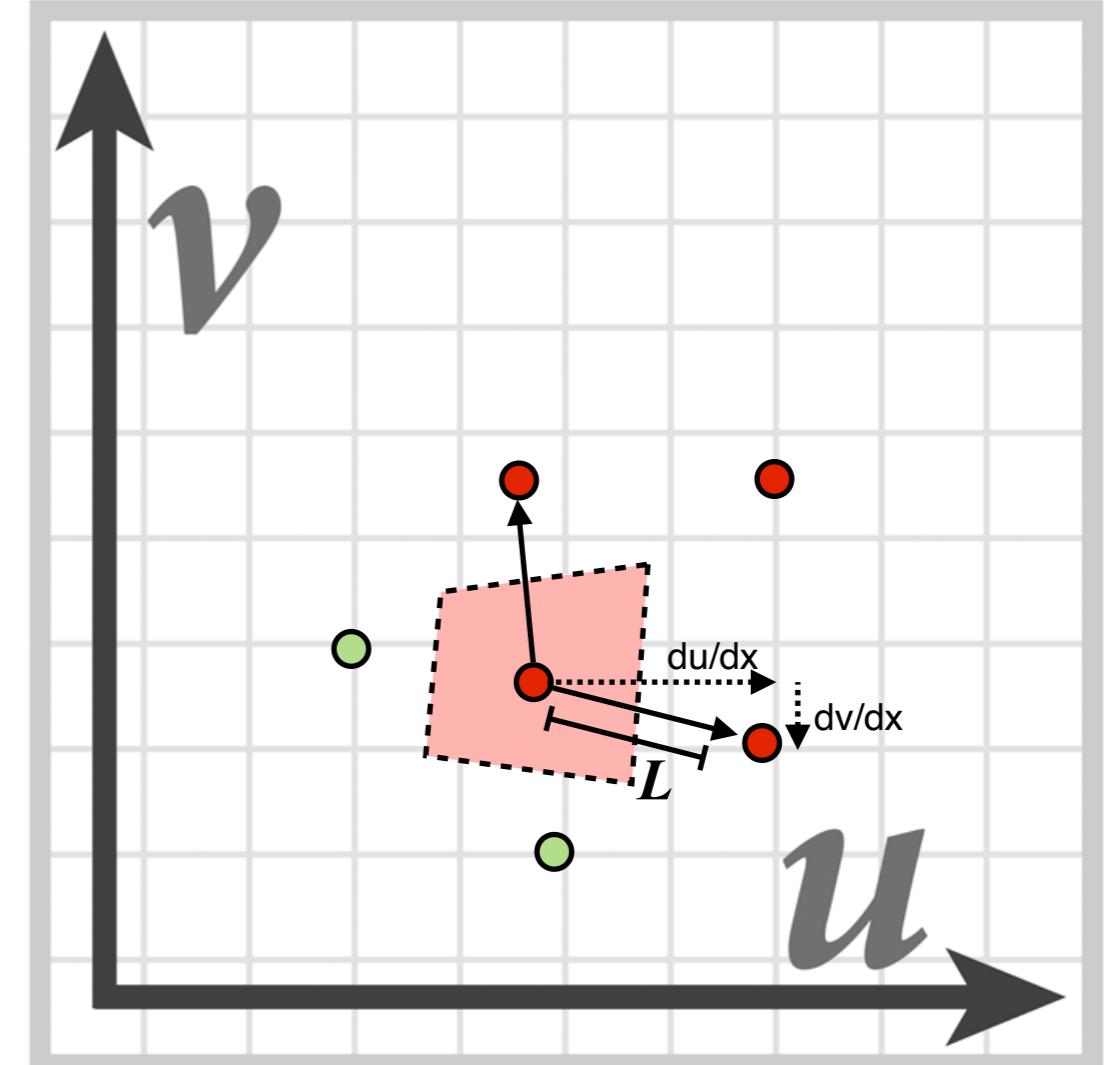
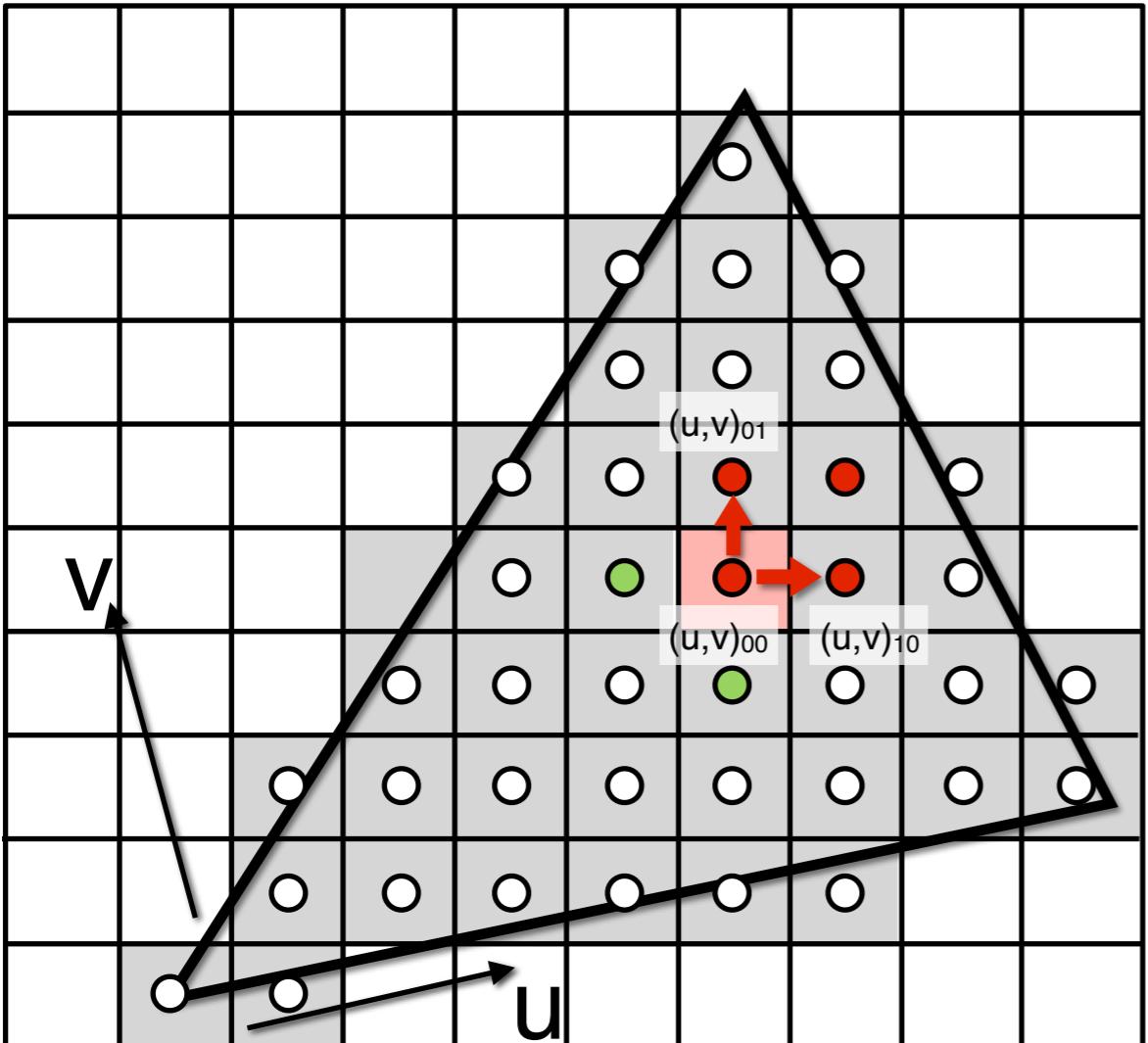
Screen space ( $x,y$ )



Texture space ( $u,v$ )

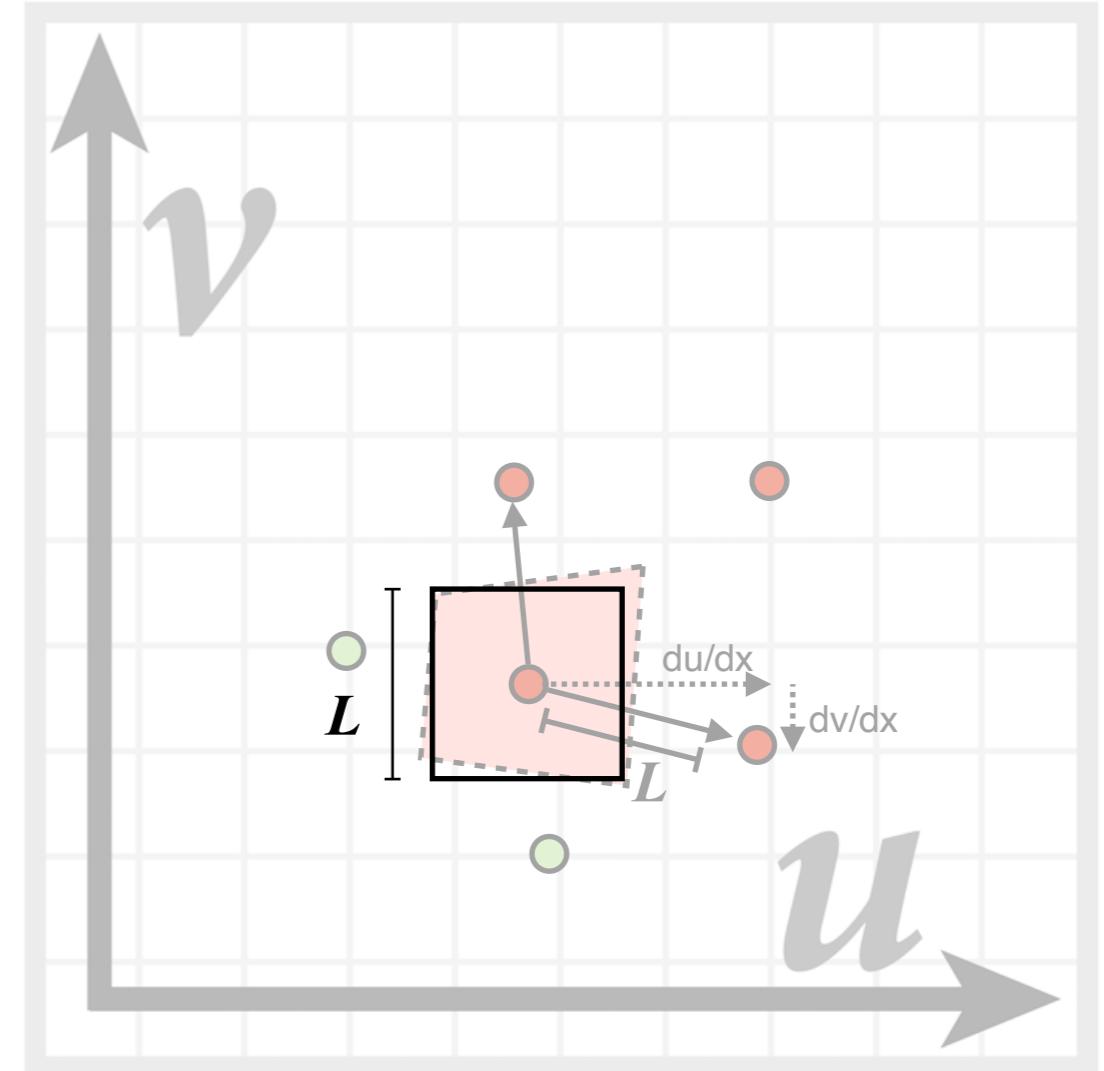
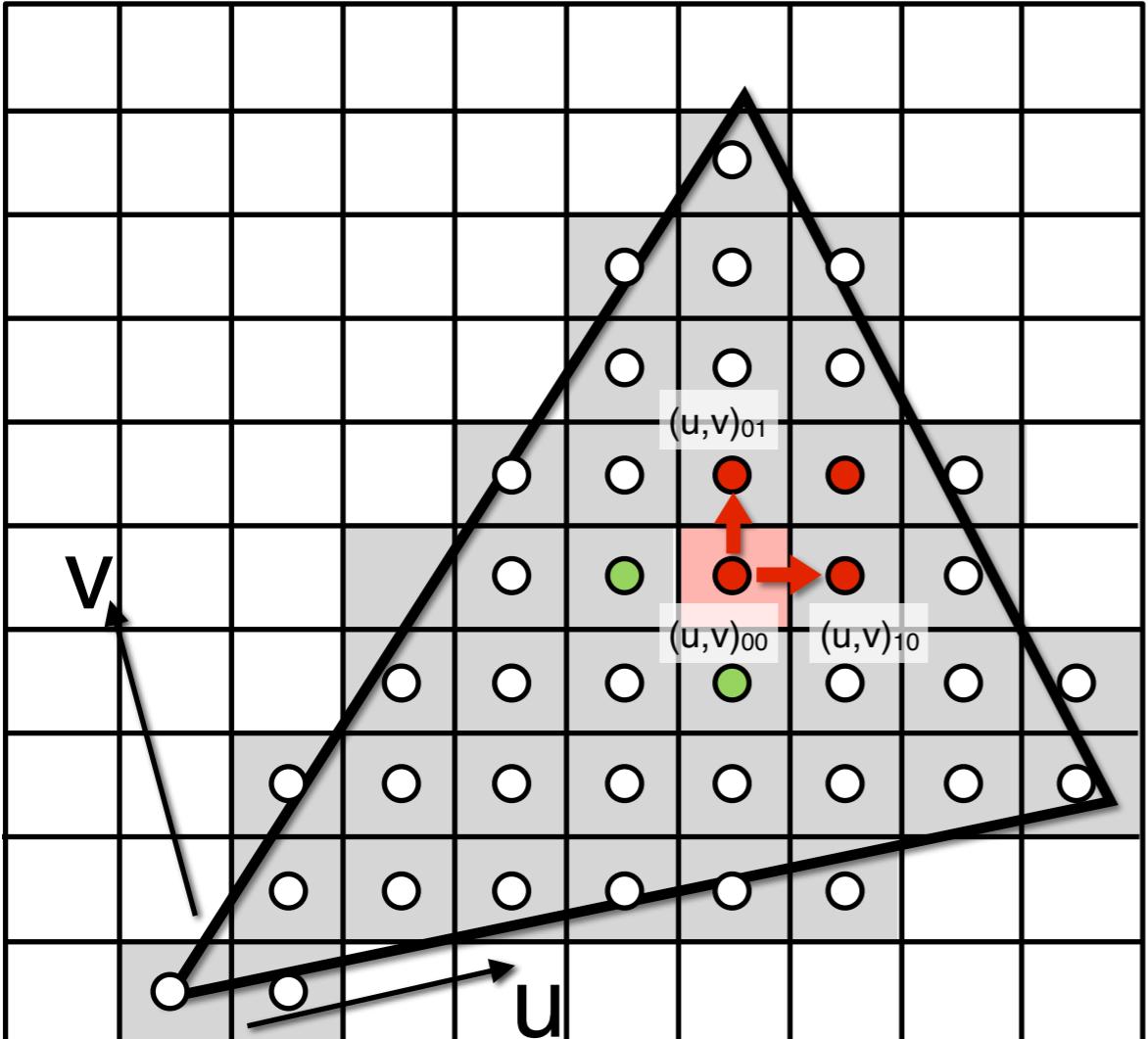
Estimate texture footprint using texture coordinates of neighboring screen samples

# Computing Mipmap Level D



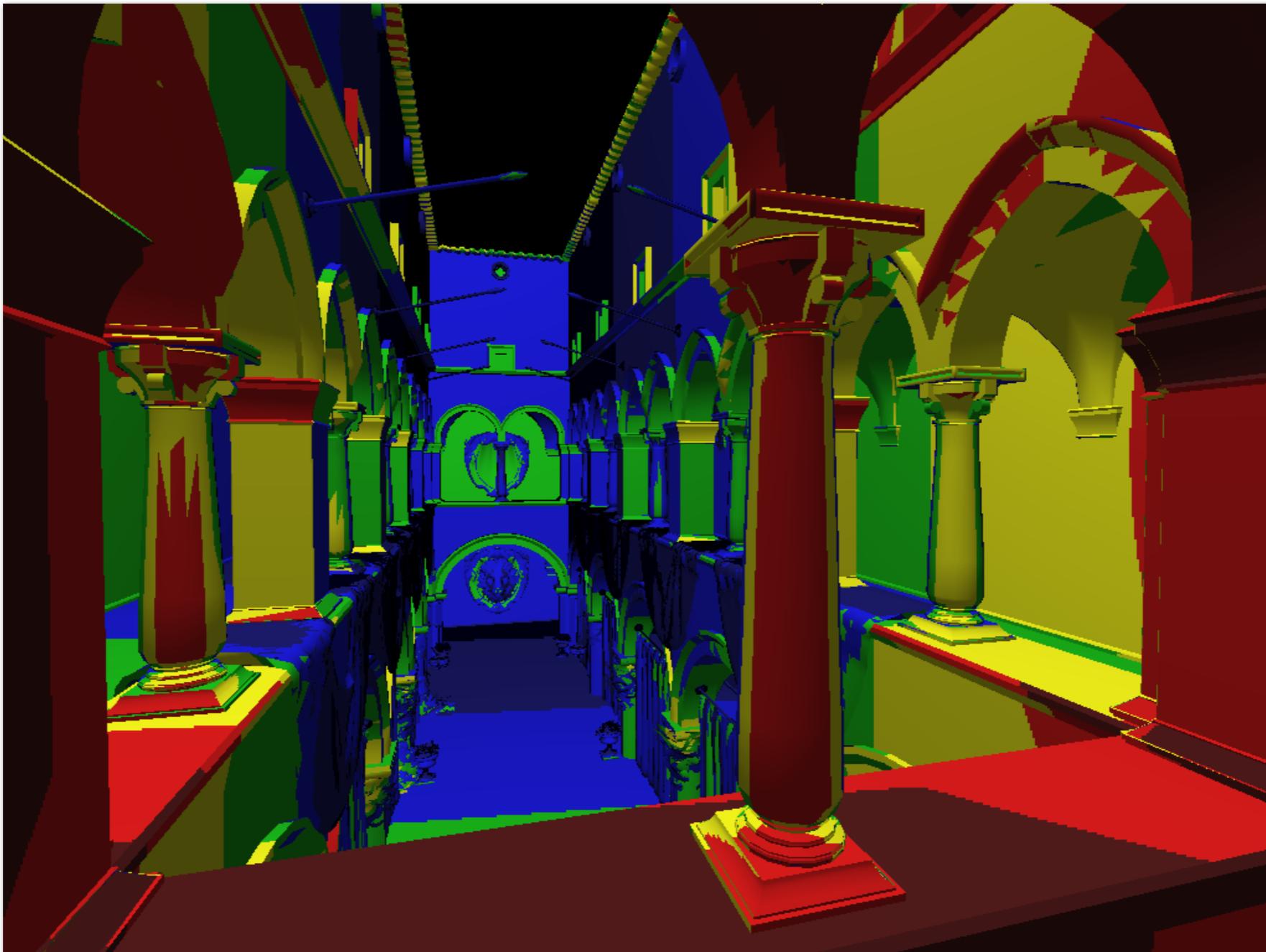
$$D = \log_2 L \quad L = \max \left( \sqrt{\left( \frac{du}{dx} \right)^2 + \left( \frac{dv}{dx} \right)^2}, \sqrt{\left( \frac{du}{dy} \right)^2 + \left( \frac{dv}{dy} \right)^2} \right)$$

# Computing Mipmap Level D



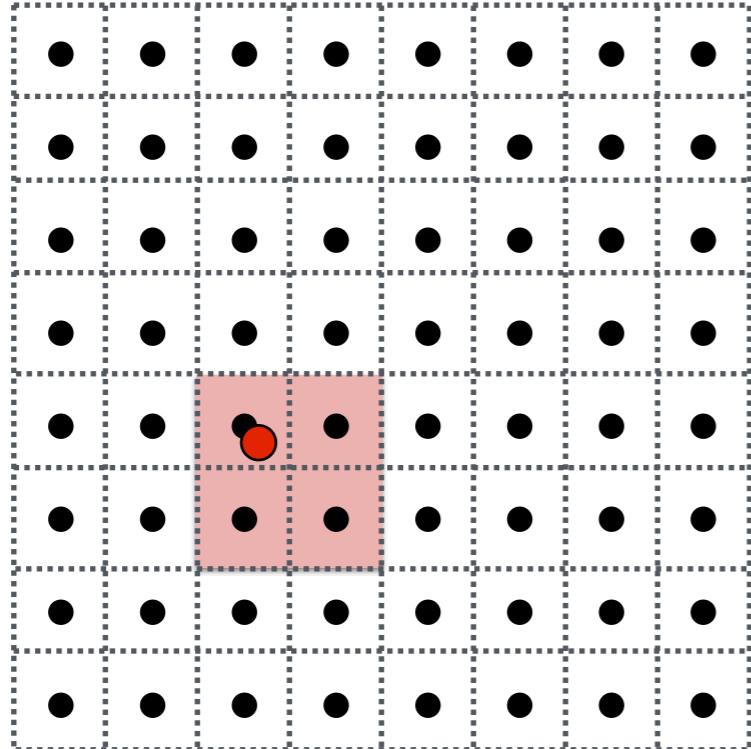
$$D = \log_2 L \quad L = \max \left( \sqrt{\left( \frac{du}{dx} \right)^2 + \left( \frac{dv}{dx} \right)^2}, \sqrt{\left( \frac{du}{dy} \right)^2 + \left( \frac{dv}{dy} \right)^2} \right)$$

# Visualization of Mipmap Level

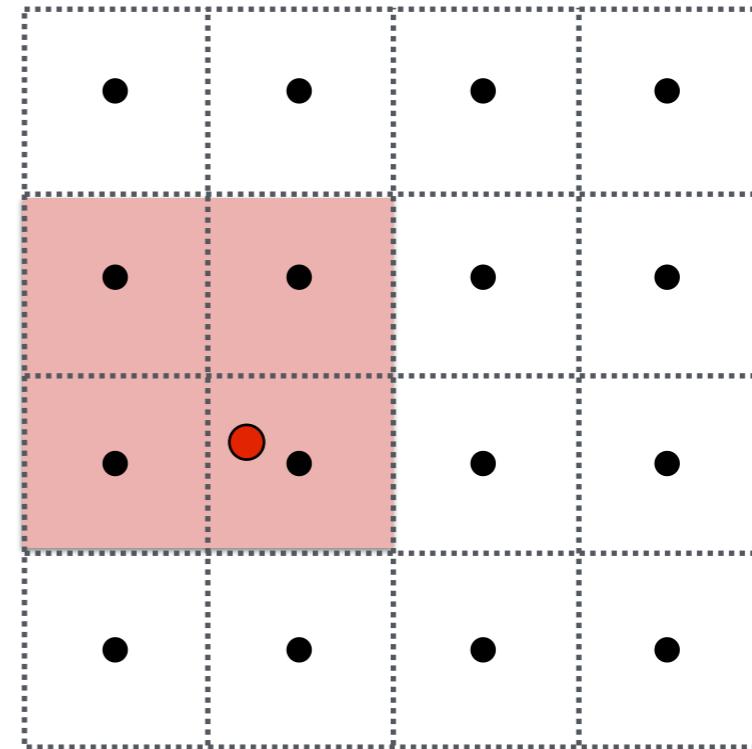


D rounded to nearest integer level

# Trilinear Interpolation



Mipmap Level D



Mipmap Level D+1

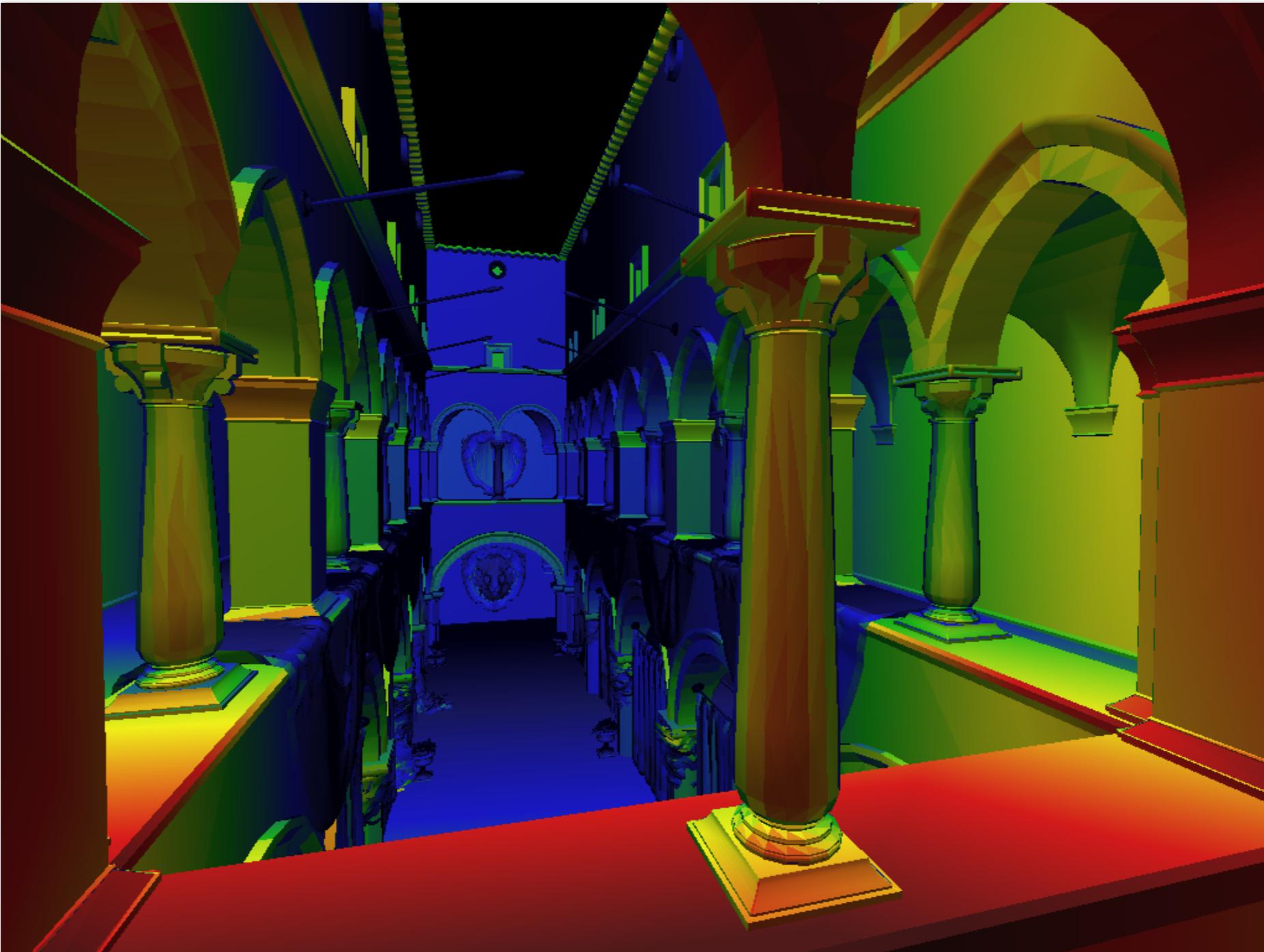
Bilinear result

层间插值

Bilinear result

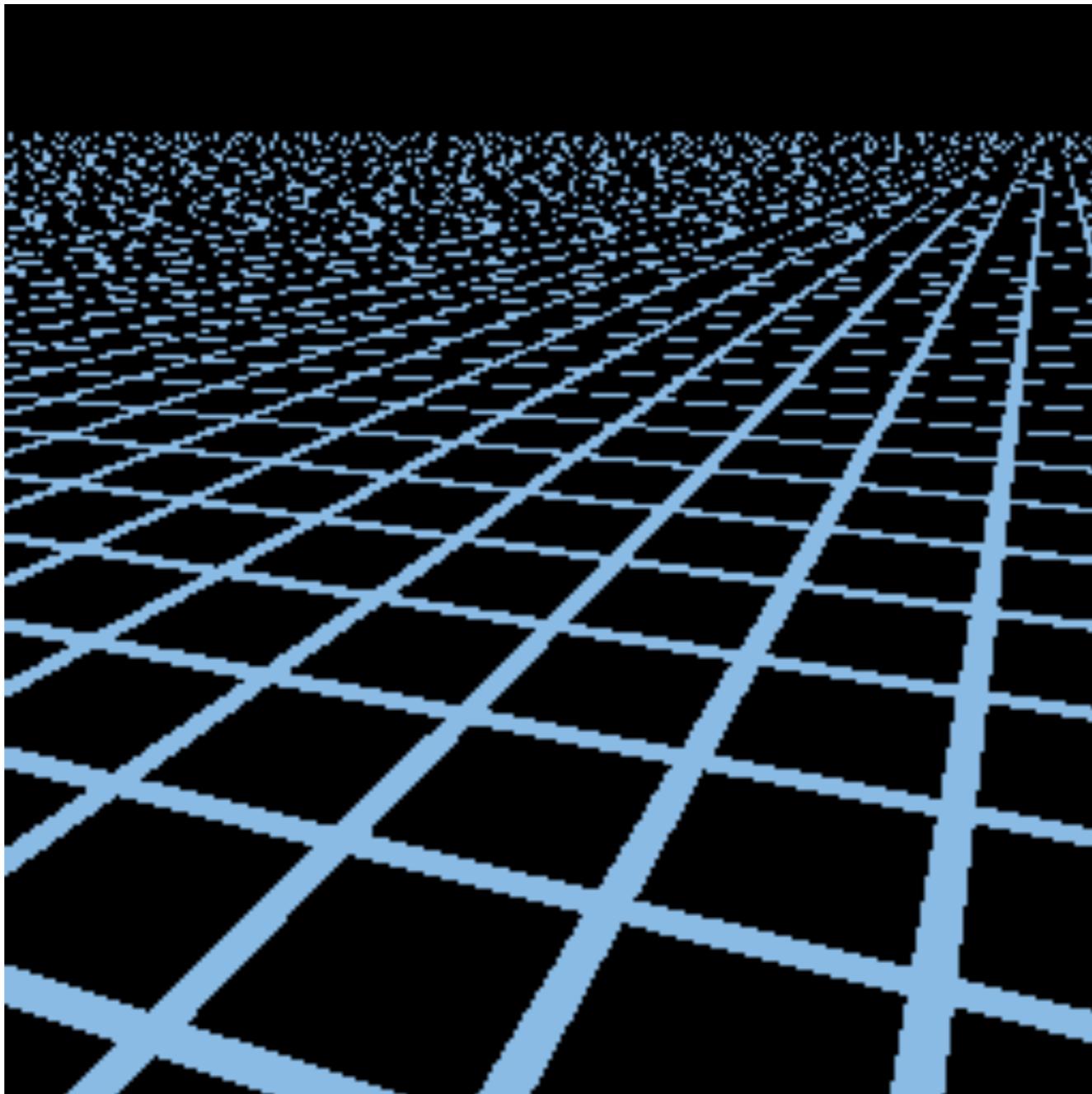
Linear interpolation based on continuous D value

# Visualization of Mipmap Level



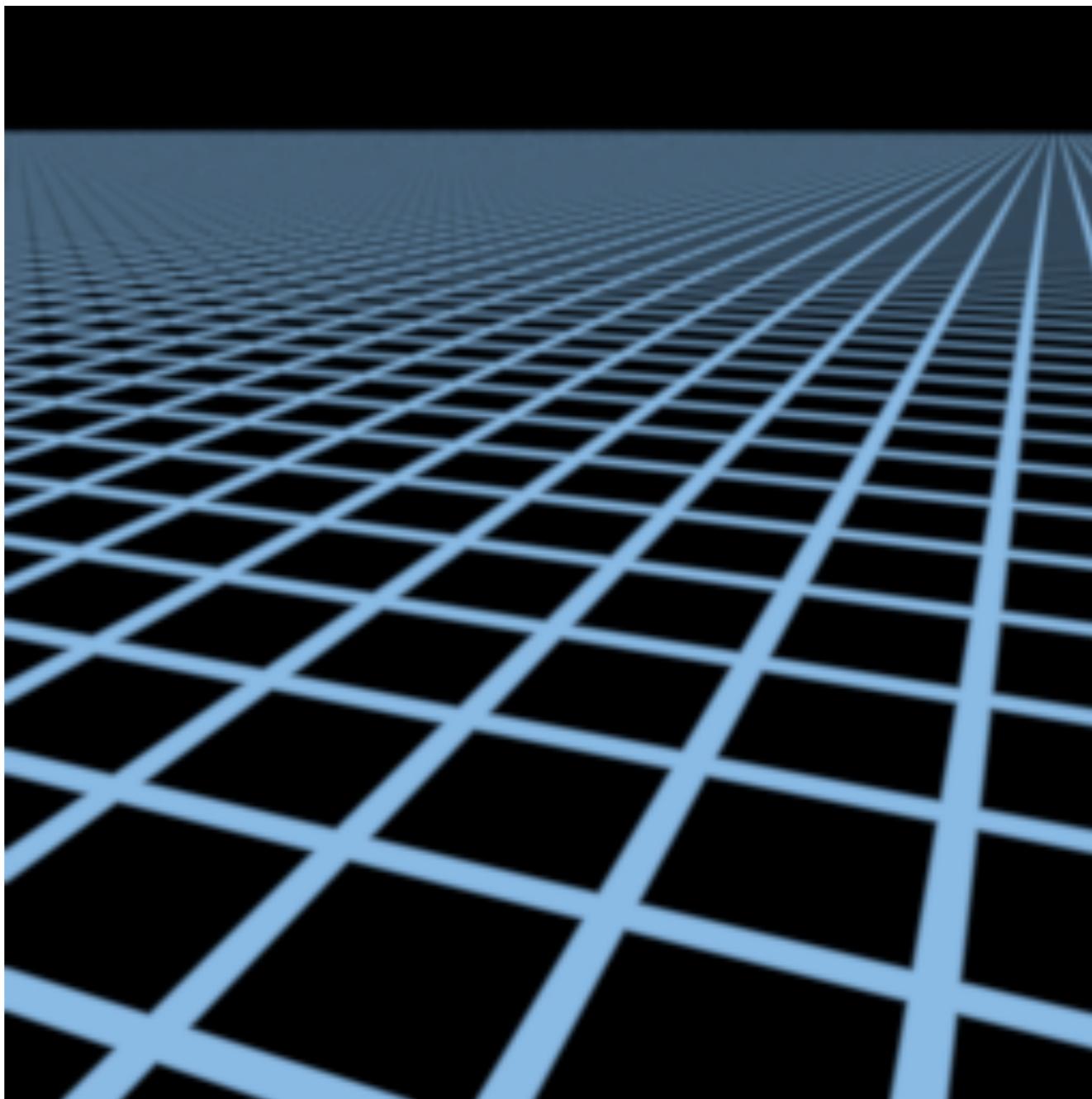
Trilinear filtering: visualization of continuous D

# Mipmap Limitations



Point sampling

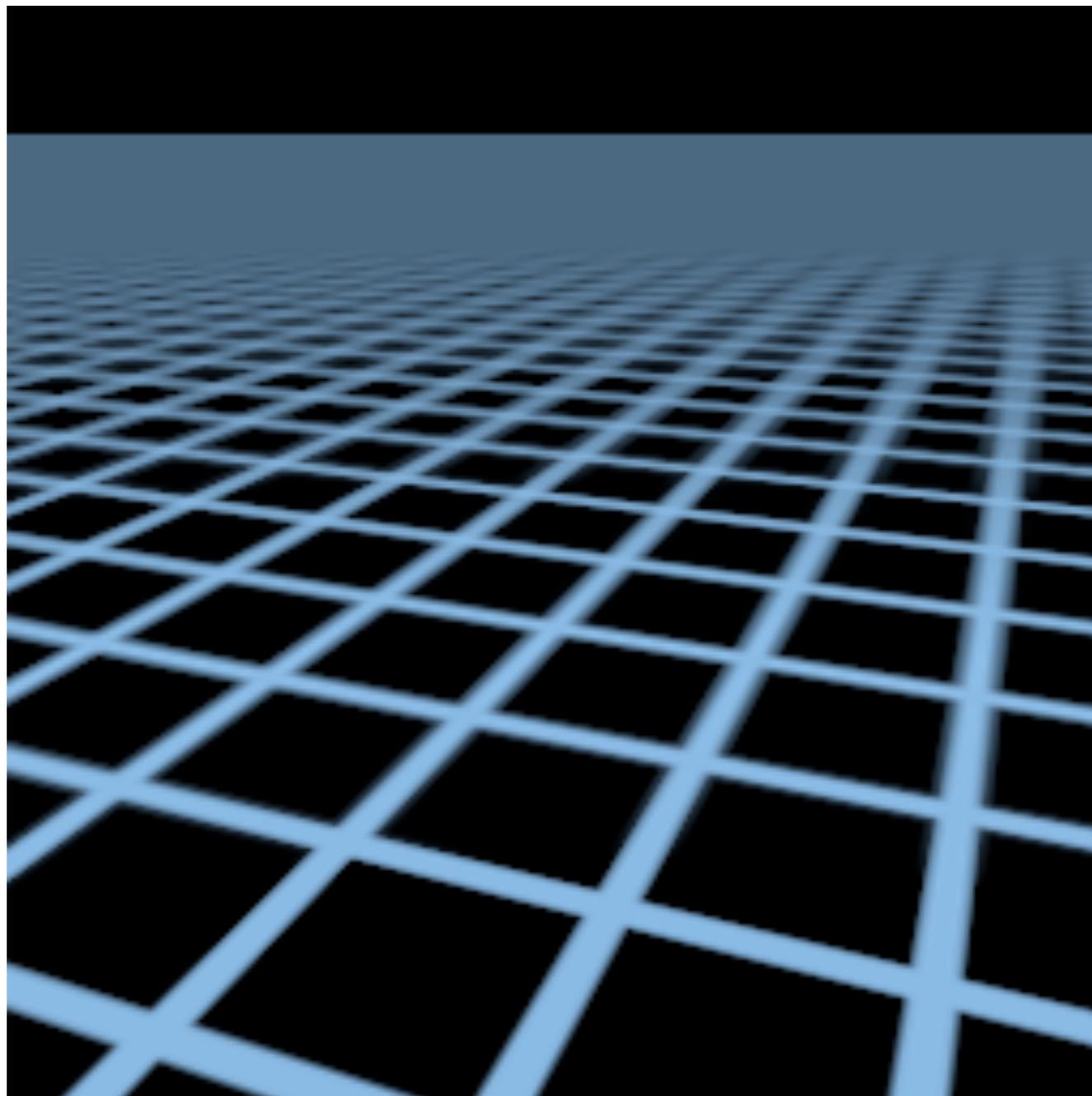
# Mipmap Limitations



Supersampling 512x (assume this is correct)

# Mipmap Limitations

Overblur  
Why?

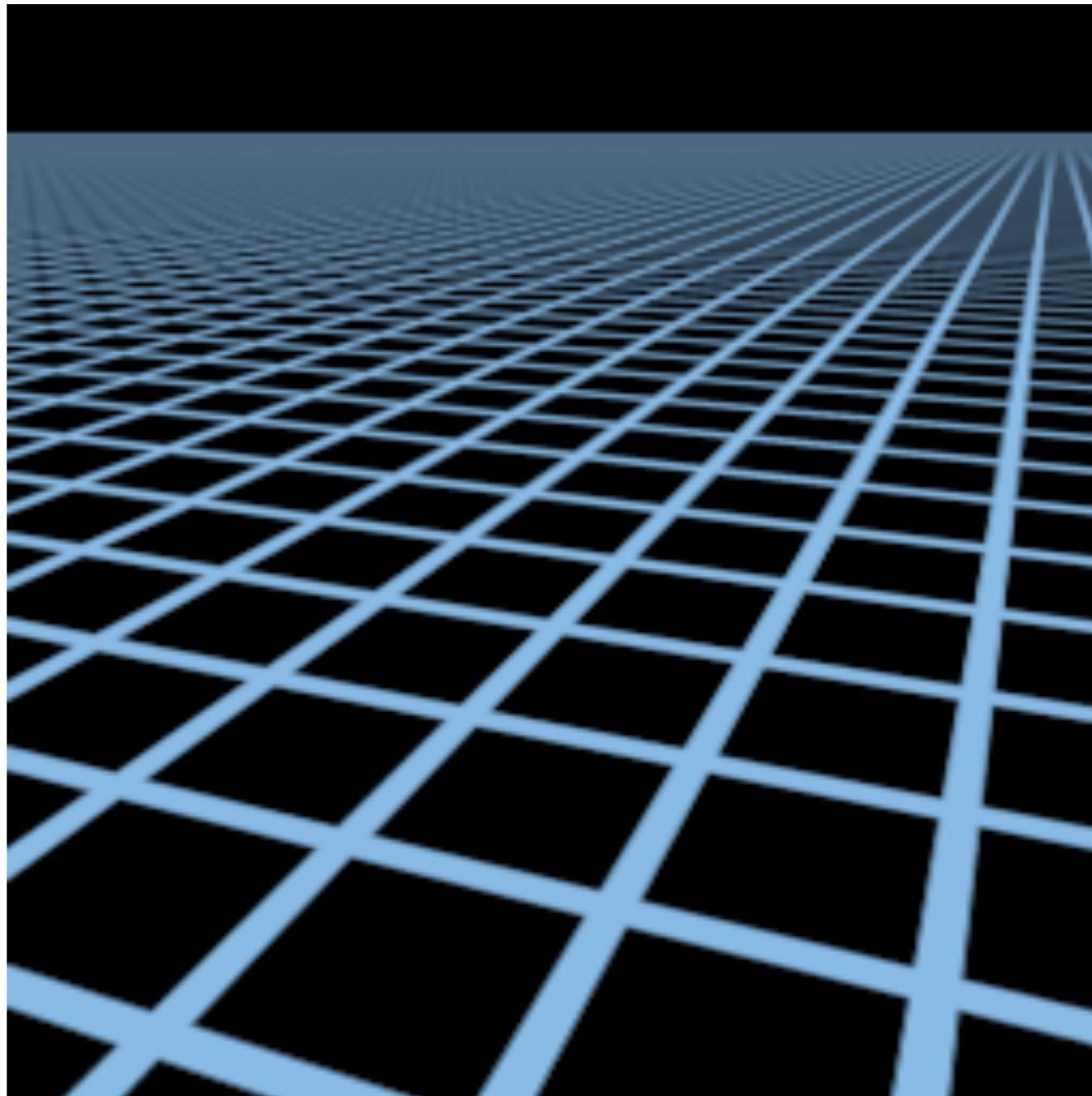


处处过度模糊

Mipmap trilinear sampling

# Anisotropic Filtering

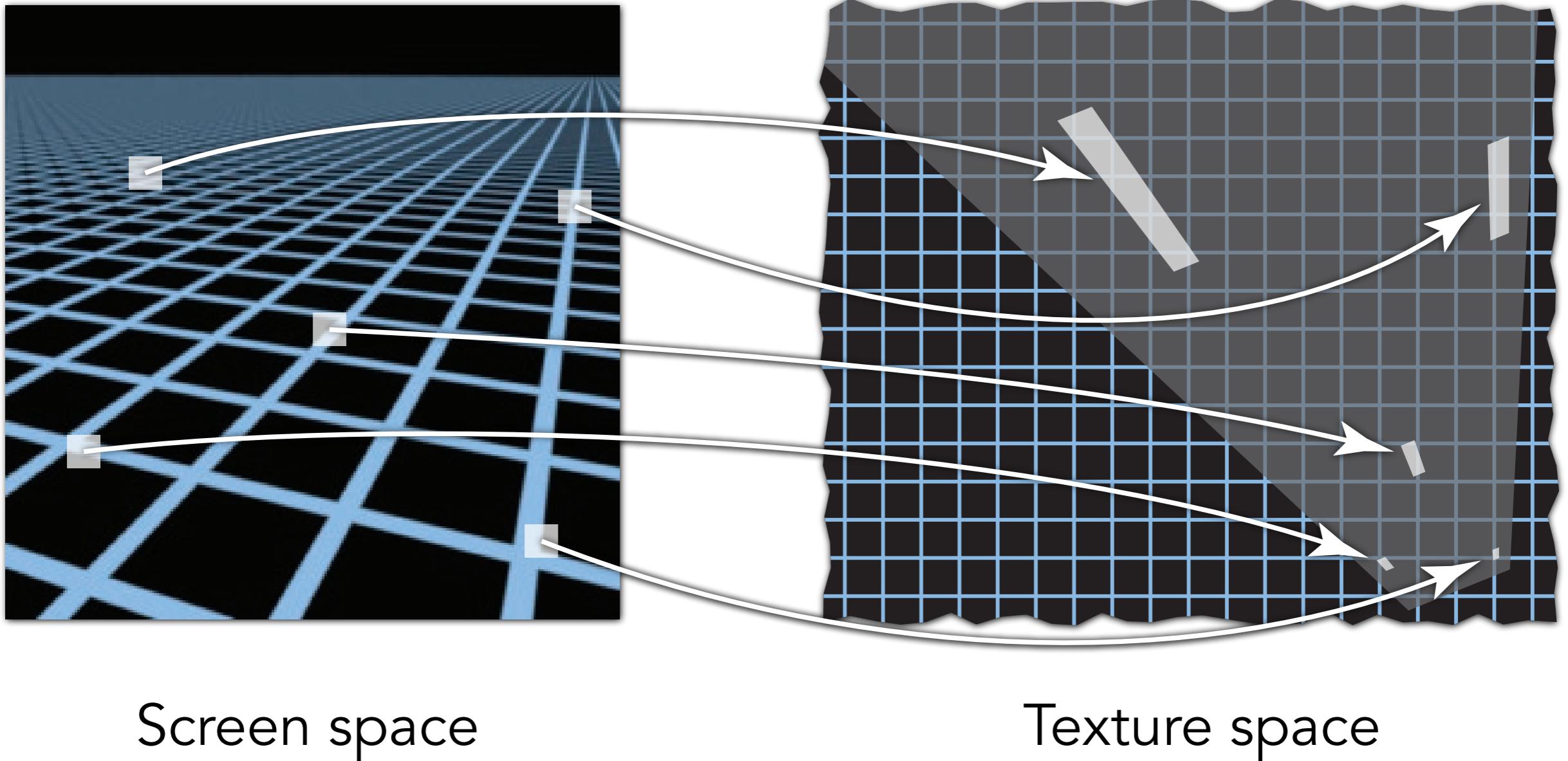
各向异性过滤(矩形)



Better than Mipmap!

# Irregular Pixel Footprint in Texture

采样不为规则矩形



# Anisotropic Filtering

Ripmaps and summed area tables

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



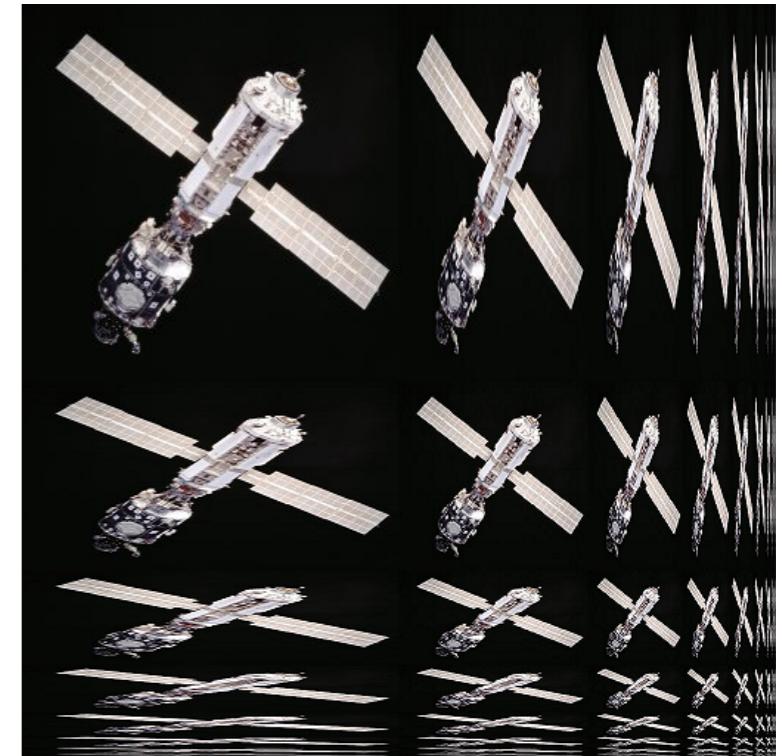
Wikipedia

# Anisotropic Filtering

开销  $\times 3$

Ripmaps and summed area tables

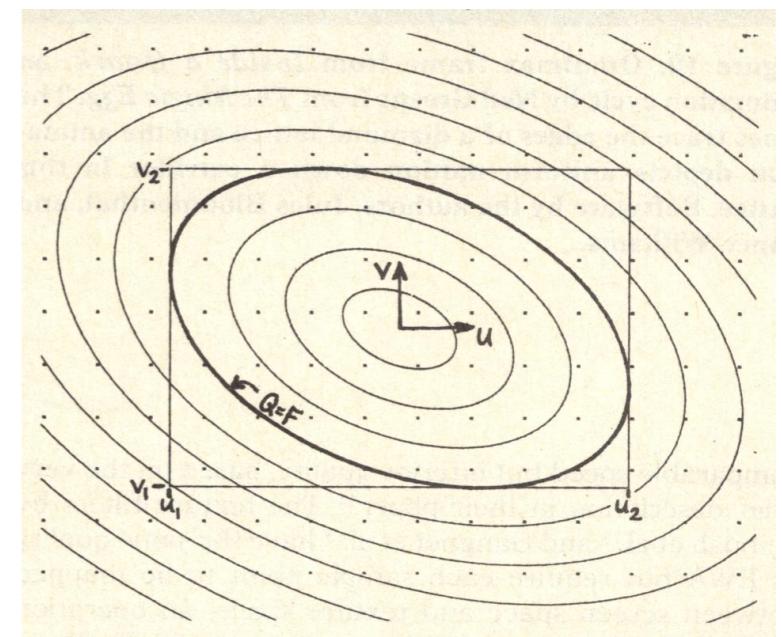
- Can look up axis-aligned rectangular zones
- Diagonal footprints still a problem



Wikipedia

EWA filtering (拆成圆形)

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



Greene & Heckbert '86

# Thank you!

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