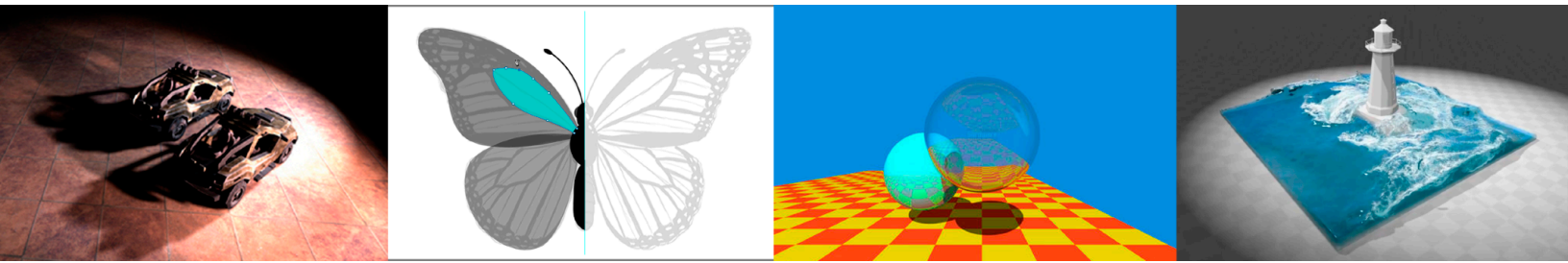
Introduction to Computer Graphics

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 重心坐标（目的：插值）
  - Texture queries 纹理查询
  - Applications of textures 纹理应用
- Shadow mapping

在三角形内部插值

Interpolation Across Triangles: Barycentric Coordinates

(重心坐标)

Interpolation Across Triangles

Why do we want to interpolate?

- Specify values **at vertices** 指定顶点的值.
- Obtain smoothly varying values **across triangles**
  在三角形上获得平滑的变化值.

What do we want to interpolate?

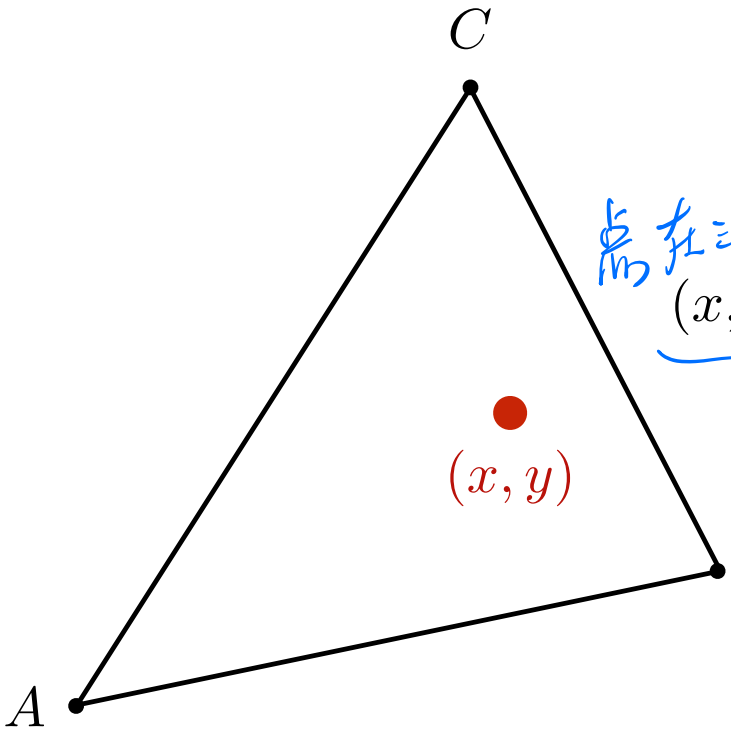
- Texture coordinates, colors, normal vectors, …
  1. 纹理坐标 2. 颜色 3. 法线向量

How do we interpolate?

- **Barycentric coordinates**
  重心坐标

Barycentric Coordinates

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



点在三角形的平面内：

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

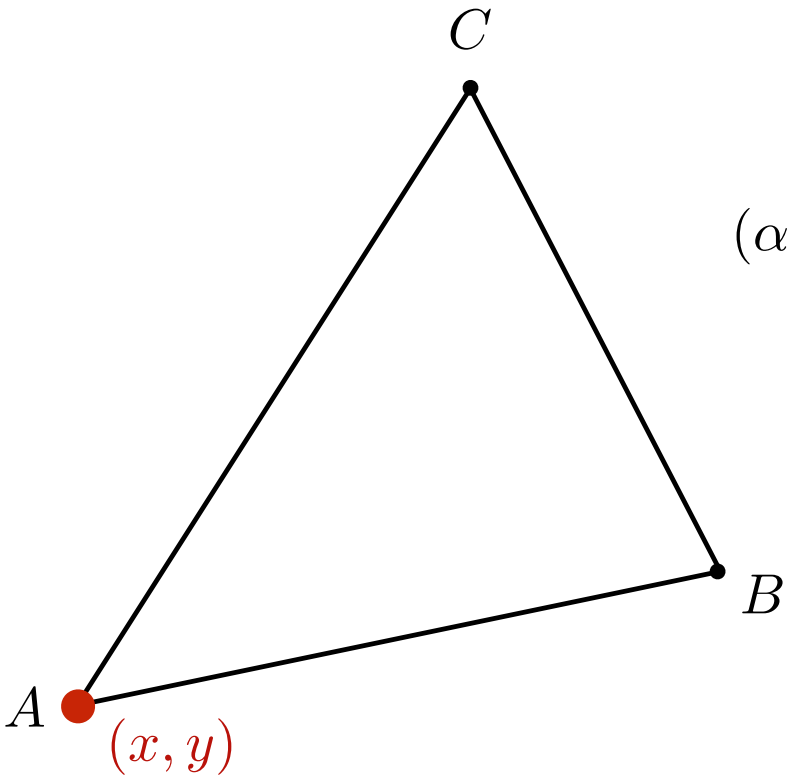
$$\alpha + \beta + \gamma = 1$$

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

如果 α, β, γ 非负
则点位于三角形内

Barycentric Coordinates

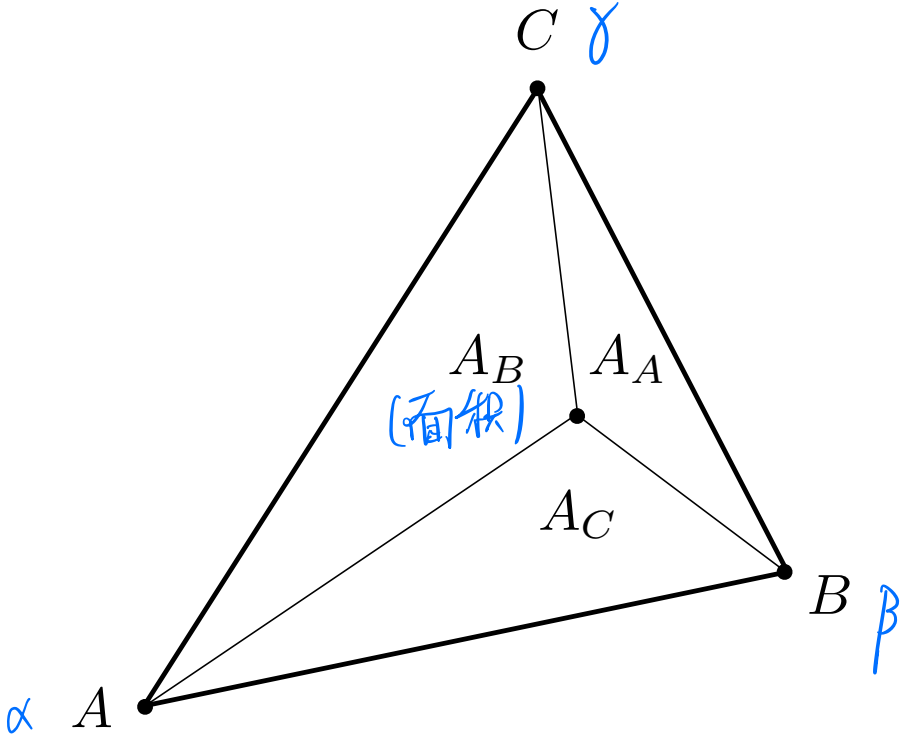
What's the barycentric coordinate of A?



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

Barycentric Coordinates

Geometric viewpoint — proportional areas



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

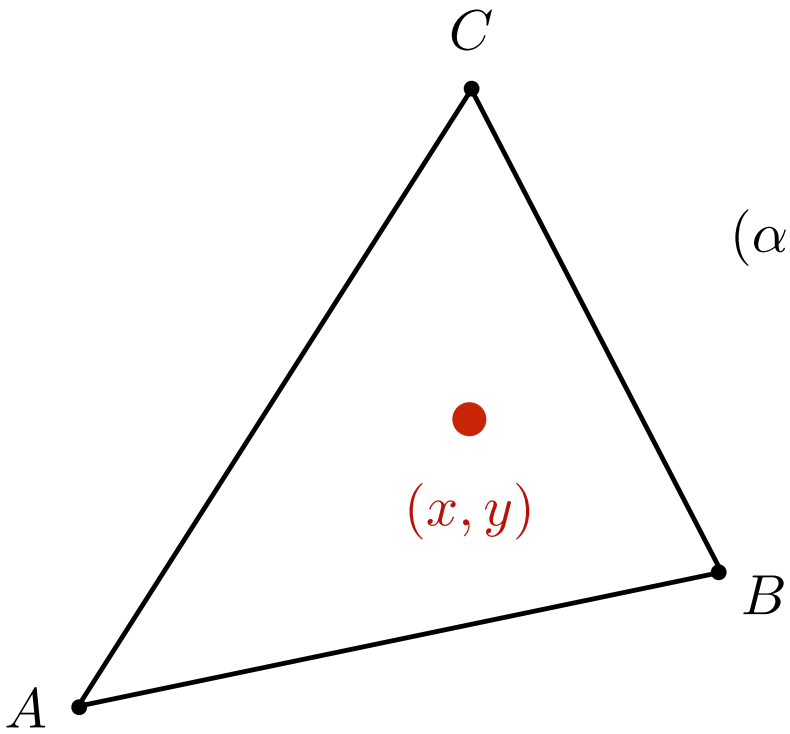
$$\beta = \frac{A_B}{A_A + A_B + A_C}$$

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

Barycentric Coordinates

三角形的重心坐标

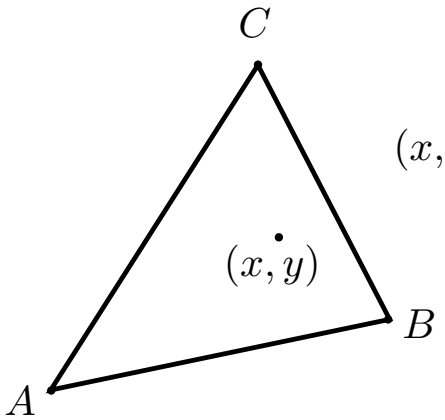
What's the barycentric coordinate of the centroid?



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

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

Barycentric Coordinates: Formulas



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

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

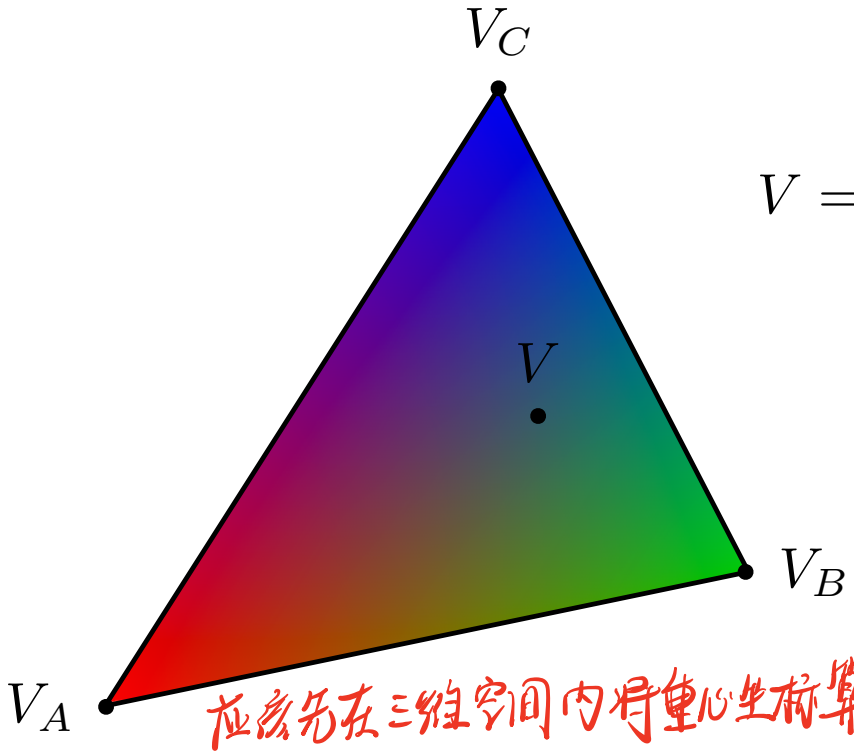
$$\beta = \frac{-(x - x_C)(y_A - y_C) + (y - y_C)(x_A - x_C)}{-(x_B - x_C)(y_A - y_C) + (y_B - y_C)(x_A - x_C)}$$

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

Using Barycentric Coordinates

在顶点上线性插值

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…

应该先在三维空间内将重心坐标算出，而非投影后在二维空间计算

However, barycentric coordinates are not invariant under projection!

注：重心坐标在投影下可能会改变.

Applying Textures

Simple Texture Mapping: Diffuse Color

Usually a pixel's center

for each rasterized screen sample (x,y):

利用插值出来的纹理坐标(重心坐标)

(u,v) = evaluate texture coordinate at (x,y)

texcolor = texture.sample(u,v);

set sample's color to texcolor;

Using barycentric coordinates!

漫反射率 Kd

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

纹理放大

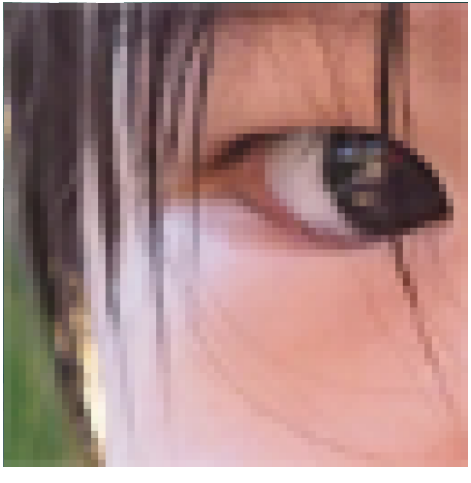
Texture Magnification

(What if the texture is too small?)

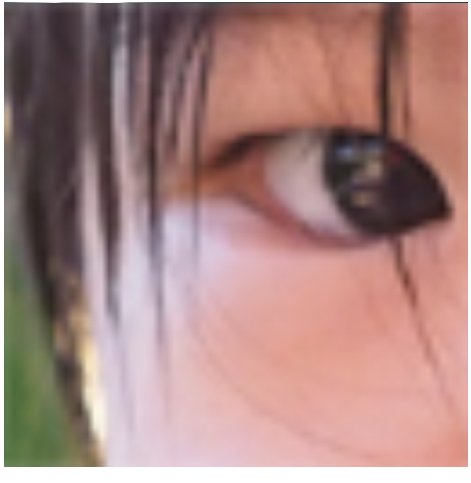
Texture Magnification - Easy Case

Generally don't want this — insufficient texture resolution

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



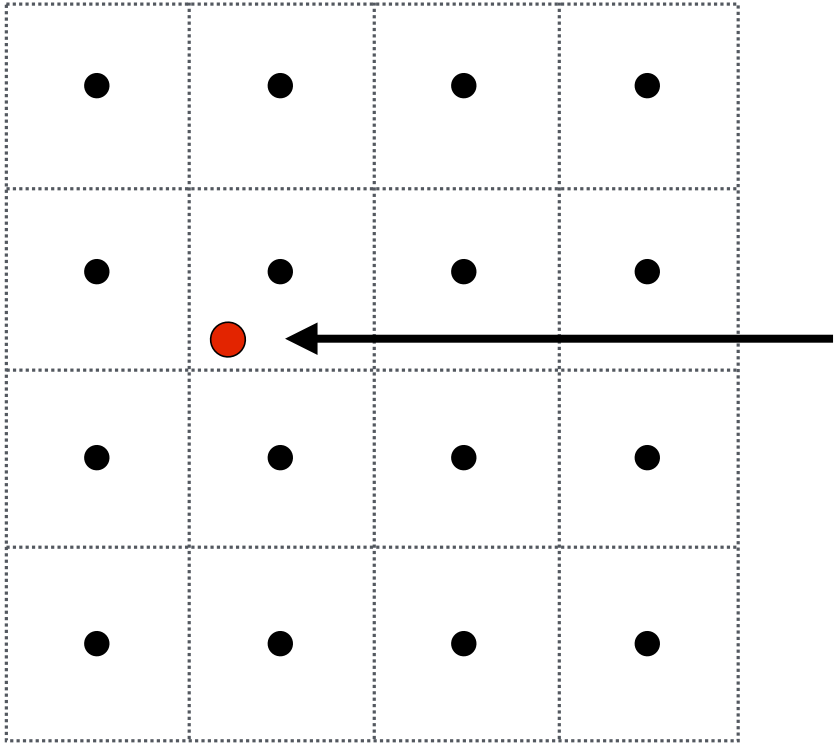
Nearest



Bilinear

Bicubic

Bilinear Interpolation 双线性插值



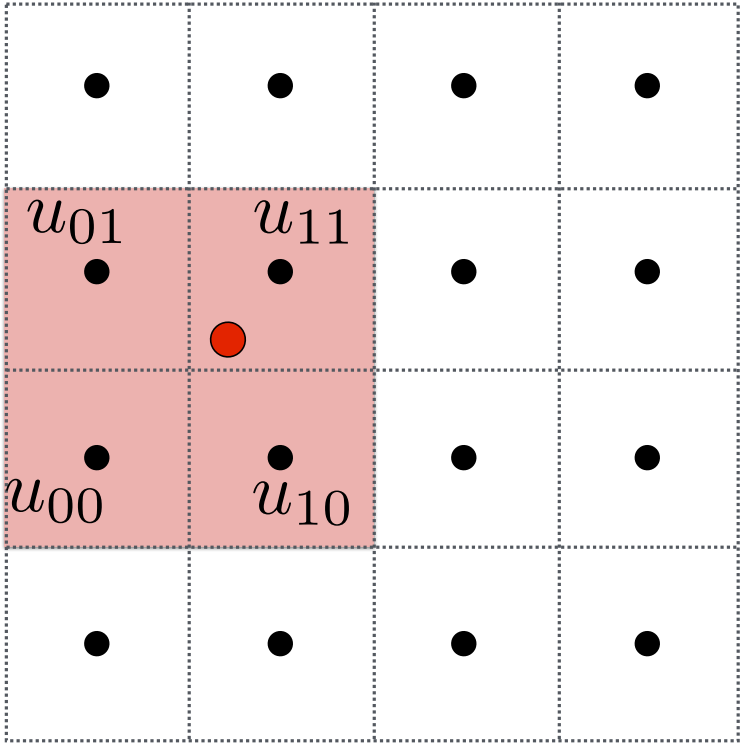
红点：等待点

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

Black points indicate texture sample locations

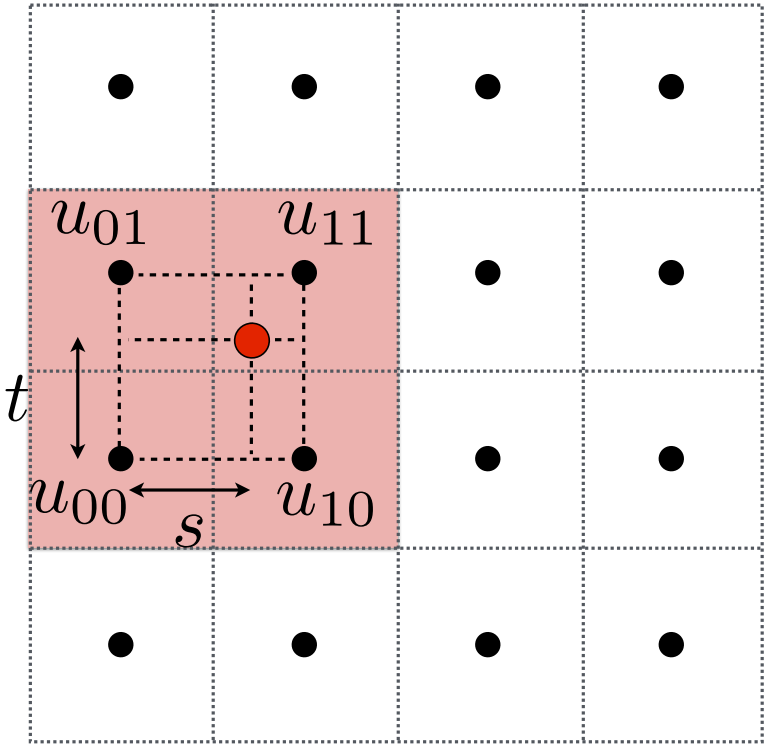
黑点：纹理样本

Bilinear Interpolation



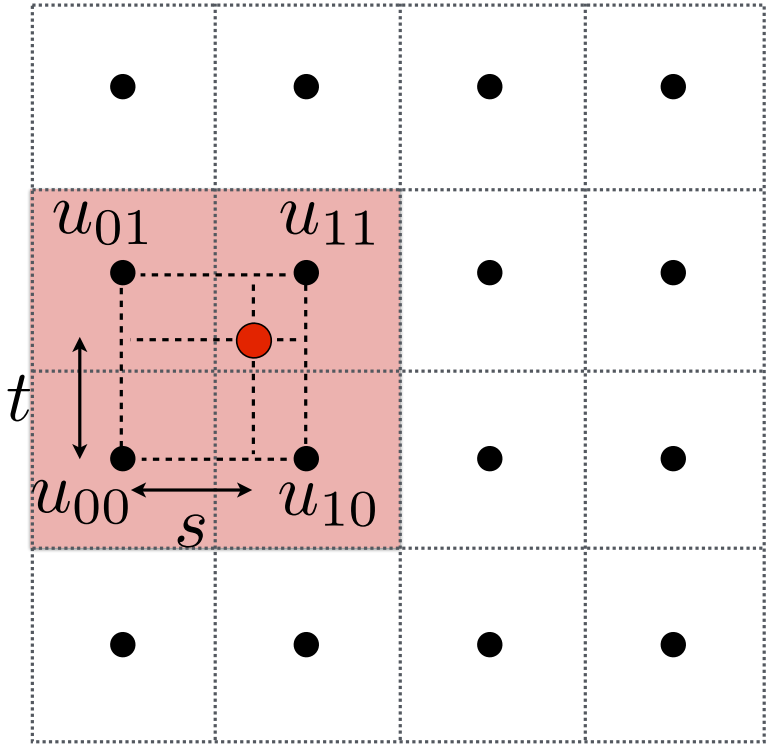
Take 4 nearest sample locations, with texture values as labeled.

Bilinear Interpolation



And fractional offsets, (s,t) as shown

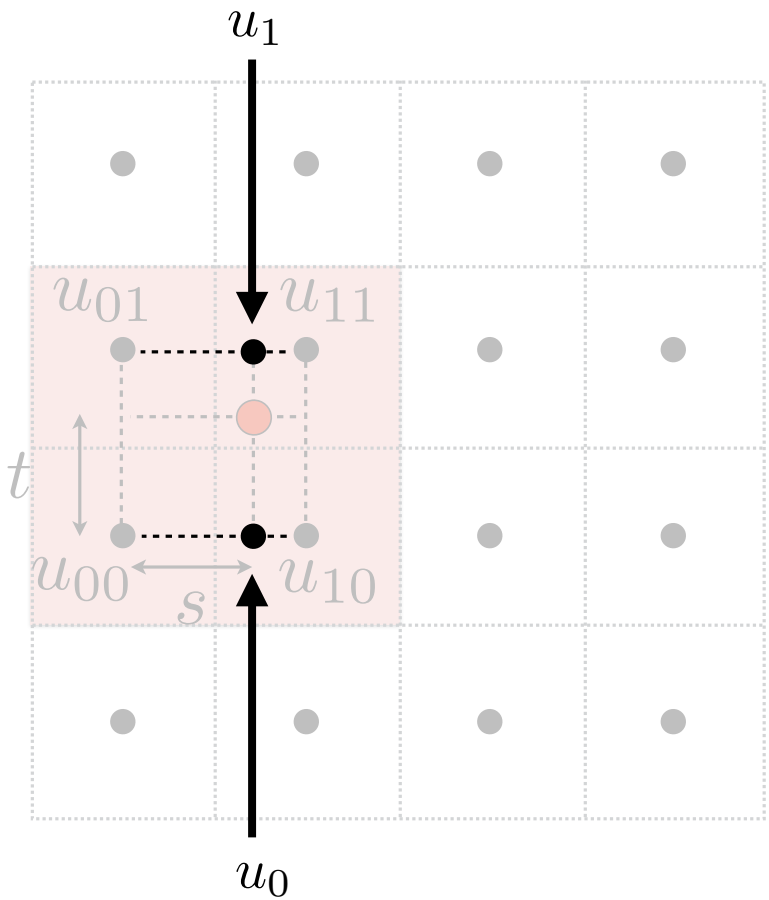
Bilinear Interpolation



Linear interpolation (1D)

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

Bilinear Interpolation



Linear interpolation (1D)

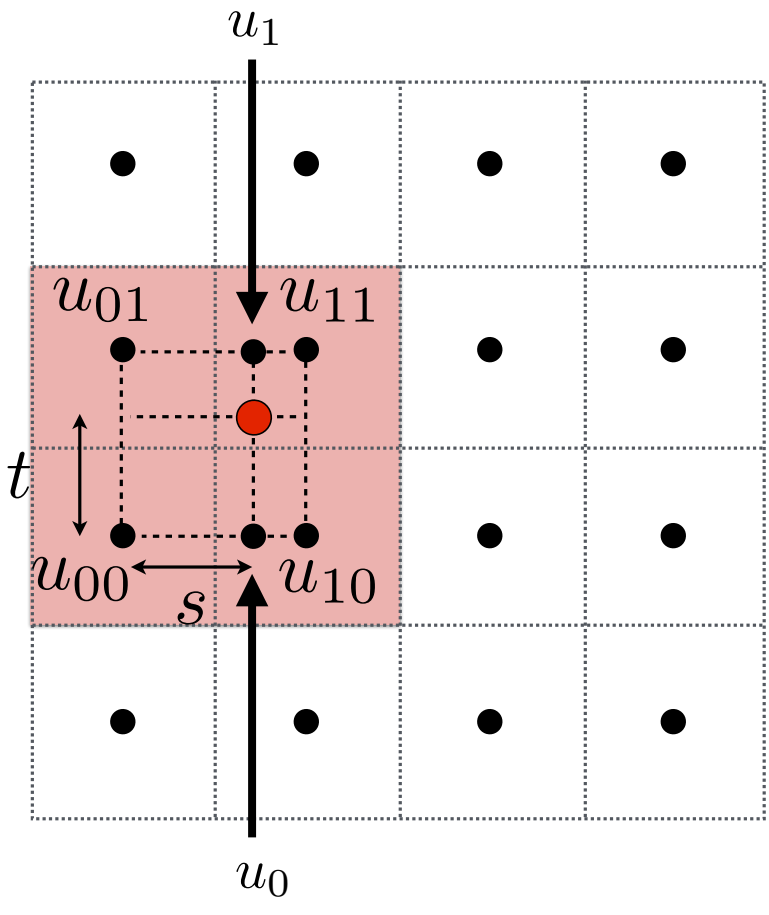
$$\text{lerp}(x, v_0, v_1) = v_0 + x(v_1 - v_0)$$

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

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

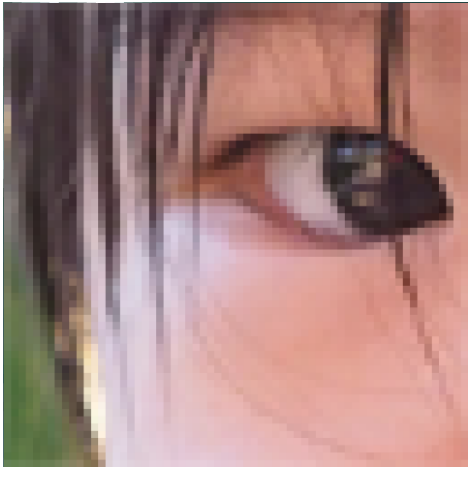
$$u_1 = \text{lerp}(s, u_{01}, u_{11})$$

Final vertical lerp, to get result:

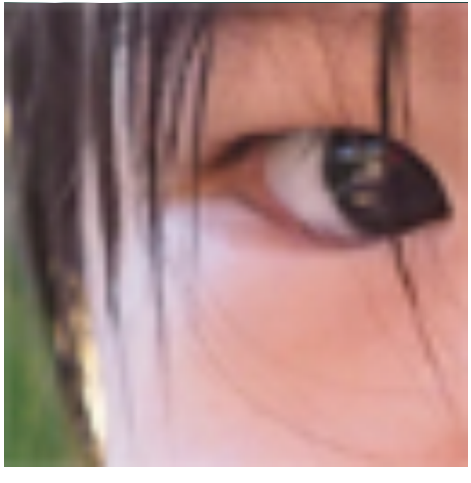
$$f(x, y) = \text{lerp}(t, u_0, u_1)$$

Texture Magnification - Easy Case

Bilinear interpolation usually gives pretty good results at reasonable costs



Nearest



Bilinear

Bicubic

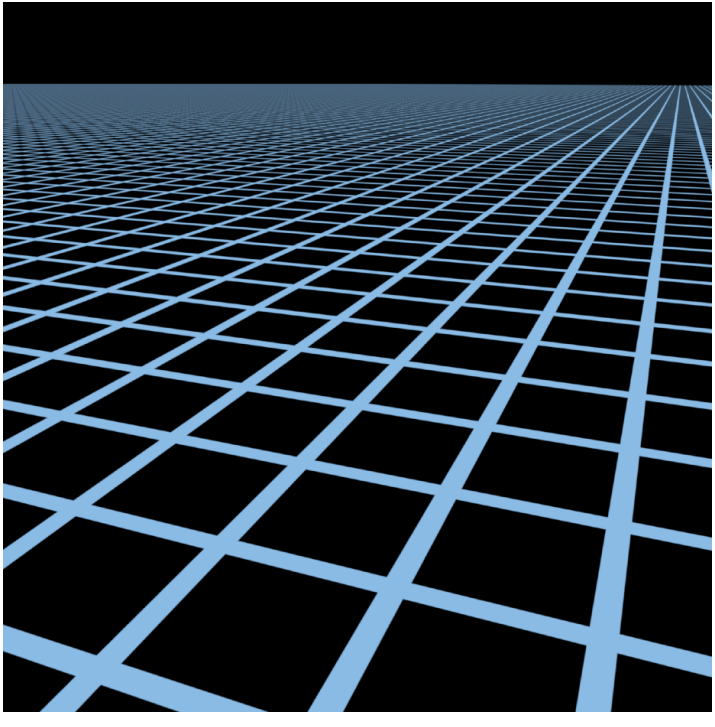
<非线性>

Texture Magnification **(hard case)**

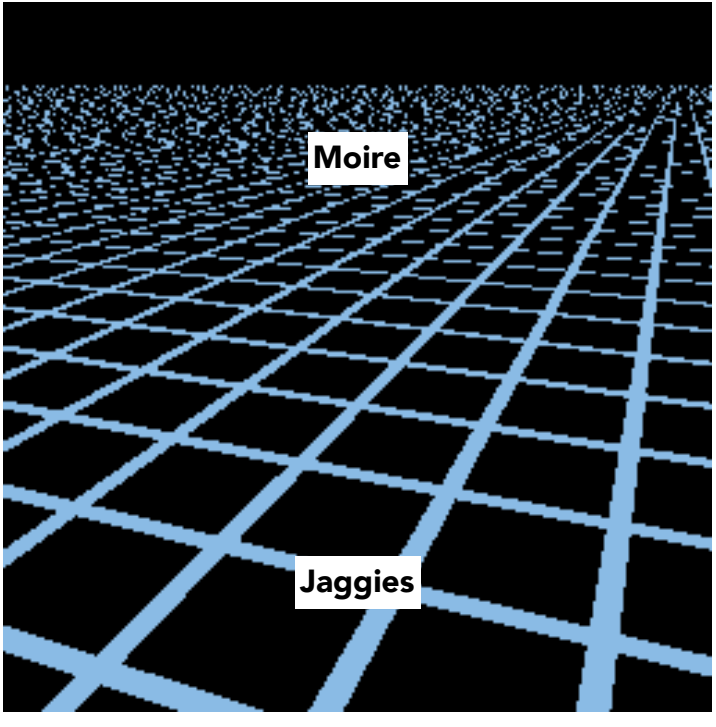
(What if the texture is too large?)

纹理太大？

Point Sampling Textures — Problem

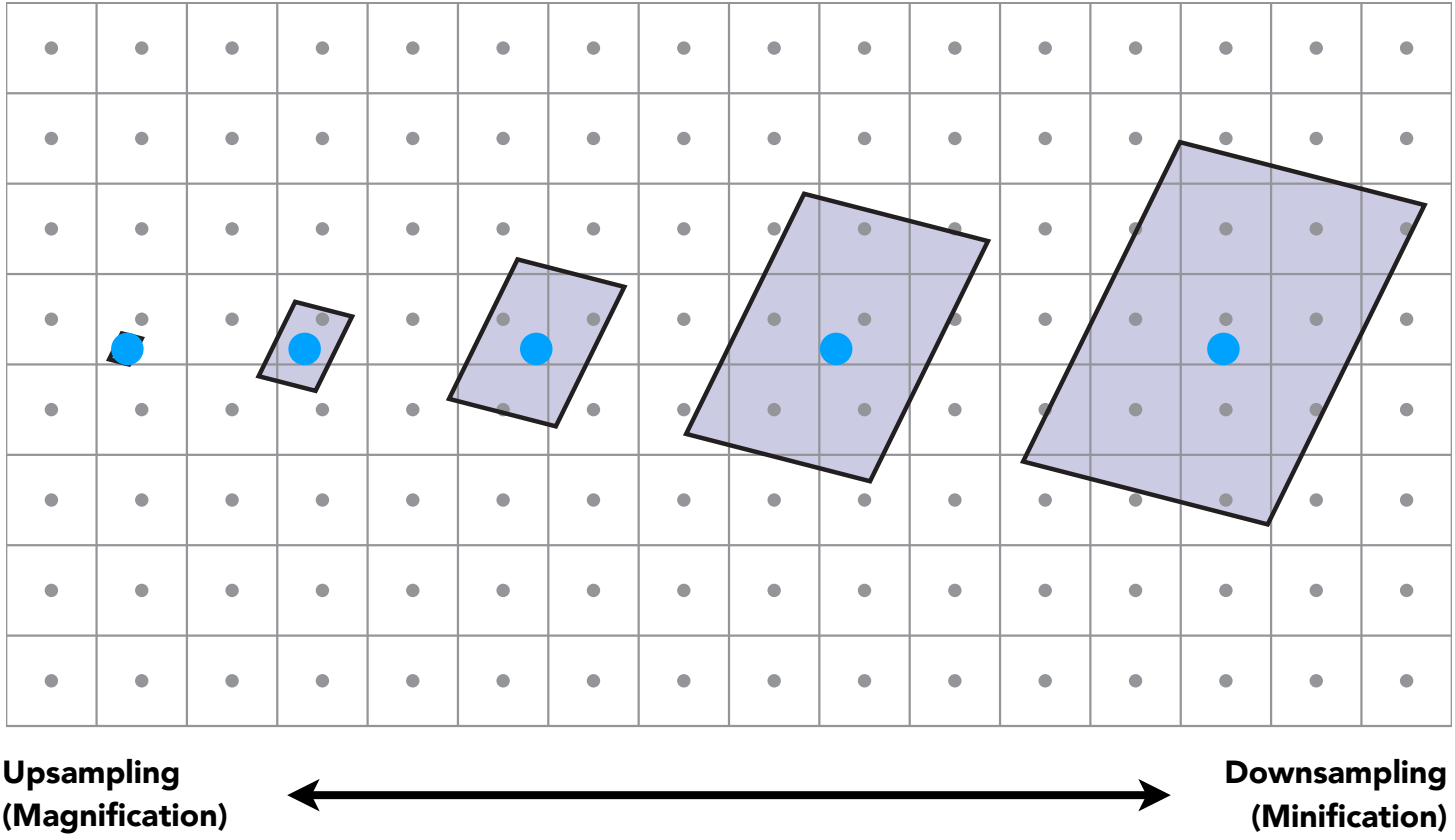


Reference

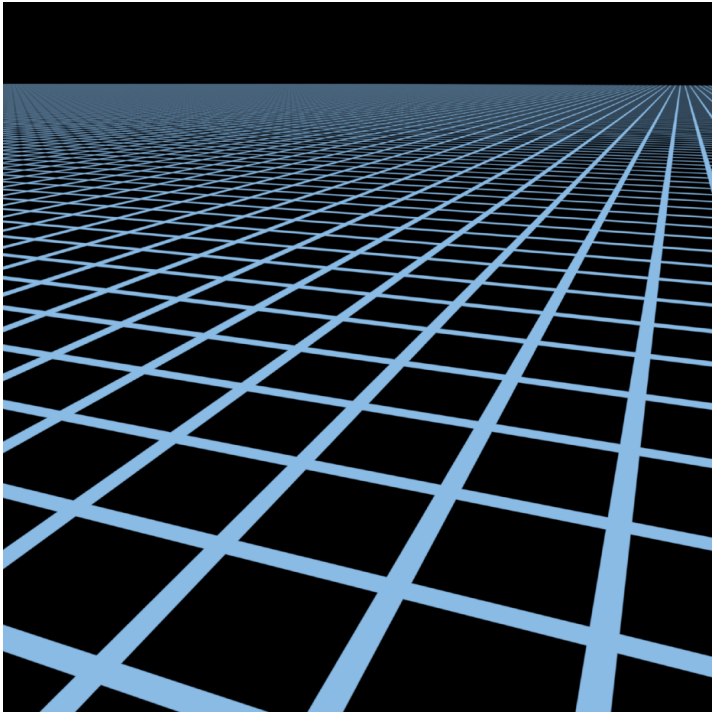


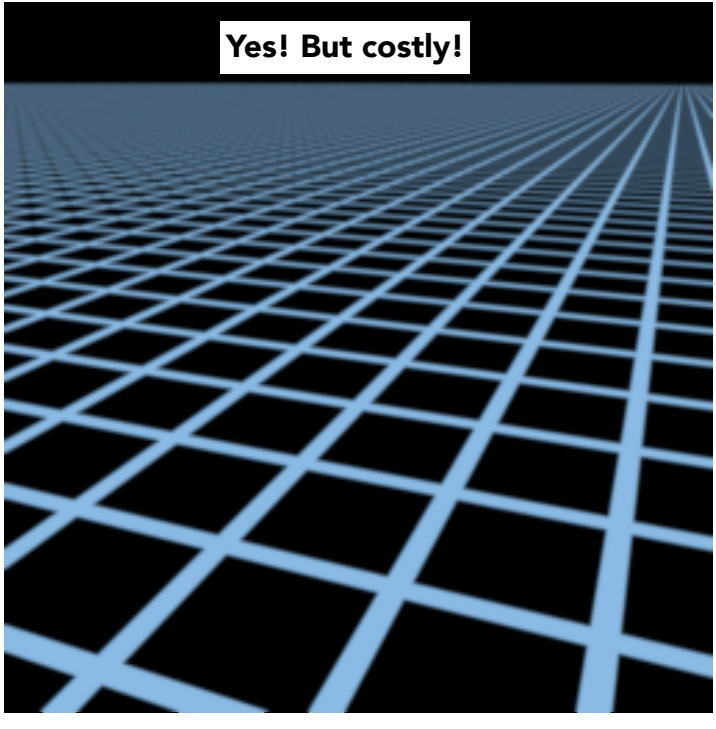
Point sampled

Screen Pixel "Footprint" in Texture



Will Supersampling Do Antialiasing?





512x supersampling

反走样

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

提高采样频率

Let's understand this problem in another way

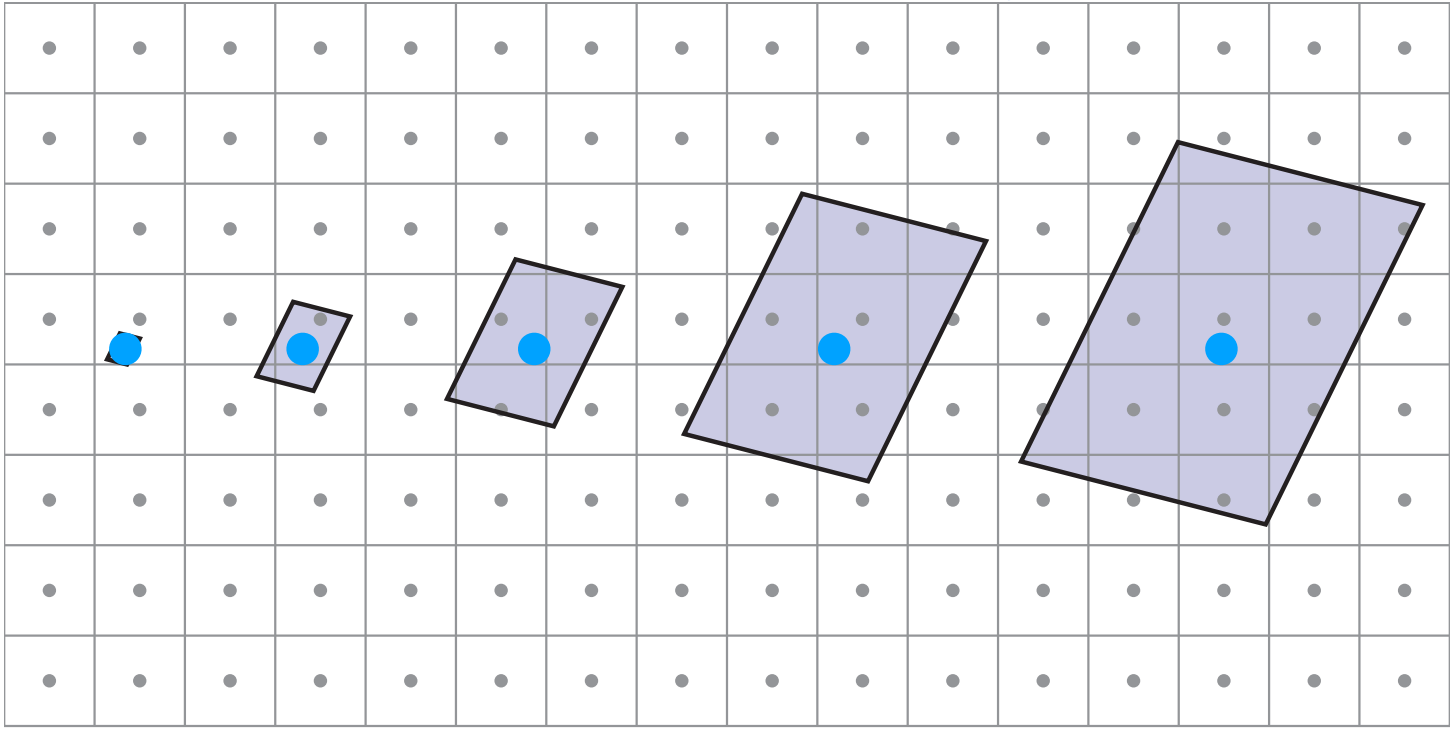
- What if we don't sample?
- Just need to **get the average value within a range**!

只需得到一个范围内的平均值

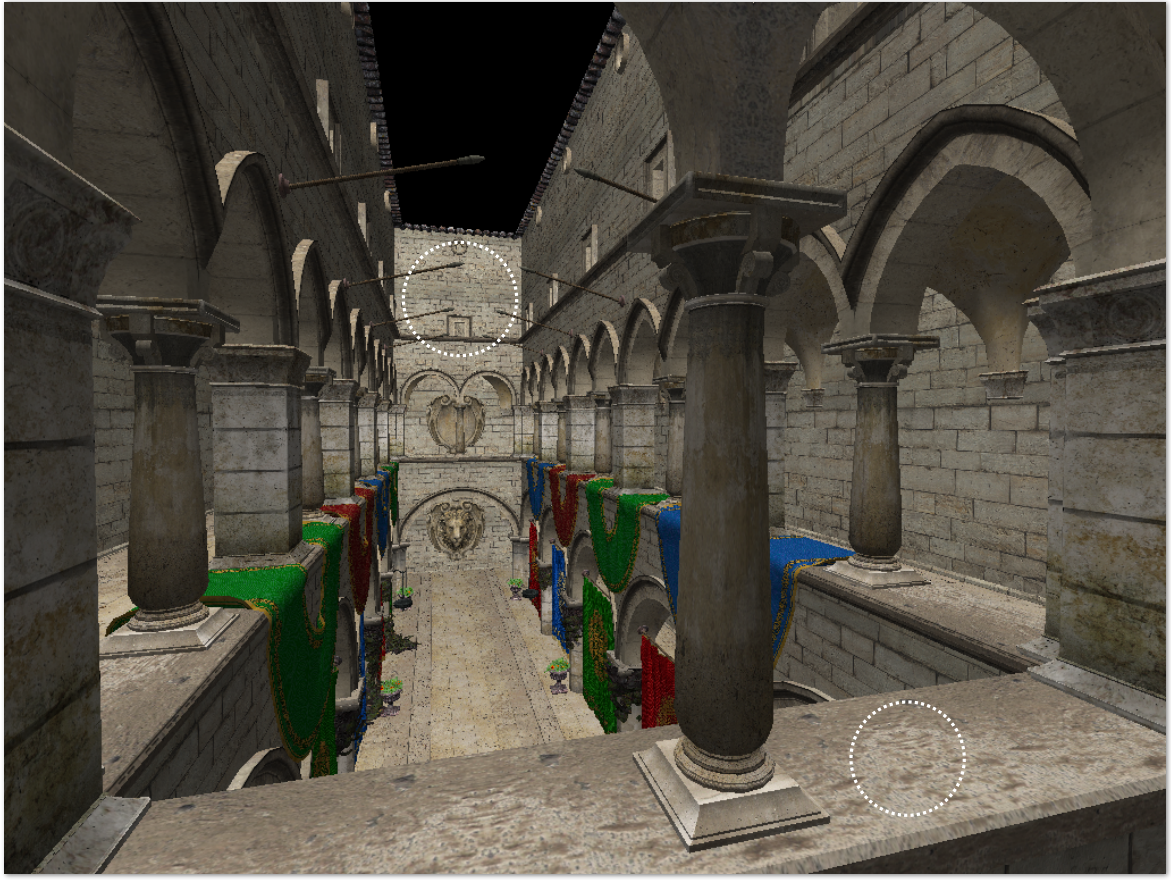
Point Query vs. (Avg.) Range Query

点查询(双线性插值)

范围查询<得到区域内平均值>



Different Pixels -> Different-Sized Footprints



Mipmap

Allowing (fast, approx., square) range queries

允许作范围查询

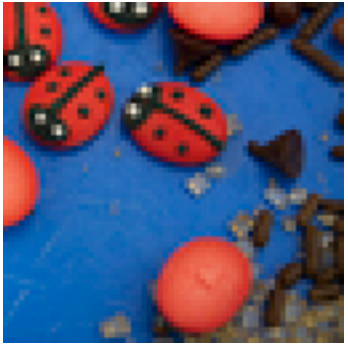
<快、不太准、正方形的范围查询>

近似

Mipmap (L. Williams 83) 额外开销：1/3

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

Level 0 = 128x128



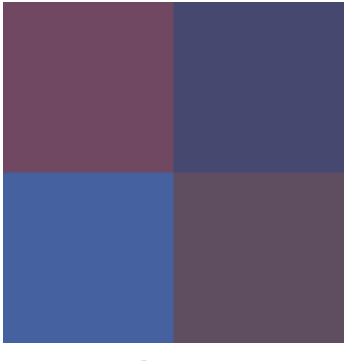
Level 1 = 64x64

Level 2 = 32x32

Level 3 = 16x16

Level 4 = 8x8

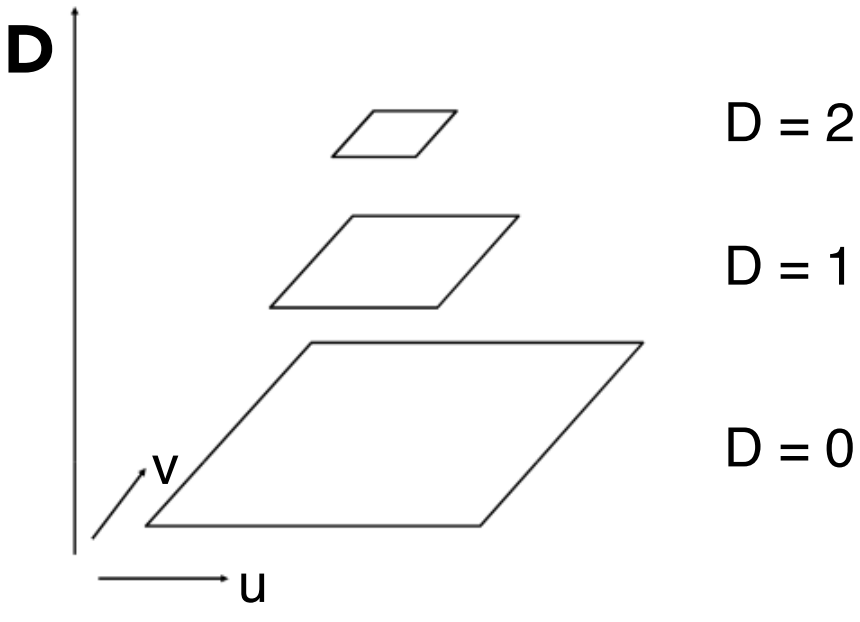
Level 5 = 4x4



Level 6 = 2x2

Level 7 = 1x1

Mipmap (L. Williams 83)

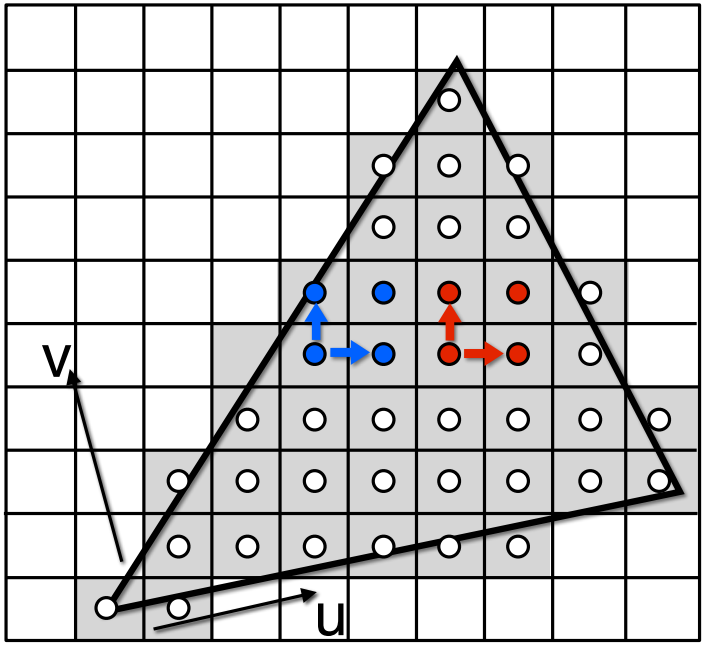


"Mip hierarchy"

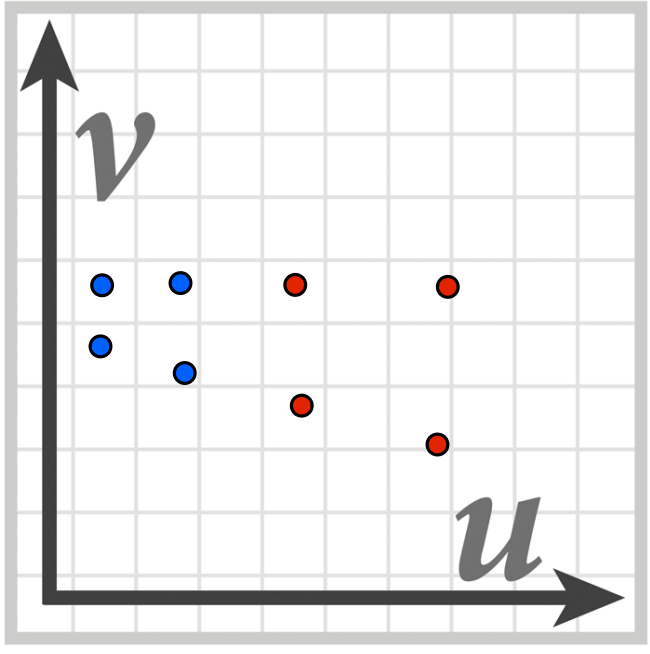
level = D

What is the storage overhead of a mipmap?

Computing Mipmap Level D



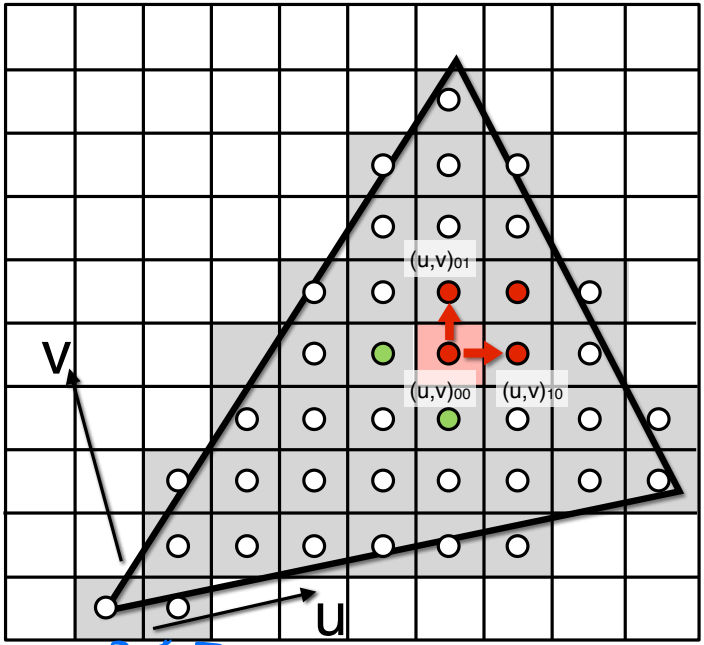
Screen space (x,y)

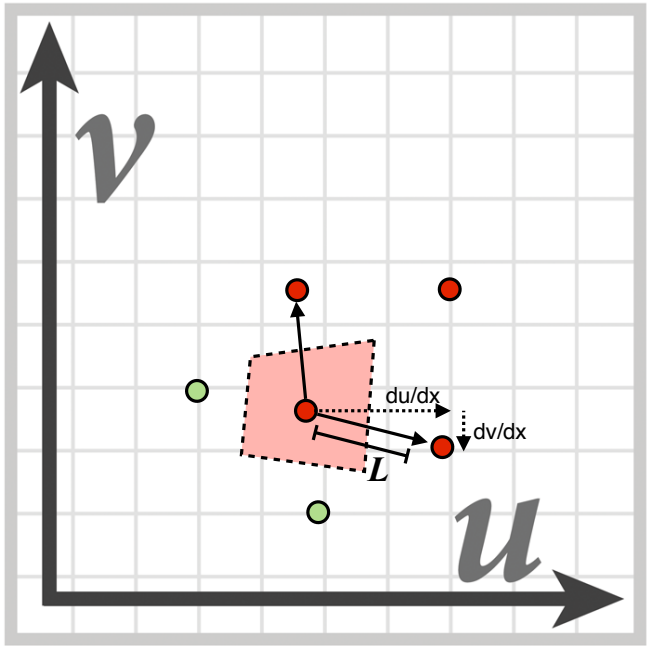


Texture space (u,v)

Estimate texture footprint using texture coordinates of neighboring screen samples

Computing Mipmap Level D



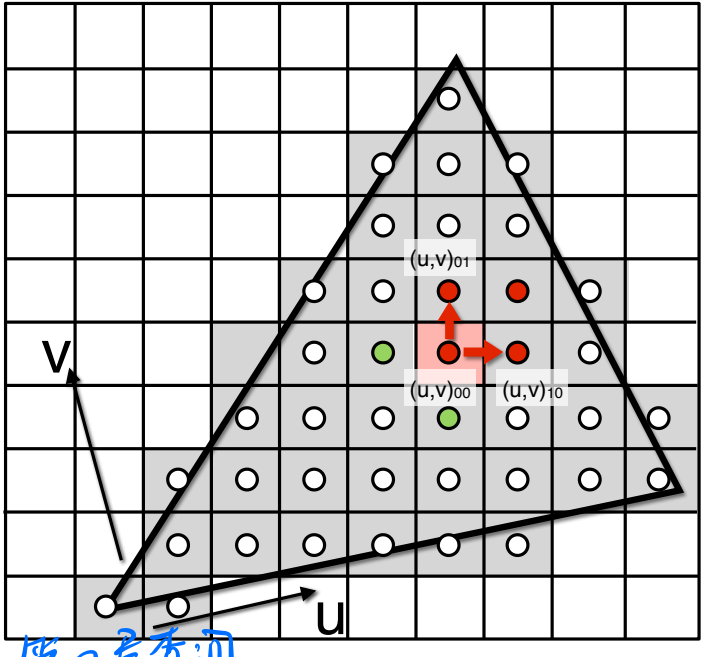


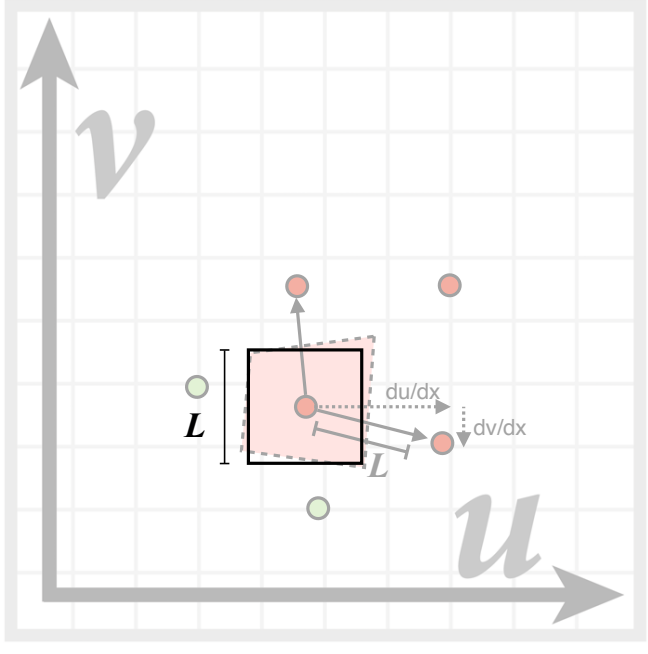
第D层查询

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

此时在第D层，LxL的正方形会变为1x1像素

Computing Mipmap Level D



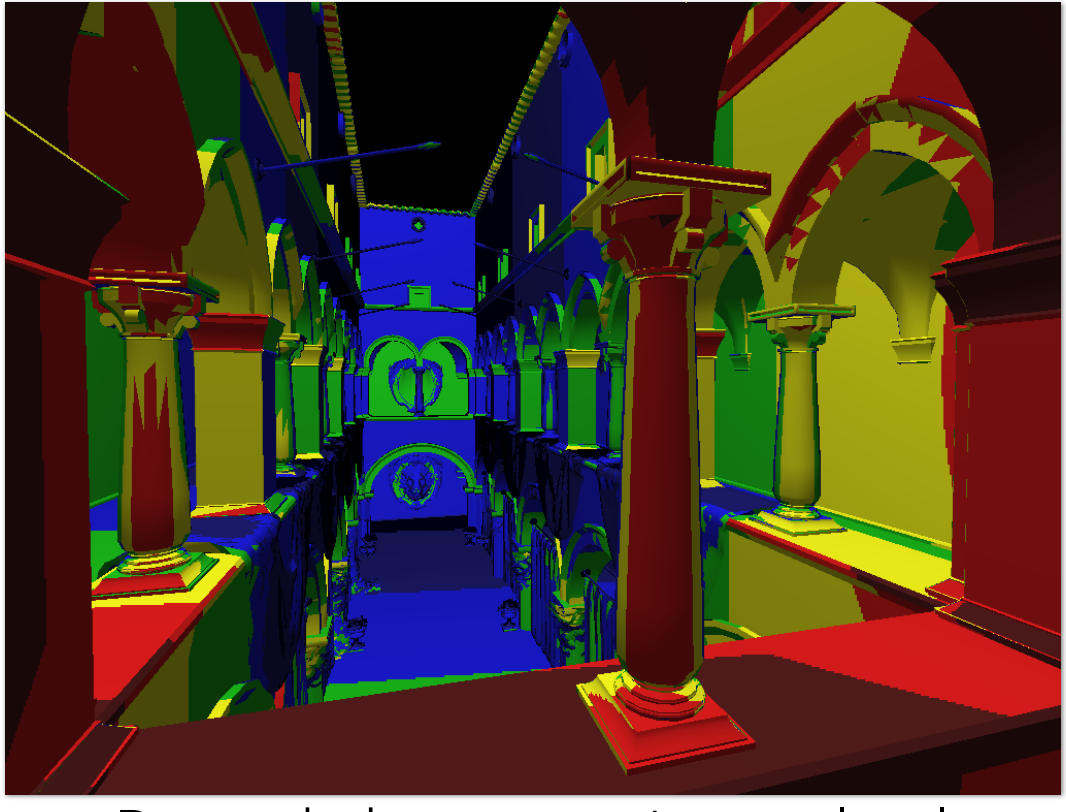


第D层查询

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

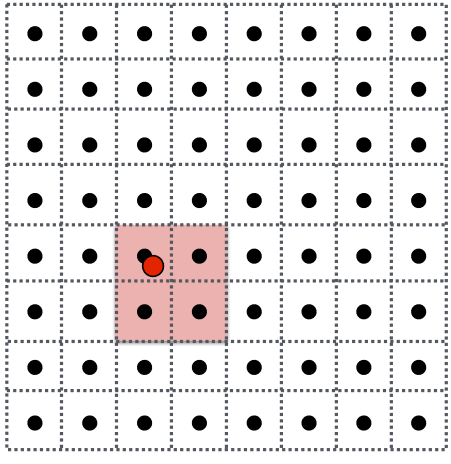
此时在第D层，LxL的正方形会变为1x1像素

Visualization of Mipmap Level

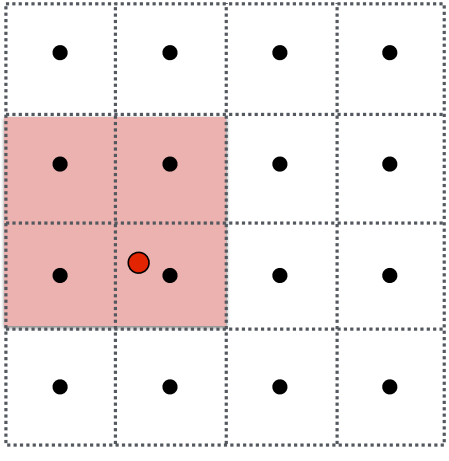


D rounded to nearest integer level

Trilinear Interpolation 三线性插值



Mipmap Level D



Mipmap Level D+1

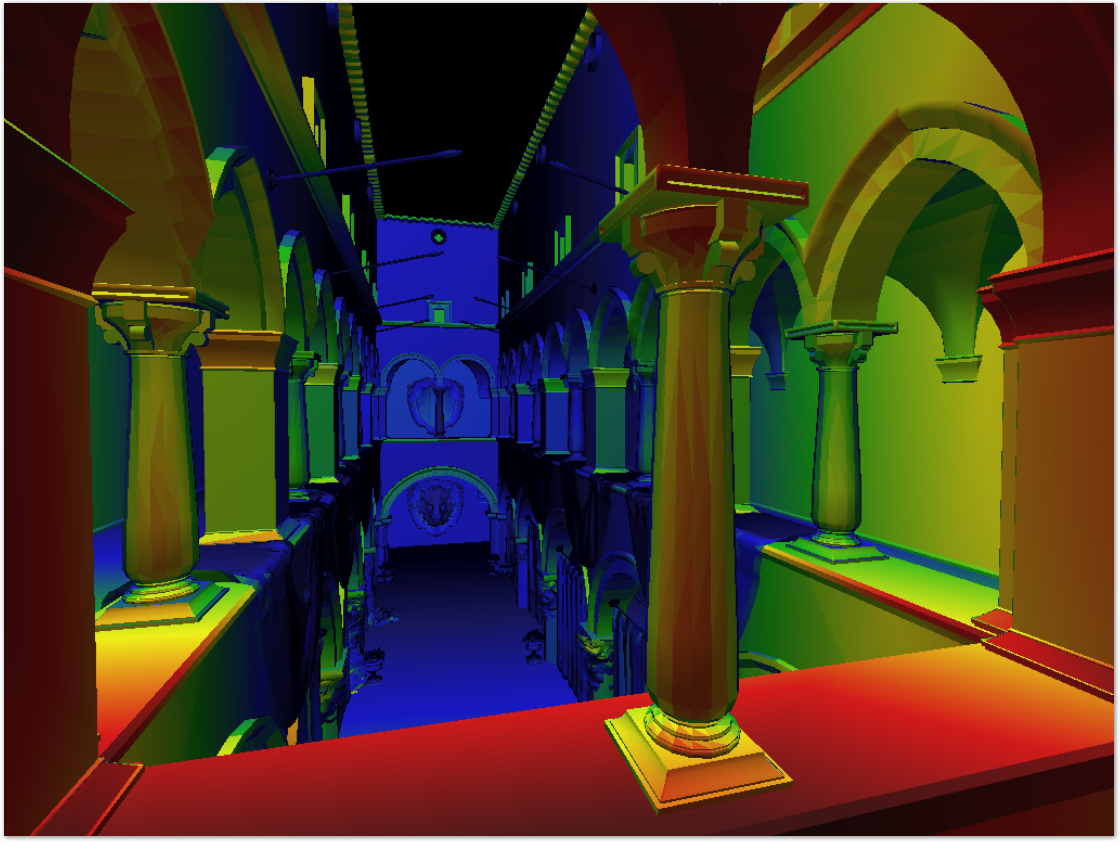
Bilinear result

Bilinear result

Linear interpolation based on continuous D value

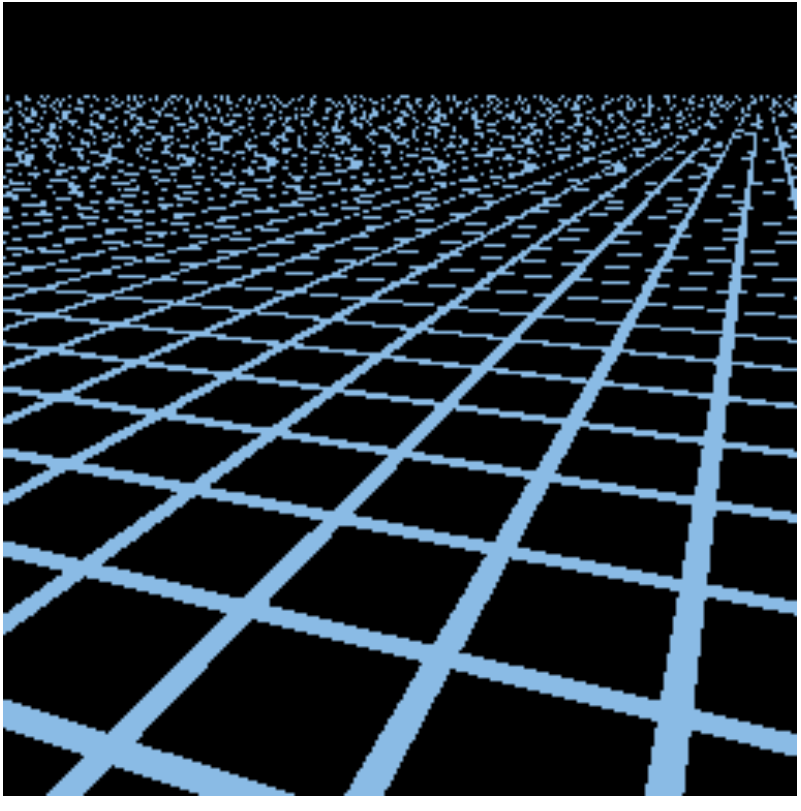
基于连续D值的线性插值

Visualization of Mipmap Level



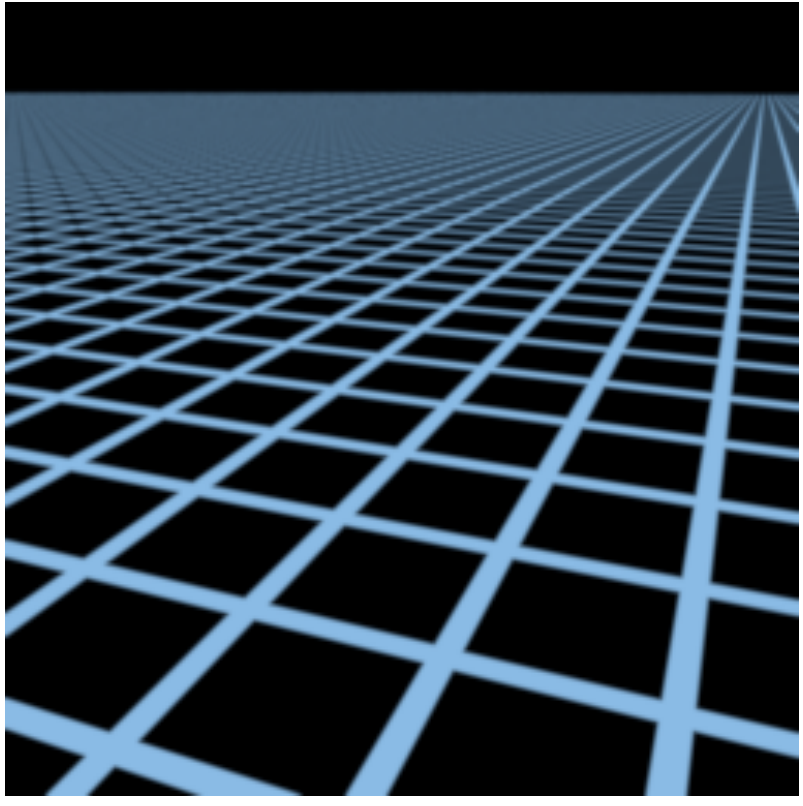
Trilinear filtering: visualization of continuous D

Mipmap Limitations



Point sampling

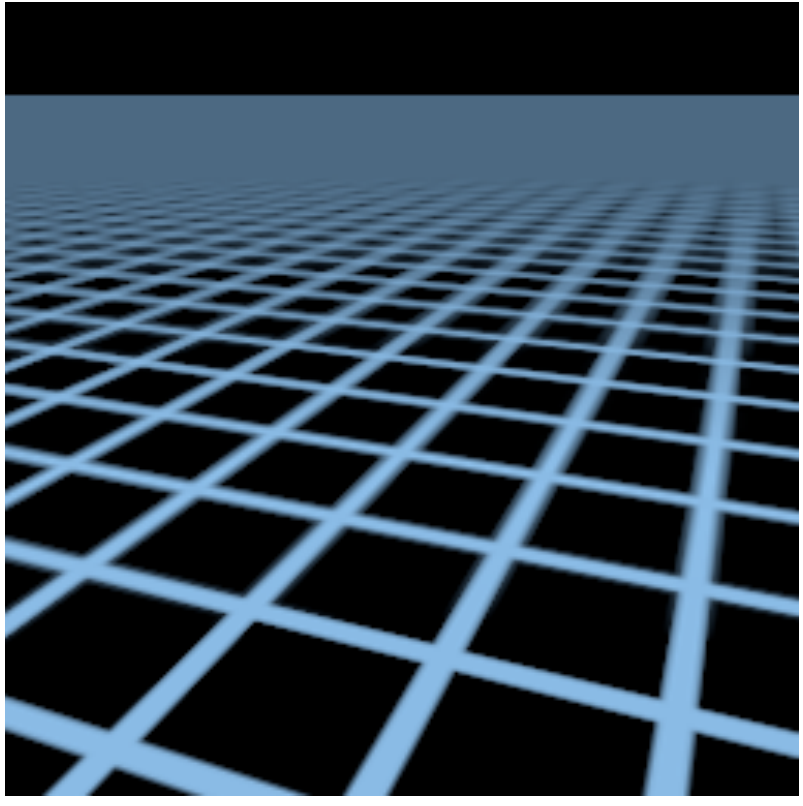
Mipmap Limitations



Supersampling 512x (assume this is correct)

Mipmap Limitations

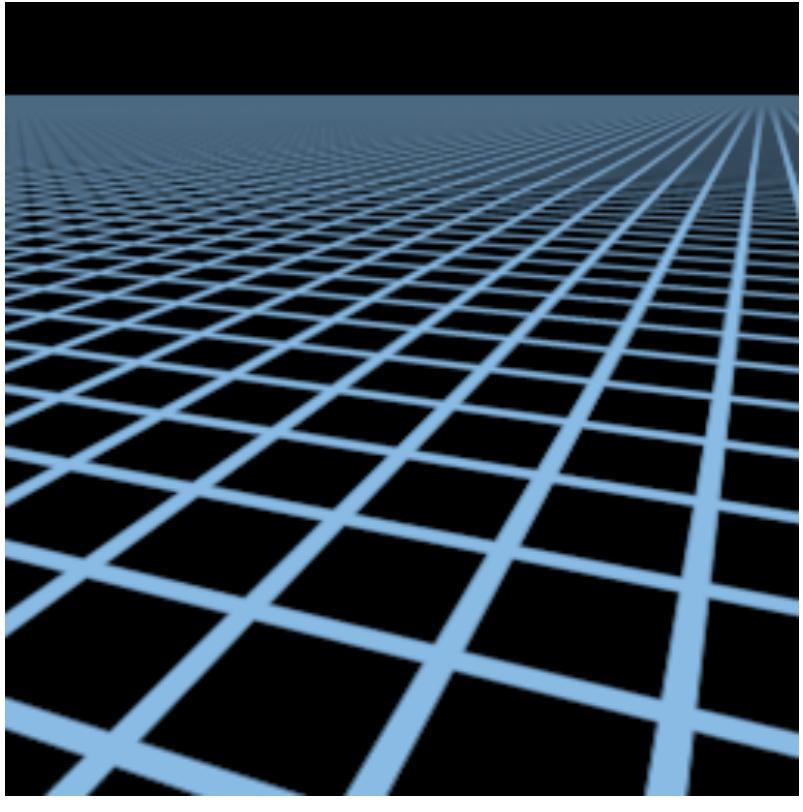
Overblur
Why?



Mipmap trilinear sampling

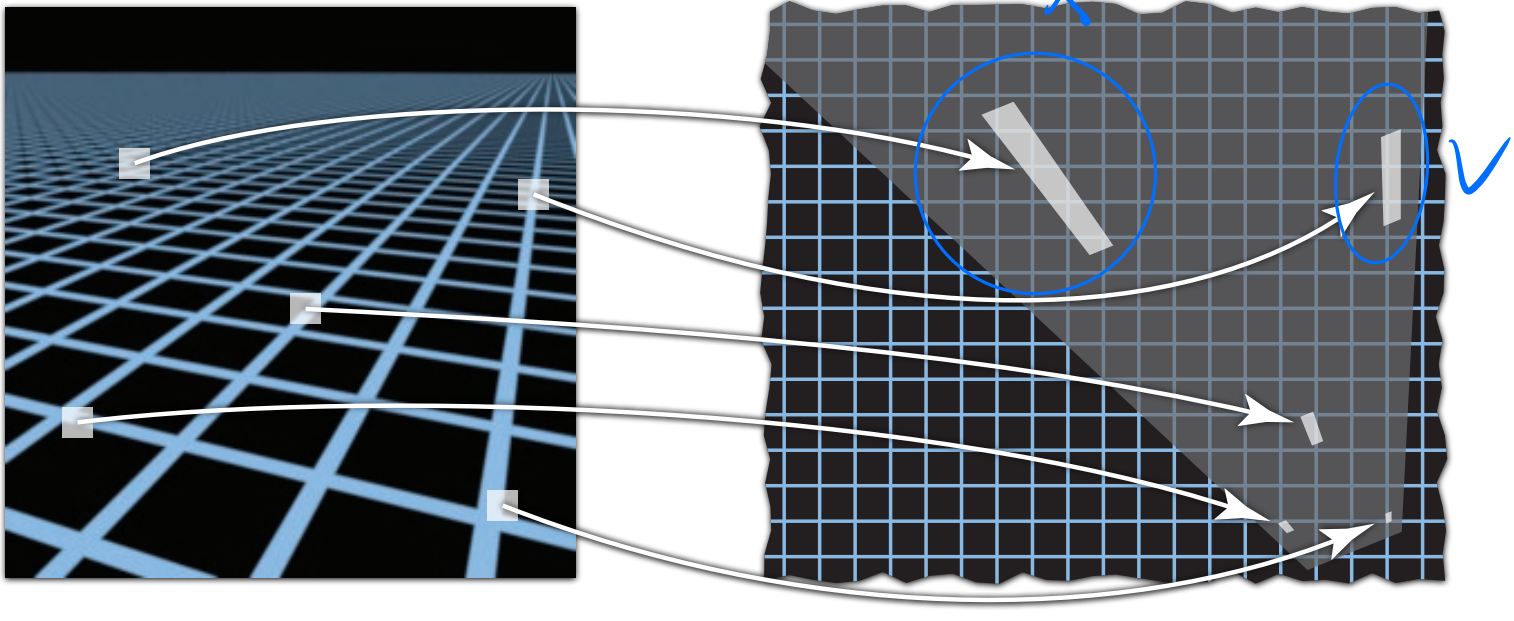
各向异性过滤

Anisotropic Filtering



Better than Mipmap!

Irregular Pixel Footprint in Texture



Screen space

Texture space

Anisotropic Filtering

Ripmaps and summed area tables

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

Wikipedia

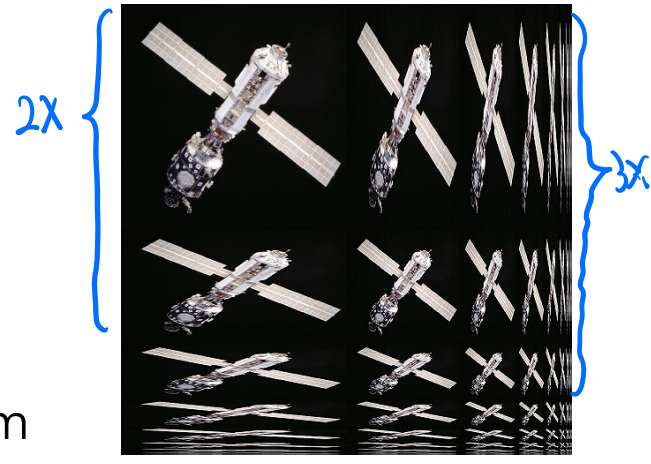
Anisotropic Filtering

额外开销：3倍（最多）

Ripmaps and summed area tables

可查询与坐标轴平行包围的区域

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

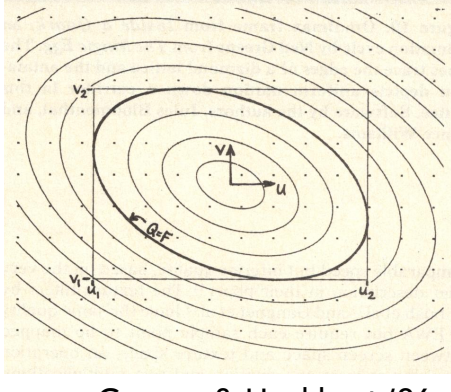


Wikipedia

EWA filtering

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

能处理不规则区域（但花费很大）



Greene & Heckbert ‘86

Thank you!

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