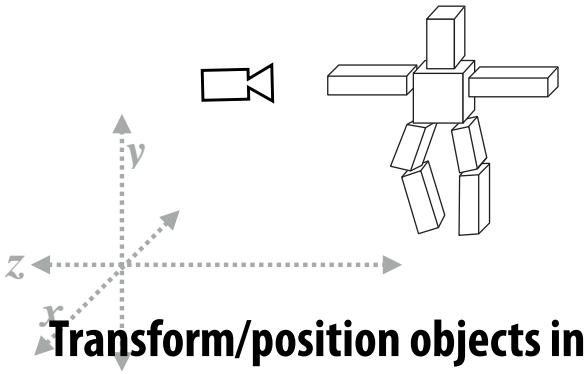
Lecture 6:

Texture Mapping

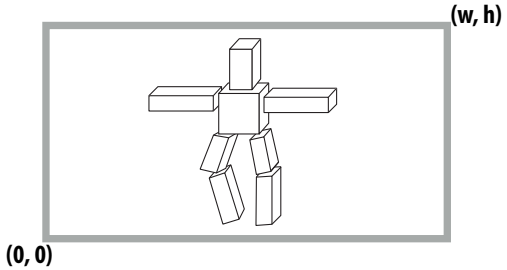
Computer Graphics 2025

Fuzhou University - Computer Science

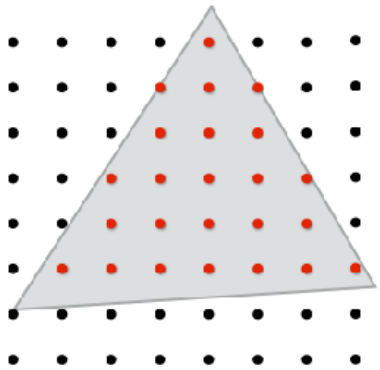
What We've Covered So Far



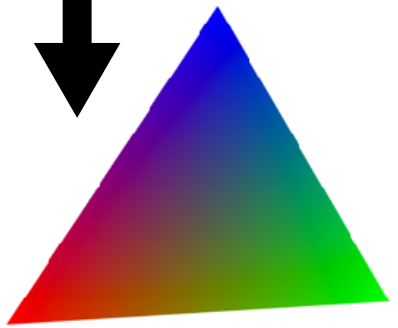
Transform/position objects in the world



Project objects onto the screen



Sample triangle coverage



Interpolate triangle attributes at covered samples

Sample texture maps / evaluate shaders

Combine samples into final image (depth, alpha, …)

Today's Topics

What is Texture Mapping?

Applying Textures

Texture Filtering

Drawing Surface with Complex Details

Texture Coordinate Mappings

Think Chocolate Wrappers

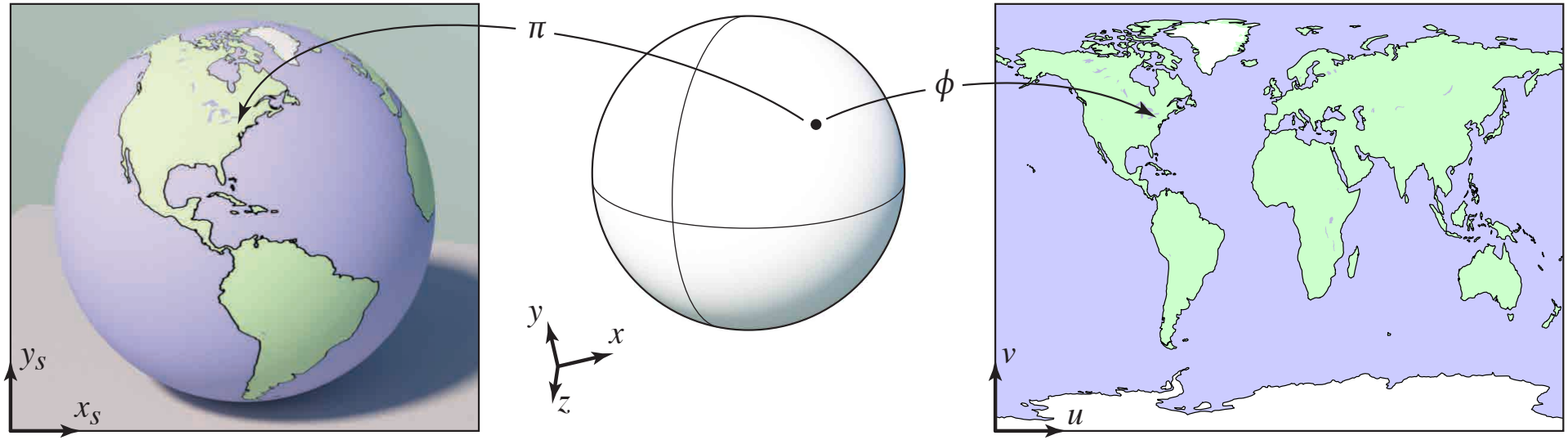
Texture image

Surfaces are 2D

三维物体的表面是二维的（可撕扯、摊开成二维图像）

Surface lives in 3D world space

Every 3D surface point also has a place where it goes in the 2D image and in the 2D texture.

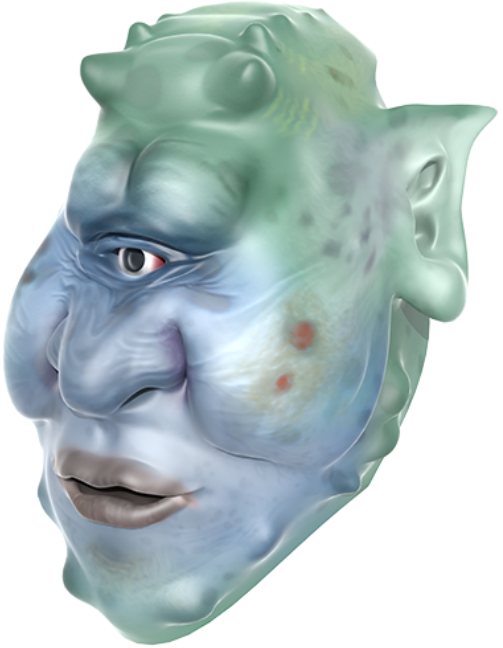


Screen space　　World space　　Texture space

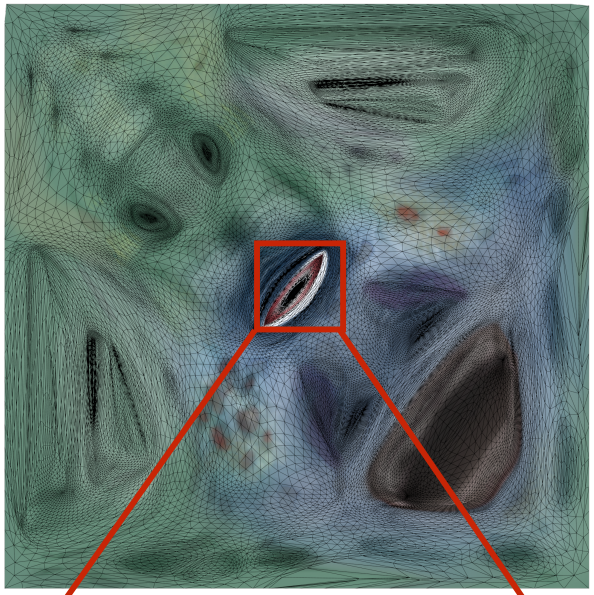
Texture Applied to Surface

Rendering without texture

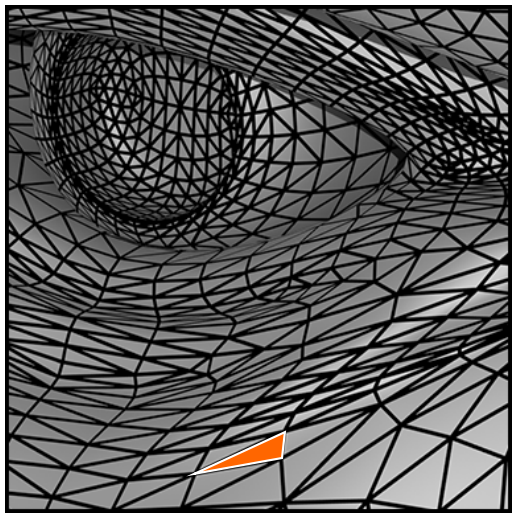
Rendering with texture

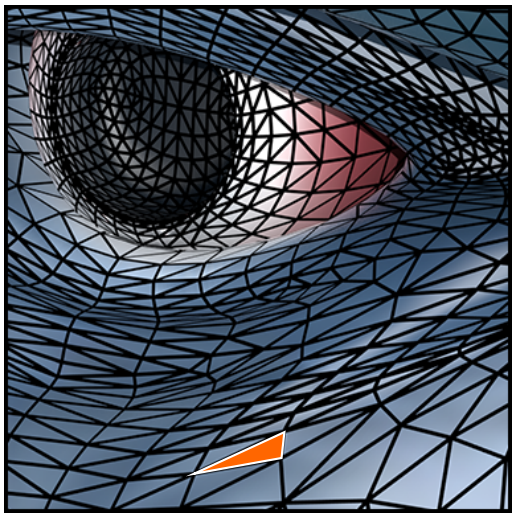


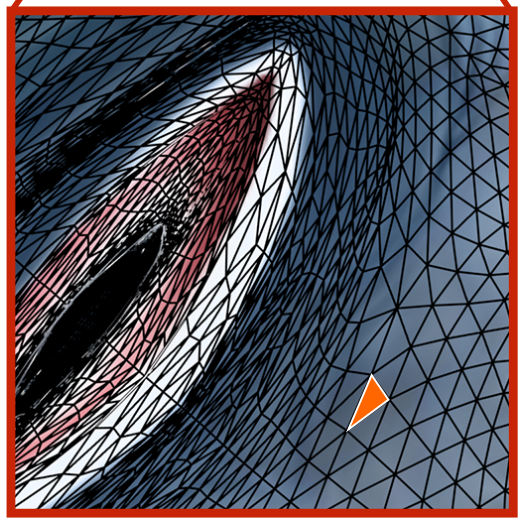
Texture image



Zoom





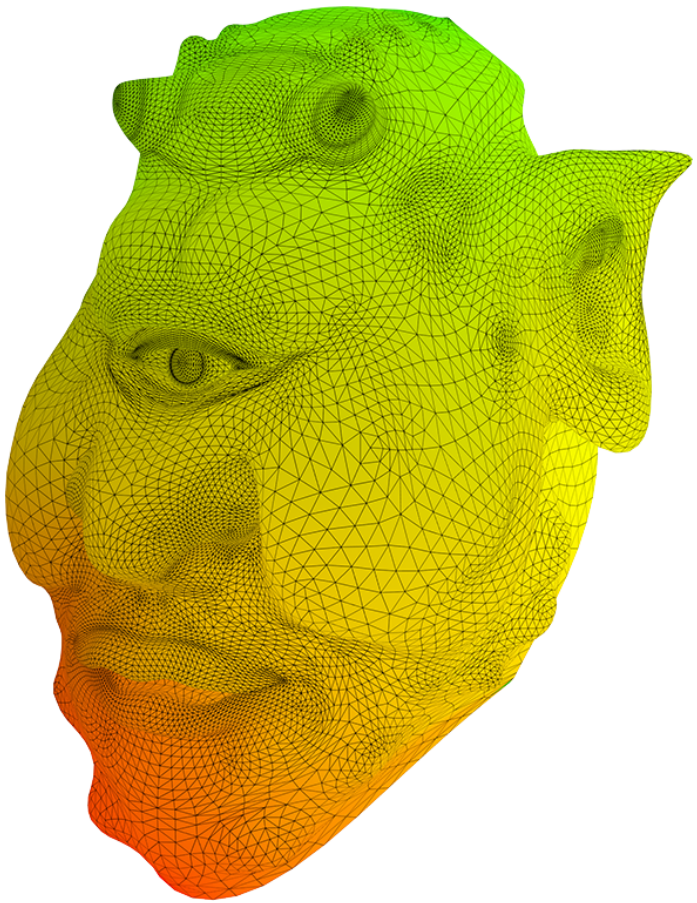


Each triangle "copies" a piece of the texture image back to the surface.

Visualization of Texture Coordinates

Each surface point is assigned a texture coordinate (u,v)

Visualization of texture coordinates



Triangle vertices in texture space

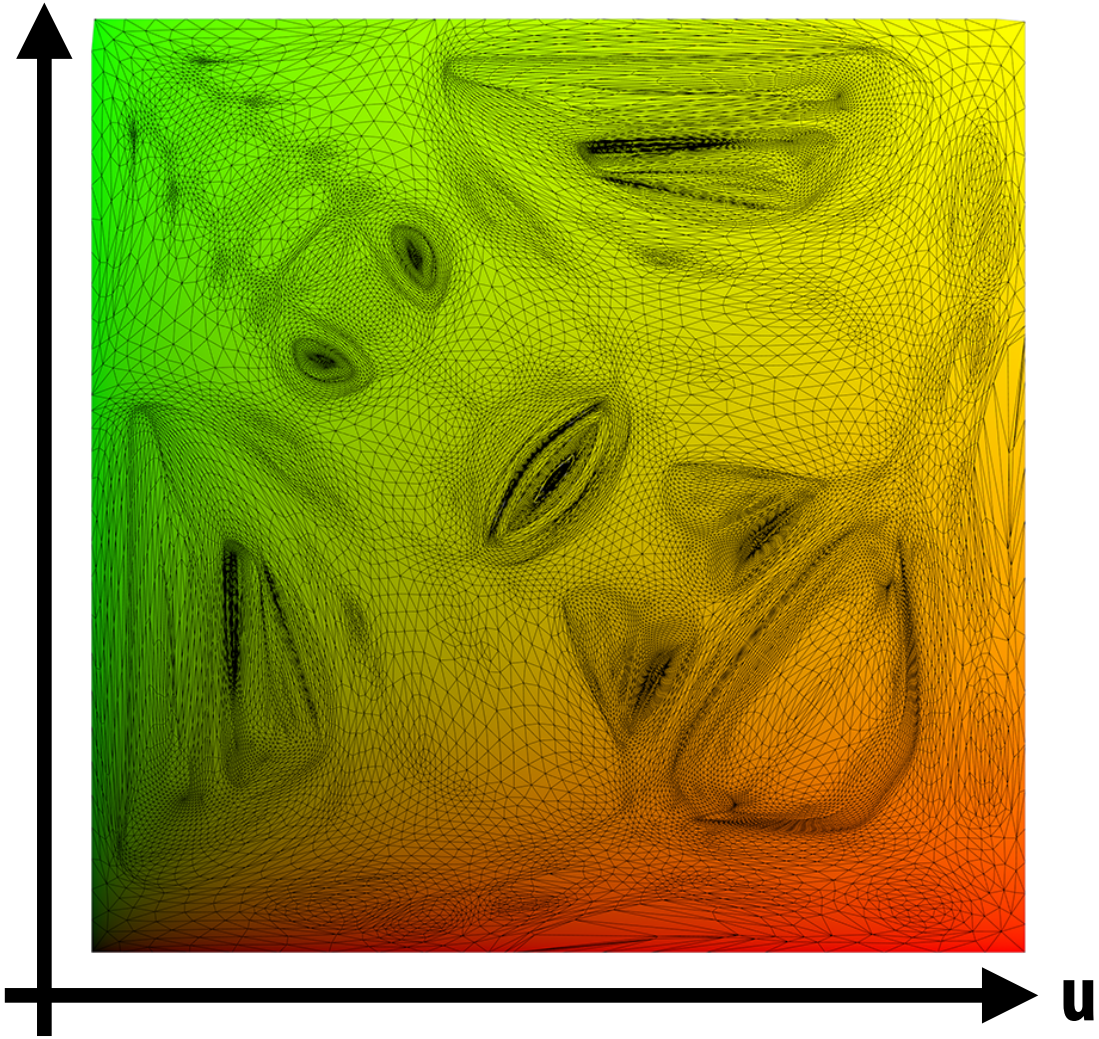
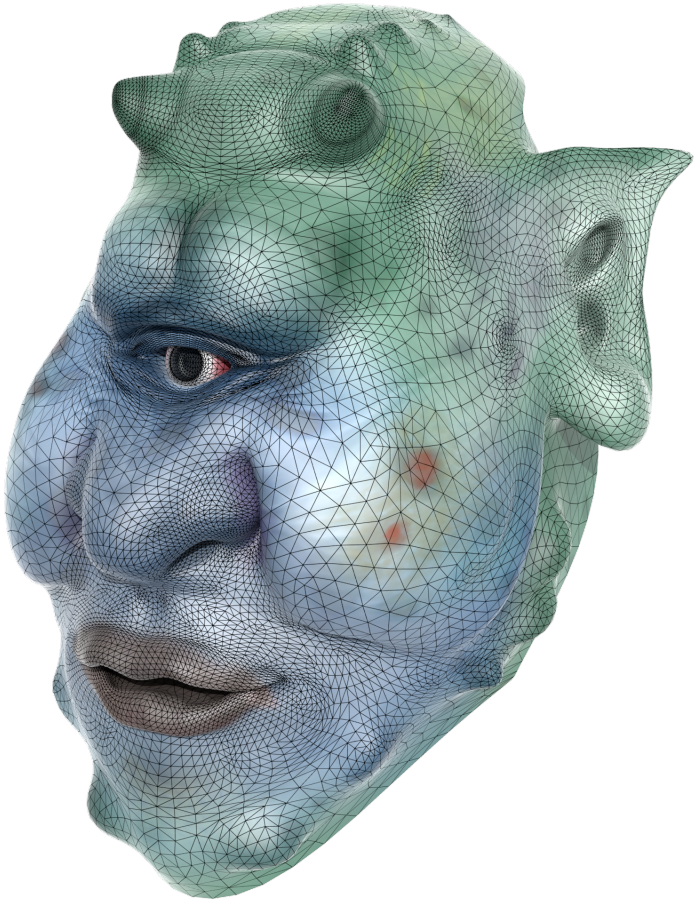


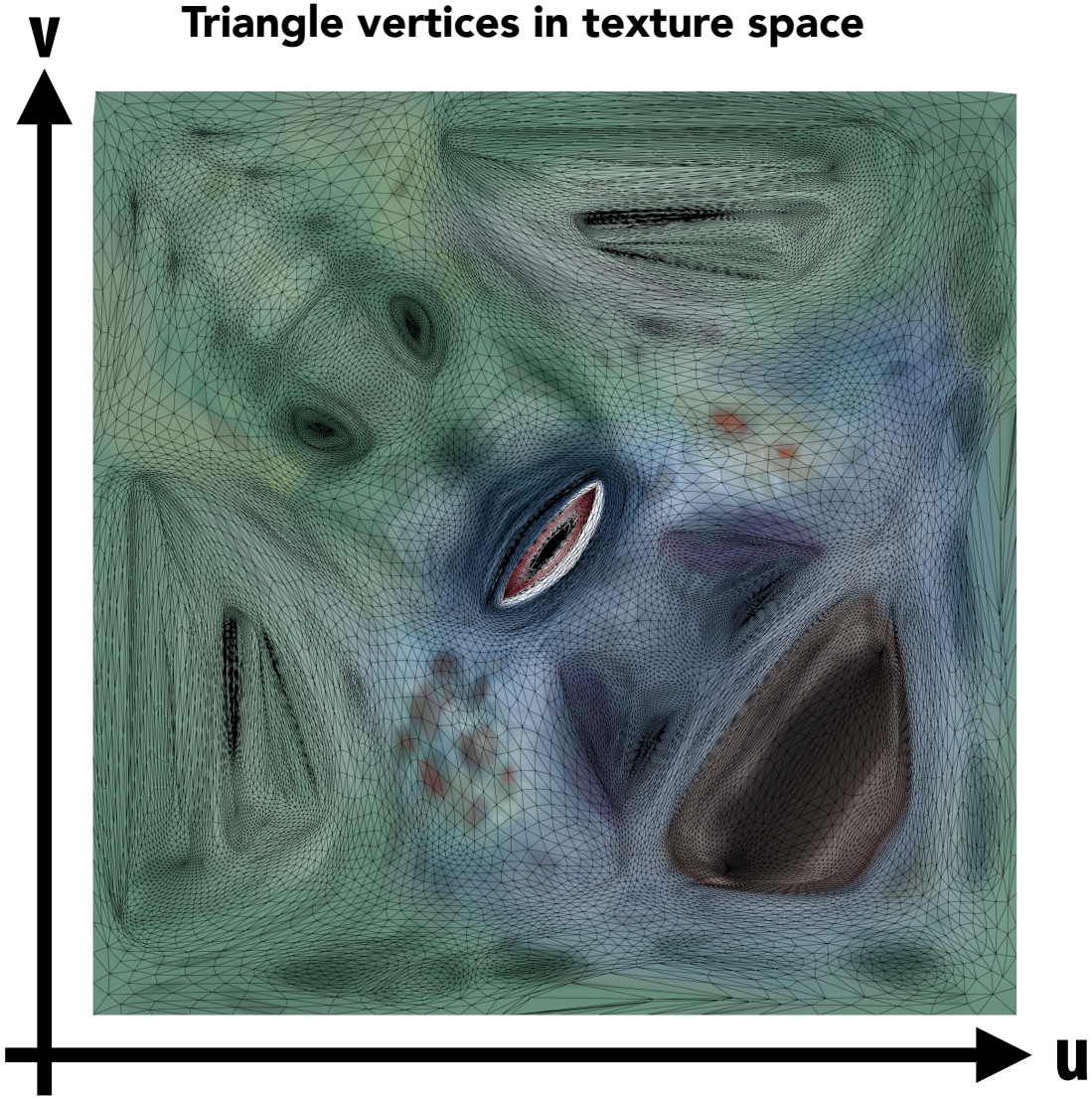
Image Texture Applied to Surface

Each surface point is assigned a texture coordinate (u,v)

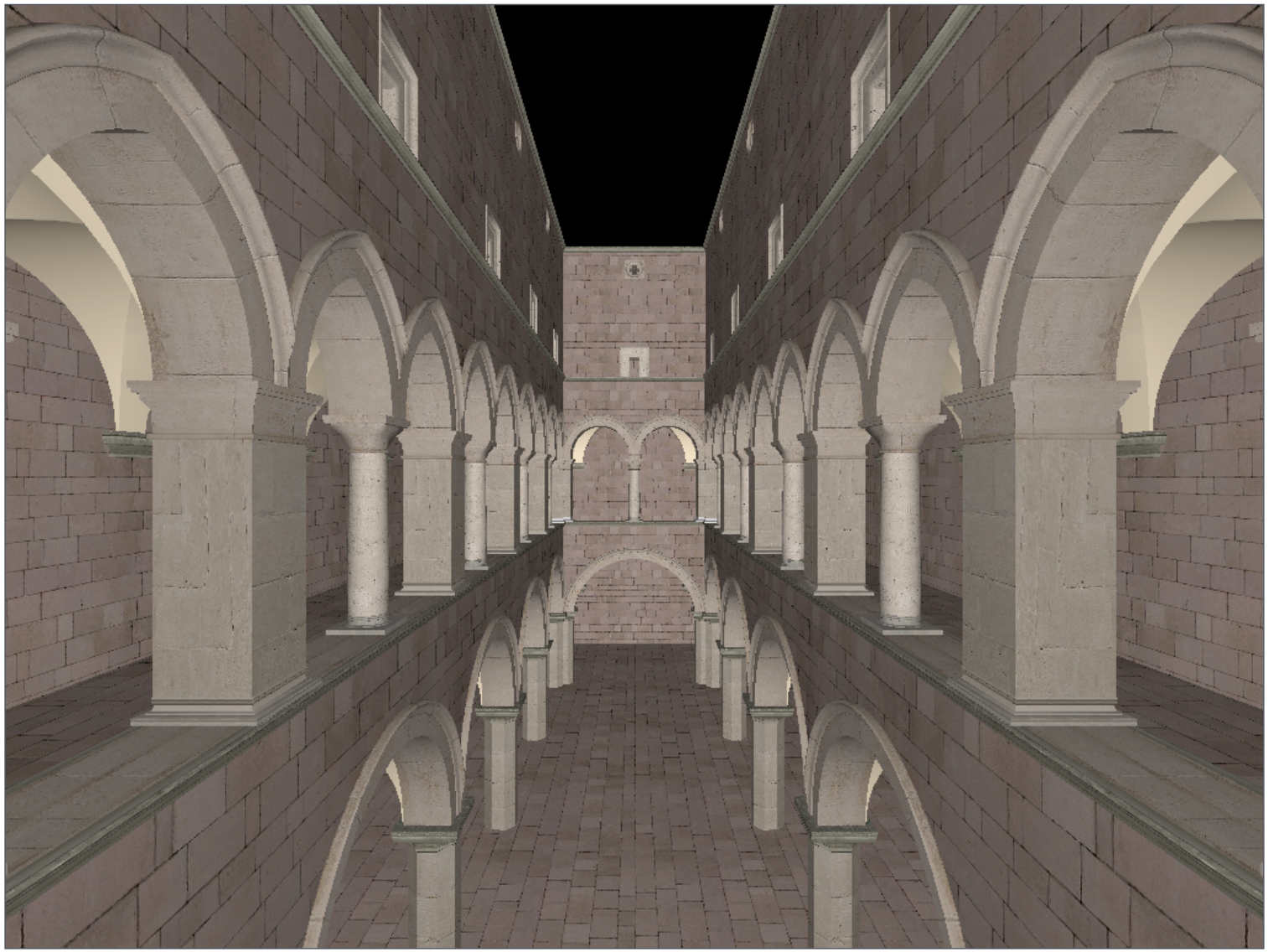
Rendered result



Triangle vertices in texture space

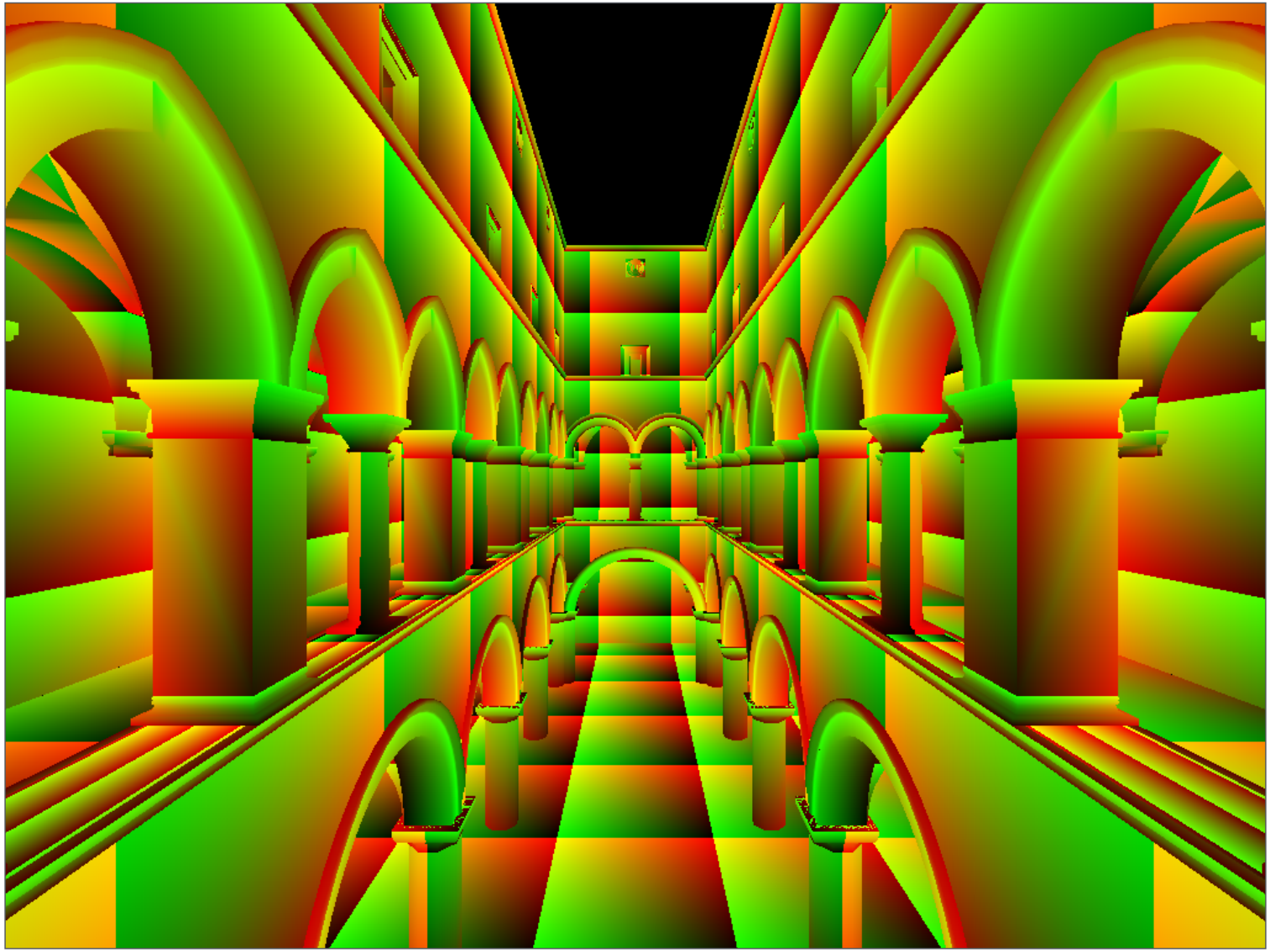


Sponza Palace Model



Textures applied to surfaces

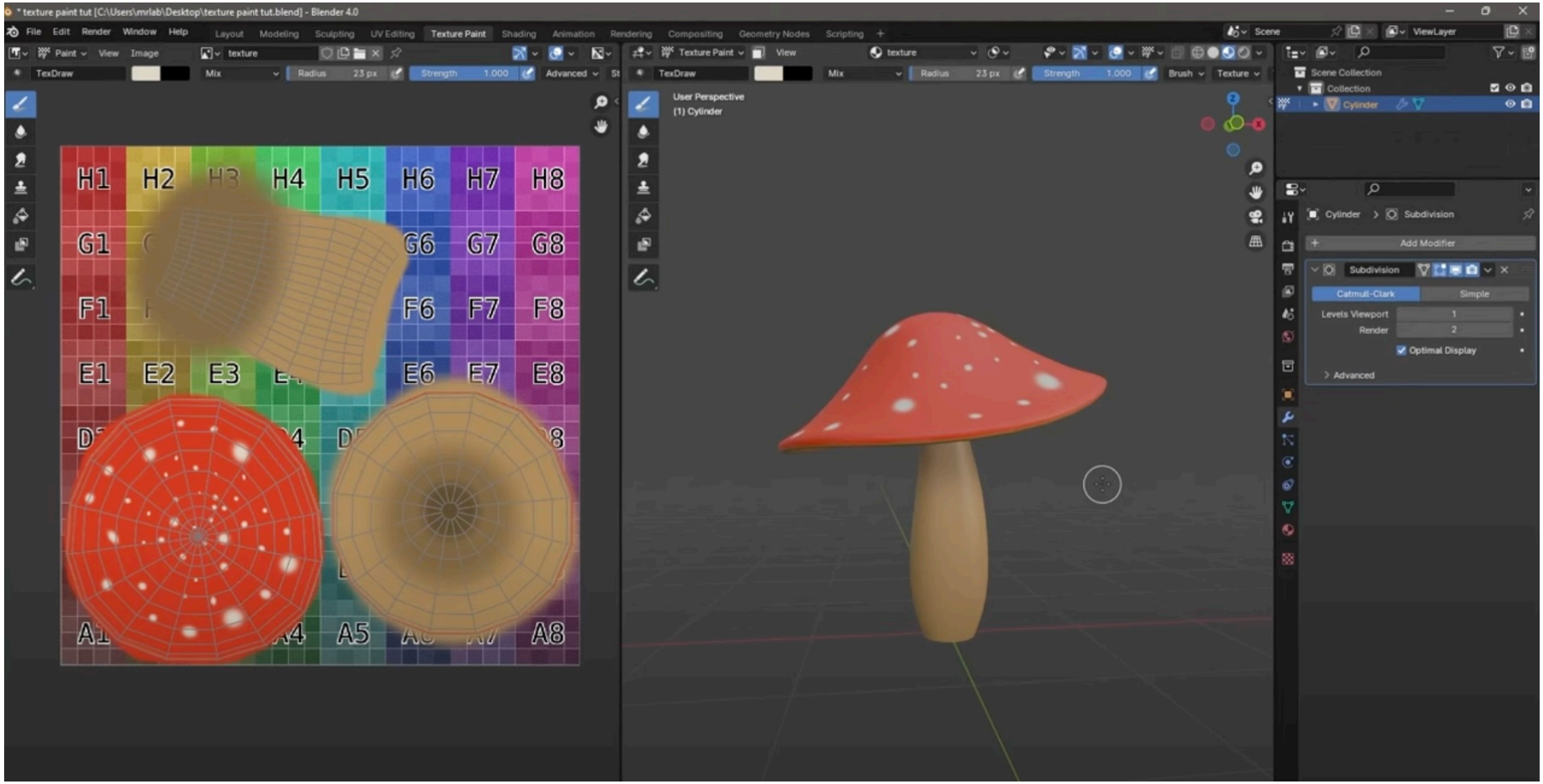
Sponza Palace Model



Visualization of texture coordinates

UV Unwrapping

Before a texture image is mapped onto the surface of a 3D object, UV unwrapping is required to flatten the surface into 2D space, and generate UV coordinates for each vertex of the object.

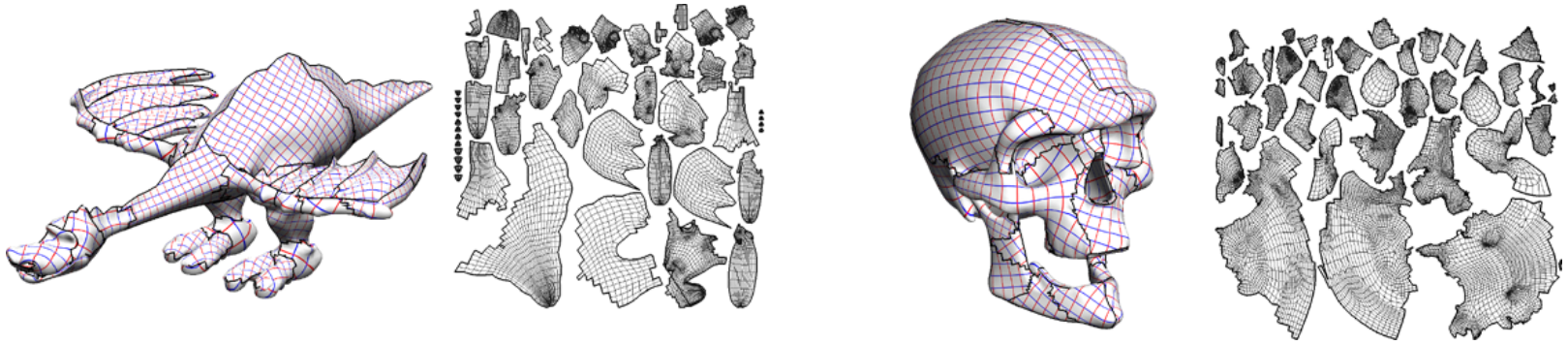


https://www.youtube.com/watch?v=-sPGZprp5JU

Creating Good Surface Coordinates is Hard

Finding cuts

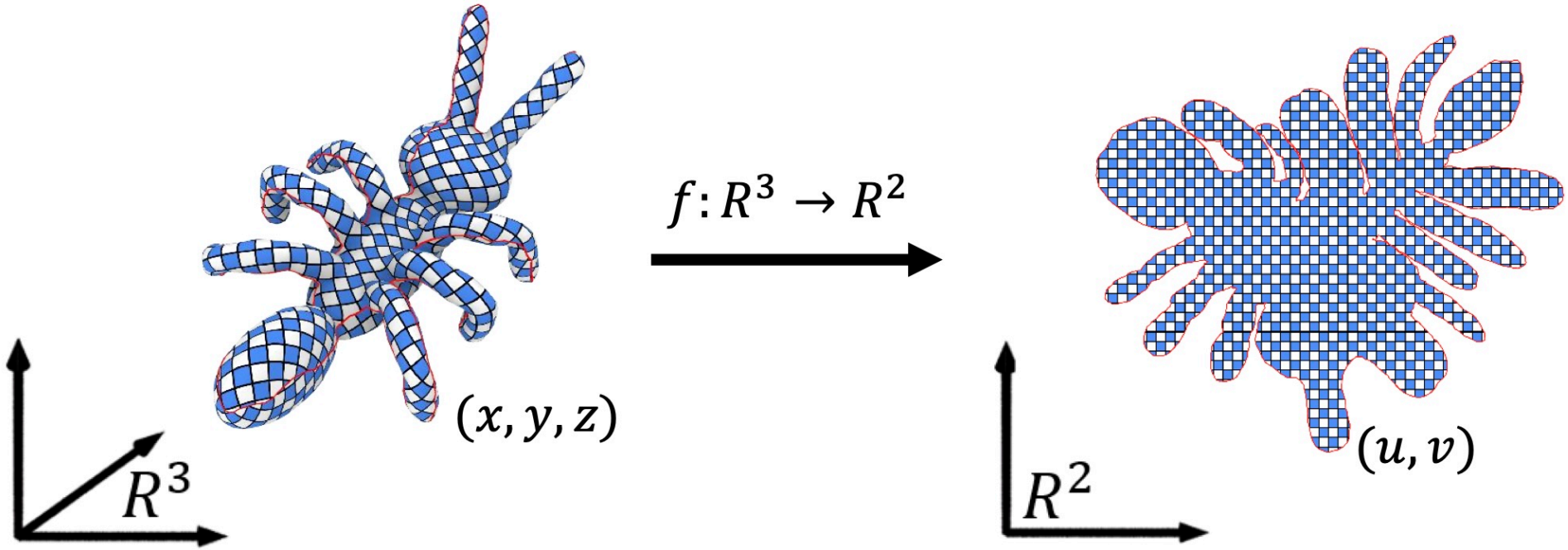
Texture atlases



Levy et al: Least Squares Conformal Maps
for Automatic Texture Atlas Generation, SIGGRAPH, 2002

曲面参数化

参考GAMES301课程



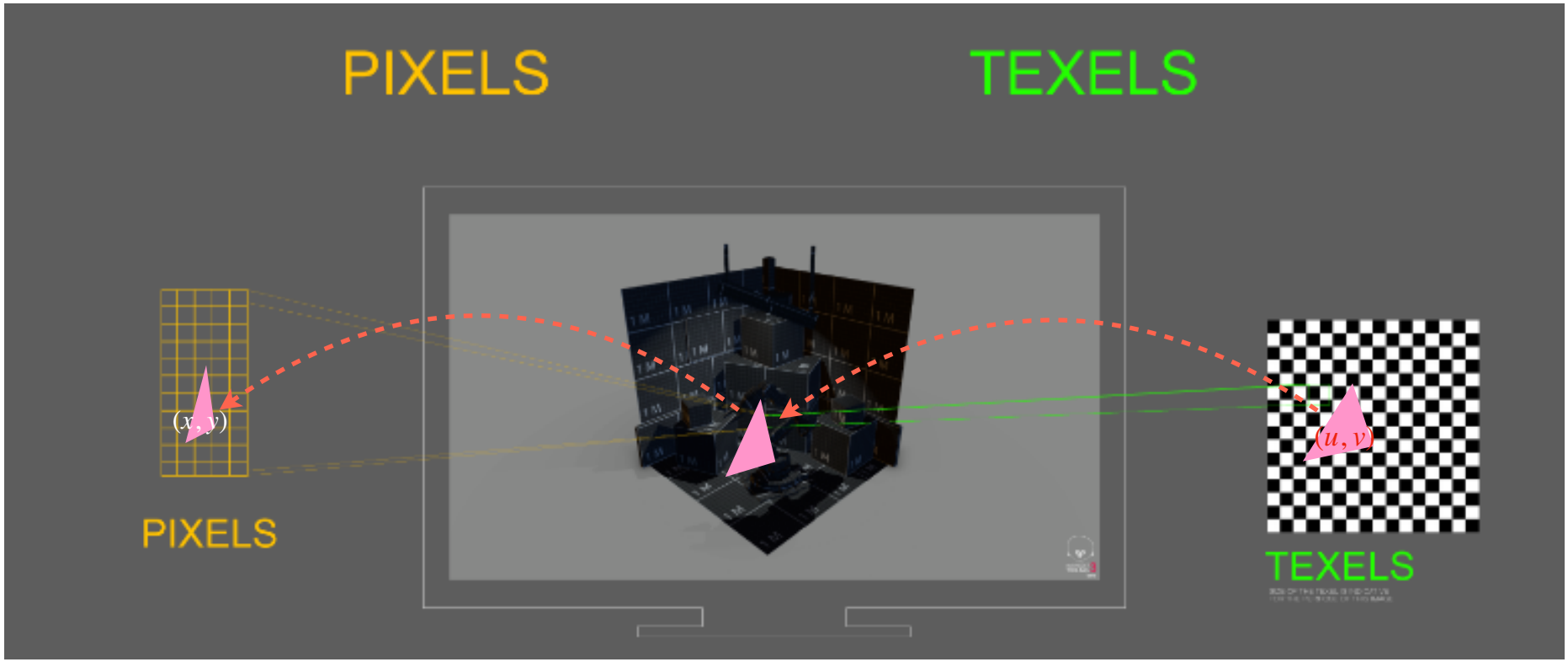
Today's Topics

What is Texture Mapping?

Applying Textures

Texture Filtering

Simple Texture Mapping Operation



Simple Texture Mapping Operation

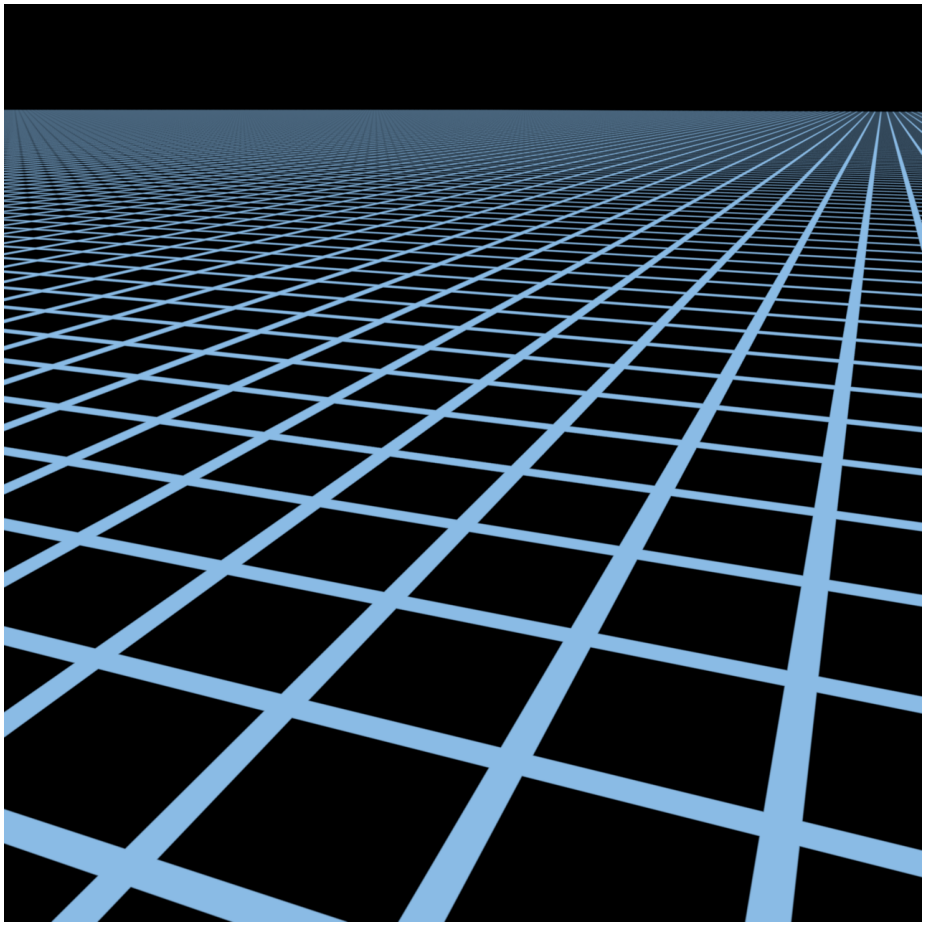
```
for each rasterized screen sample (x,y):
    (u,v) = evaluate texcoord value at (x,y)
    float3 texcolor = texture.sample(u,v);
    set sample's color to texcolor;
```

Interpolation using barycentric coordinates!

Usually the diffuse coefficient k_d

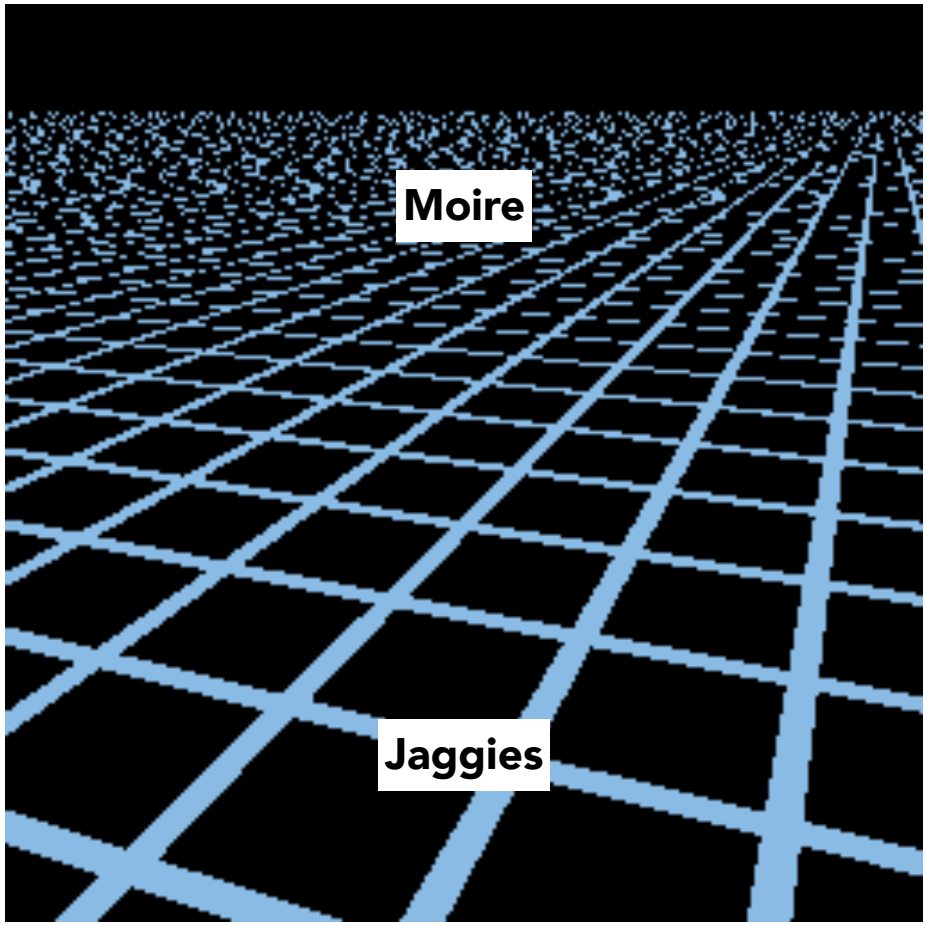
(recall the Blinn-Phong reflectance model)

Point Sampling Textures



High-res reference

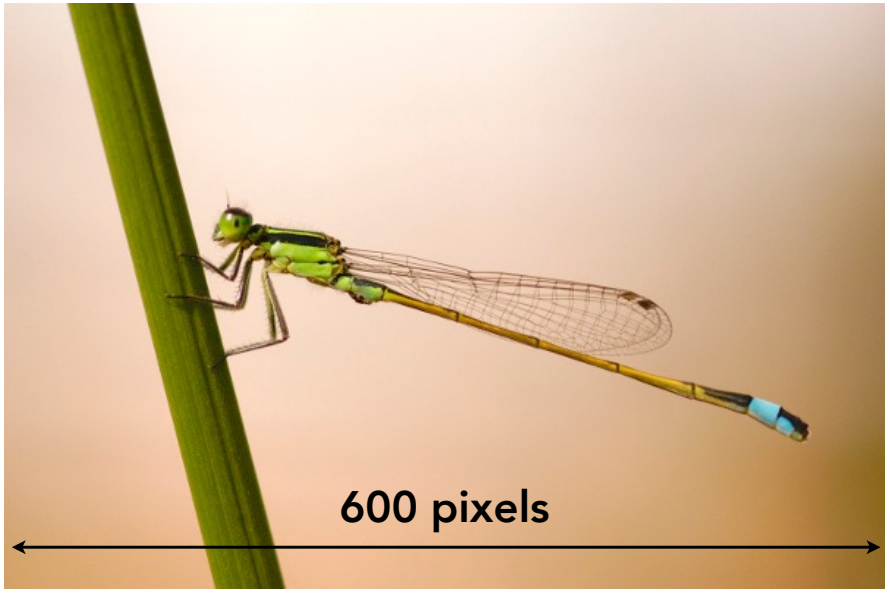
Source image: 1280x1280 pixels

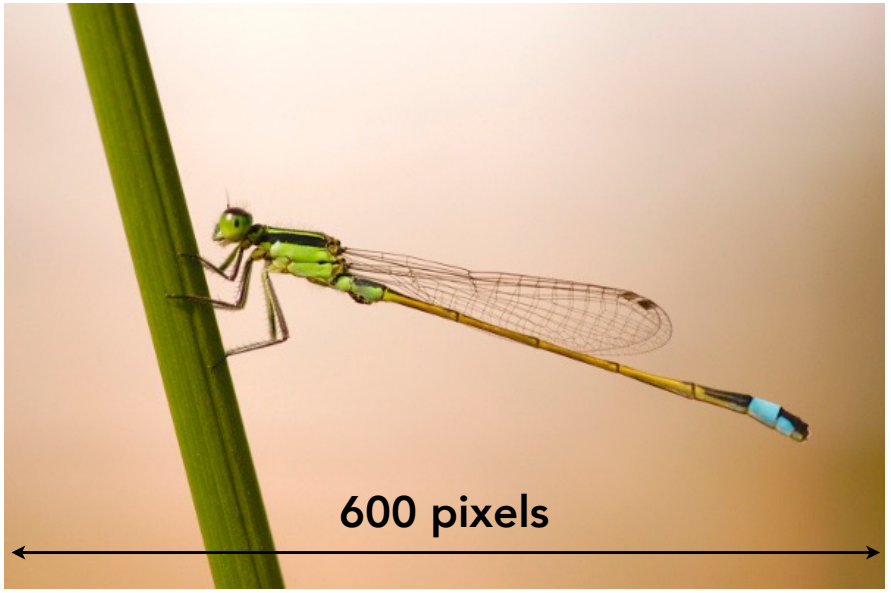


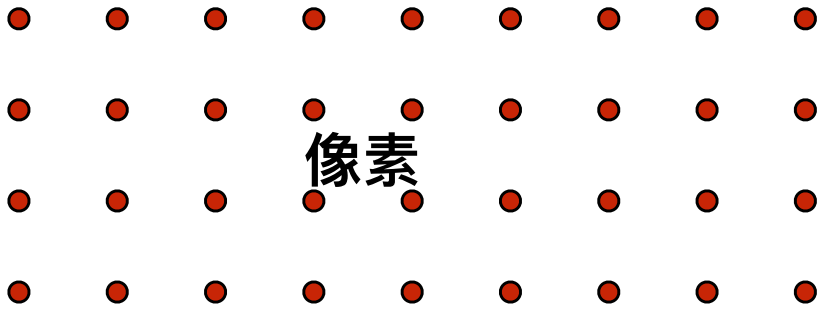
Point sampling

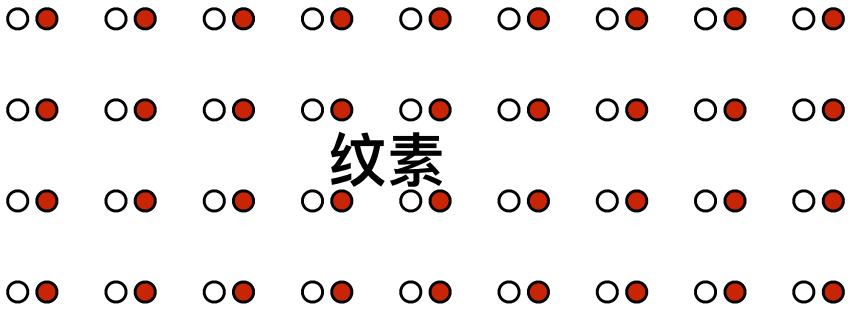
256x256 pixels

Sampling Rate on Screen vs Texture







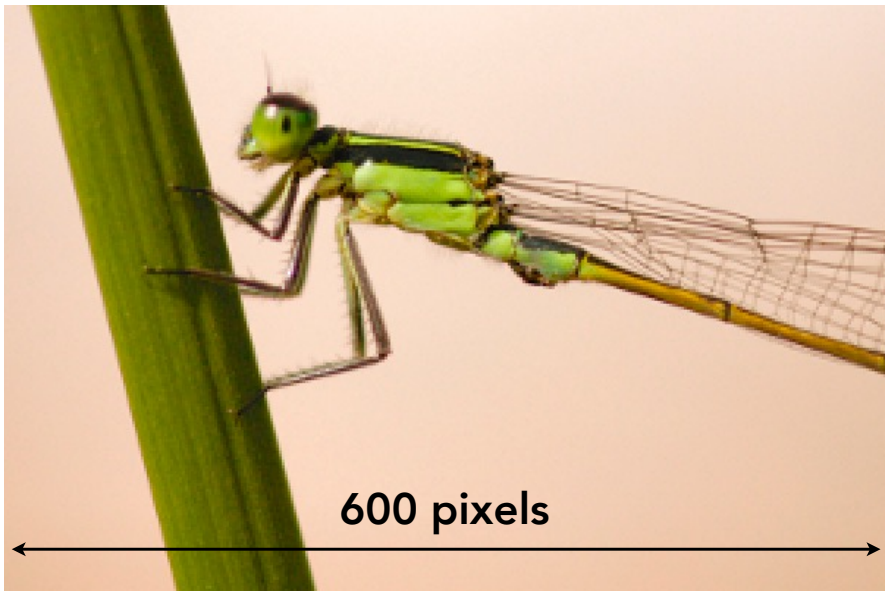


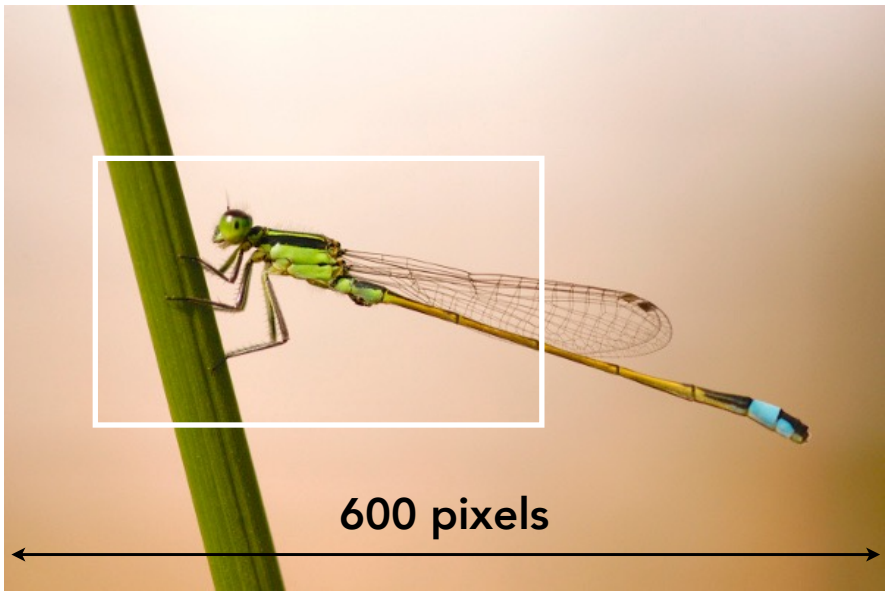
Screen space (x,y)

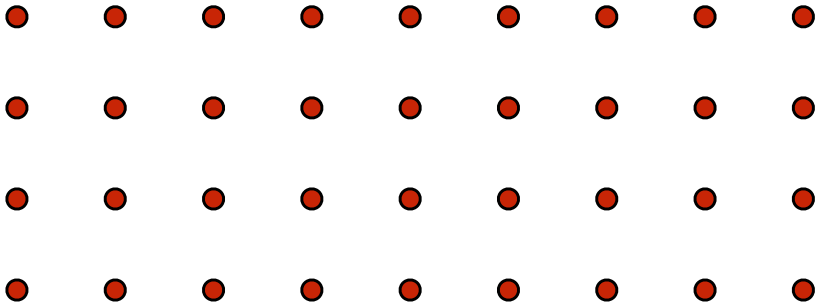
Texture space (u,v)

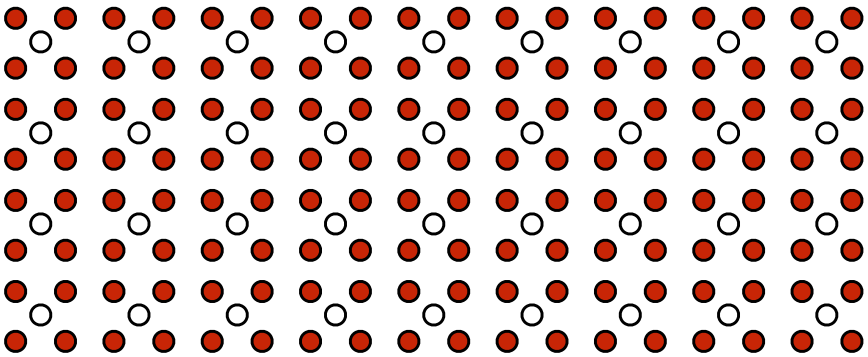
1:1 mapping

Sampling Rate on Screen vs Texture







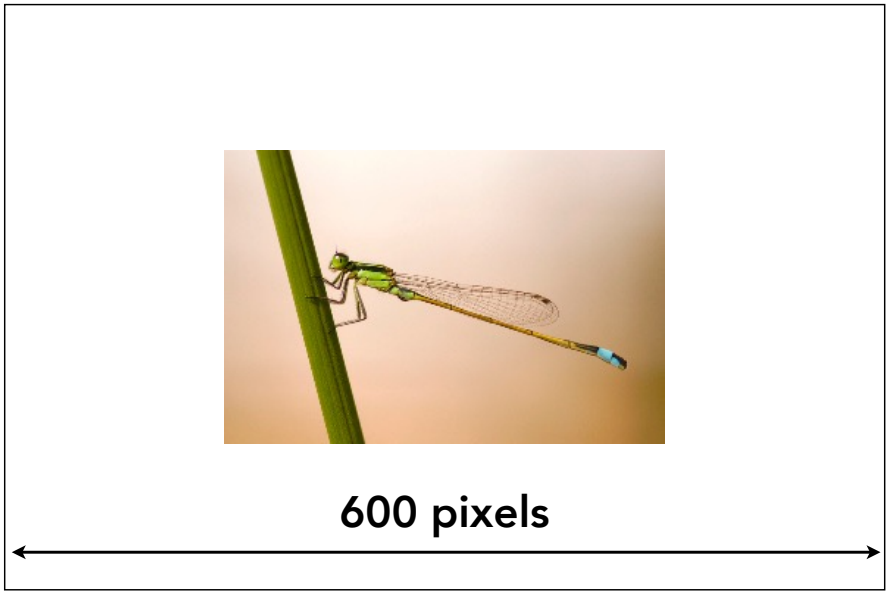


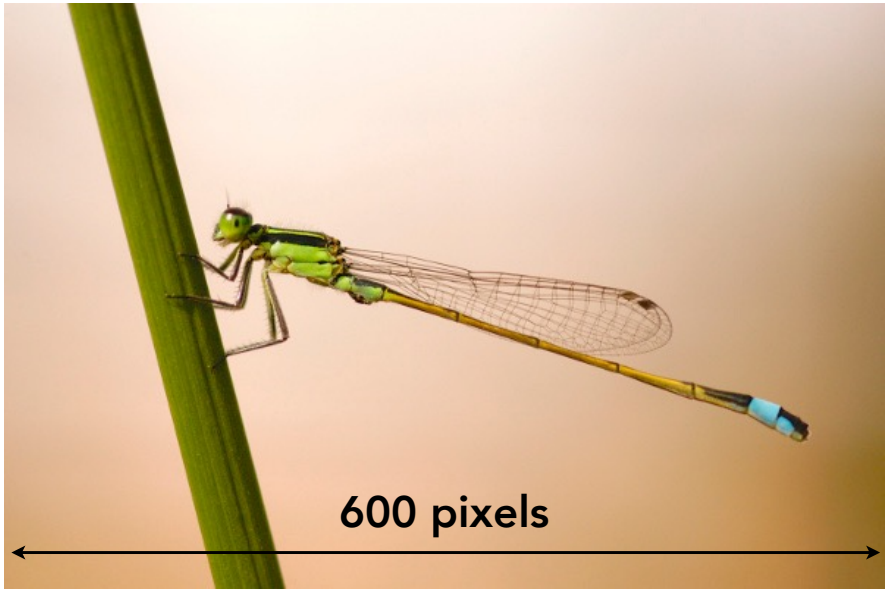
Screen space (x,y)

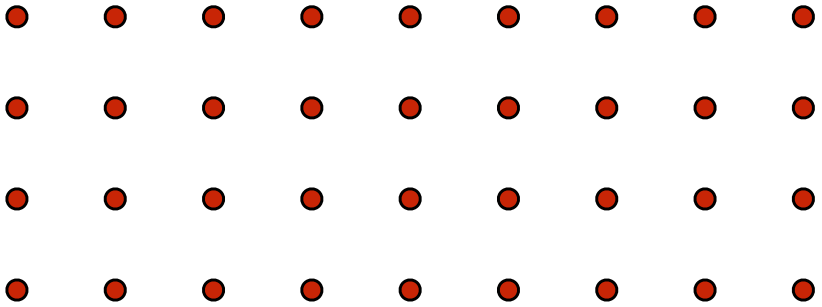
Texture space (u,v)

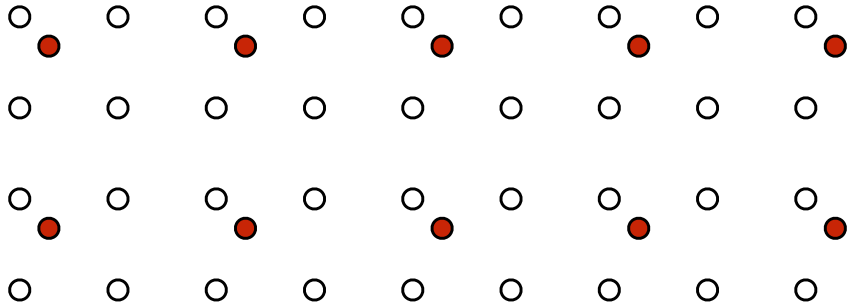
Magnified

Sampling Rate on Screen vs Texture









Screen space (x,y)

Texture space (u,v)

"Minified"

Screen Pixel Area vs Texel Area

At optimal viewing size:

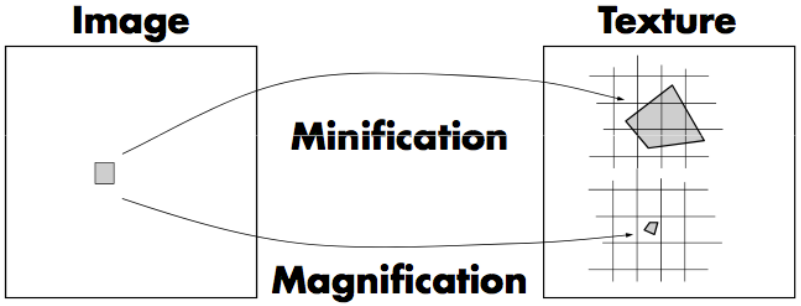
- 1:1 mapping between pixel sampling rate and texel sampling rate
- Dependent on texture resolution! e.g. 512x512

When larger (magnification)

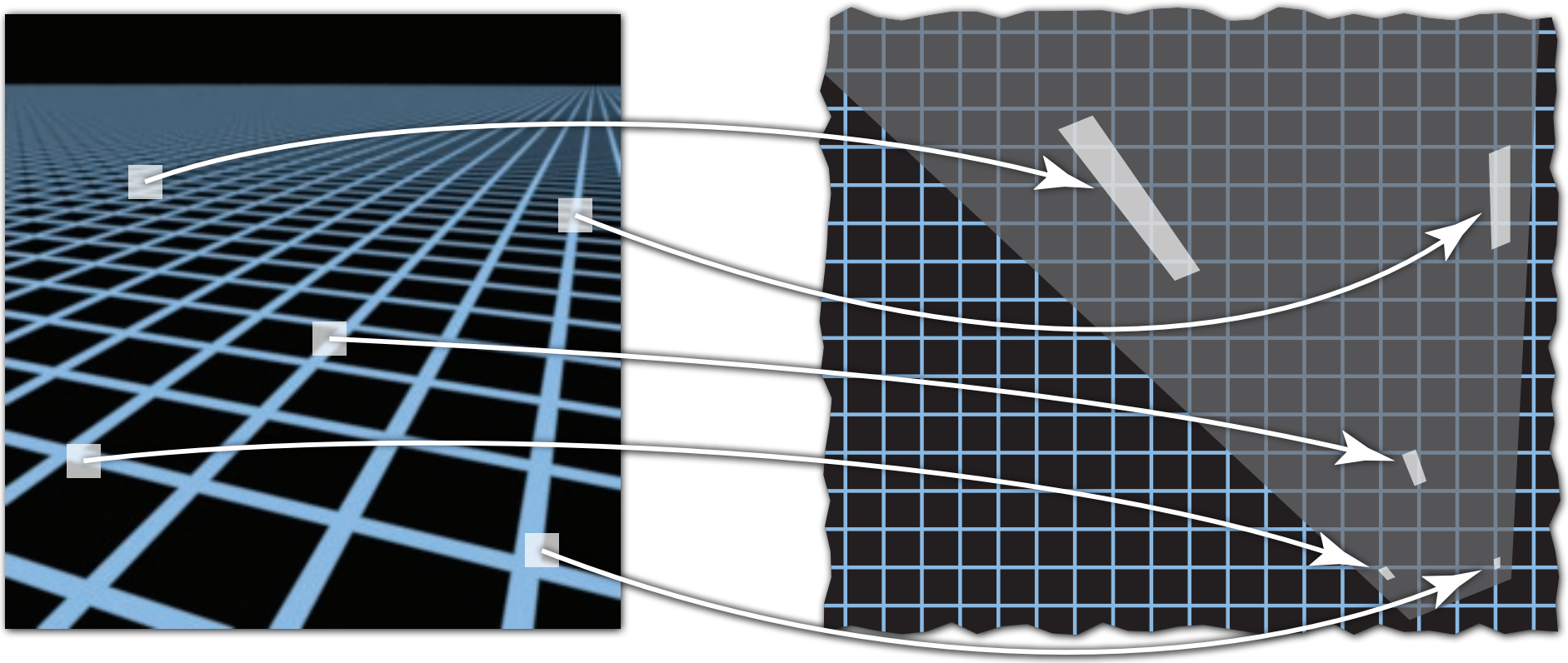
- Multiple pixel samples per texel sample

When smaller (minification)

- One pixel sample per multiple texel samples



Screen Pixel Footprint in Texture



Screen space

Texture space

NB: texture sampling pattern not rectilinear or isotropic

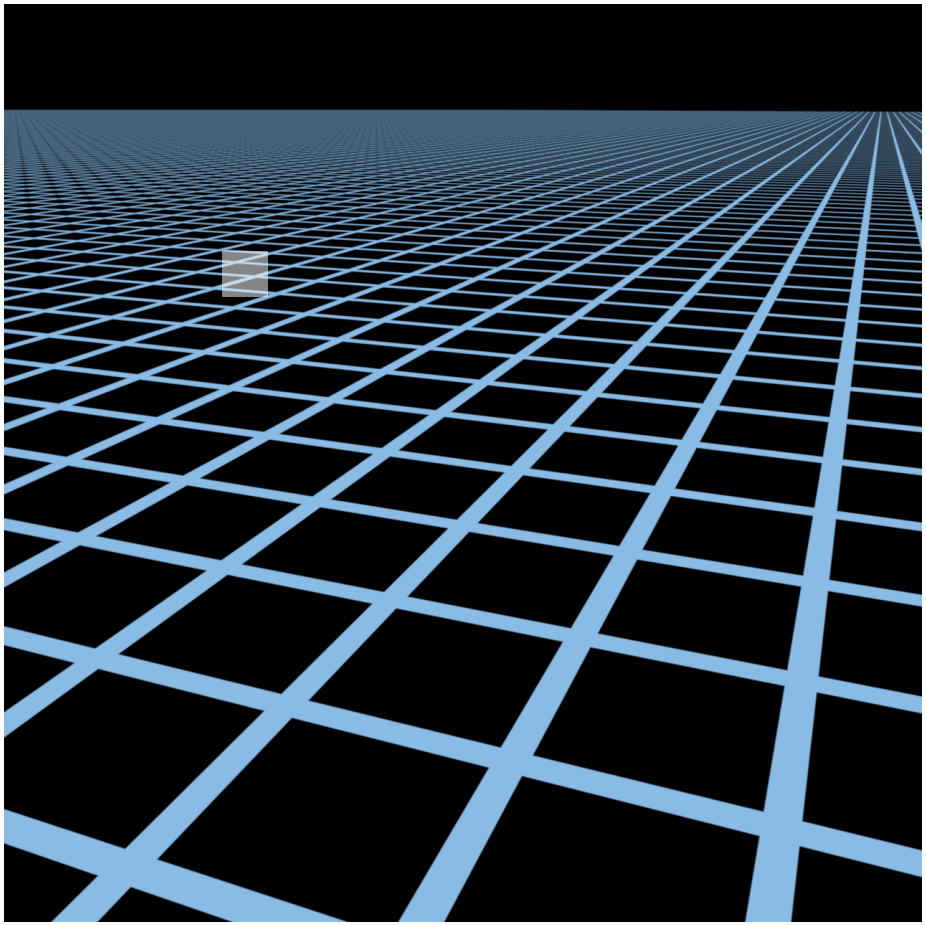
Today's Topics

What is Texture Mapping?

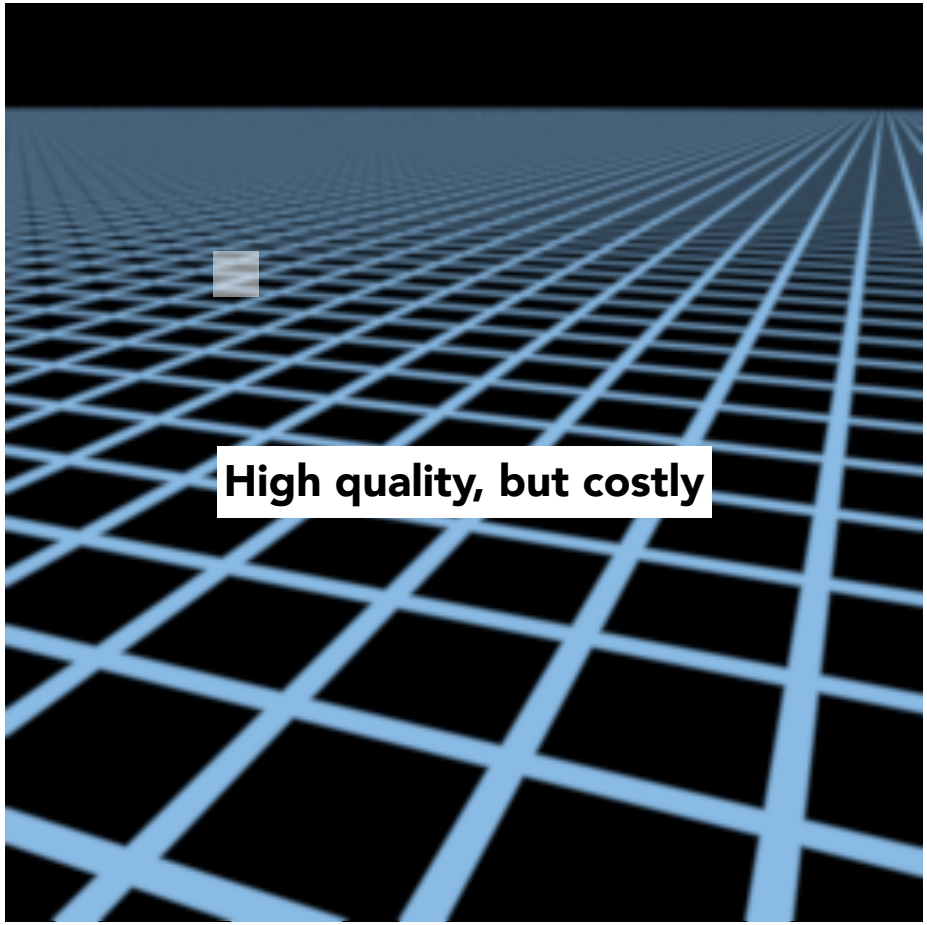
Applying Textures

Texture Filtering

Will Supersampling Antialias?



High-res reference



512x supersampling

Texture Antialiasing

Will supersampling work?

- Yes, high quality, but costly
- When highly minified, many texels in pixel footprint

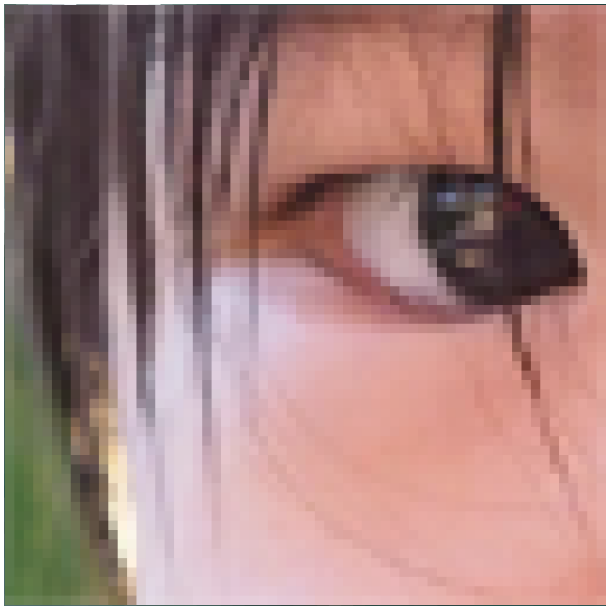
Goal: efficient texture antialiasing

- Want antialiasing with one/few texels per pixel
- How? Antialiasing = filtering before sampling!

Texture Magnification - Easy Case

(Generally don't want this — insufficient resolution)

This is image interpolation (will see kernel function)

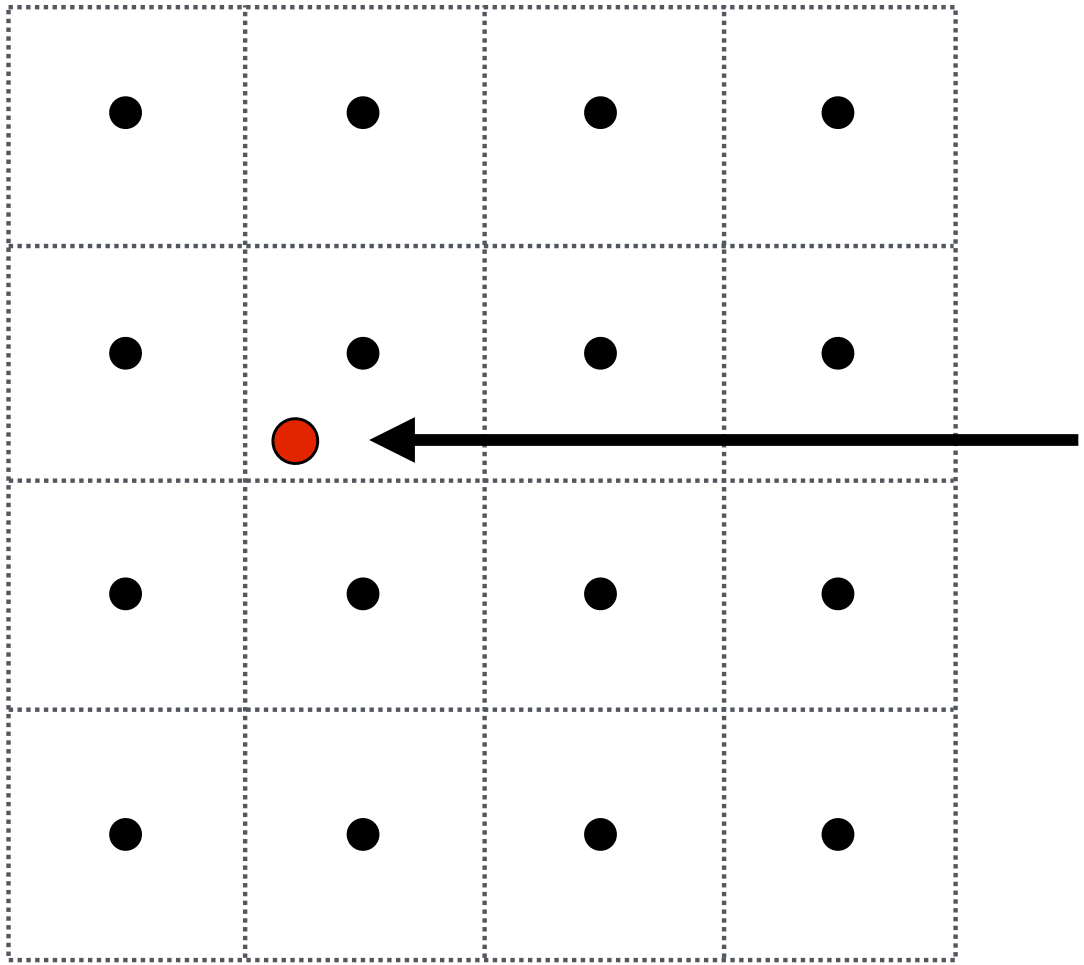


Nearest

Bilinear

Bicubic

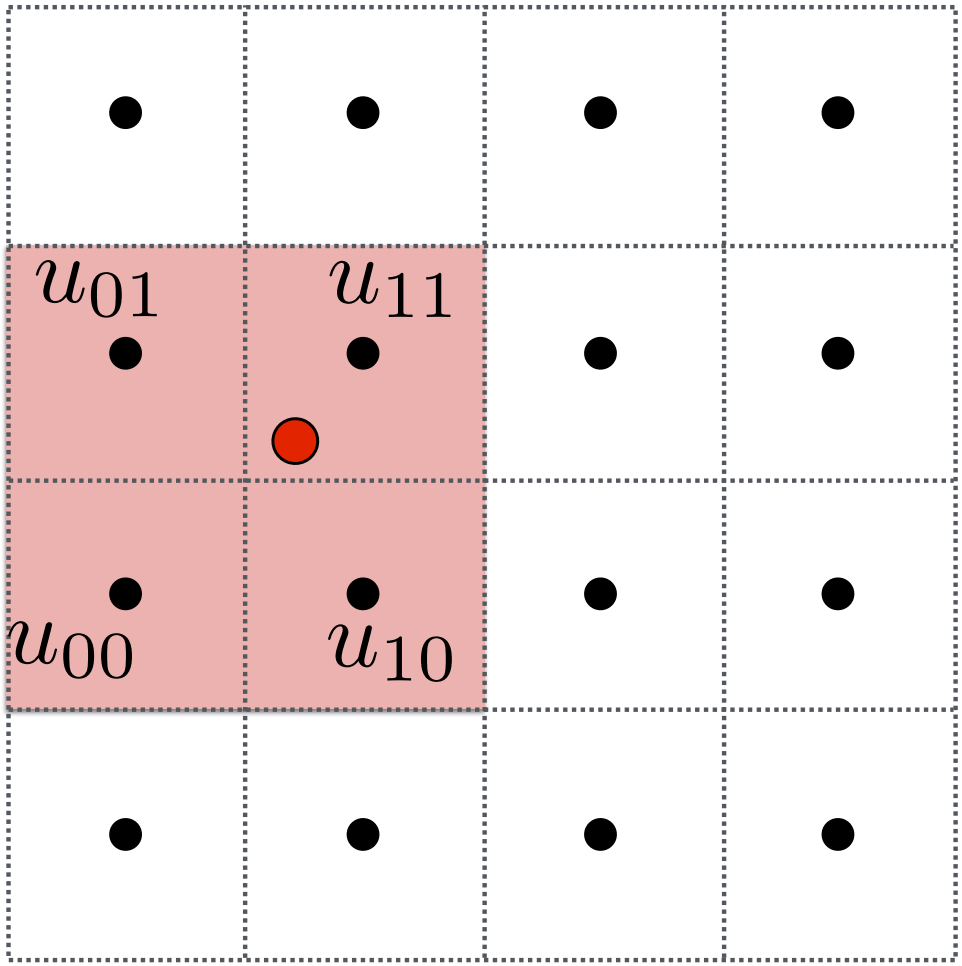
Bilinear Filtering



Want to sample texture value f(u,v) at red point

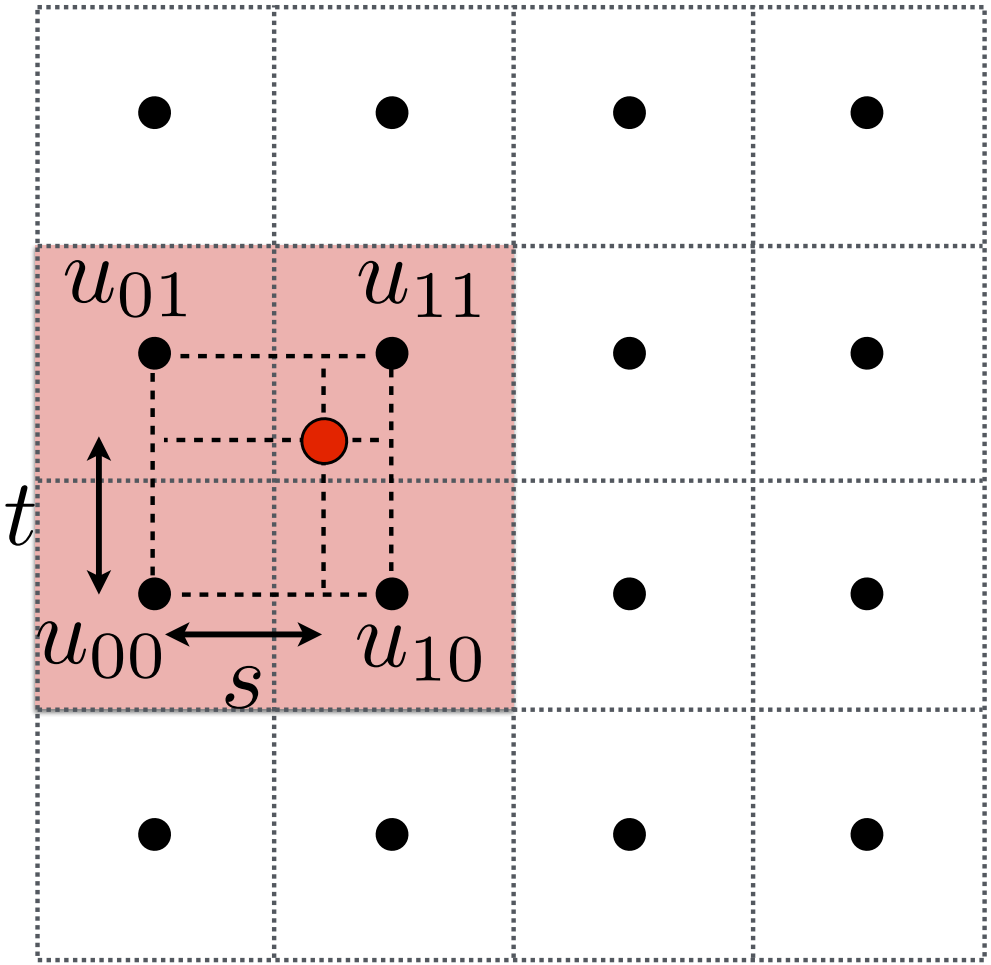
Black points indicate texture sample locations

Bilinear Filtering



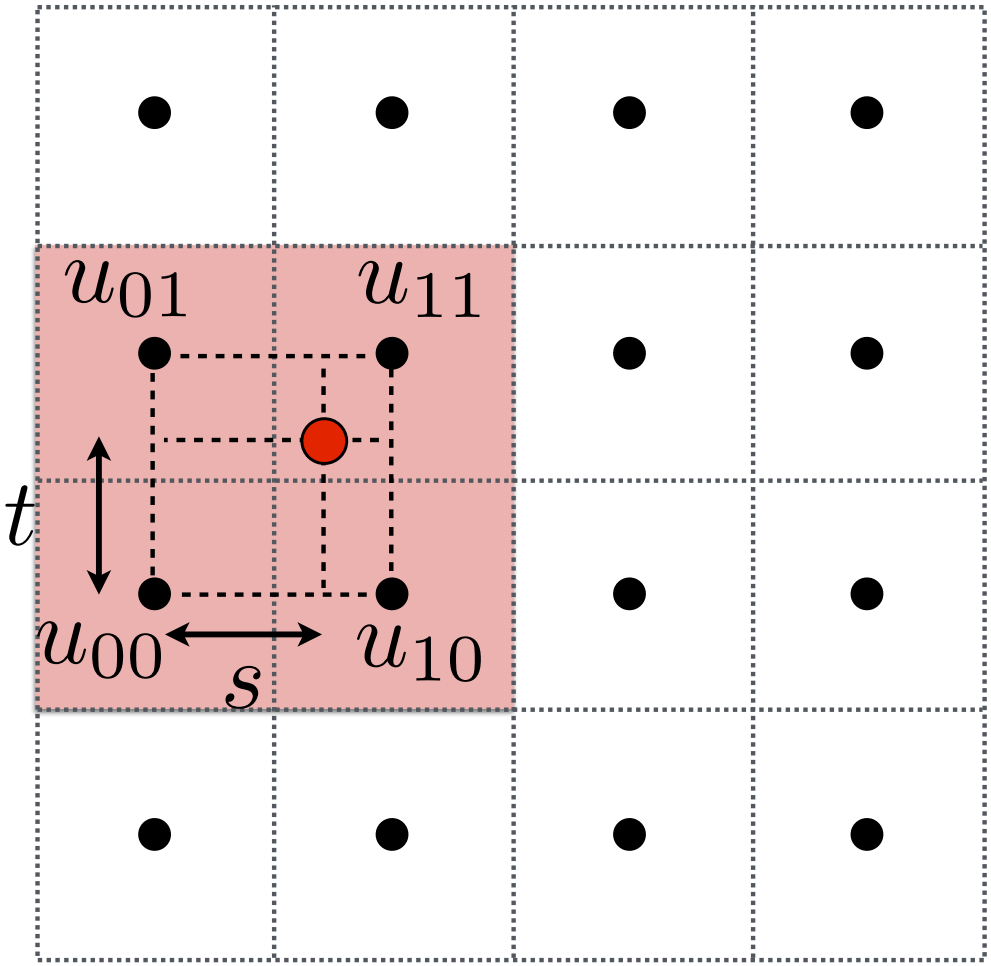
Take 4 nearest sample locations, with texture values as labeled.

Bilinear Filtering



And fractional offsets, (s,t) as shown

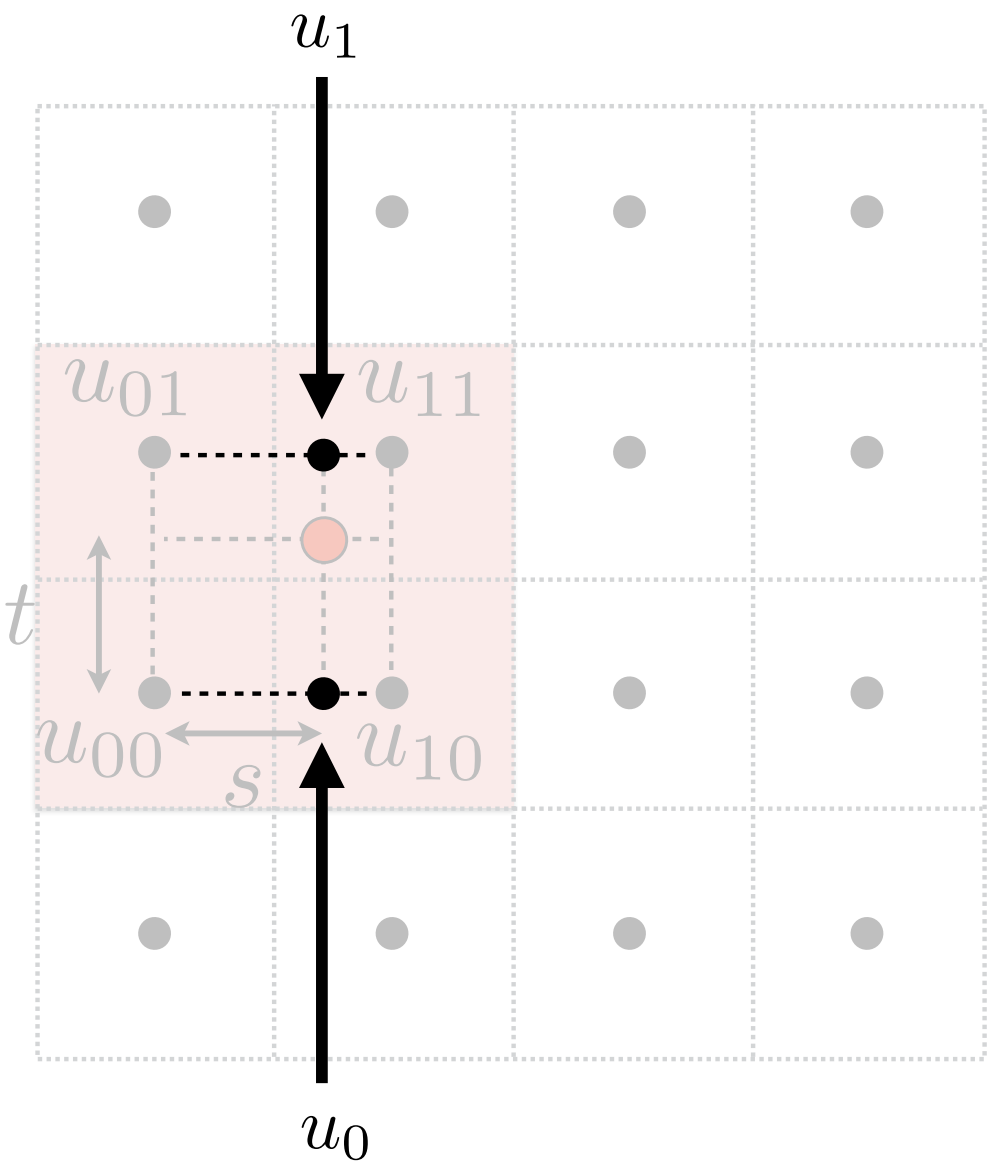
Bilinear Filtering



Linear interpolation (1D)

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

Bilinear Filtering



Linear interpolation (1D)

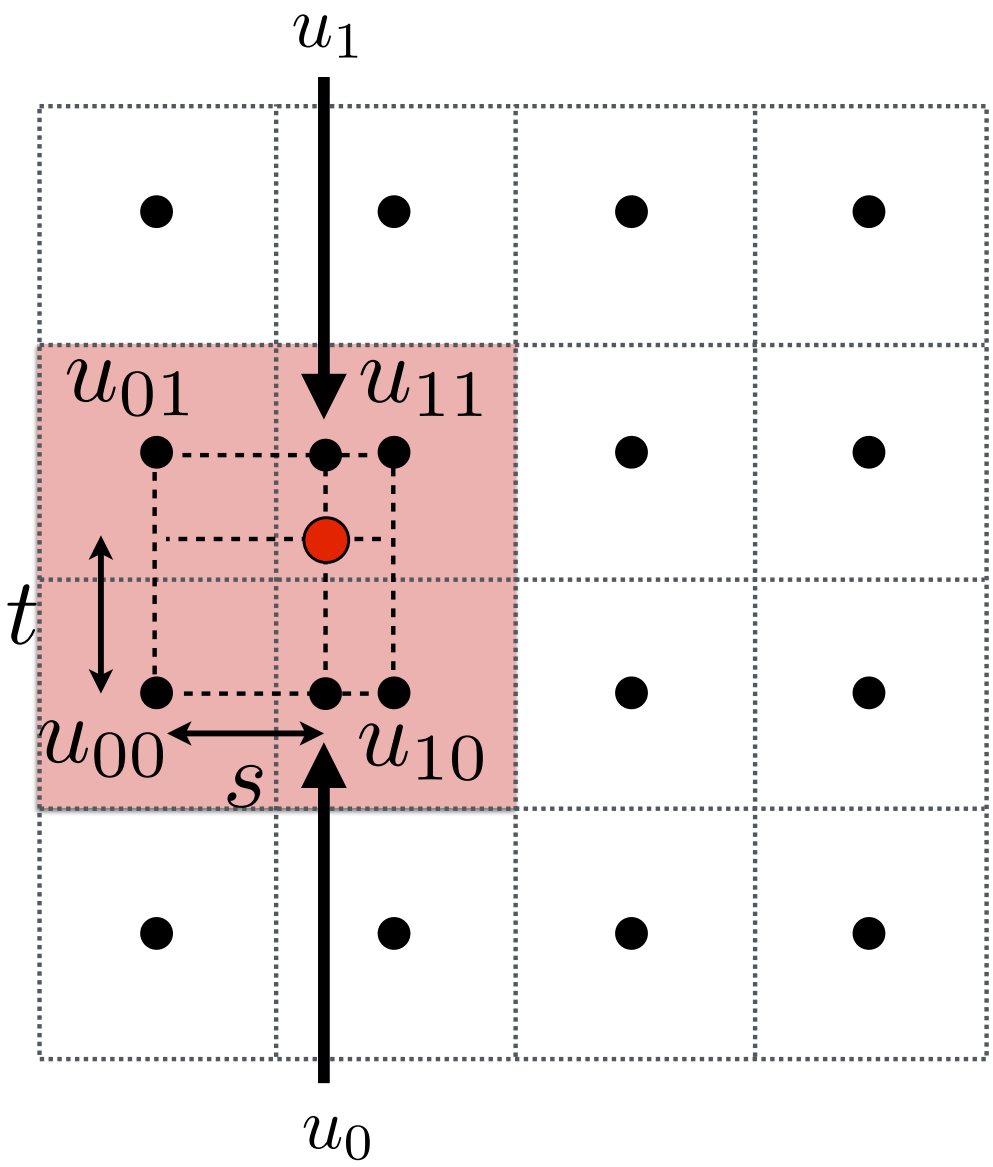
$$\text{lerp}(x, v_0, v_1) = v_0 + x(v_1 - v_0)$$

Two helper lerps (horizontal)

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

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

Bilinear Filtering



Linear interpolation (1D)

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

Two helper lerps

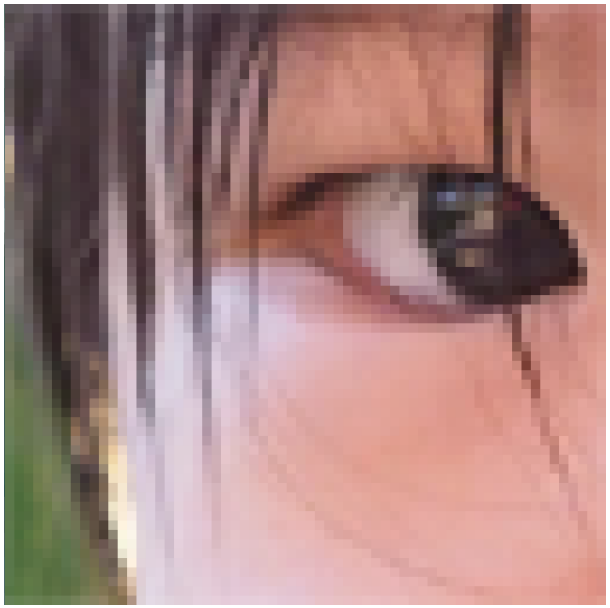
$$u_0 = \text{lerp}(s, u_{00}, u_{10})$$

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

Final vertical lerp, to get result:

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

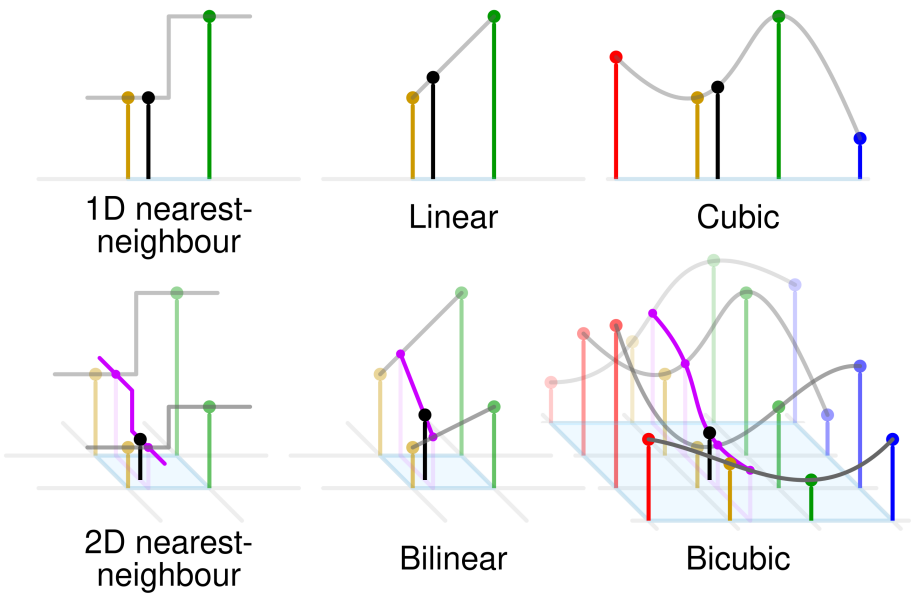
Texture Magnification - Easy Case



Nearest

Bilinear

Bicubic



Texture Minification - Hard Case

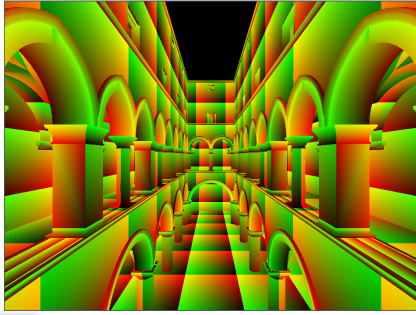
Challenging

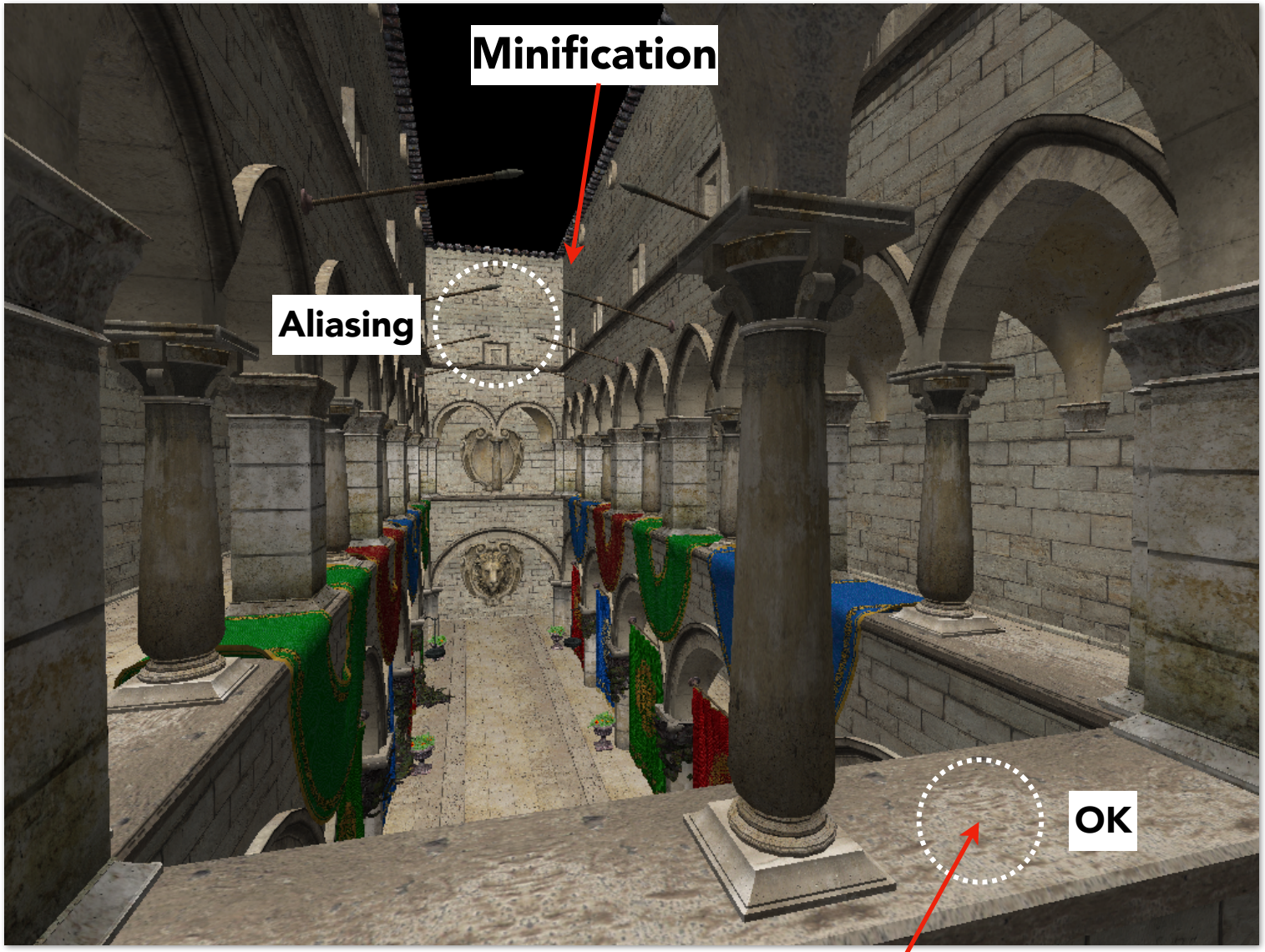
- Many texels can contribute to pixel footprint
- Shape of pixel footprint can be complex

Idea:

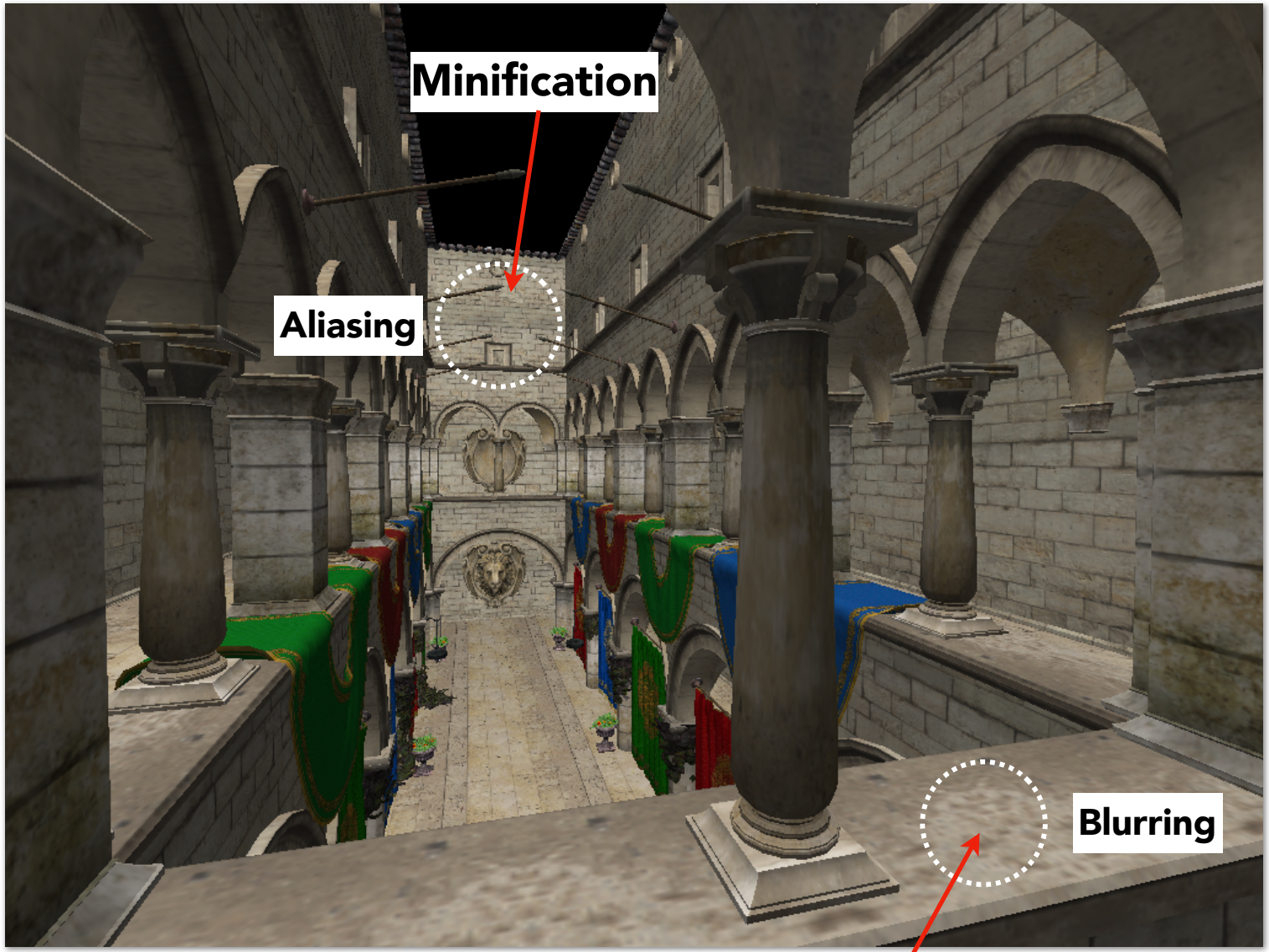
- Take texture image file, then low-pass filter it (i.e. filter out high frequencies) and downsample it (i.e. sample at a lower resolution) texture file. Do this recursively, and store successively lower resolution, each with successively lower maximum signal frequency.
- For each sample, use the texture file whose resolution approximates the screen sampling rate

Level 0 - Full Resolution Texture

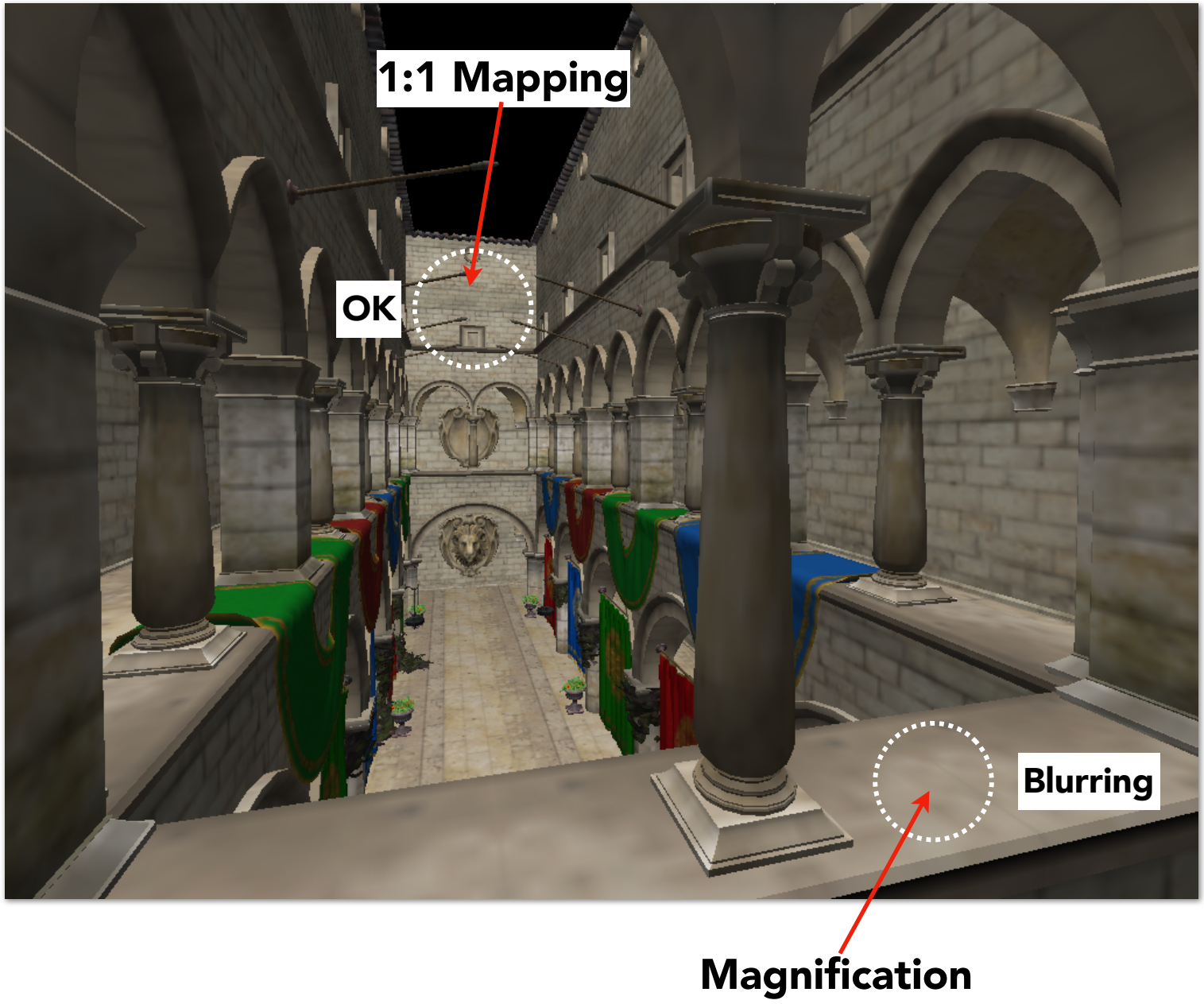




Level 2 - Downsample 4x4 (bilinear resampling)

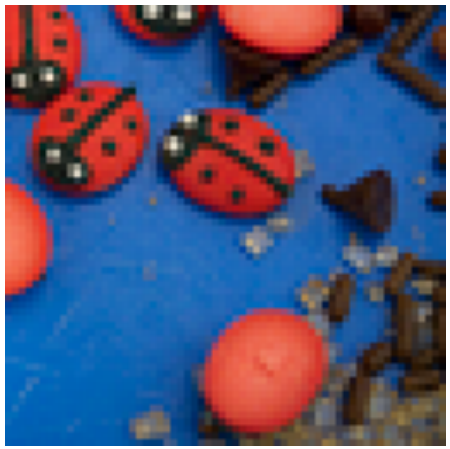


Level 4 - Downsample 16x16 (bilinear resampling)



Mipmap (L. Williams 83)

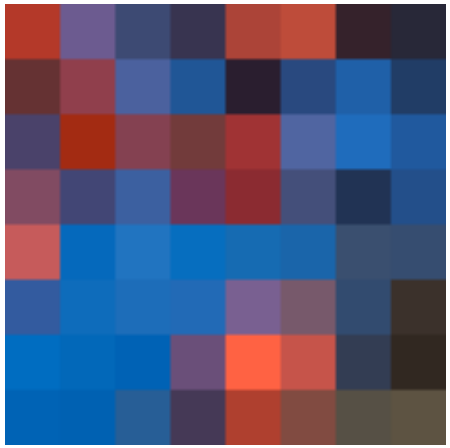
Level 0 = 128x128



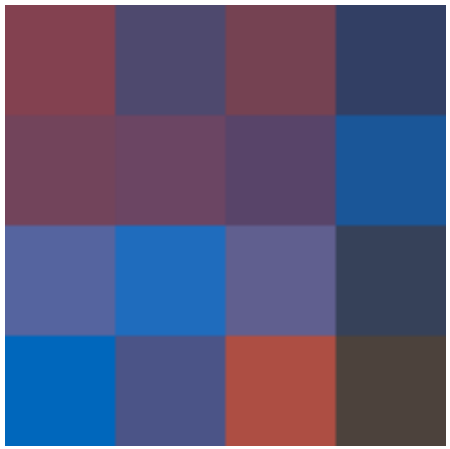
Level 1 = 64x64

Level 2 = 32x32

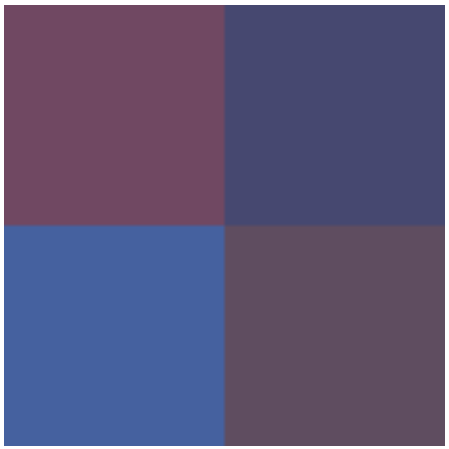
Level 3 = 16x16



Level 4 = 8x8



Level 5 = 4x4



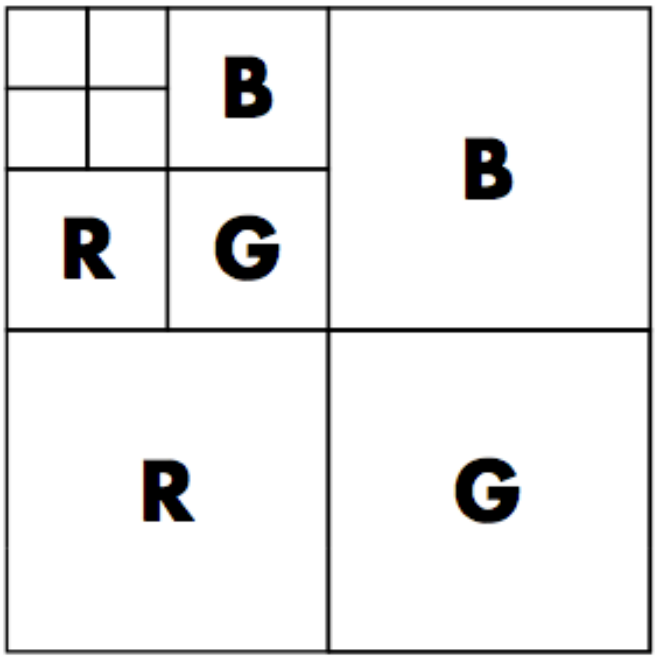
Level 6 = 2x2

Level 7 = 1x1

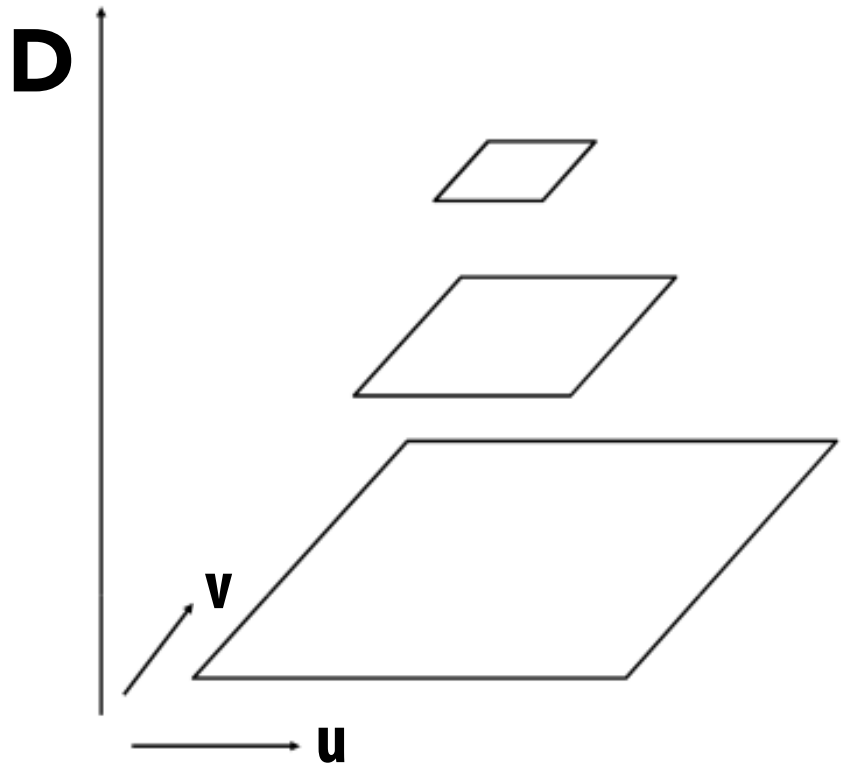
“Mip” comes from the Latin “multum in parvo", meaning a multitude in a small space

‘Much in little’，既“放置很多东西的小空间”

Mipmap (L. Williams 83)



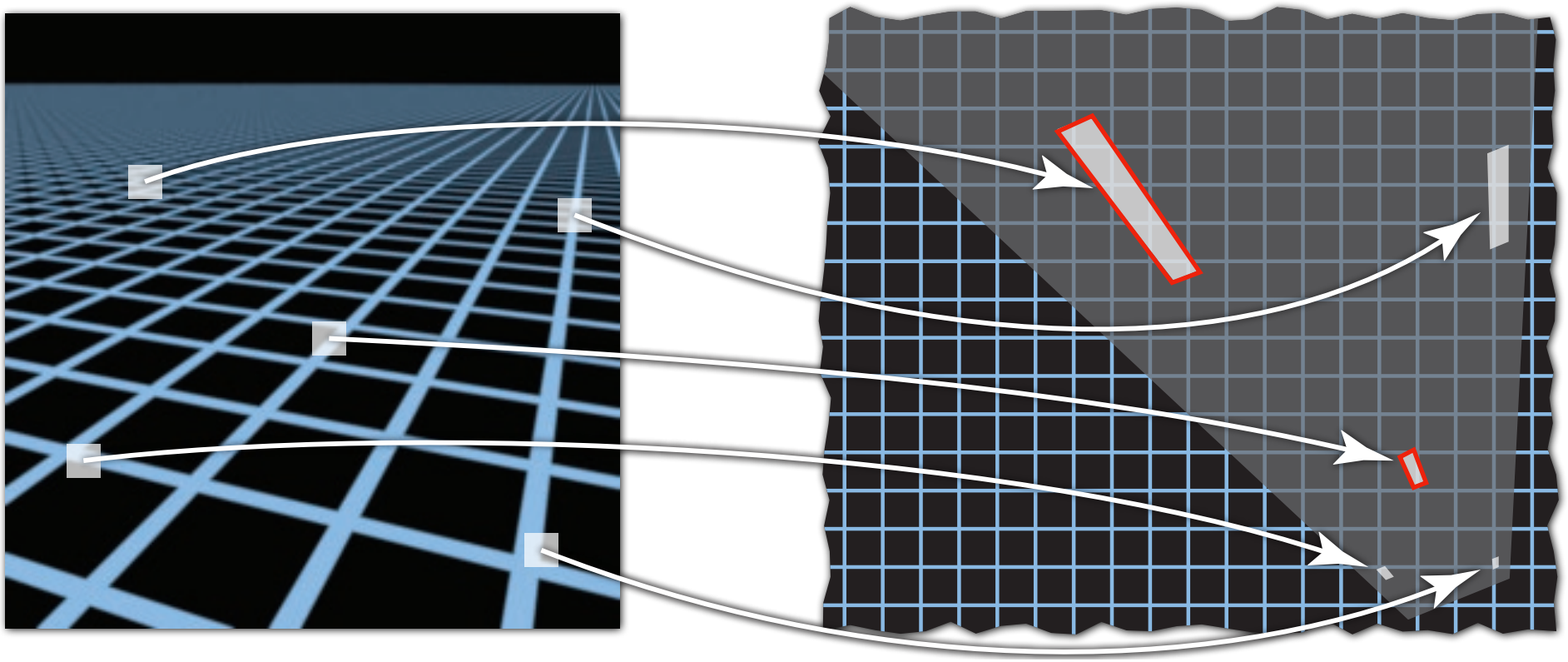
Williams' original
proposed mipmap layout



"Mip hierarchy"
level = D

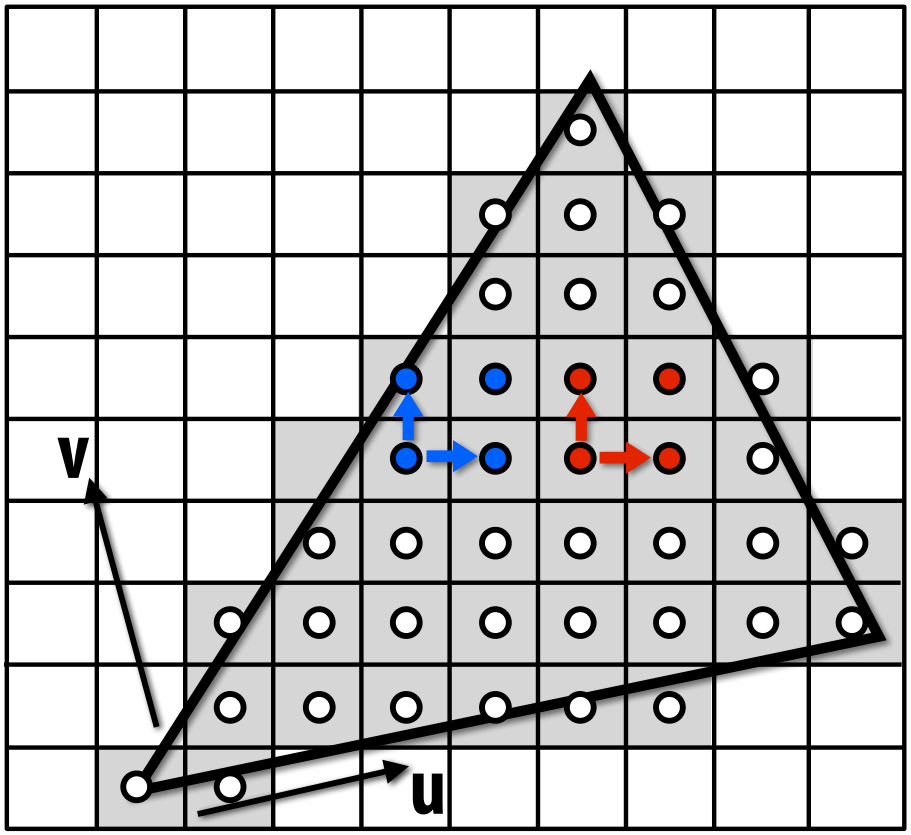
What is the storage overhead of a mipmap?

How to compute Mipmap Level?

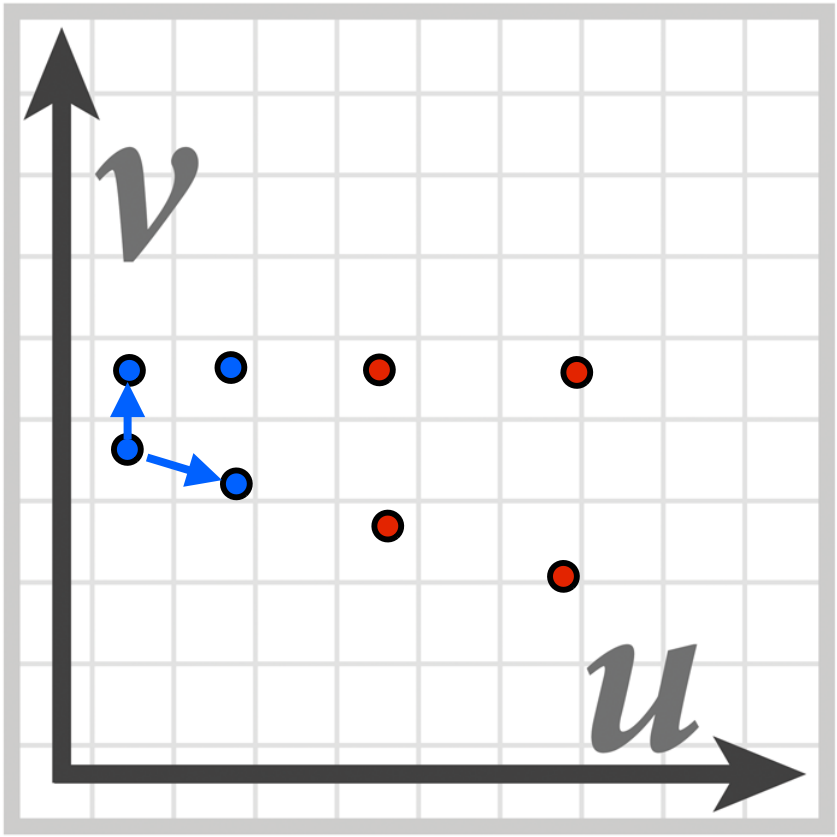


Squared pixels from screen space map to irregular regions in texture space.

Computing Mipmap Level D



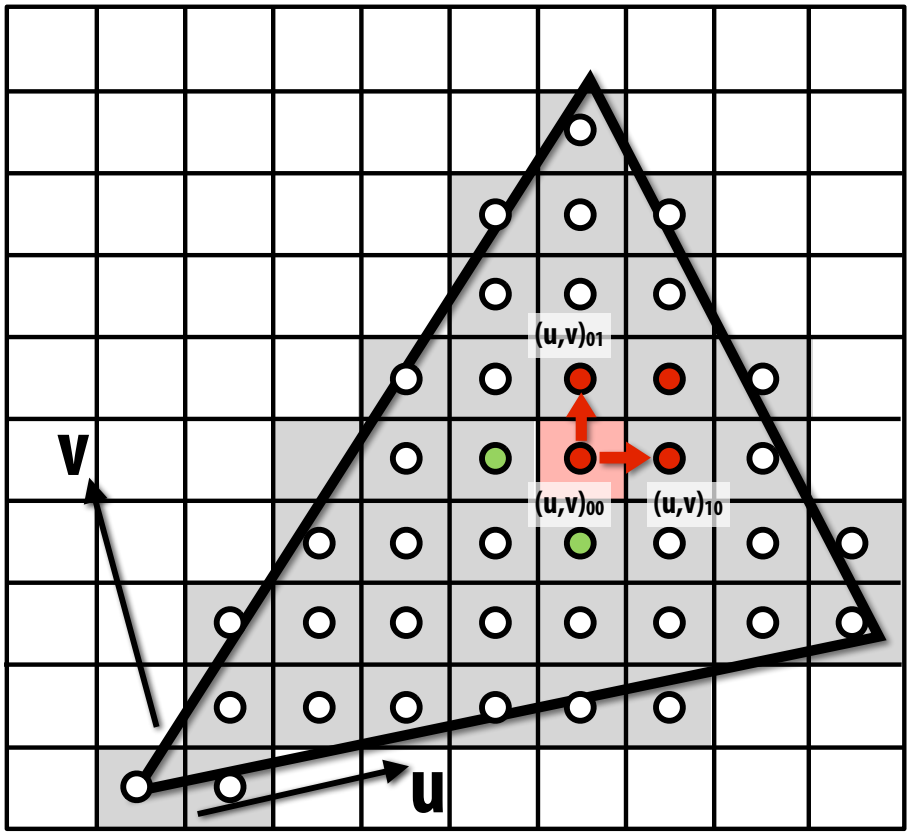
Screen space (x,y)

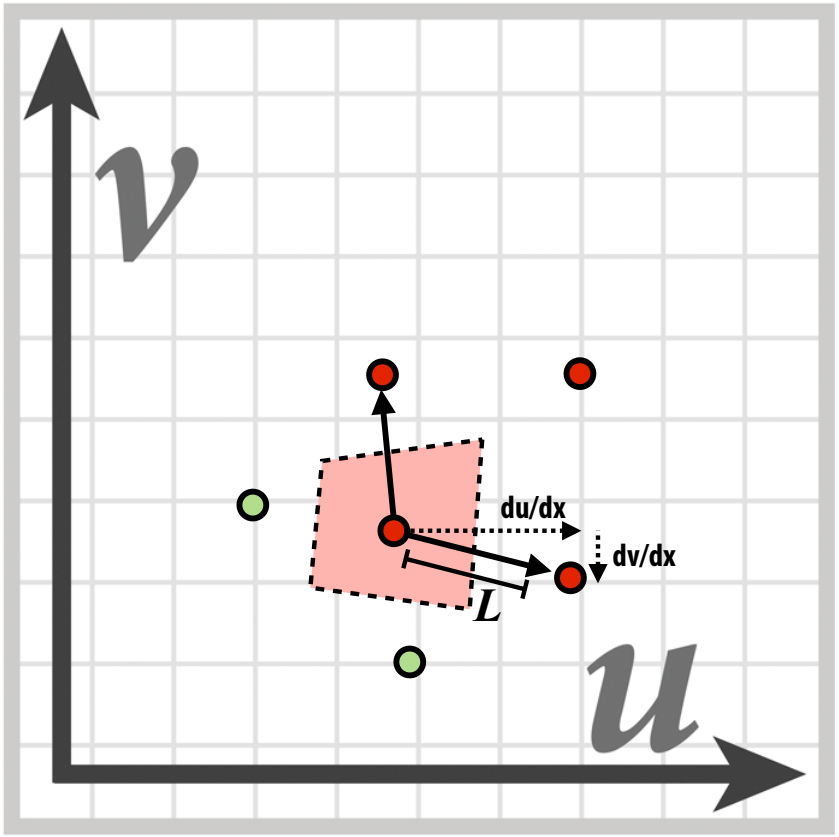


Texture space (u,v)

Estimate texture footprint using texture coordinates of neighboring screen samples

Computing Mipmap Level D



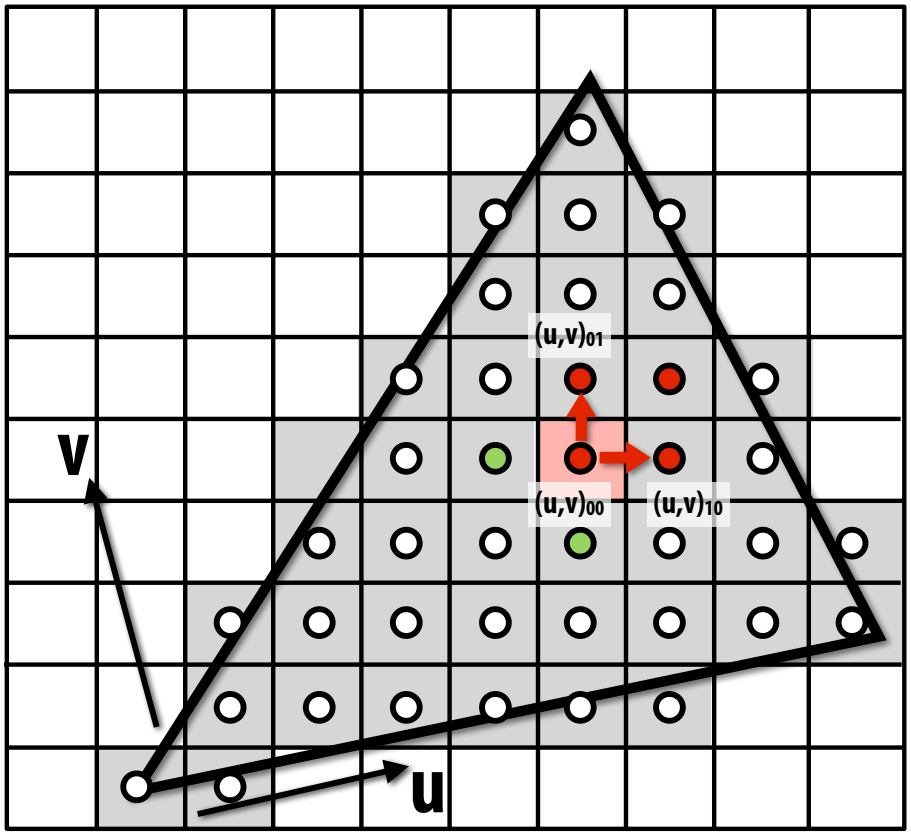


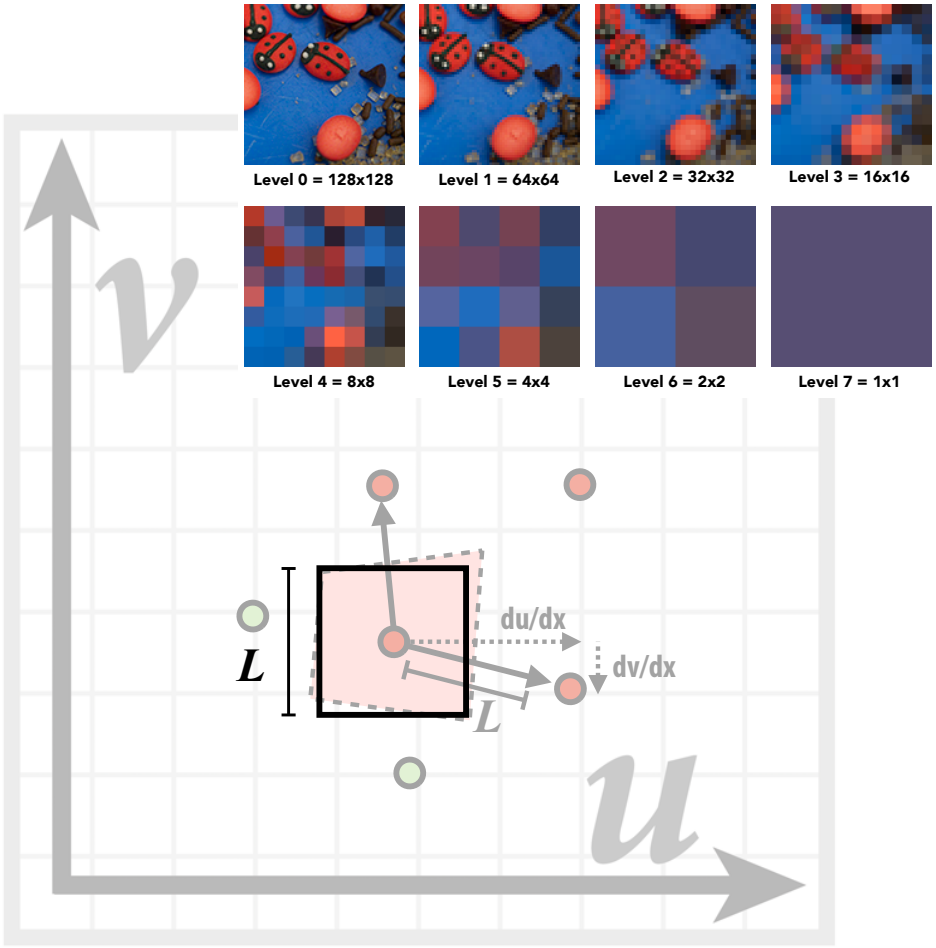
$$\frac{du}{dx} = u_{10} - u_{00} \qquad \frac{dv}{dx} = v_{10} - v_{00}$$

$$\frac{du}{dy} = u_{01} - u_{00} \qquad \frac{dv}{dy} = v_{01} - v_{00}$$

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

Computing Mipmap Level D

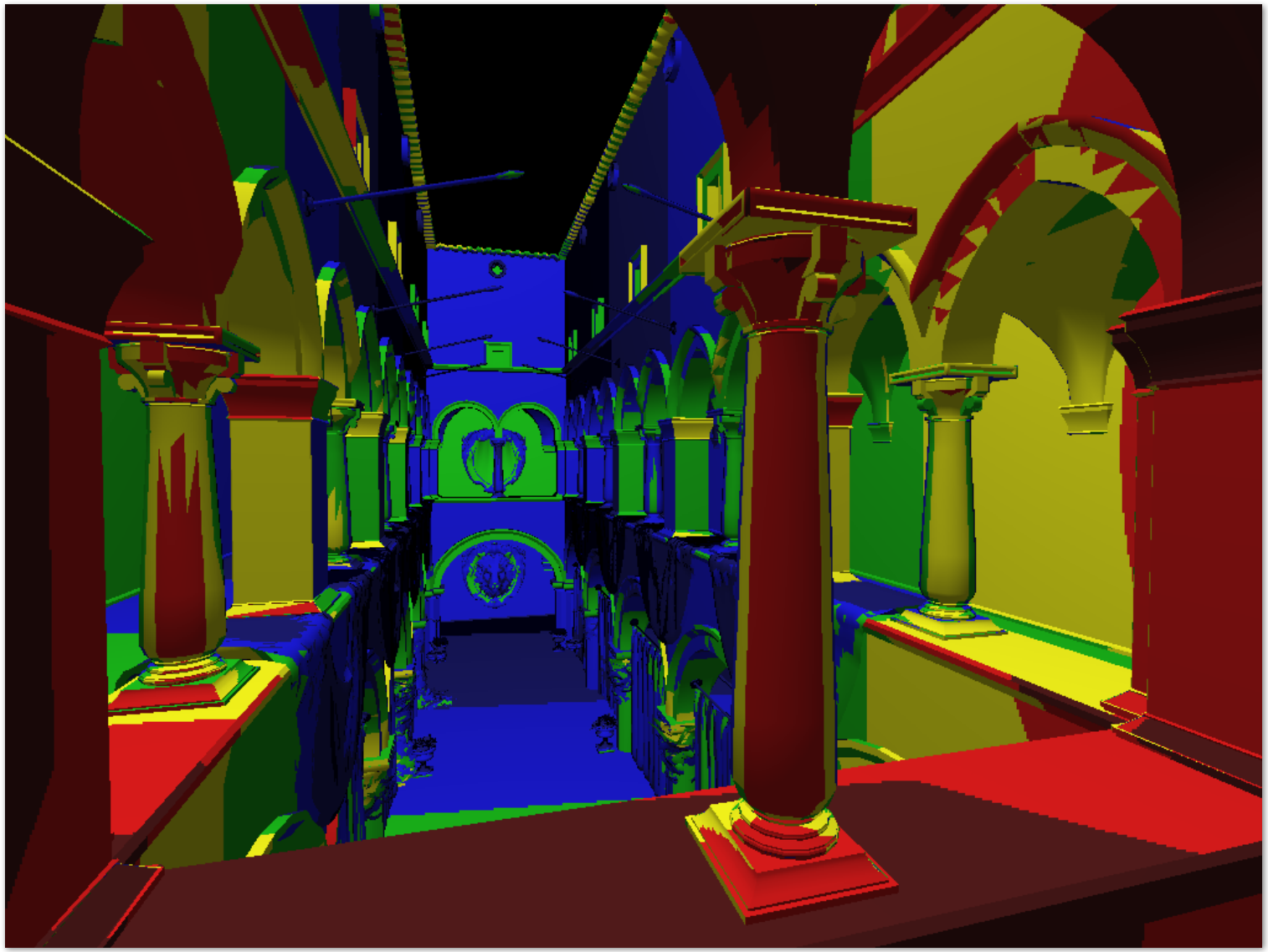




$$\text{Mip Map Level } D = \log_2 L$$

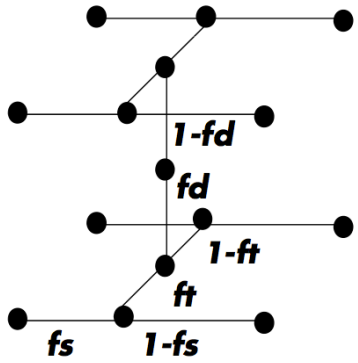
$$L = \max\left(\sqrt{\left(\frac{du}{dx}\right)^2 + \left(\frac{dv}{dx}\right)^2}, \sqrt{\left(\frac{du}{dy}\right)^2 + \left(\frac{dv}{dy}\right)^2}\right)$$

Visualization of Mipmap Level

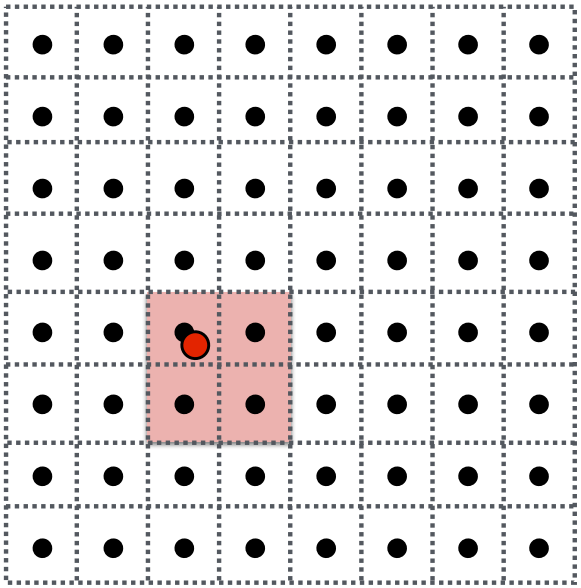


D rounded to nearest integer level

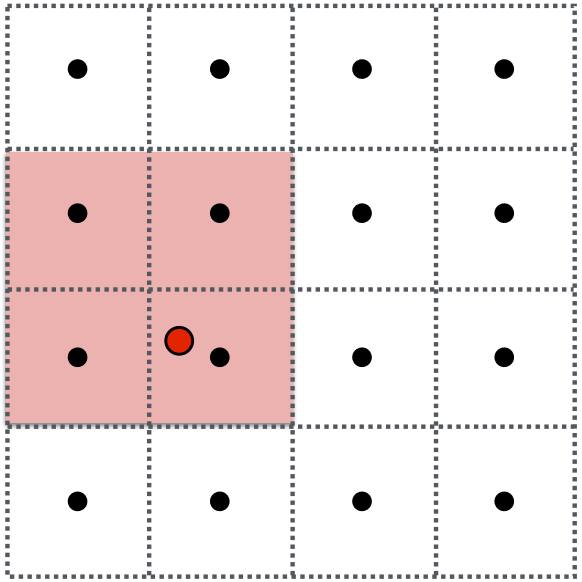
Trilinear Filtering



$$rp(t, v_1, v_2) = v_1 + t(v_2 - v_1)$$



Mipmap Level D



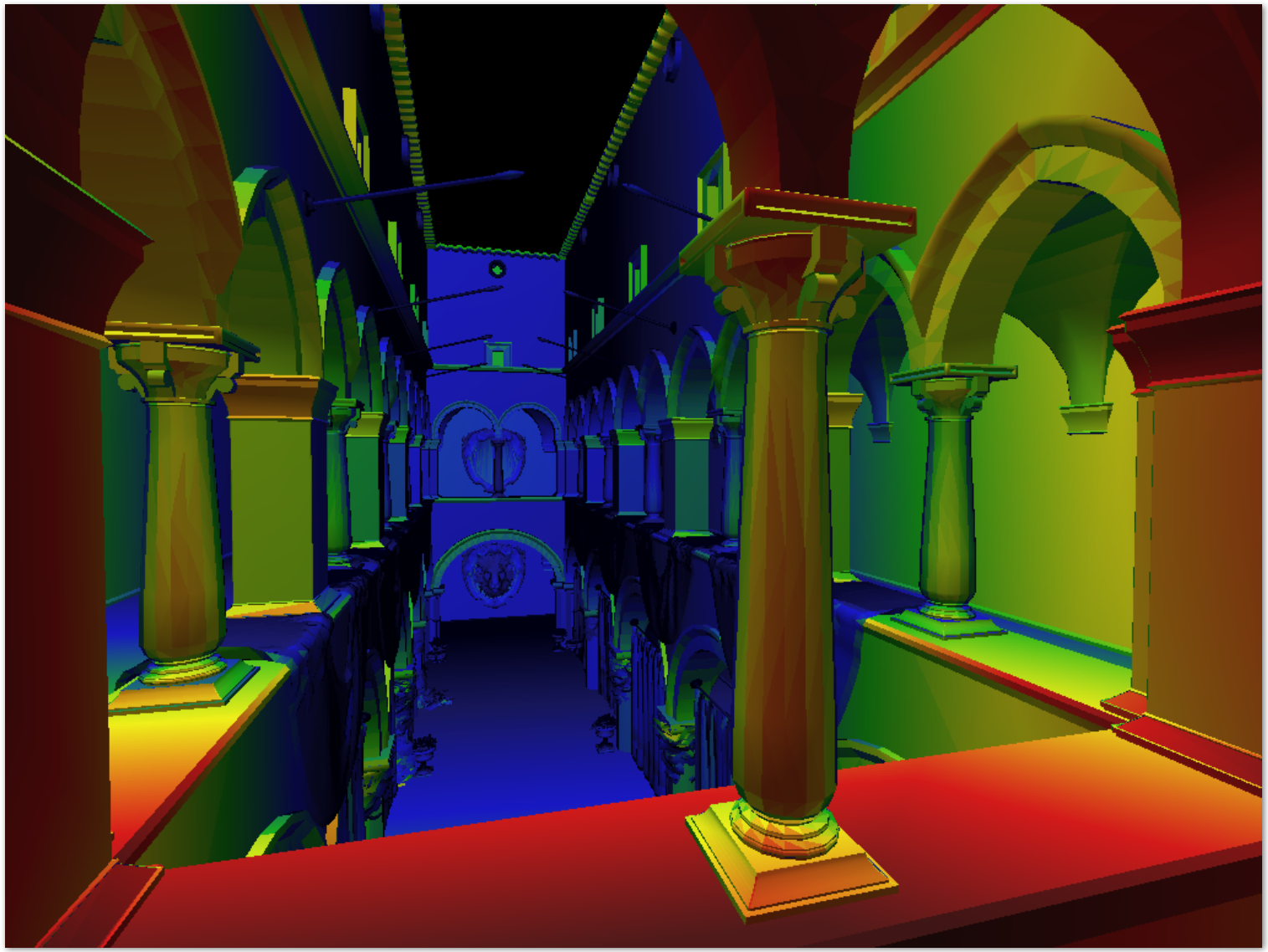
Mipmap Level D+1

Bilinear result

Bilinear result

Linear interpolation based on continuous D value

Visualization of Mipmap Level



Trilinear filtering: visualization of continuous D

Texture Filtering

Image resampling choices

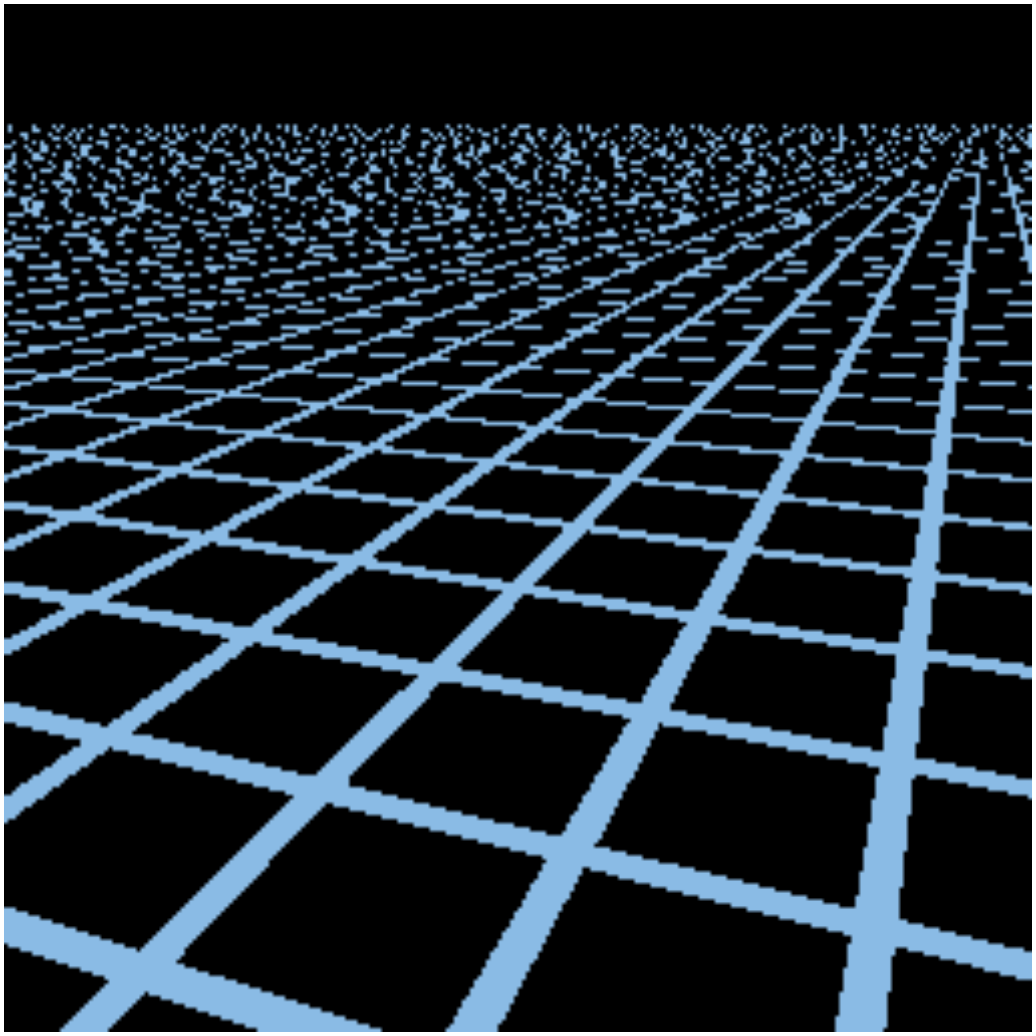
- Nearest
- Bilinear interpolation

Mipmap level resampling choices

- Always level 0
- Nearest D
- Linear interpolation

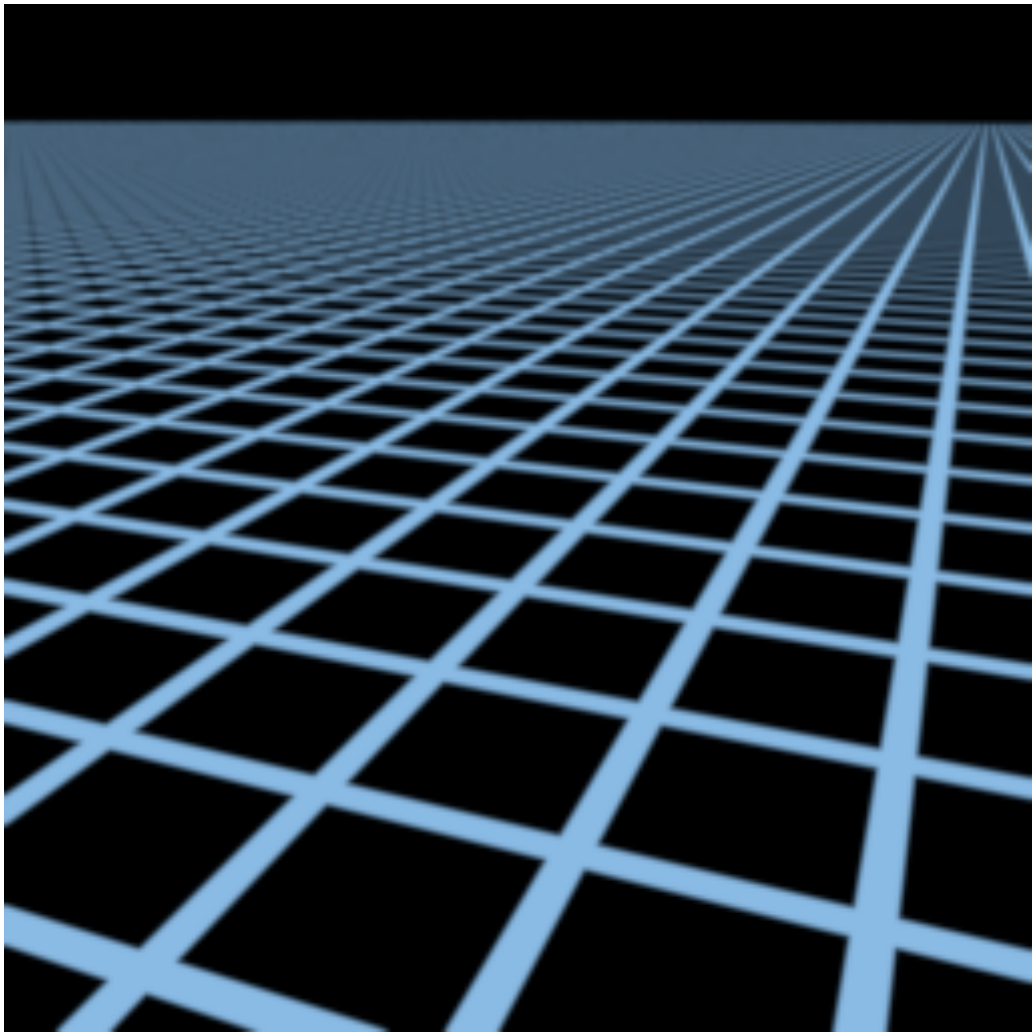
2 x 3 = 6 choices

Mipmap Limitations



Point sampling

Mipmap Limitations

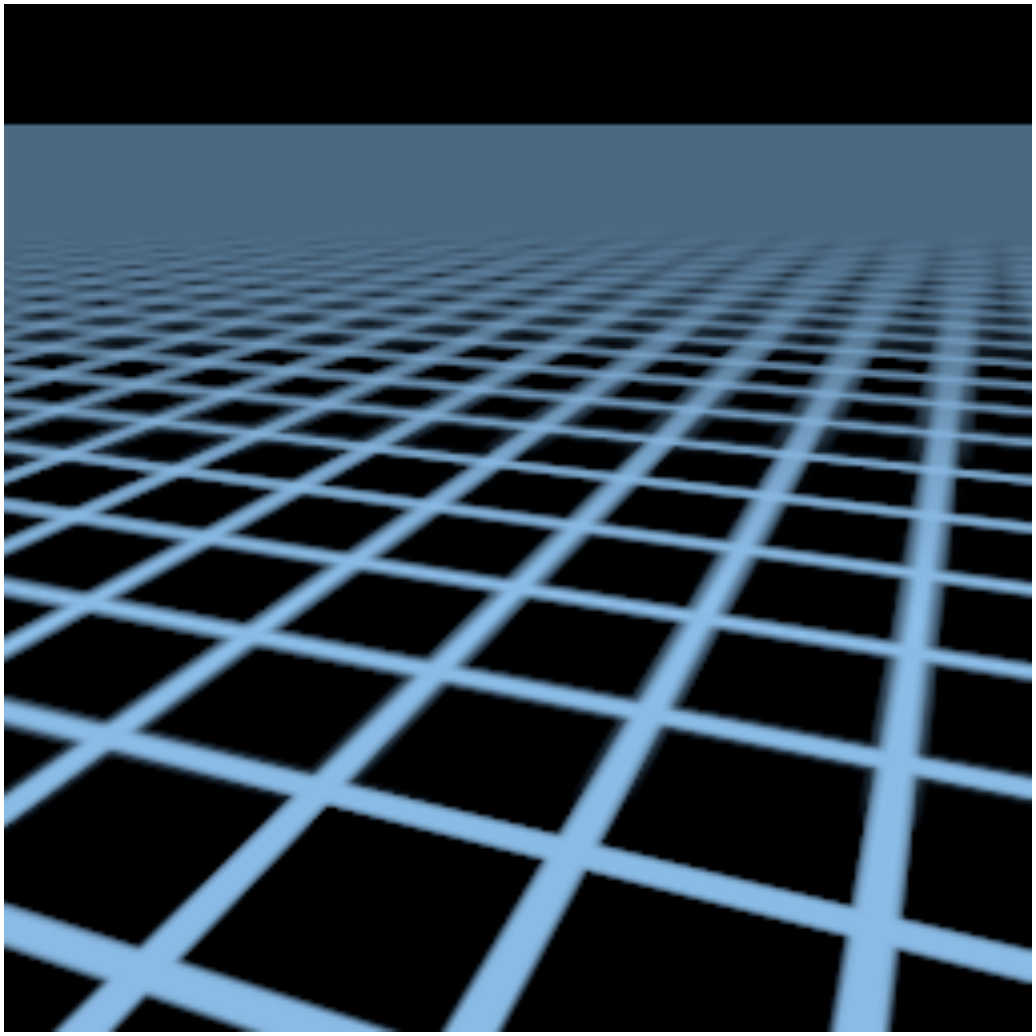


Supersampling 512x

Mipmap Limitations

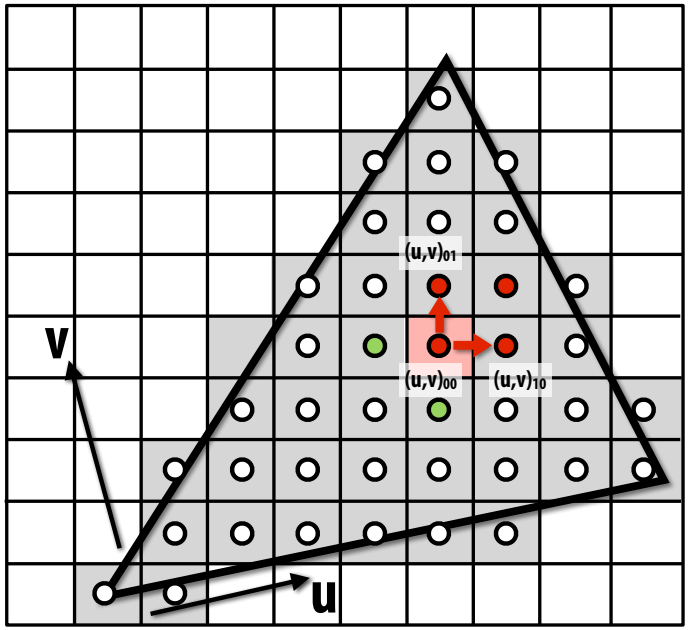
Overblur

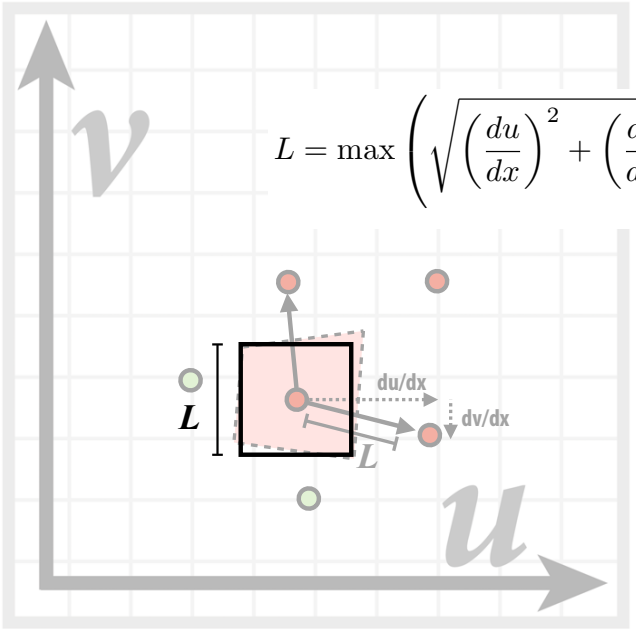
Why?



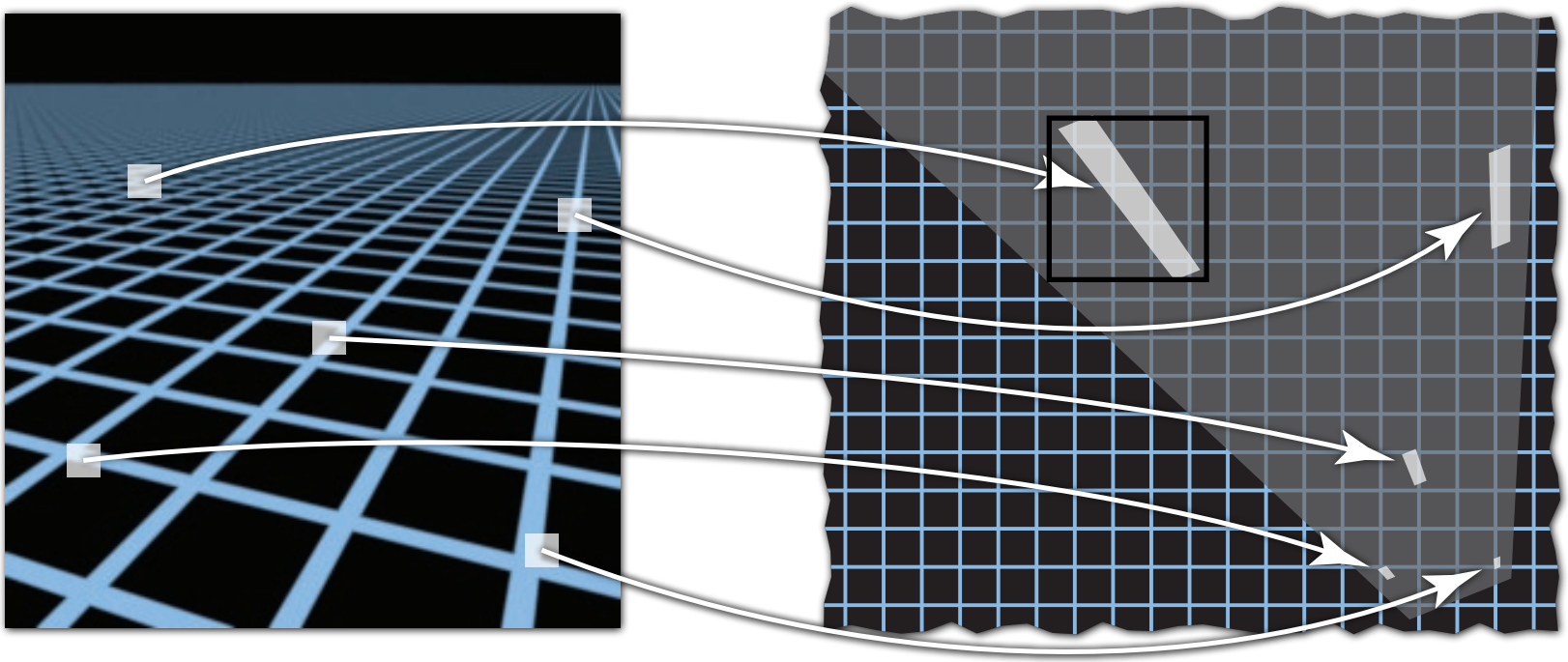
Mipmap trilinear sampling

Mipmap Assumes Regular Footprint in Texture



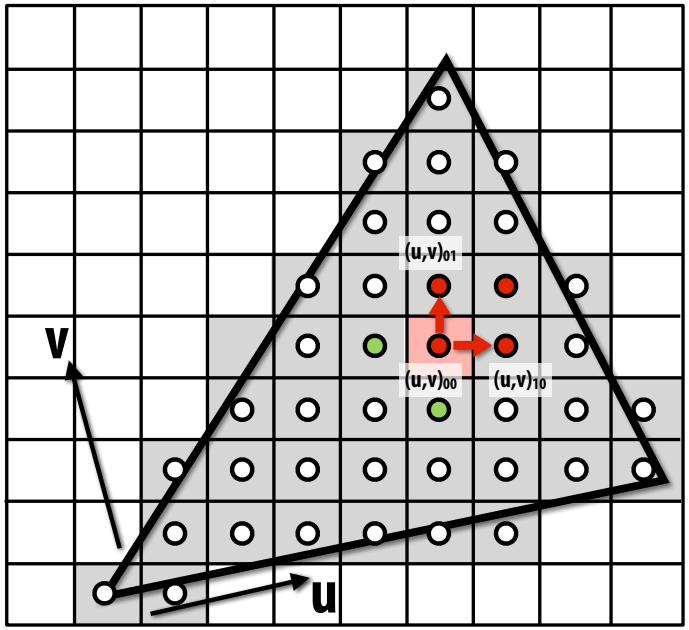


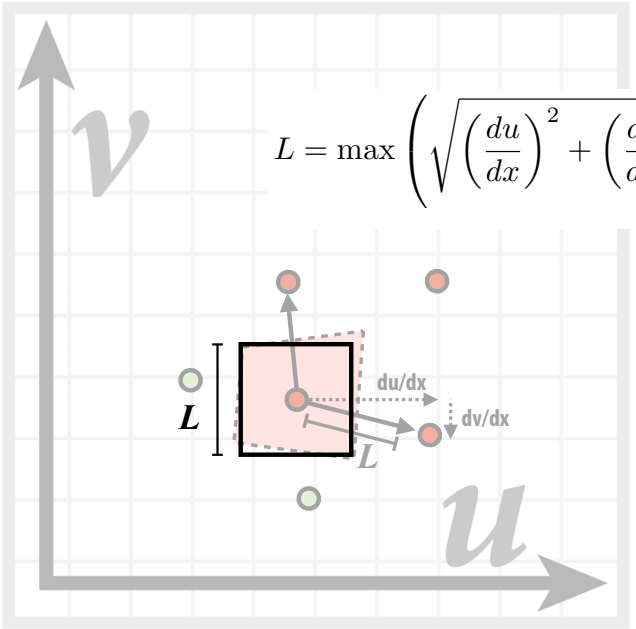
$$L = \max\left(\sqrt{\left(\frac{du}{dx}\right)^2 + \left(\frac{dv}{dx}\right)^2}, \sqrt{\left(\frac{du}{dy}\right)^2 + \left(\frac{dv}{dy}\right)^2}\right)$$



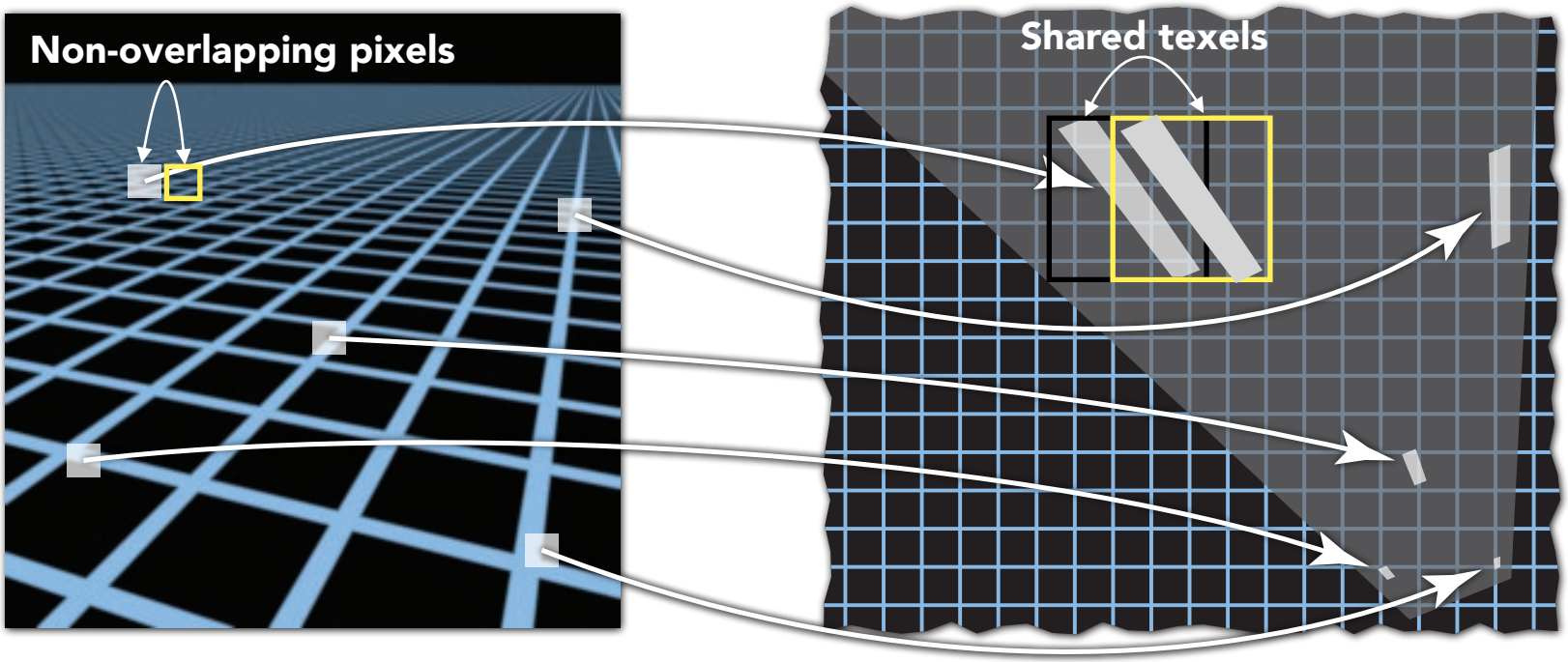
Irregular pixel footprint!

Mipmap Assumes Regular Footprint in Texture



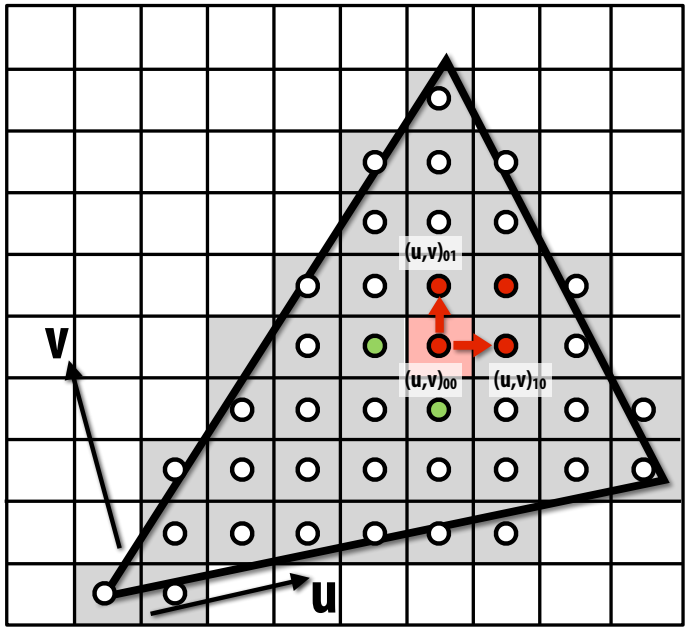


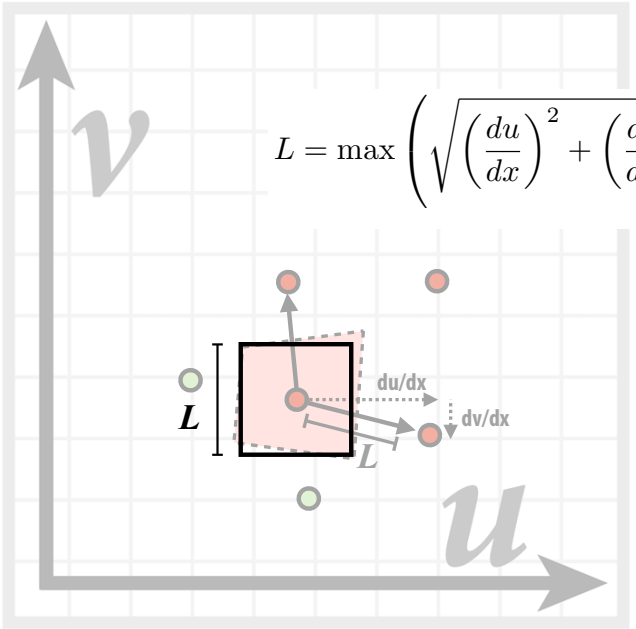
$$L = \max\left(\sqrt{\left(\frac{du}{dx}\right)^2 + \left(\frac{dv}{dx}\right)^2}, \sqrt{\left(\frac{du}{dy}\right)^2 + \left(\frac{dv}{dy}\right)^2}\right)$$



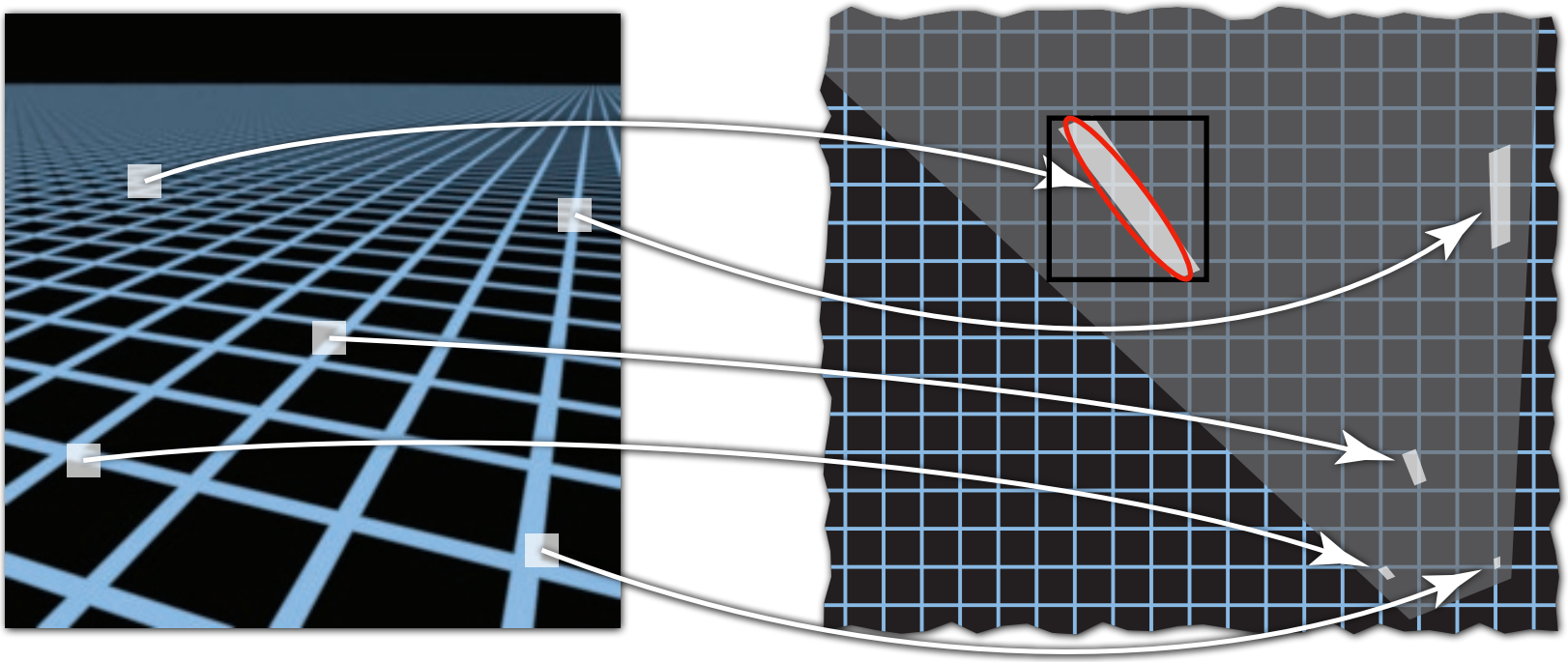
Irregular pixel footprint!

Mipmap Assumes Regular Footprint in Texture





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



Irregular pixel footprint!

Anisotropic Filtering

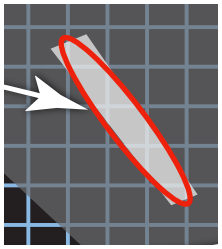
各向异性滤波：相对于Mipmap各向同性（isotropic）而言的改进

Ripmaps and summed area tables

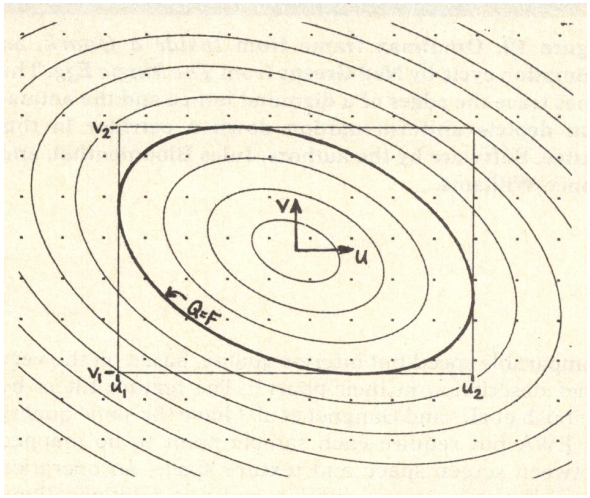
- Can look up axis-aligned rectangular zones
- Diagonal footprints still a problem

Wikipedia

EWA filtering



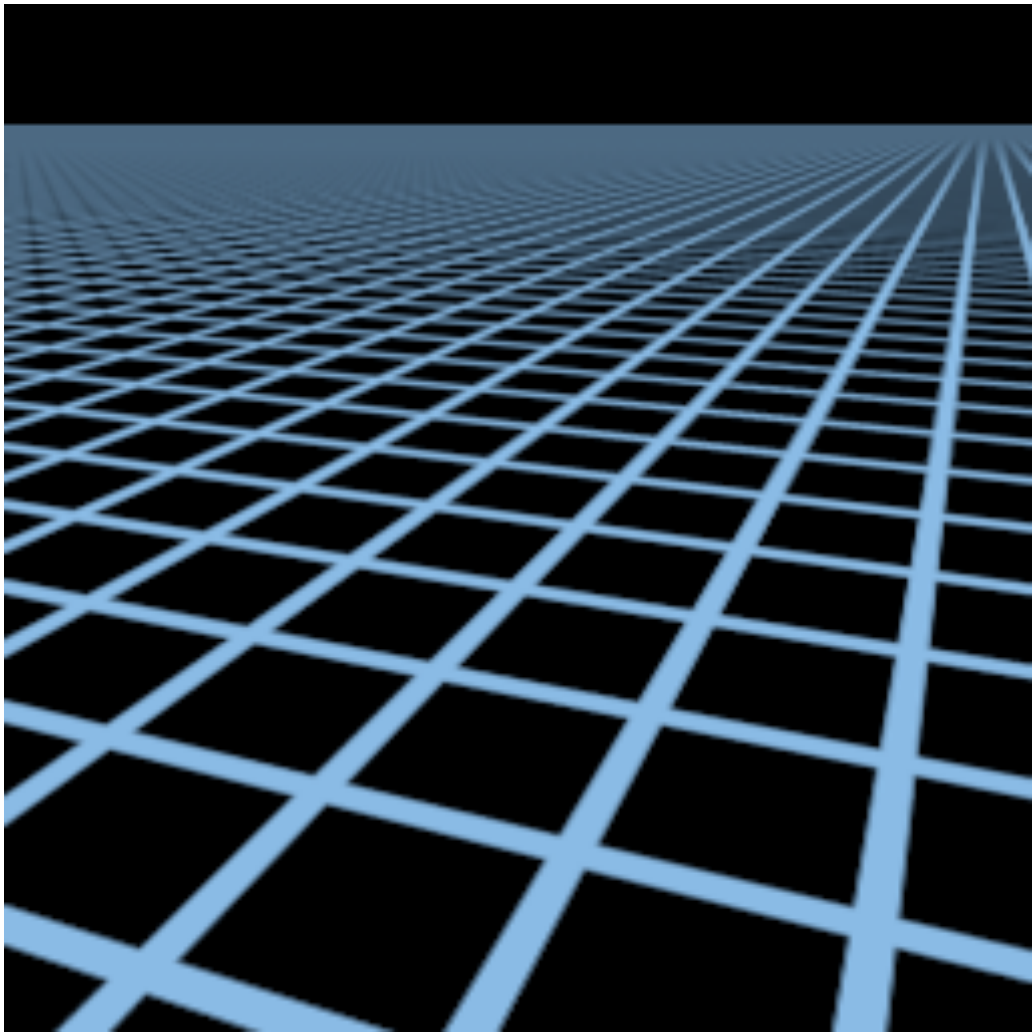
- Use multiple lookups
- Weighted average
- Mipmap hierarchy still helps



Greene & Heckbert ‘86

Elliptical Weighted Average

Anisotropic Filtering



Elliptical weighted average (EWA) filtering

算法实现原理介绍可参考：https://zhuanlan.zhihu.com/p/105167411

Things to Remember

Many uses of texturing

- Bring high-resolution data to fragment calculations
- Colors, normals, lighting on sphere, volumetric data, …

How does texturing work?

- Texture coordinate parameterization
- Barycentric interpolation of coordinates
- Texture sampling pattern and frequency
- Mipmaps: texture filtering hierarchy, level calculation, trilinear interpolation
- Anisotropic sampling

Course Roadmap

Rasterization Pipeline

Core Concepts

- Sampling
- Antialiasing
- Transforms

Intro

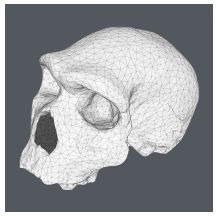
Rasterization

Transforms & Projection

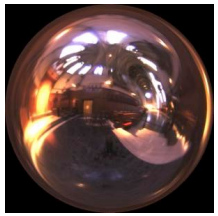
Visibility, Shading

Texture Mapping

Geometric Modeling

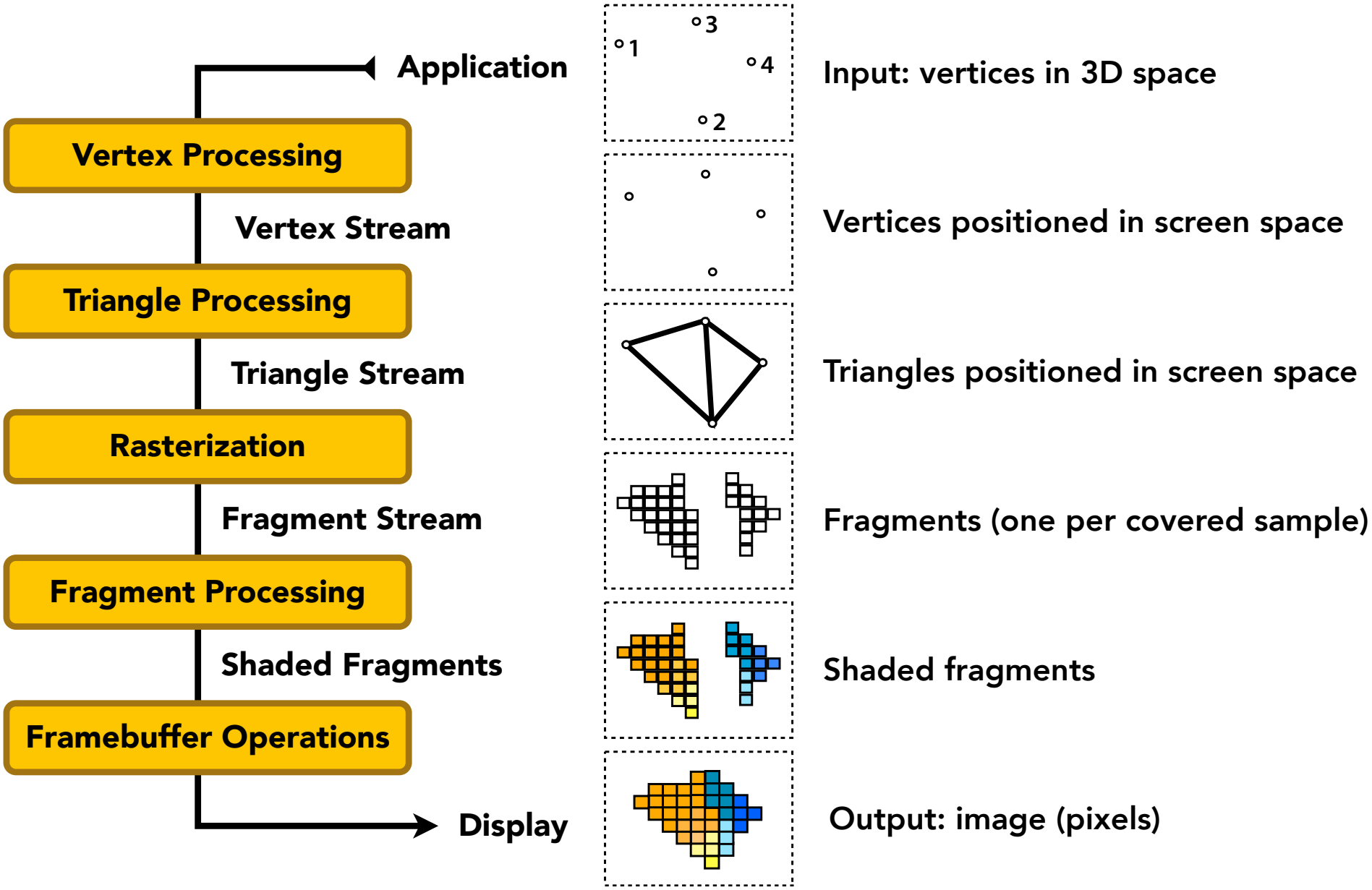


Lighting & Materials

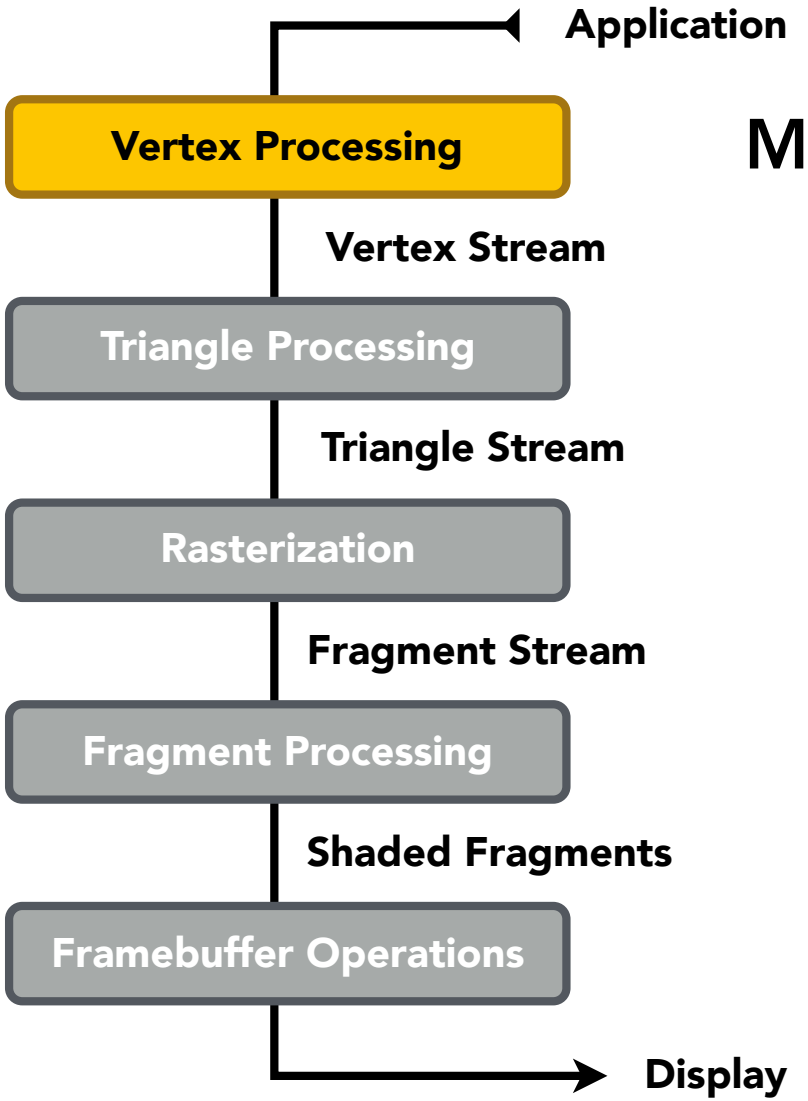


Cameras & Imaging

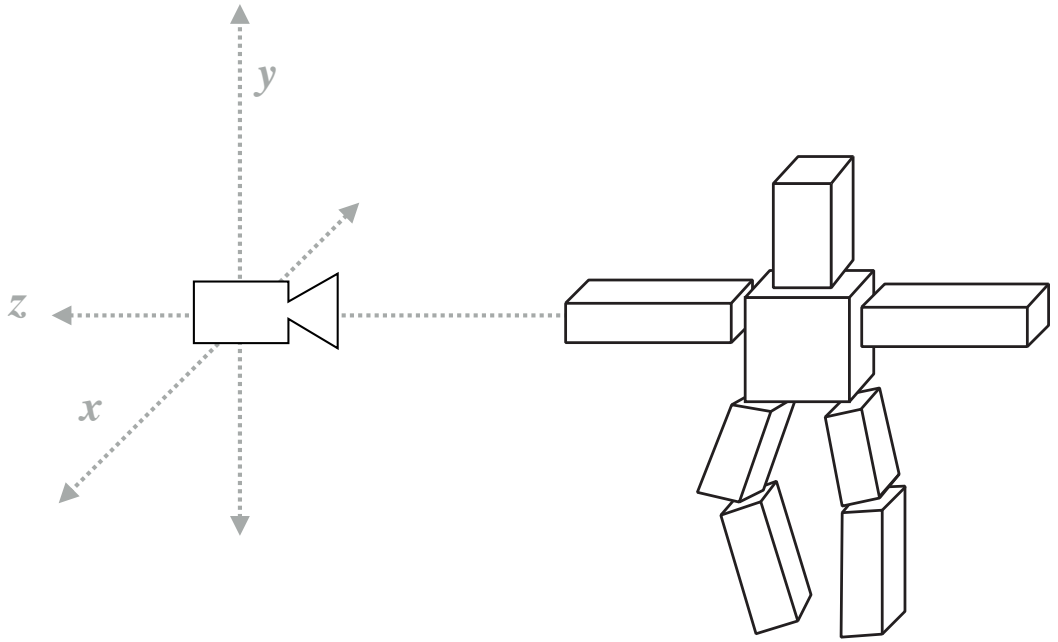
Rasterization Pipeline



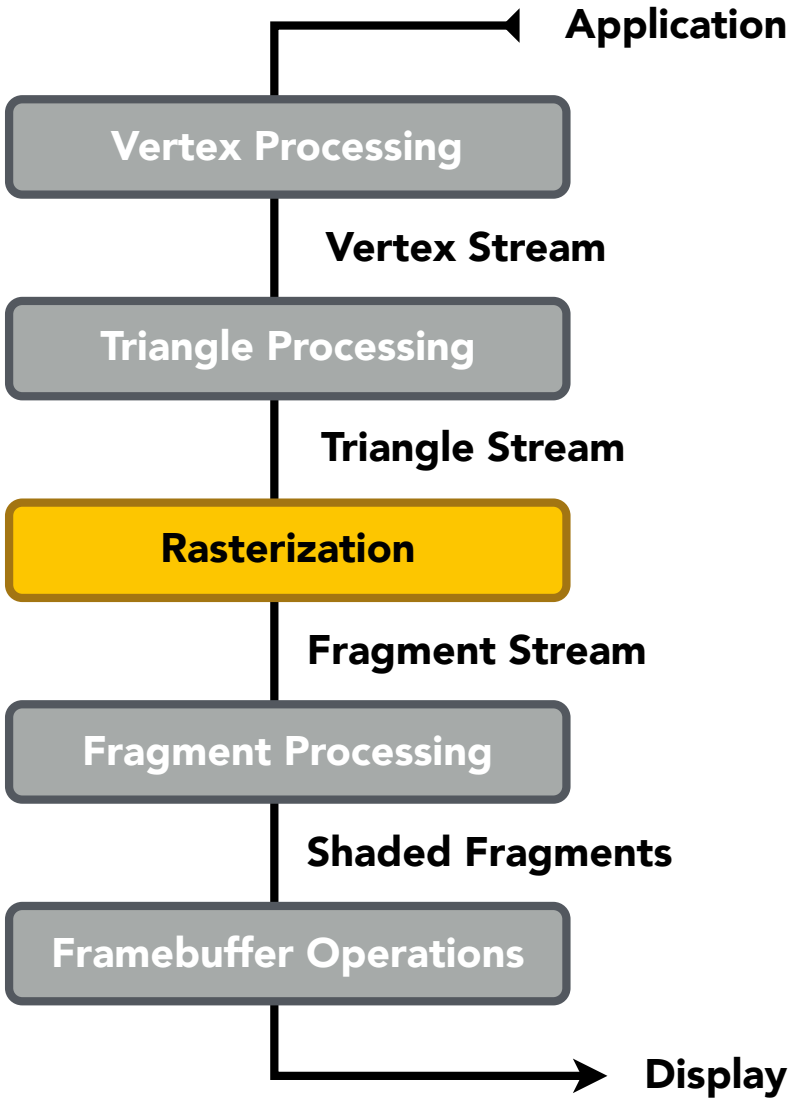
Rasterization Pipeline



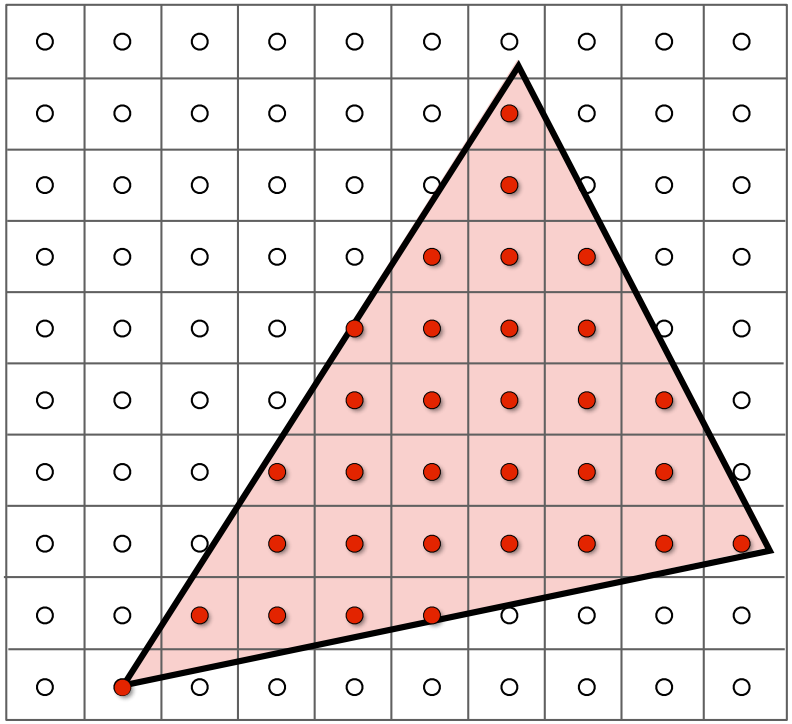
Model, View, Projection transforms



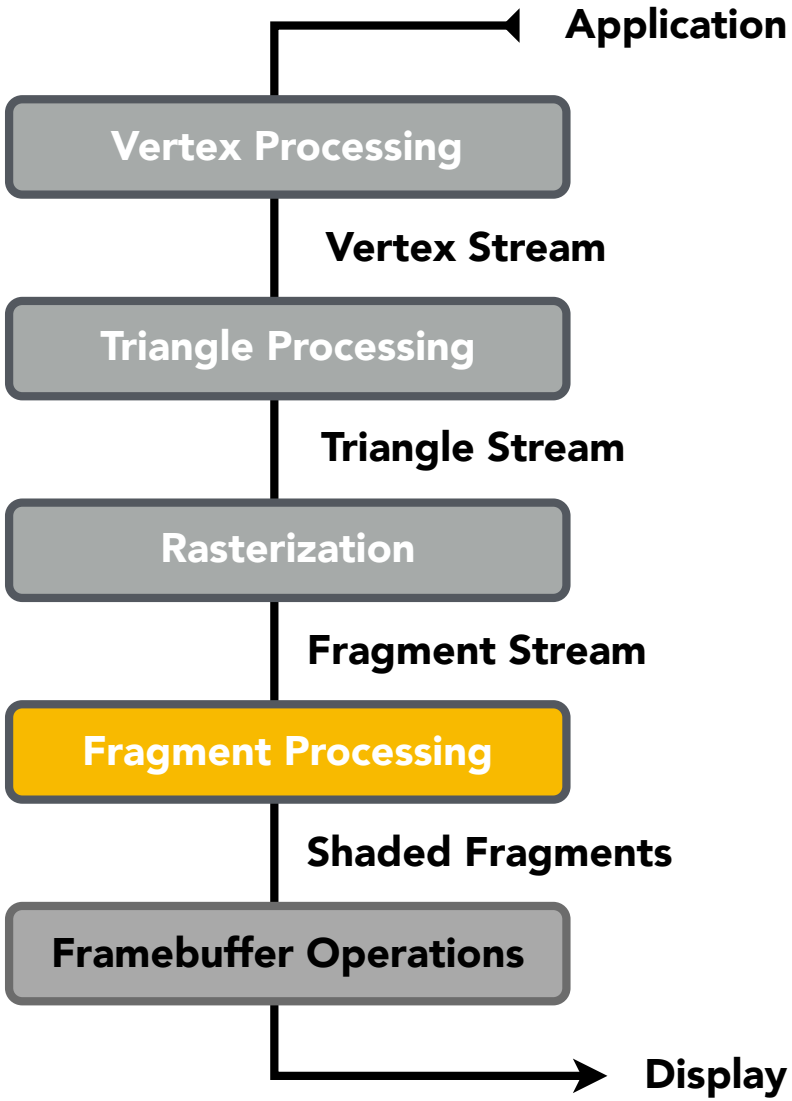
Rasterization Pipeline



Sampling triangle coverage

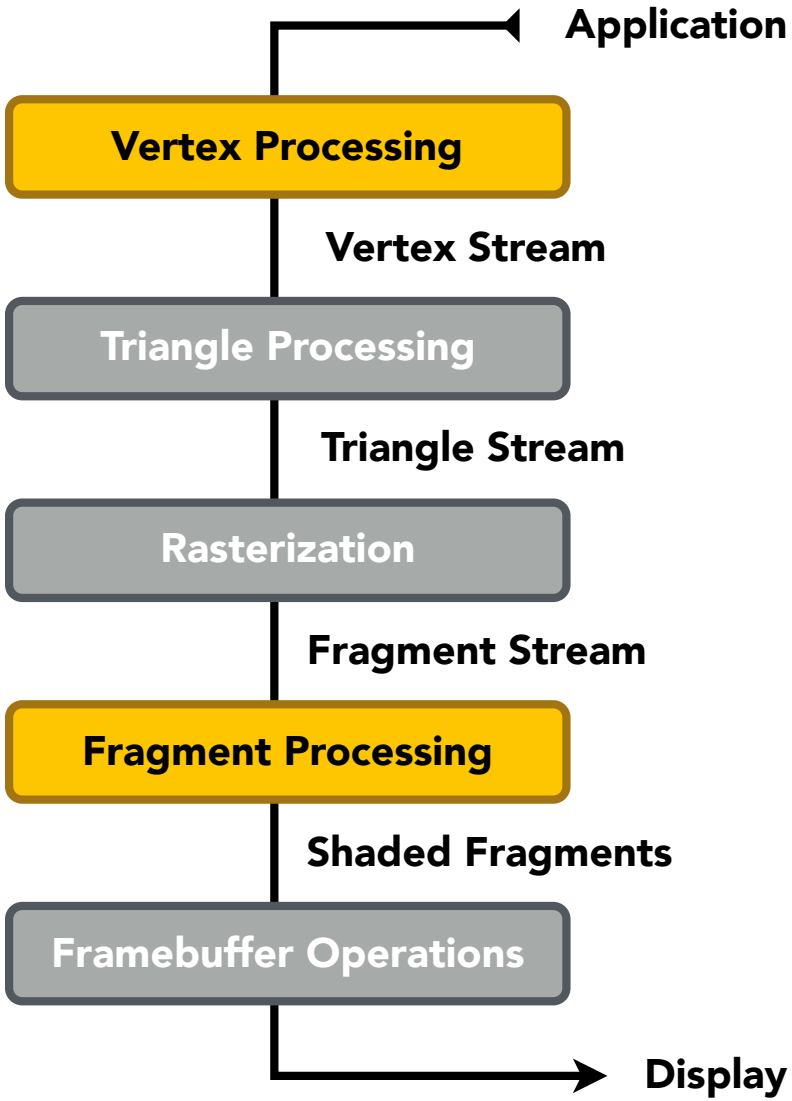


Rasterization Pipeline

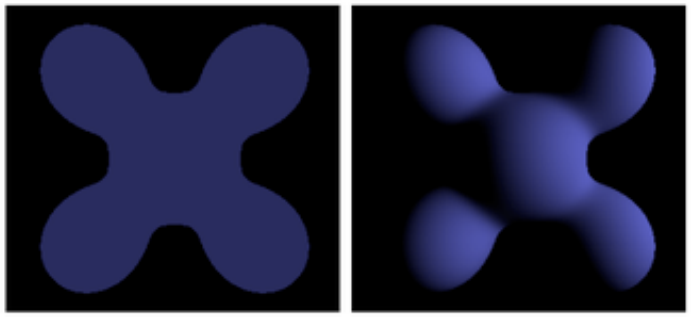


Z-Buffer Visibility Tests

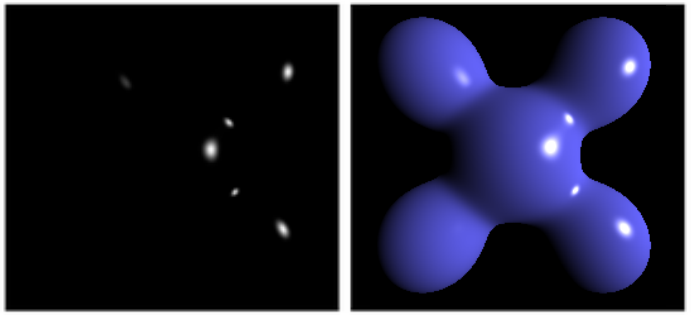
Rasterization Pipeline



Shading

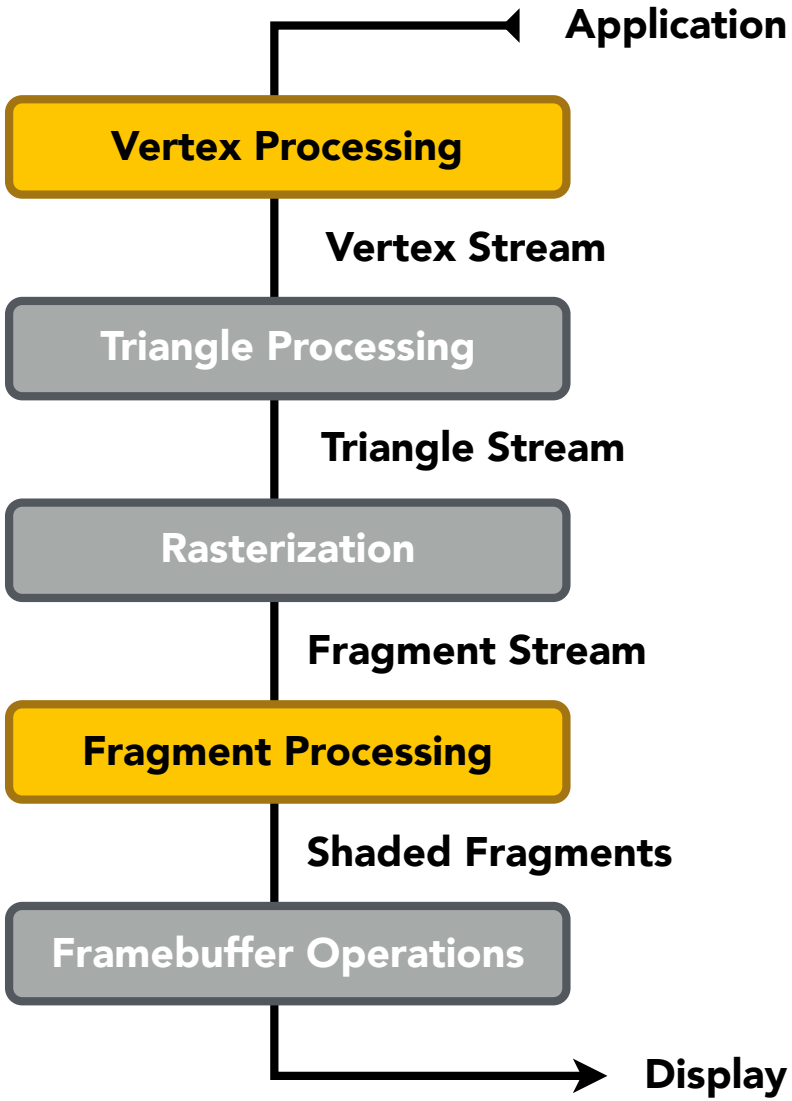


Ambient + Diffuse

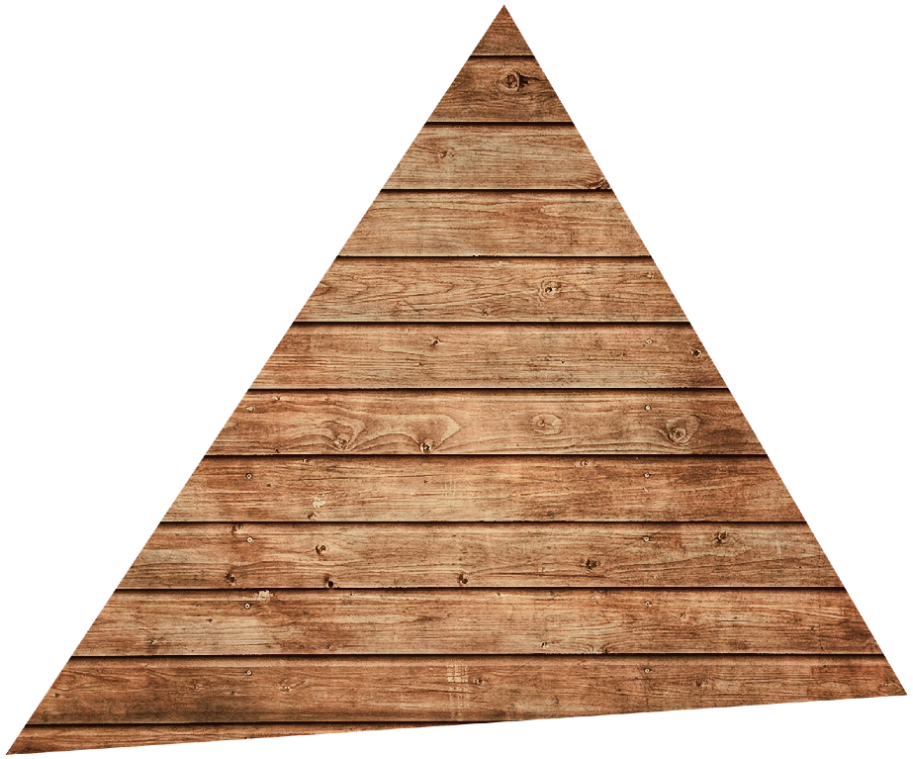


+ Specular = Blinn-Phong Reflectance Model

Rasterization Pipeline



Texture Mapping



Shader Programs

- Shader programs are the programmable parts of the rendering pipeline
- Runs on the GPU (Graphics Processing Unit).
- Written in a shading language (like GLSL, HLSL, or Metal Shading Language)
- Responsible for controlling how vertices and pixels (fragments) are processed during rendering.
    - Vertex Shader: Runs once per vertex. Transforms vertex positions (e.g., from model space → world space → screen space). Passes data (like color, normals, texture coordinates) to the next stage.
    - Fragment Shader (Pixel Shader): Runs once per fragment (potential pixel). Determines the final color, texture, lighting, transparency, etc.

Shader Programs

- Program vertex and fragment processing stages
- Describe operation on a single vertex (or fragment)

Example GLSL fragment shader program

```
uniform sampler2D myTexture;
uniform vec3 lightDir;
varying vec2 uv;
varying vec3 norm;

void diffuseShader()
{
  vec3 kd;
  kd = texture2d(myTexture, uv);
  kd *= clamp(dot(–lightDir, norm), 0.0, 1.0);
  gl_FragColor = vec4(kd, 1.0);
}
```

- Shader function executes once per fragment.
- Outputs color of surface at the current fragment's screen sample position.
- This shader performs a texture lookup to obtain the surface's material color at this point, then performs a diffuse lighting calculation.

Shader Programs

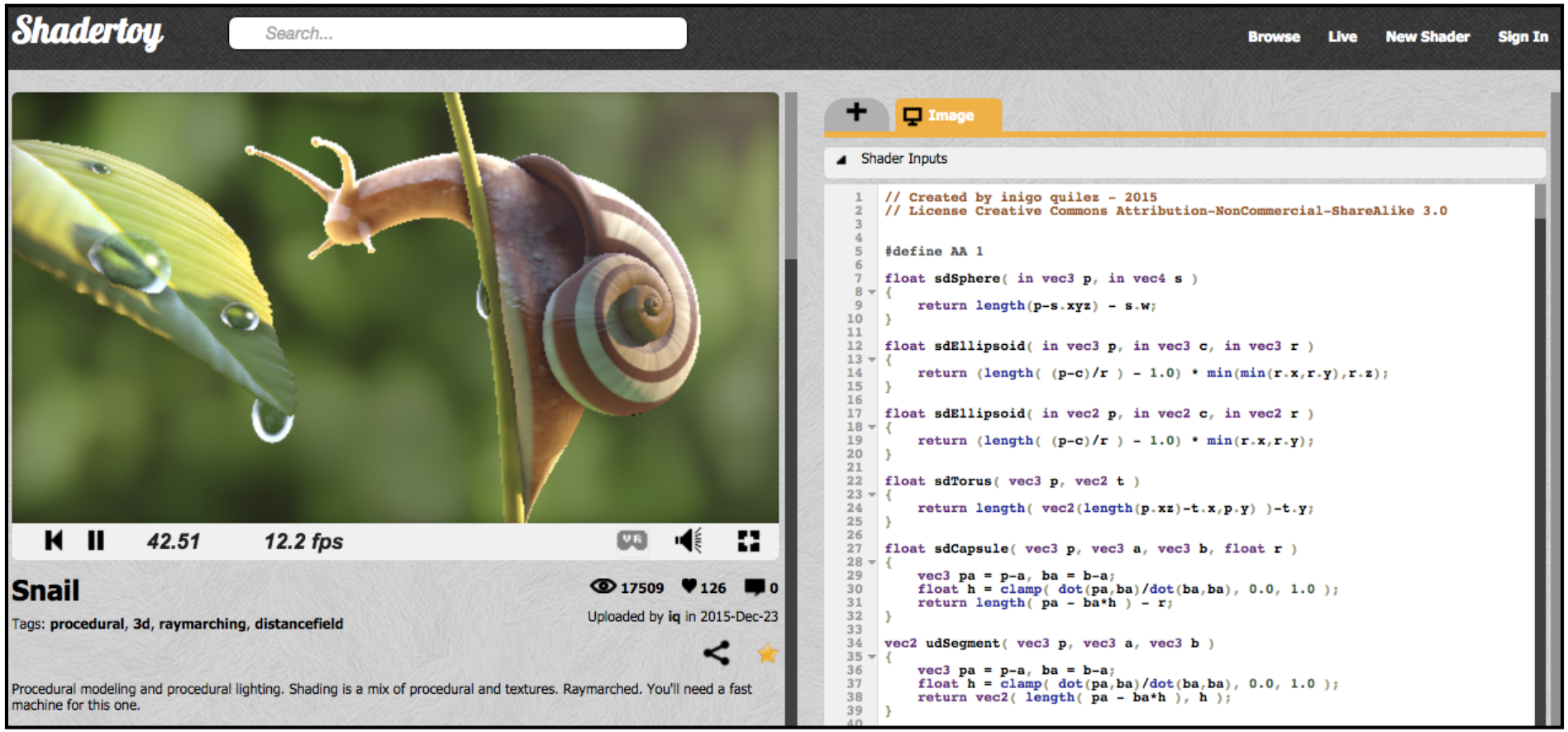
- Program vertex and fragment processing stages
- Describe operation on a single vertex (or fragment)

Example GLSL fragment shader program

```
uniform sampler2D myTexture;      // program parameter
uniform vec3 lightDir;            // program parameter
varying vec2 uv;                  // per fragment value (interp. by rasterizer)
varying vec3 norm;                // per fragment value (interp. by rasterizer)

void diffuseShader()
{
  vec3 kd;
  kd = texture2d(myTexture, uv);                  // material color from texture
  kd *= clamp(dot(–lightDir, norm), 0.0, 1.0);    // Lambertian shading model
  gl_FragColor = vec4(kd, 1.0);                   // output fragment color
}
```

Snail Shader Programs



Inigo Quilez

Procedurally modeled, 800 line shader.

http://shadertoy.com/view/ld3Gz2

Goal: Highly Complex 3D Scenes in Realtime

Unreal Engine Kite Demo (Epic Games 2015)

Homework 1

- ## 实现一个光栅化器

  本次作业包括6个部分，覆盖了第 2-6 讲的内容。

  - 1. 绘制三角形
  - 2. 基于超采样的反走样算法
  - 3. 几何变换
  - 4. 质心坐标
  - 5. 基于像素采样的纹理映射
  - 6. MipMap纹理映射

- ## 作业完成日期：10.17

What's More?

What is Texture Mapping?

Applying Textures

Texture Filtering

Advanced Texturing Methods

Many, Many Uses for Texturing

In modern GPUs, texture = memory + filtering

- General method to bring data to fragment calculations

Many applications

- Environment lighting
- Store microgeometry
- Procedural textures
- Solid modeling
- Volume rendering
- …

Environment Map 环境光贴图

A function from the sphere to colors, stored as a texture.

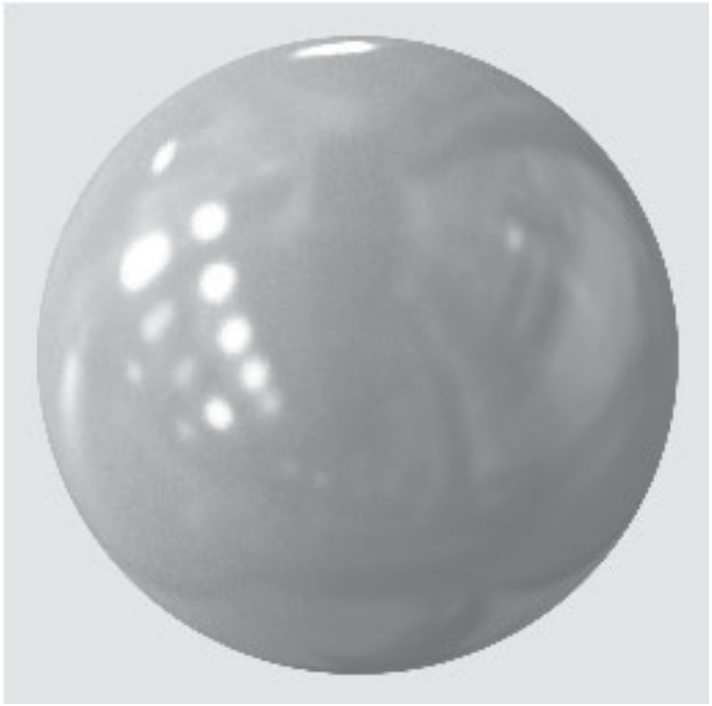
Lat / long texture map

Reflection vector indexes into texture map

[Blinn & Newell 1976]

模型烘焙：提供足够的细节同时保持性能。在游戏和VR中做为模型渲染的替代。

Environment Map



With environment map

Without environment map

[Dror, Willsky, & Adelson 2004]

Spherical Environment Map

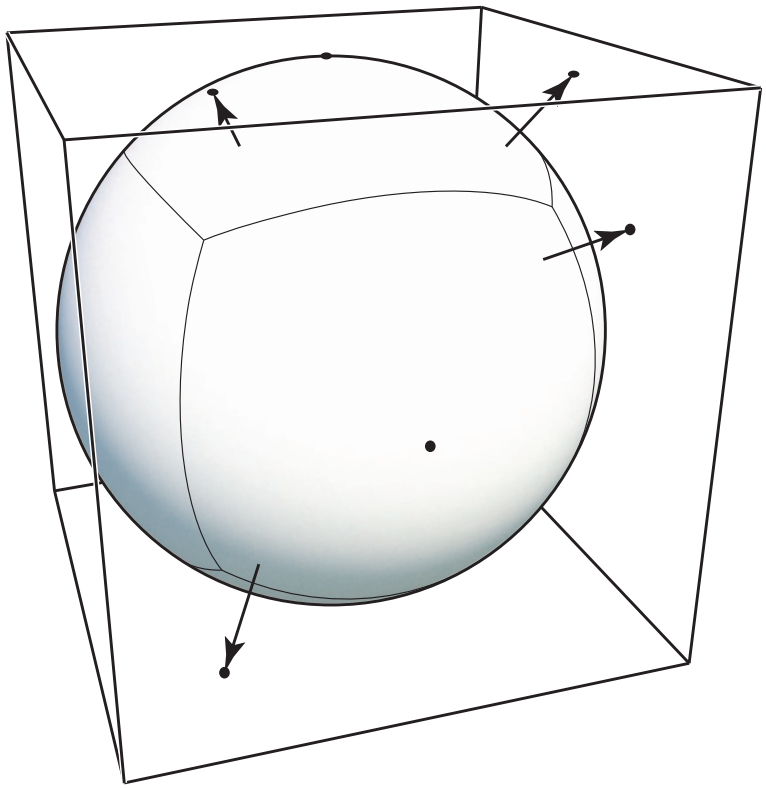
Hand with Reflecting Sphere. M. C. Escher, 1935. lithograph

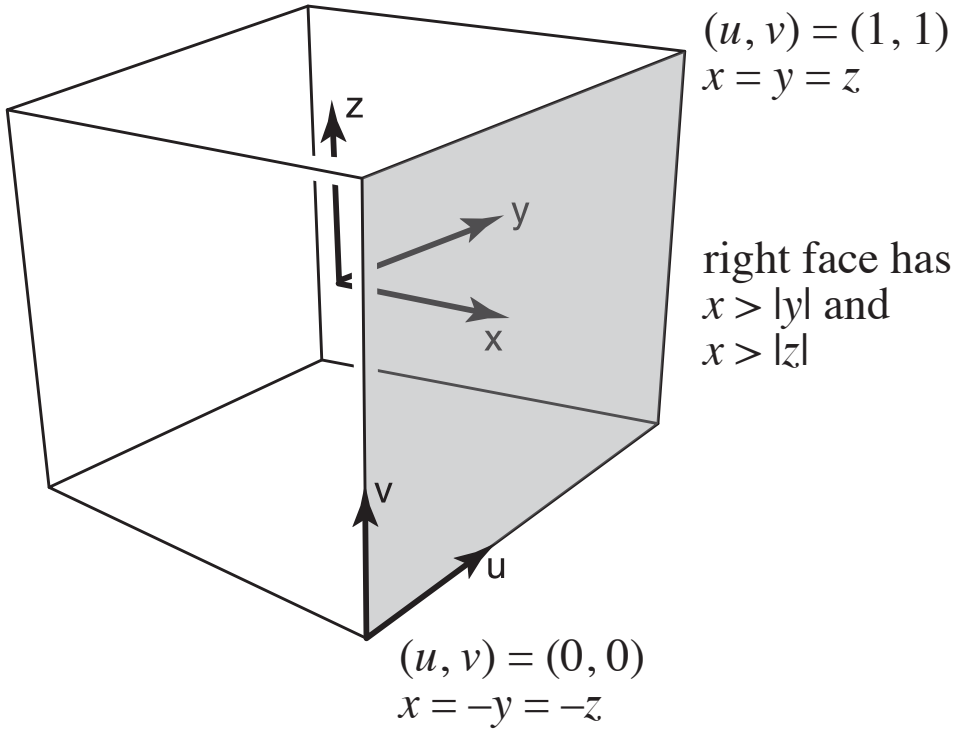
Light Probes, Paul Debevec

Spherical Environment Map

Prone to distortion (top and bottom parts)!

Cube Map





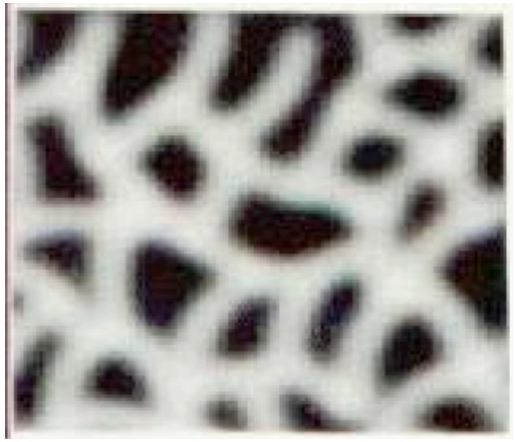
A vector maps to cube point along that direction.

The cube is textured with 6 square texture maps.

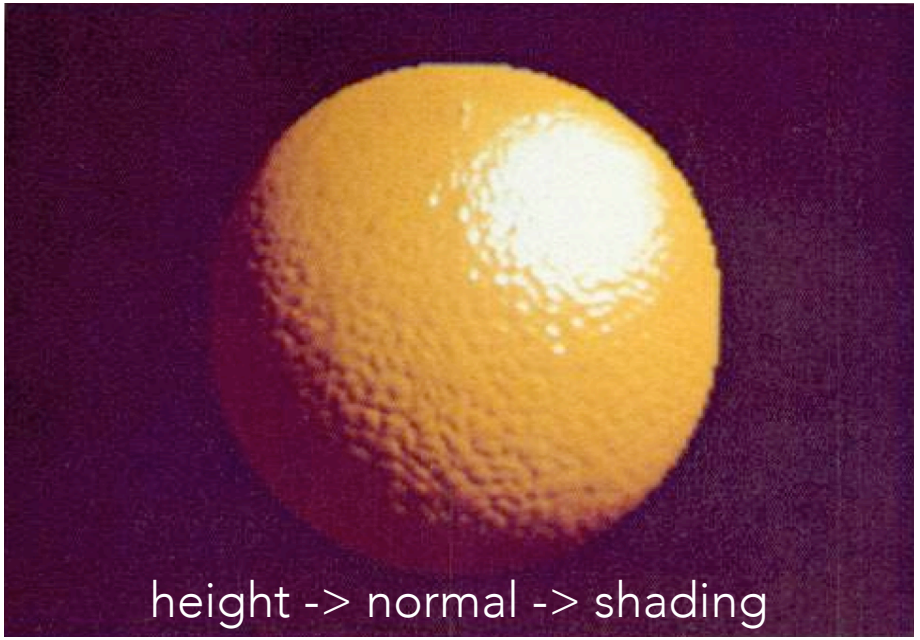
[Emil Persson]

Textures can affect shading!

- Textures doesn't have to only represent colors
  - What if it stores the height / normal?
  - Bump / normal mapping
  - Fake the detailed geometry



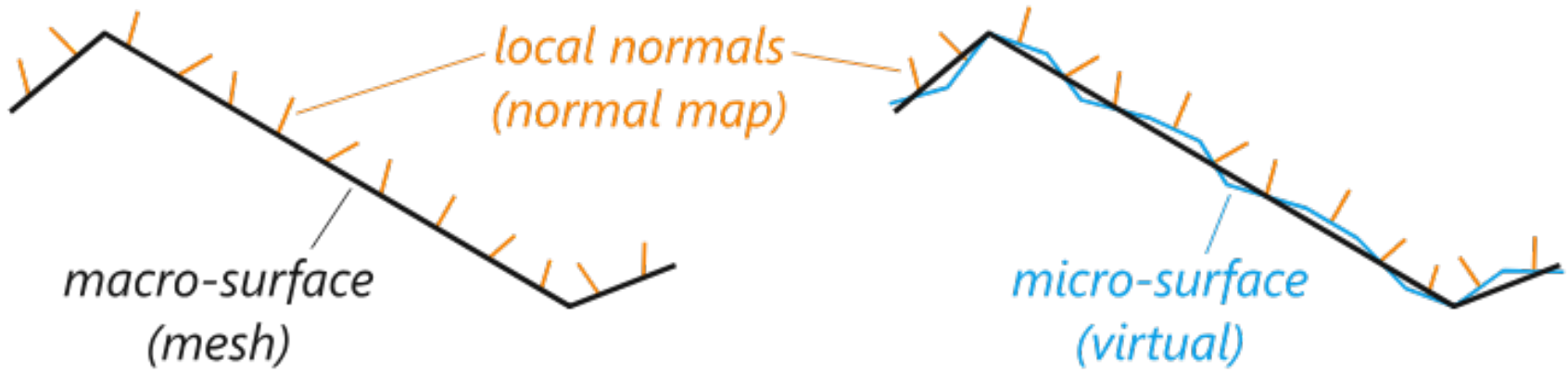
Relative height to the underlying surface



height -> normal -> shading

Textures can affect shading!

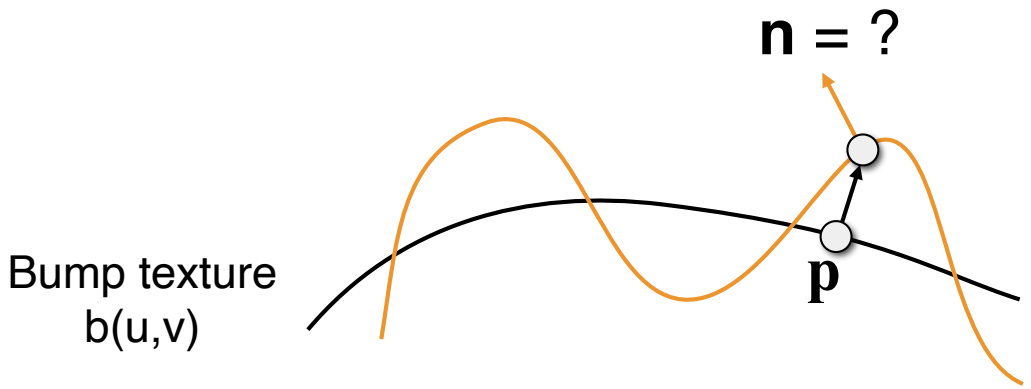
- Textures doesn't have to only represent colors
  - What if it stores the height / normal?
  - Bump / normal mapping
  - Fake the detailed geometry



法向贴图

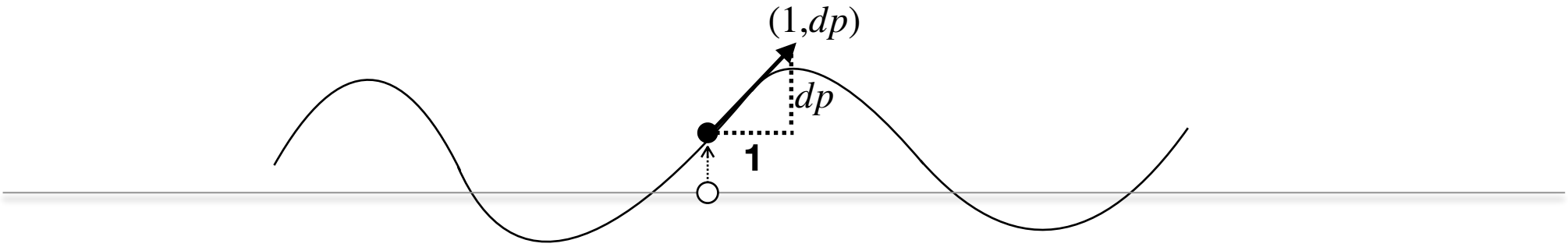
Bump Mapping (Normal Mapping)

- Adding surface details without adding more triangles
  - Perturb surface normal per pixel (for shading computations only)
  - "Height shift" per texel defined by a texture
  - How to modify normal vector?



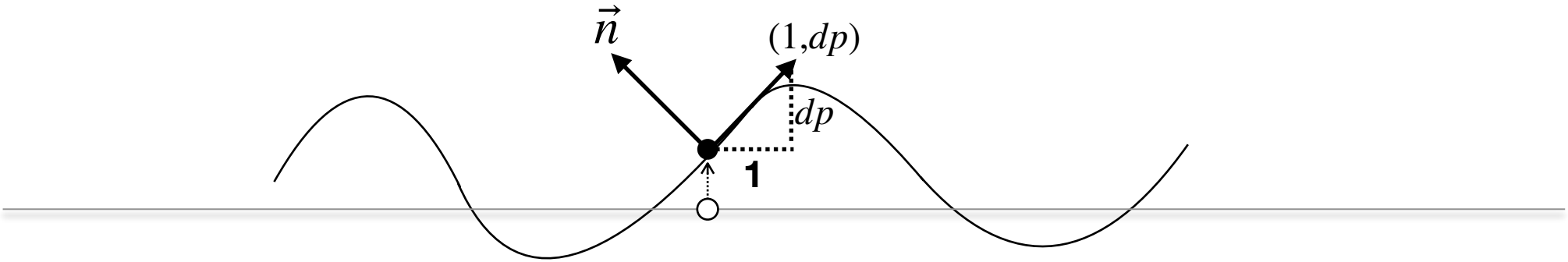
How to perturb the normal (in flatland)

- Original surface normal $n(p) = (0,1)$
- Derivative at p is $dp = c * [h(p+1) - h(p)]$



How to perturb the normal (in flatland)

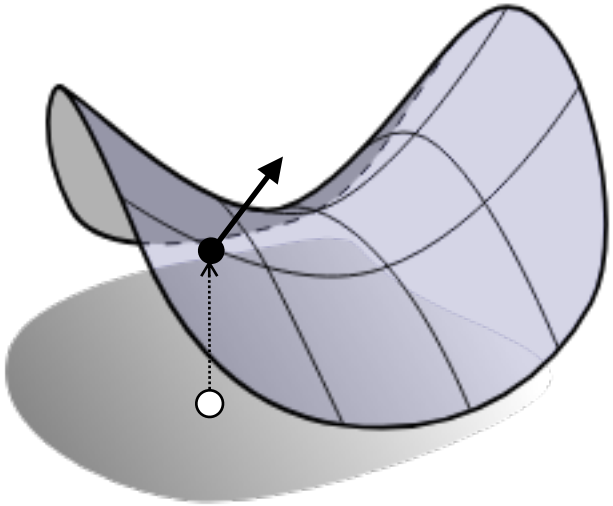
- Original surface normal $n(p) = (0,1)$
- Derivative at p is $dp = c * [h(p+1) - h(p)]$
- Perturbed normal is then $n(p) = (-dp,1)$.normalized()



How to perturb the normal (in 3D)

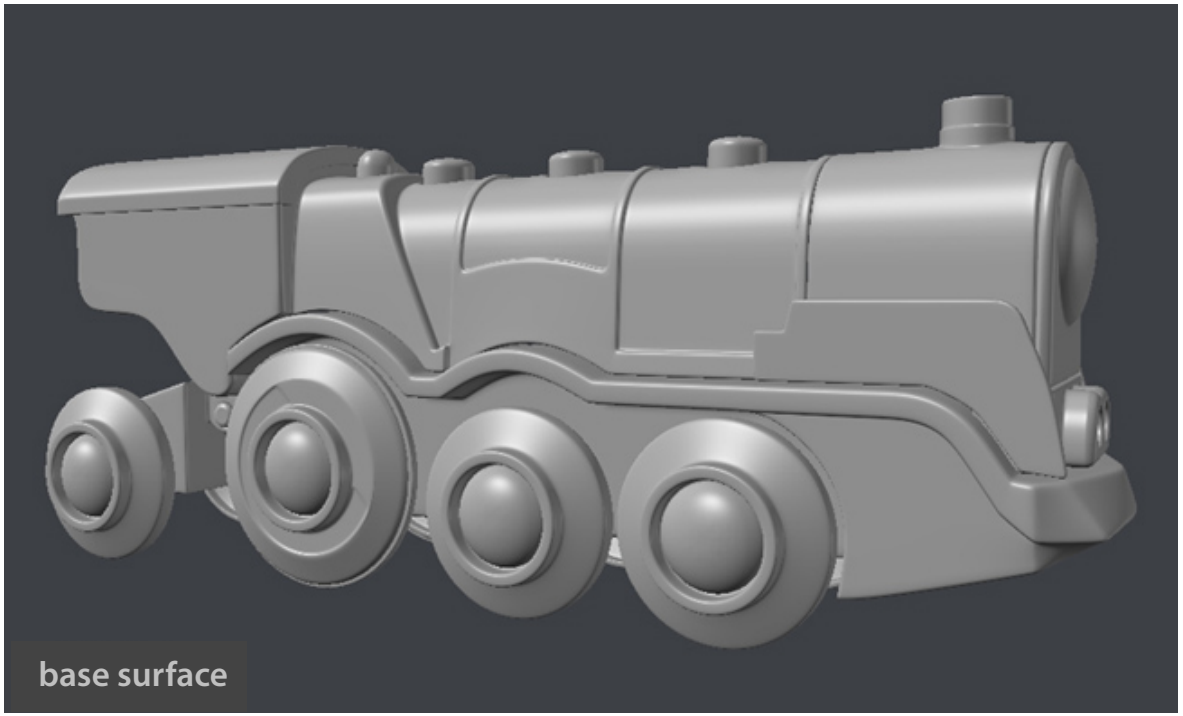
- Original surface normal $n(p) = (0,0,1)$
- Derivative at p are
  - $\frac{dp}{du} = c_1 * [h(u+1) - h(u)]$
  - $\frac{dp}{dv} = c_2 * [h(v+1) - h(v)]$
- Perturbed normal is then

  $n(p) = (-dp/du, -dp/dv, 1)$.normalized()
- Note that this is in local coordinate!

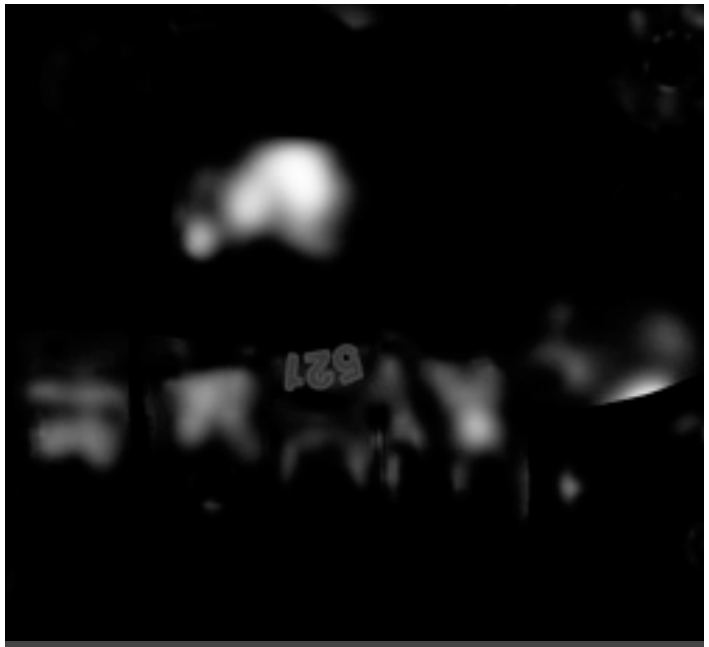


Displacement Mapping 位移贴图

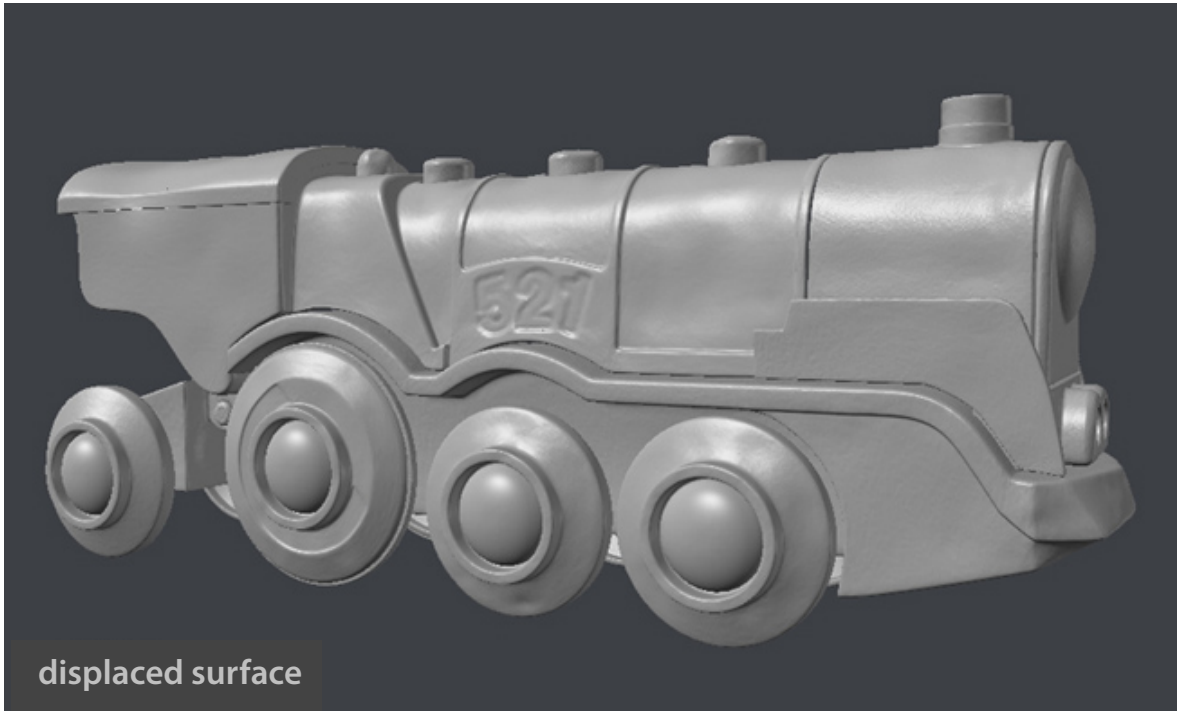
Texture stores perturbation to surface position



base surface



hand-painted displacement map (detail)

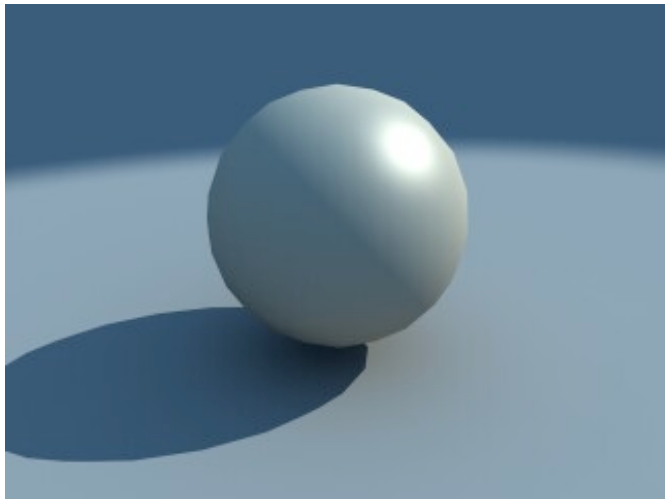


displaced surface

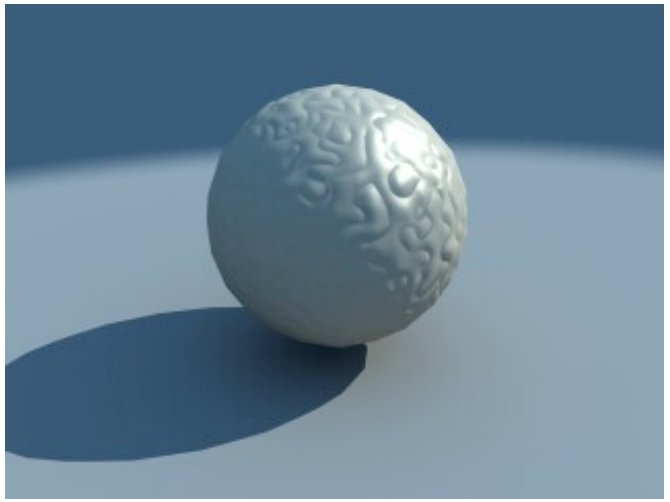
Paweł Filip
tolas.wordpress.com

Textures can affect shading!

- Displacement mapping — a more advanced approach
  - Uses the same texture as in bumping mapping
  - Actually moves the vertices

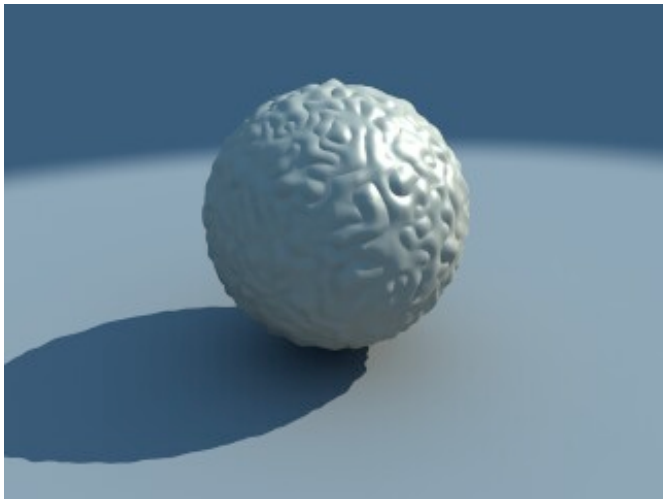


Geometry



Bump mapping

Perturbs normals



Displacement mapping

Perturbs positions

Provide Precomputed Shading

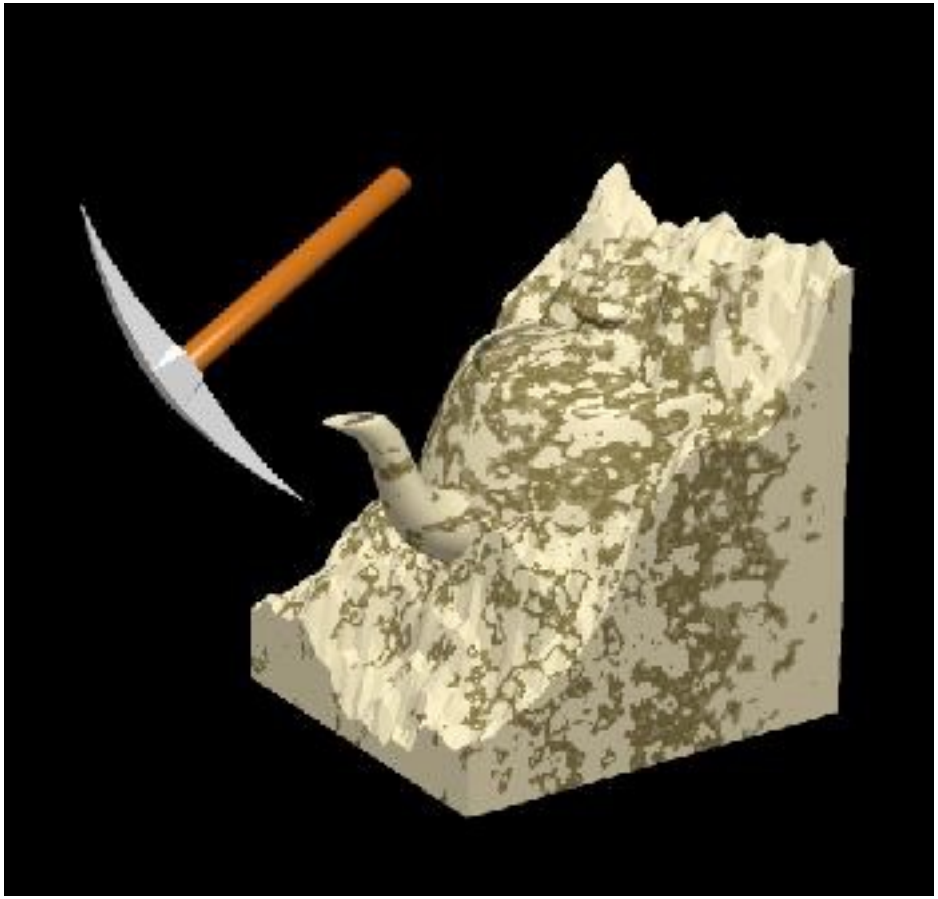
Simple shading

Ambient occlusion texture map

With ambient occlusion

Autodesk

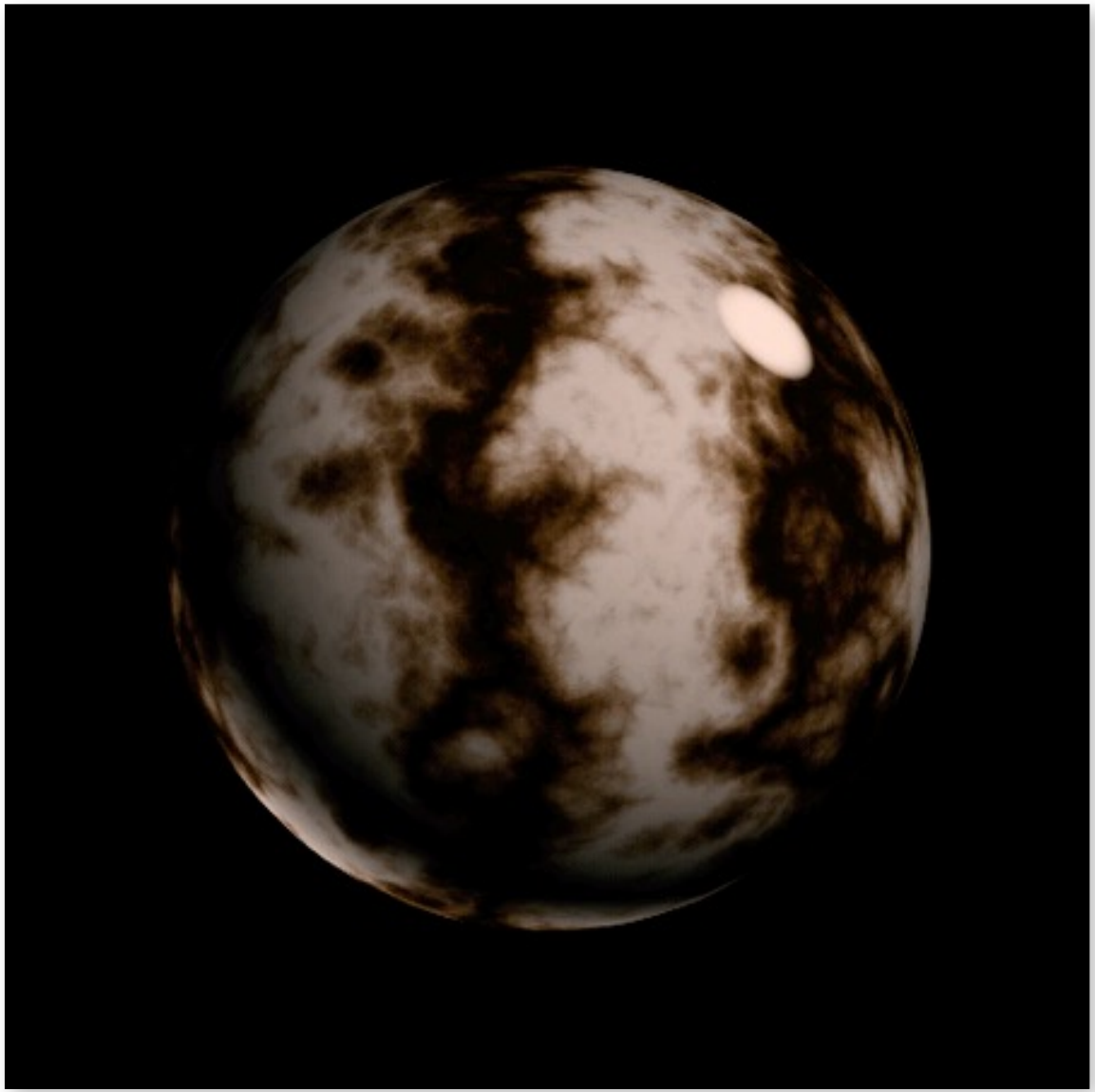
3D Textures

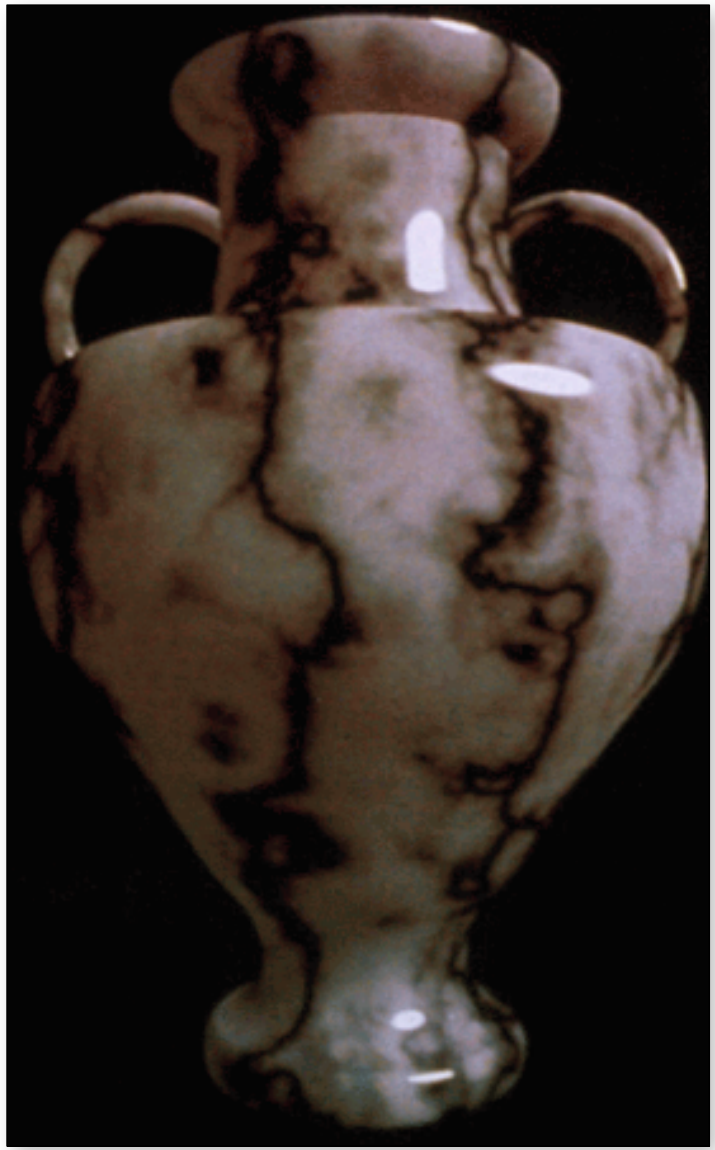


[Wolfe / SG97 Slide set]

Textures is a function of (u,v,w). Solid modeling.

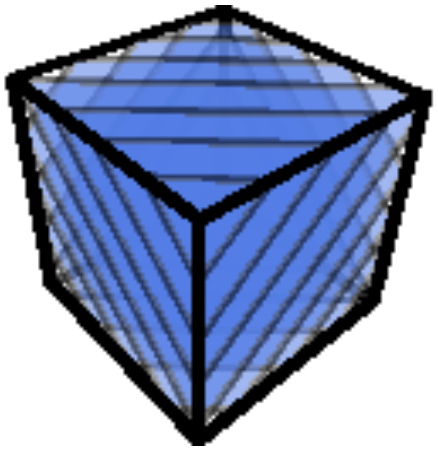
3D Procedural Noise + Solid Modeling

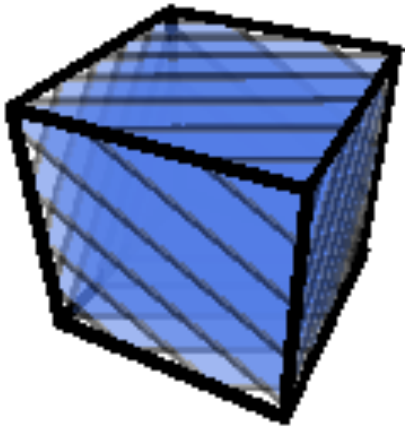




Perlin noise, Ken Perlin

3D Textures and Volume Rendering





Marc Levoy