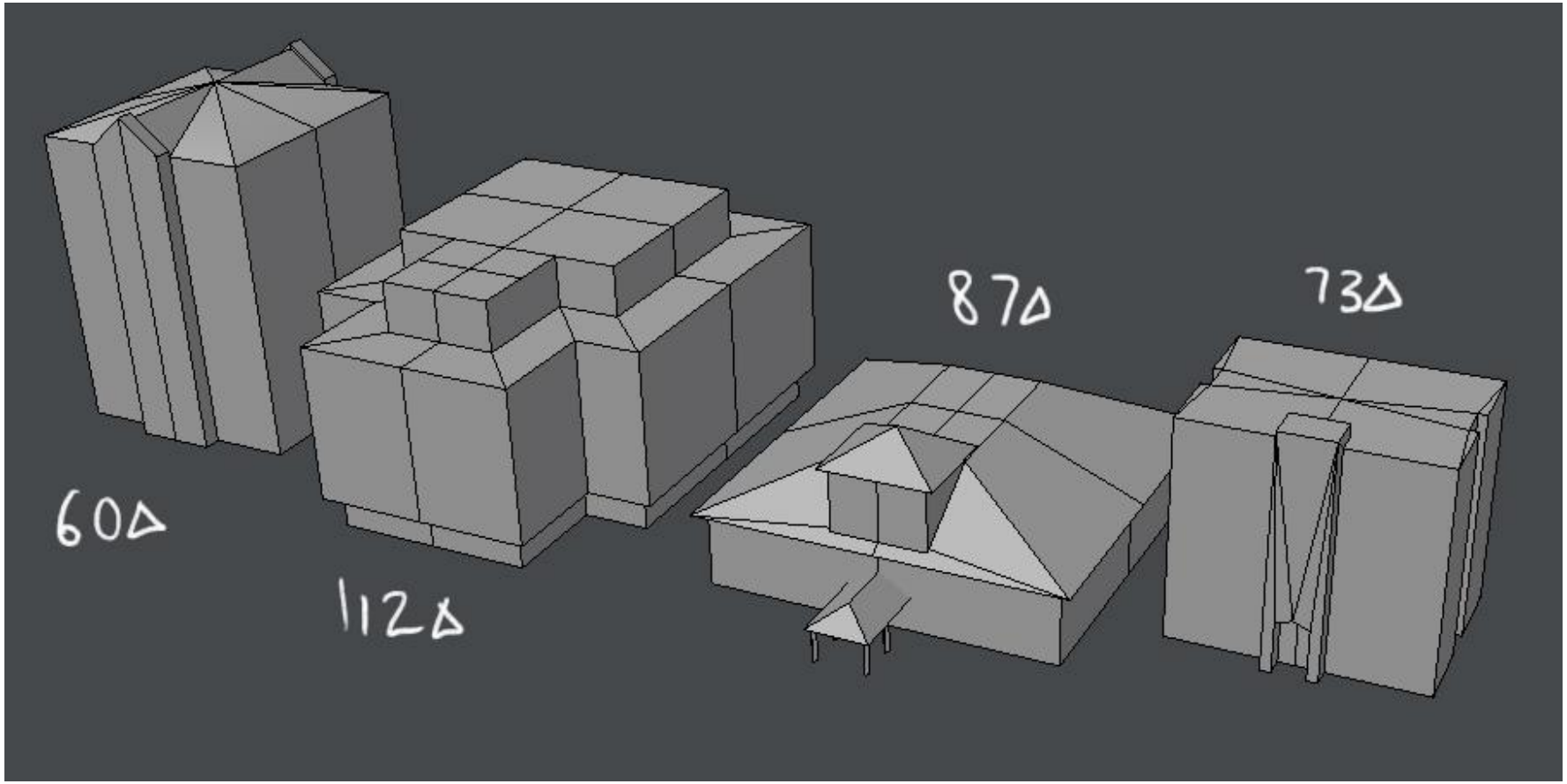
Textures

CSU44052 Computer Graphics

Binh-Son Hua

Motivation

- 3D modeling so far results in some geometry

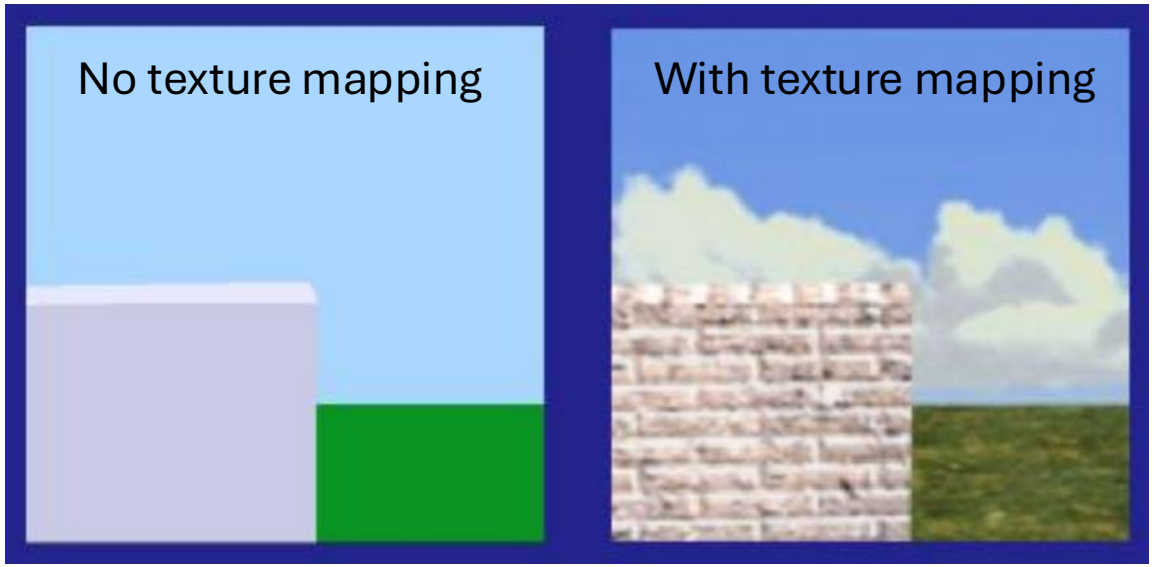


Motivation

- But real building has very different appearance!

Texture Mapping

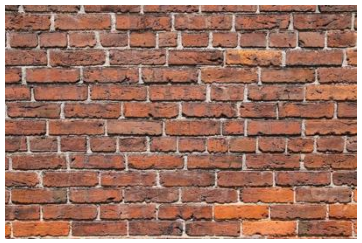
- Texture mapping allows you to take a simple polygon and give it the appearance of something much more complex
  - Due to Ed Catmull, PhD thesis, 1974
- We modulate colours from images to our objects in order to create the illusion of realism



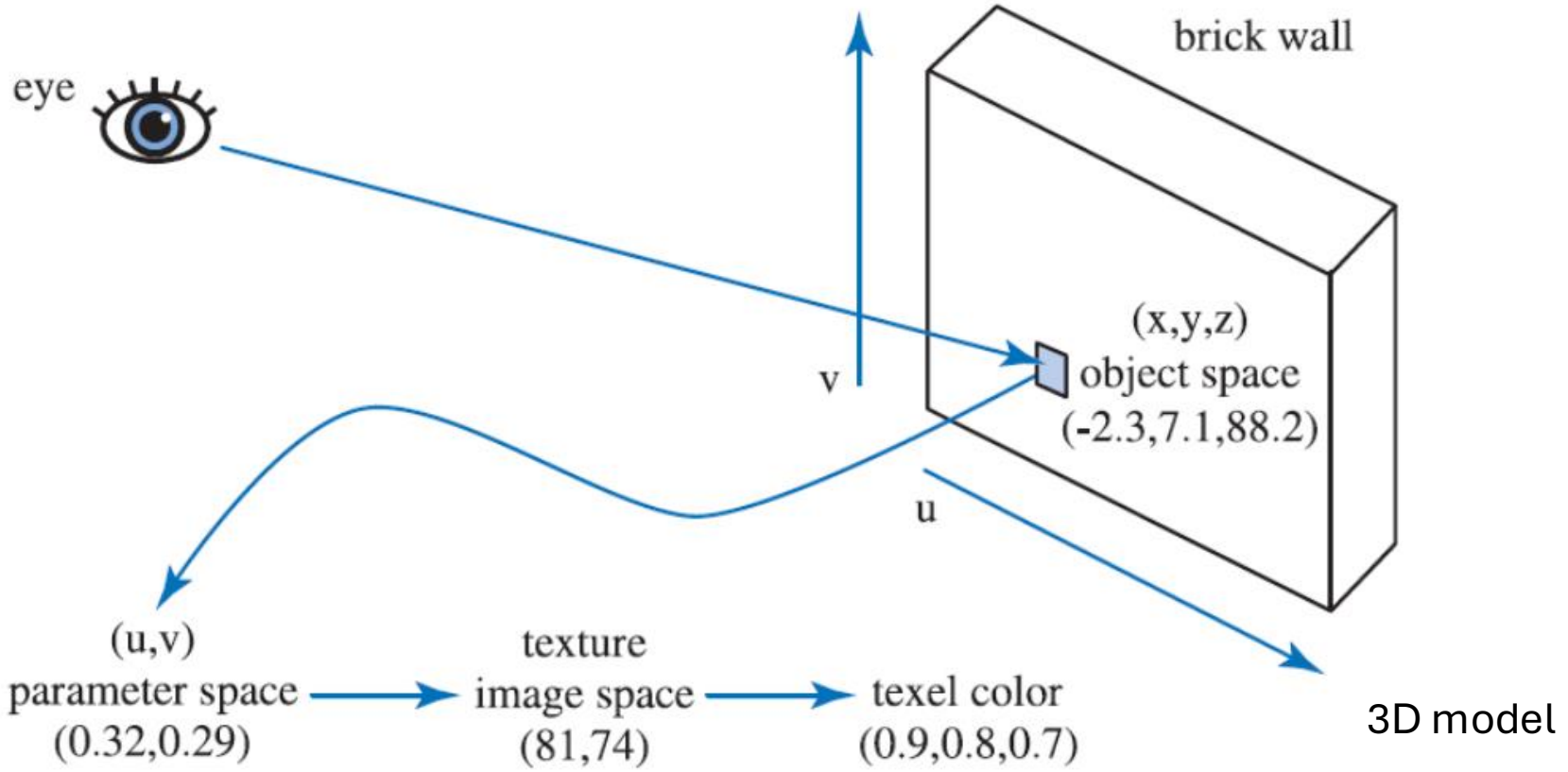
Texturing in OpenGL

```
GLuint texture;
glGenTextures(1, &texture);
glBindTexture(GL_TEXTURE_2D, texture);

// Texture parameters
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);

// Load image into texture
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, w, h, 0, GL_RGB, GL_UNSIGNED_BYTE, img);
```

How does it work?

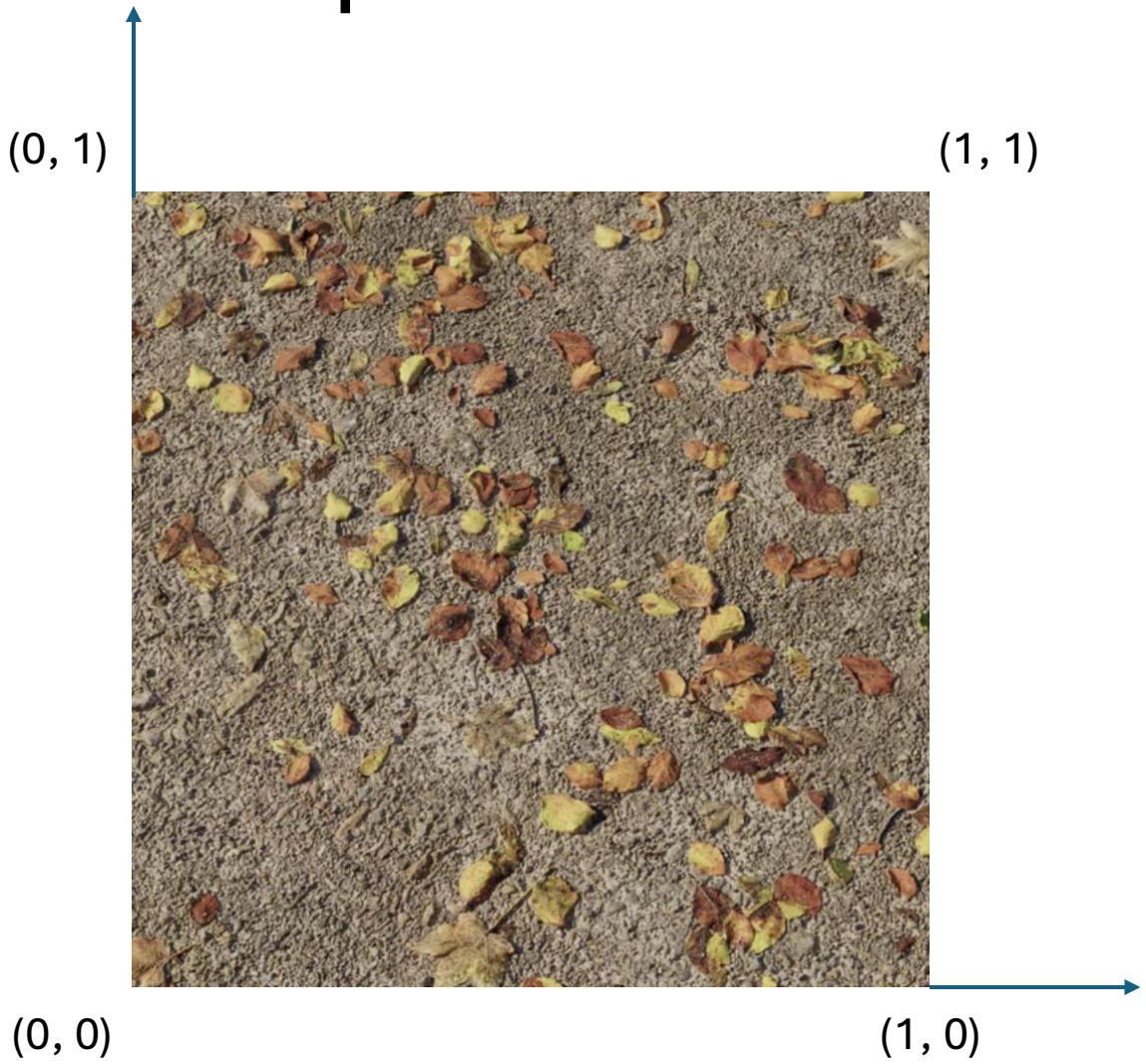


Texture

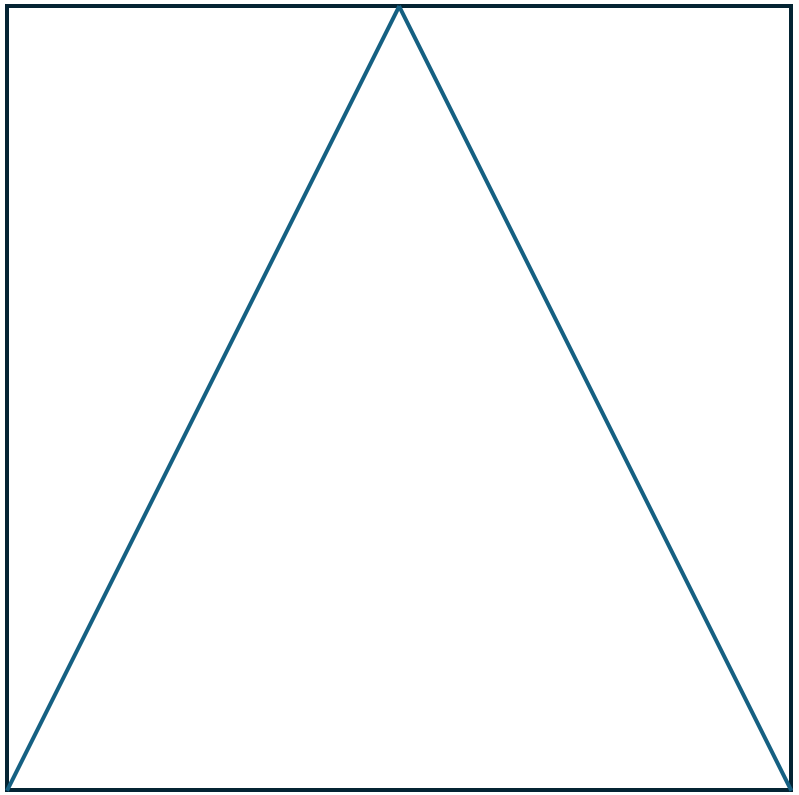


The pixels in the image texture are often called **texels**, to differentiate them from the pixels on the screen

Example: Texture a Triangle

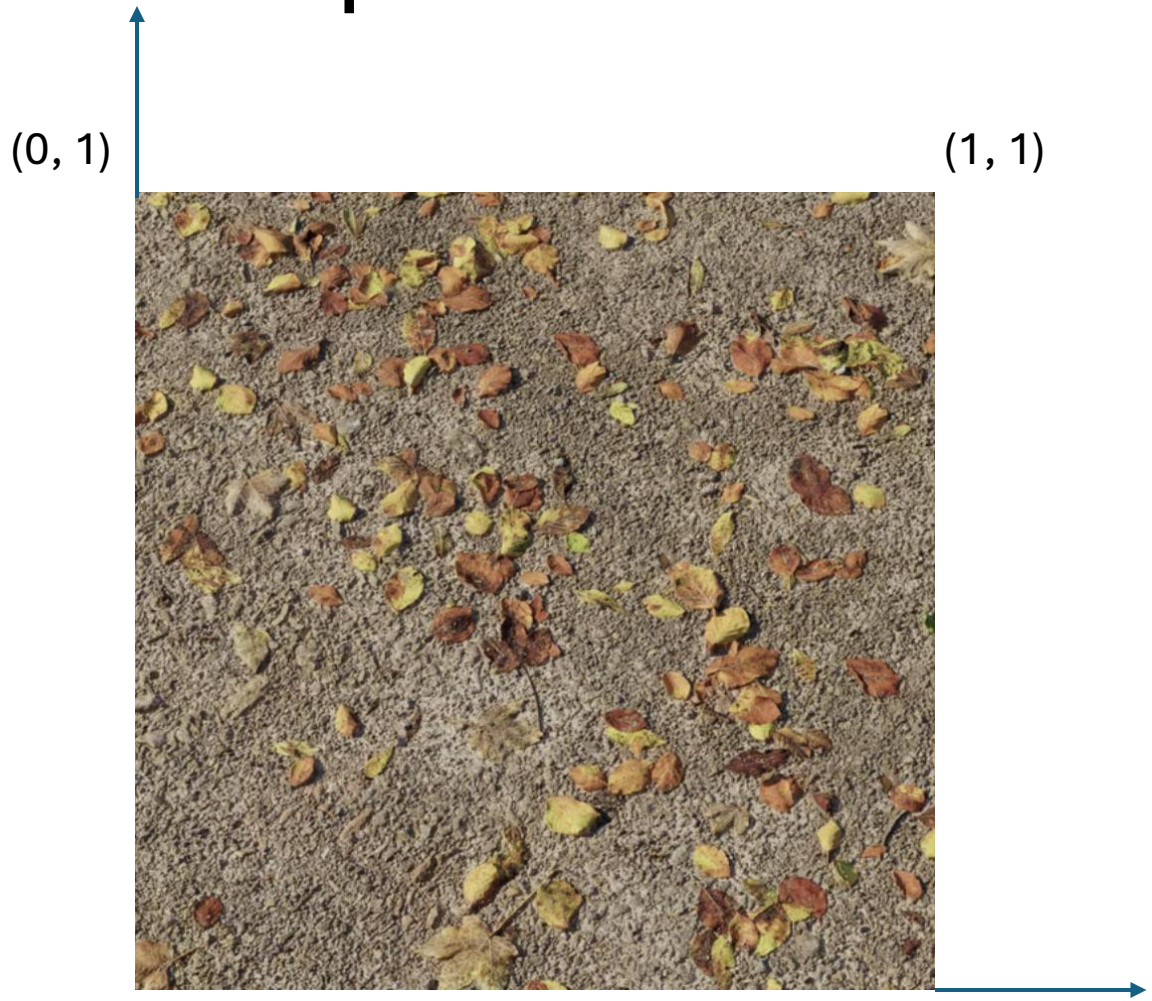


Source texture

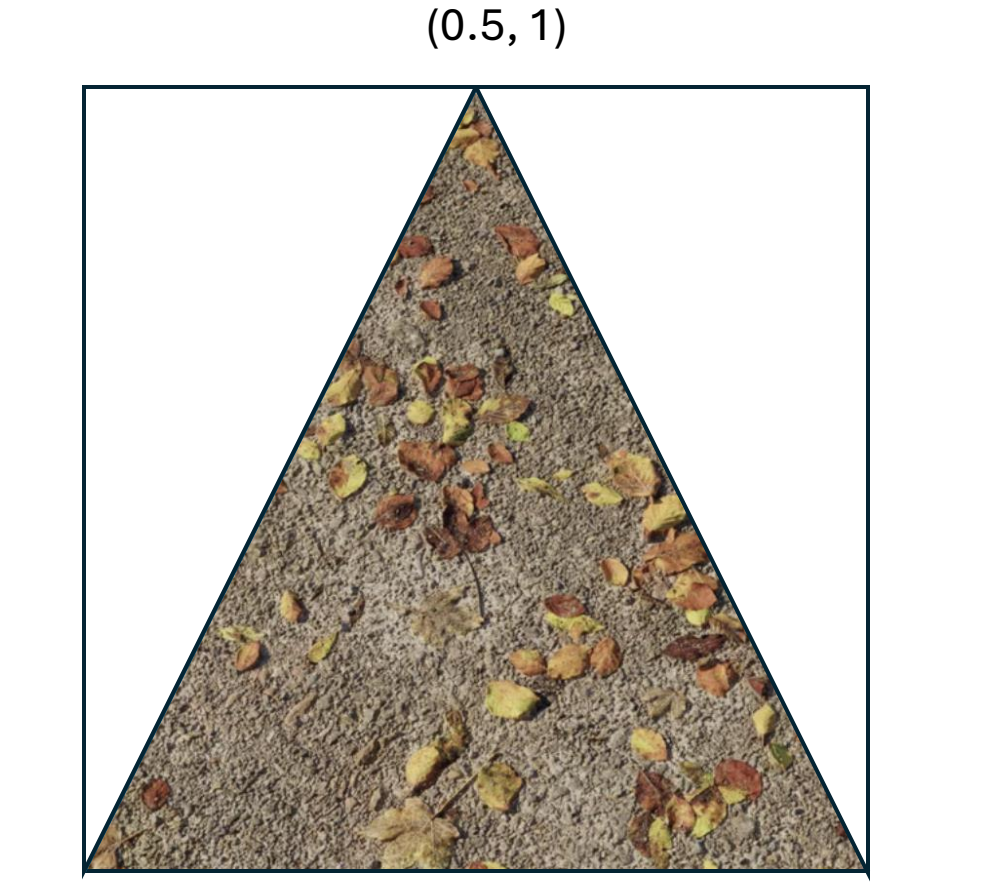


Triangle

Example: Texture a Triangle

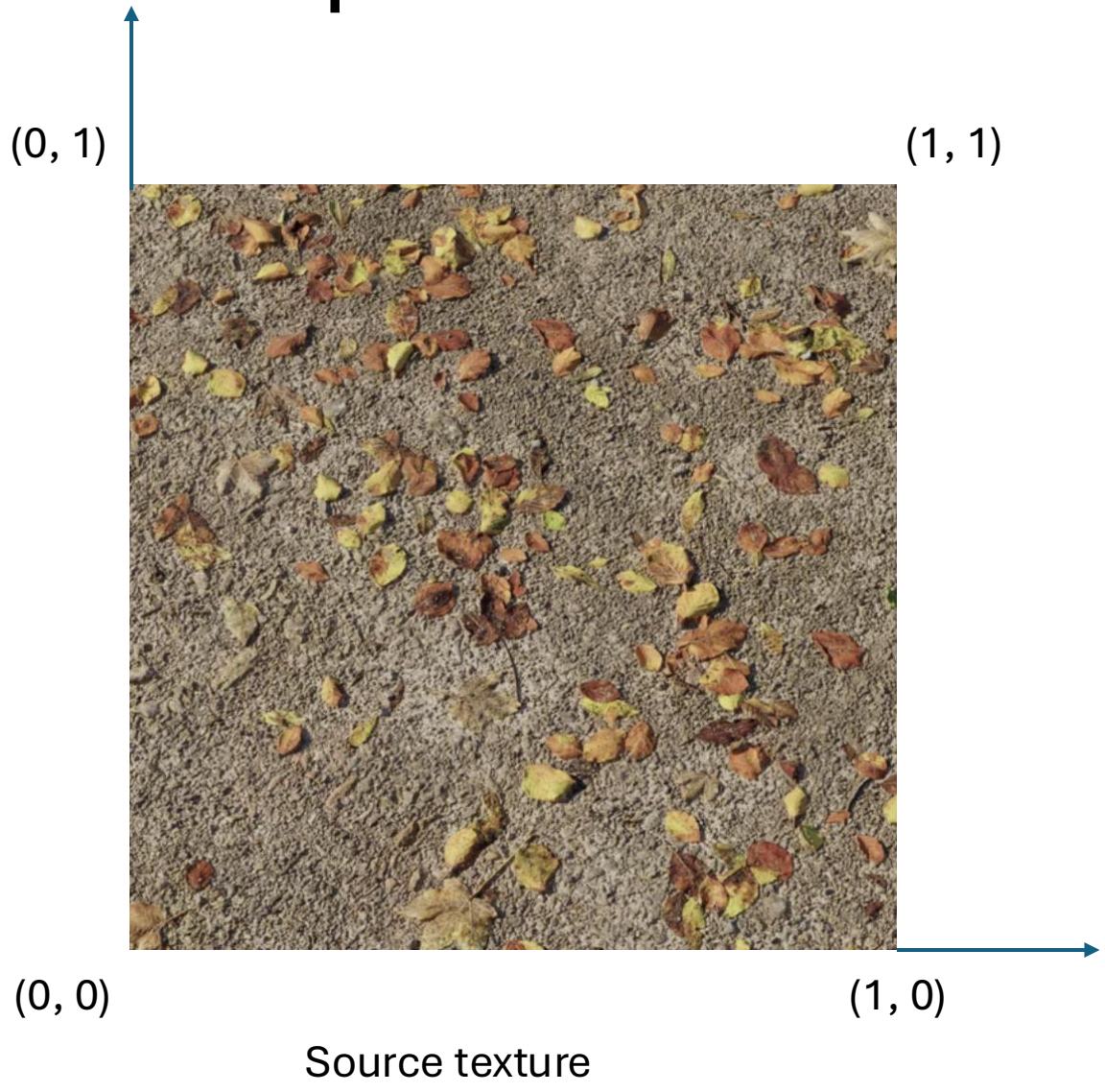


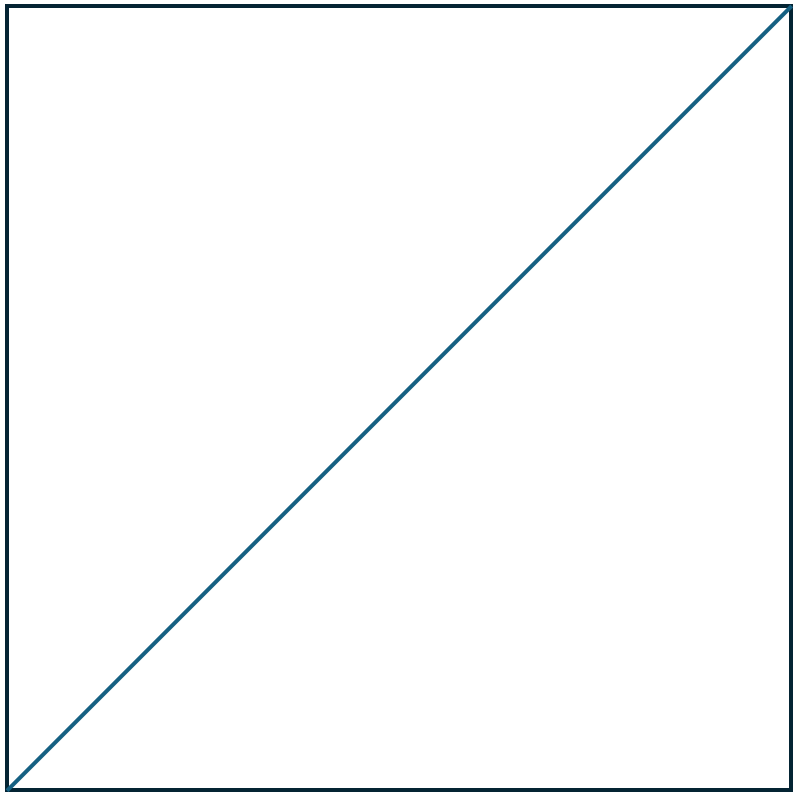
Source texture



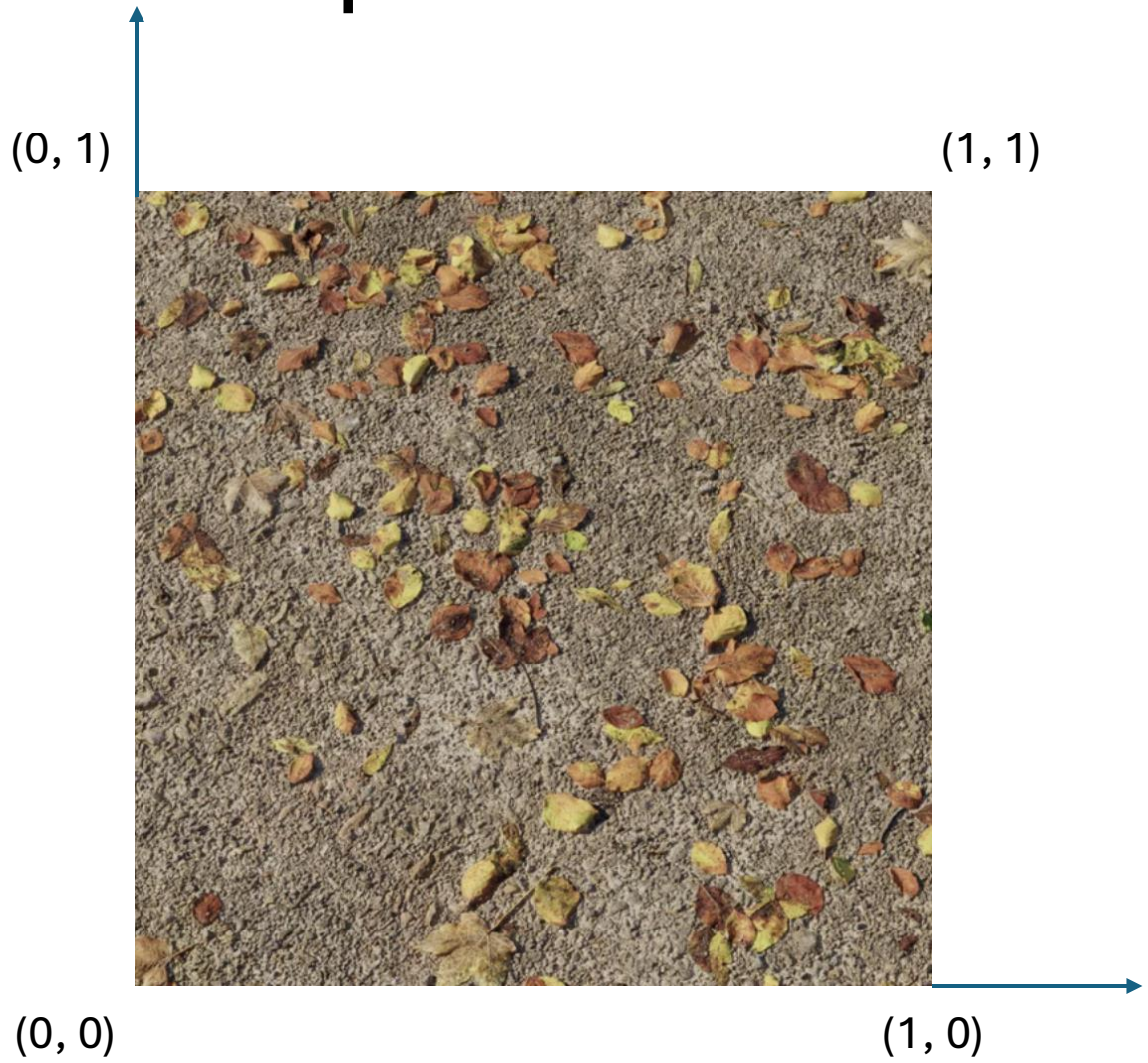
Triangle

Example: Texture a Rectangle

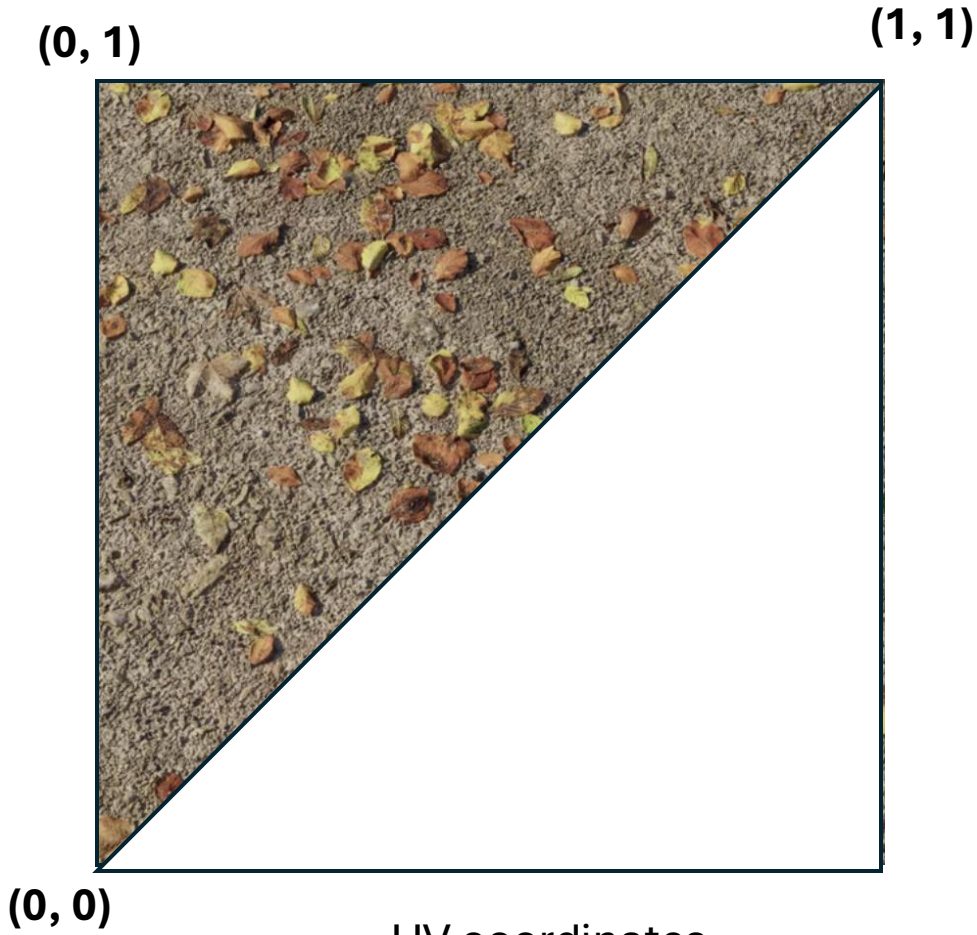




Example: Texture a Rectangle



Source texture



UV coordinates
of 3D vertices

Example: Texture a Rectangle

(0, 1)

(1, 1)

(0, 0)

(1, 0)

Source texture

(0, 1)

(1, 1)

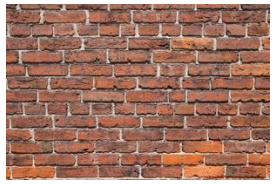
(0, 0)

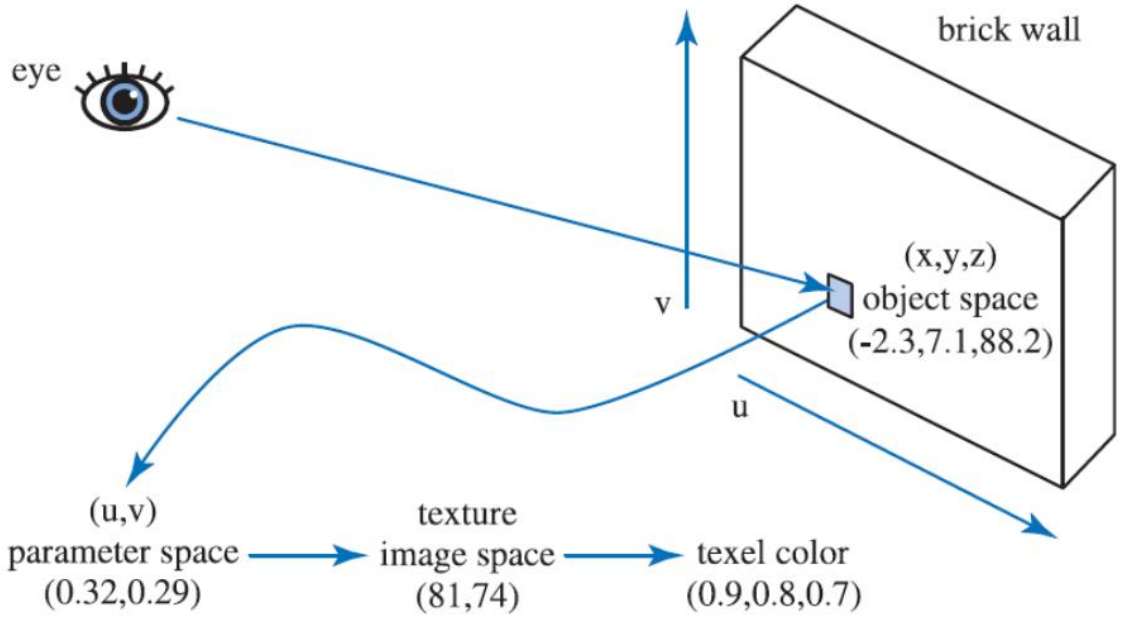
(1, 0)

UV coordinates
of 3D vertices

Texture Mapping Steps

- Create a texture map

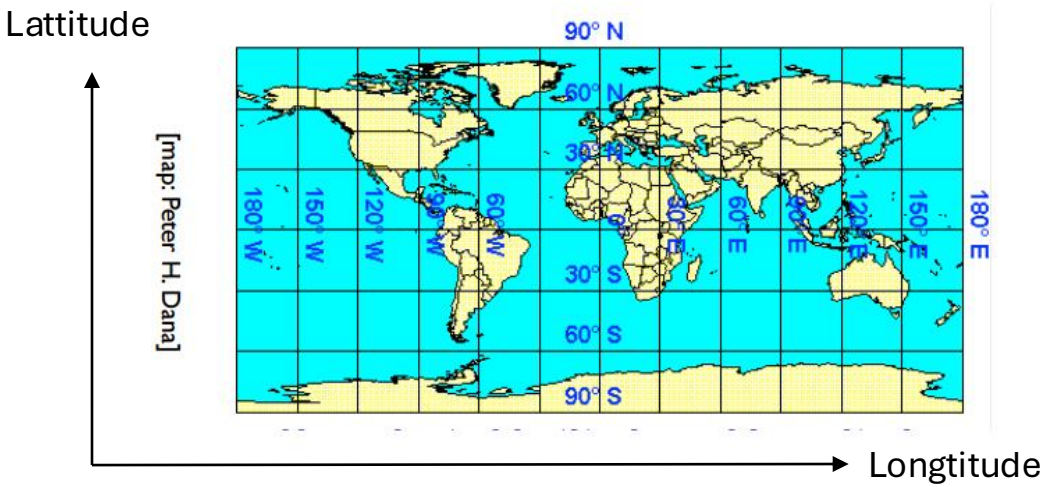




- Define texture coordinates (UV coordinates) on the 3D vertices
- Load texture, set texture parameters
- Sample the texture in a fragment shader using the UV coordinates and calculate the output color

Projector Functions

- Parametric function to map a 3D point into texture coordinates
- Sphere: map polar coordinates (latitude θ and longtitude ϕ) to (x, y, z)



$$x = R \sin\theta \cos\varphi$$

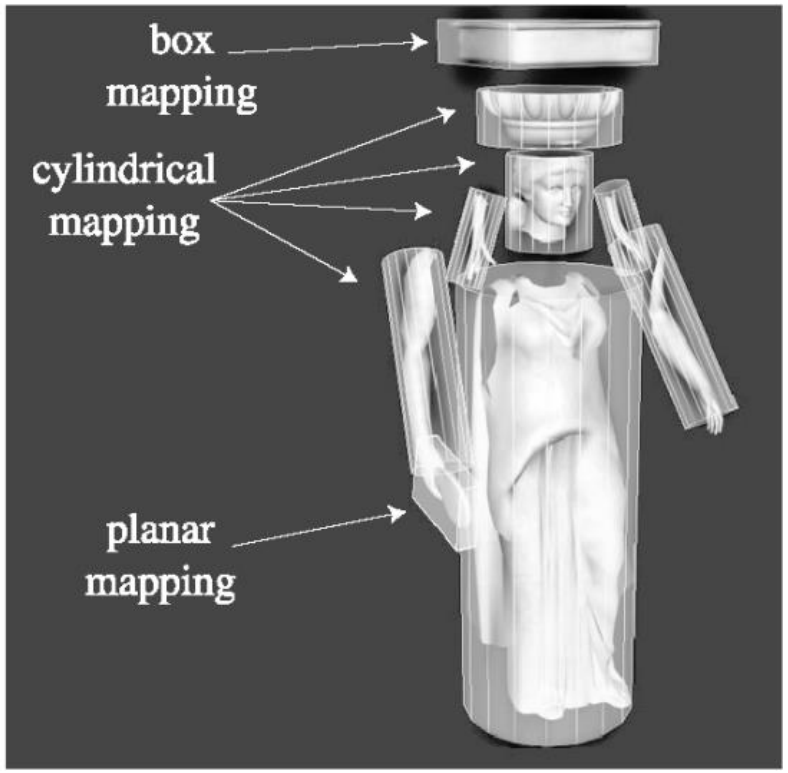
$$y = R \sin\theta \sin\varphi$$

$$z = R \cos\theta$$

- Usually done offline and stored with the vertex.
- But can be applied in the vertex shader, giving different effects such as animation (e.g., fire, water, blending between marble and skin textures to make a statue come to life

Arbitrary Surfaces

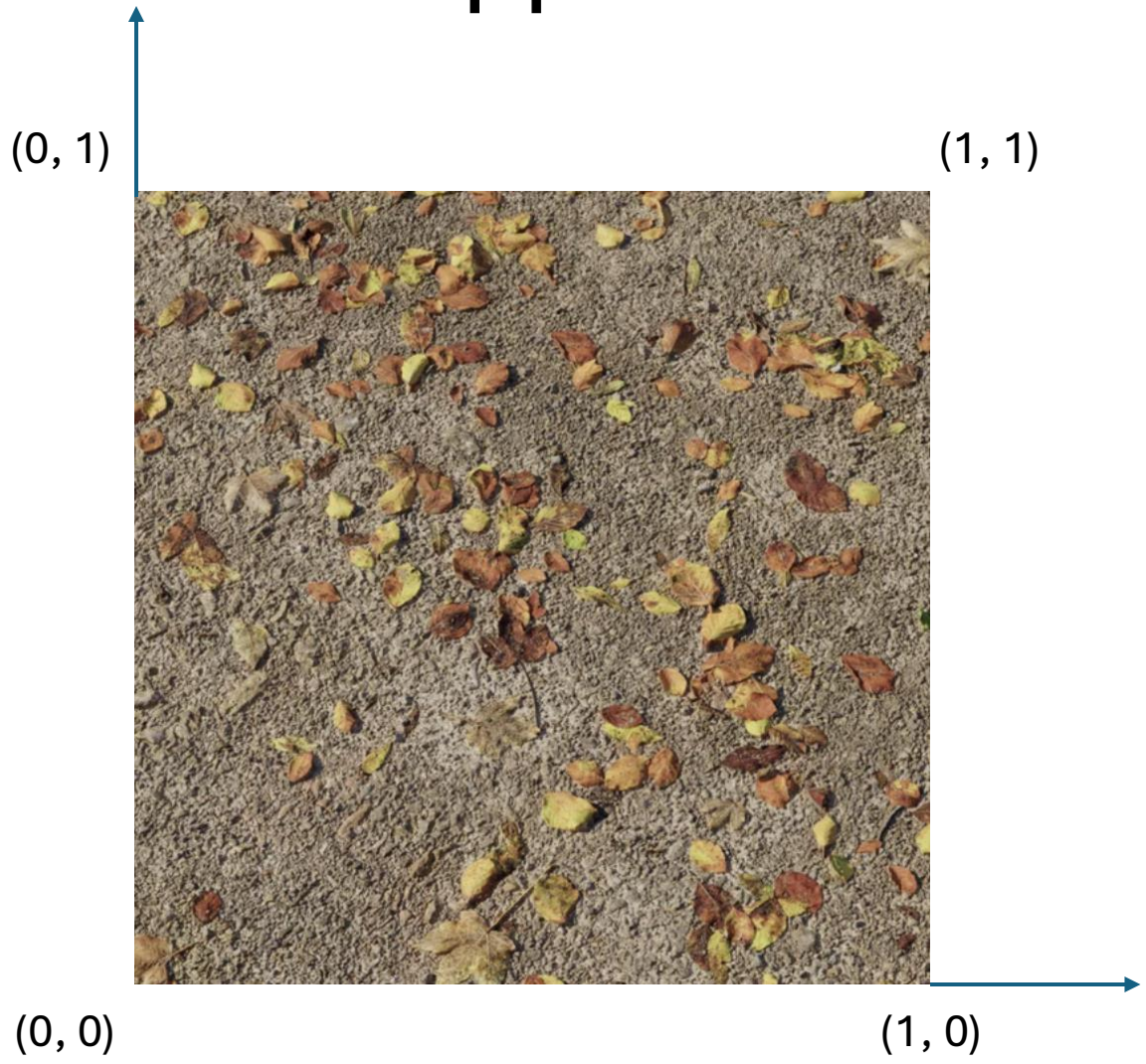
- Non-parametric surfaces: project to parametric surface



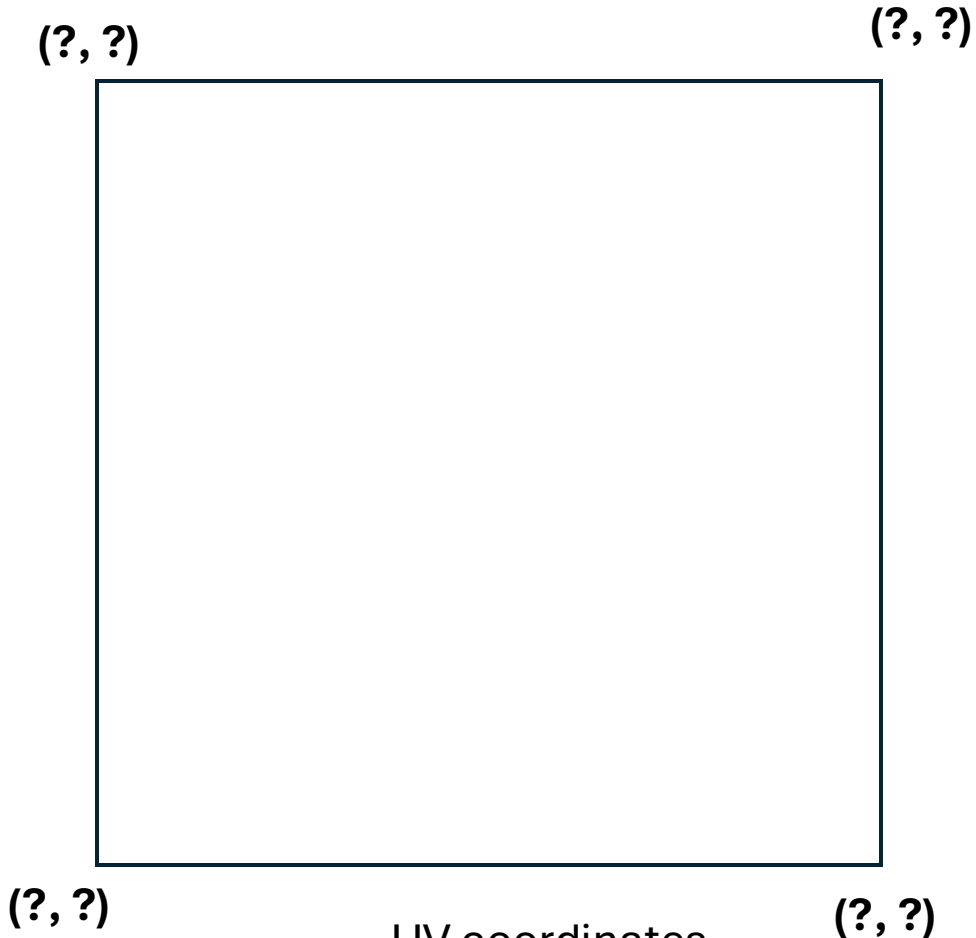
Creating Textures using 3D Modeling Software

- Artist often manually decomposes the model into near-planar pieces
- Tools to help minimize distortion by unwrapping the mesh
- Goal: have each polygon be given a fairer share of a texture's area, while also maintaining as much mesh connectivity as possible

What happens if UV not in [0, 1]?



Source texture



UV coordinates
of 3D vertices

Texture Wrapping

- Configure the behaviour when UV coordinates are out of range (when out of [0, 1])

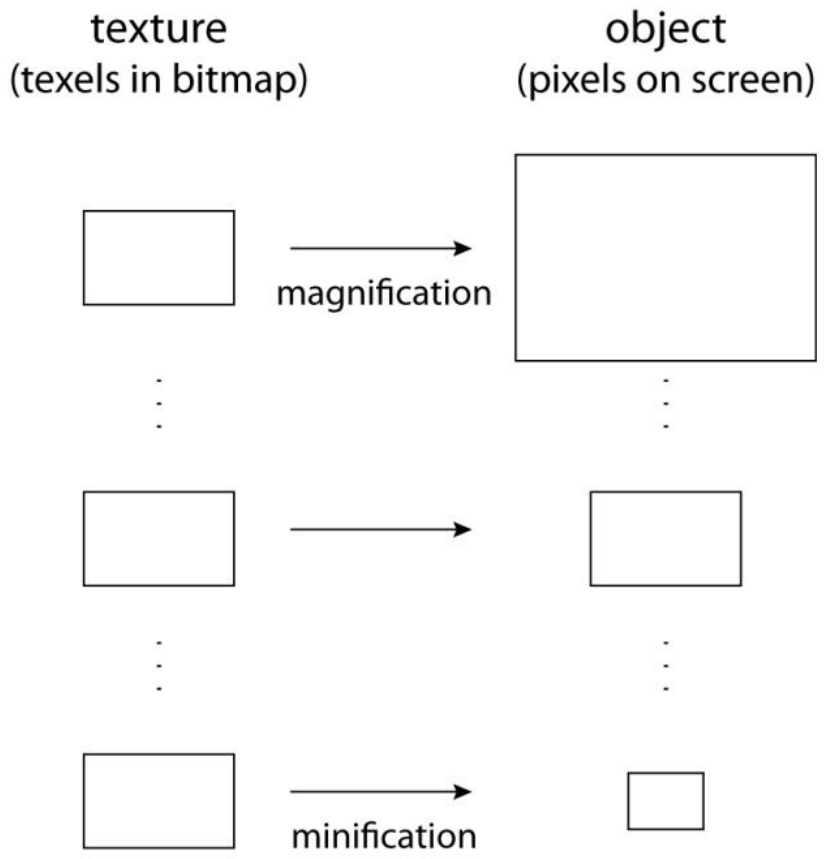
- *Wrap: Repeats*
- *Mirror – Repeats but mirrored every other time; continuity across edges*
- *Clamp: Clamped to edge of texture*
- *Border: Clamped to border colour*

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
```

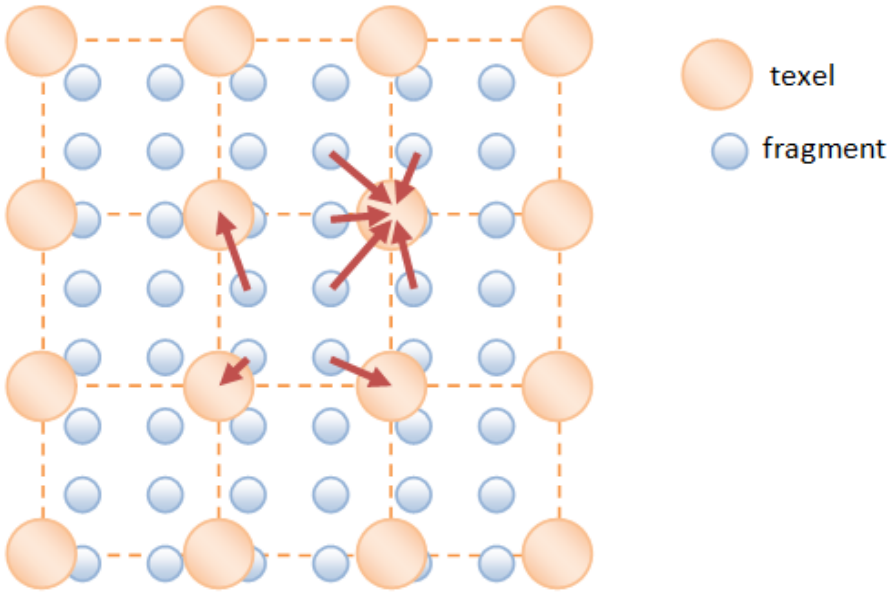
Magnification & Minification

- **Sample** the texture map:
- If viewer is close: Object gets larger → Magnify texture
- “Perfect” distance: Not always “perfect” match (misalignment, etc.)
- If viewer is further away: Object gets smaller → Minify texture

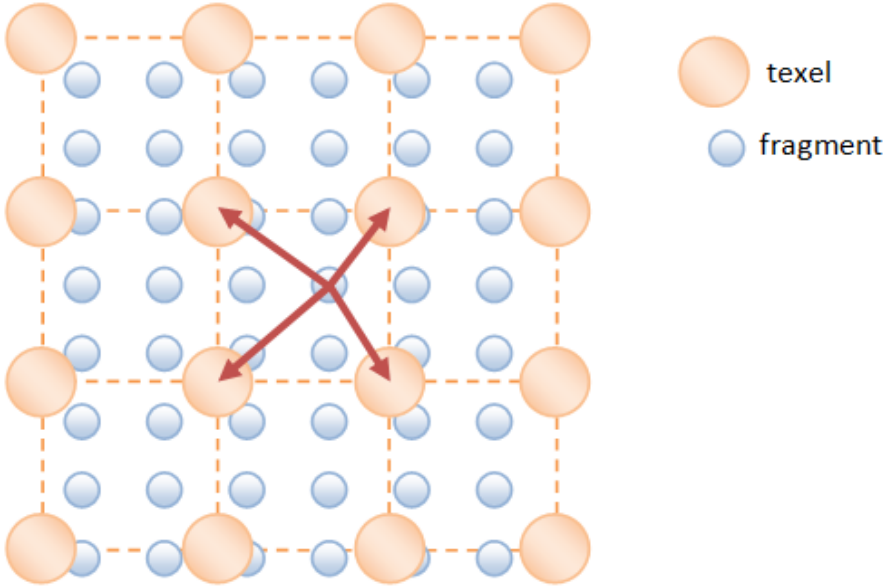
What if the projected texture square does not match the screen resolution?



Texture Filtering



Nearest point sampling



Bilinear interpolation

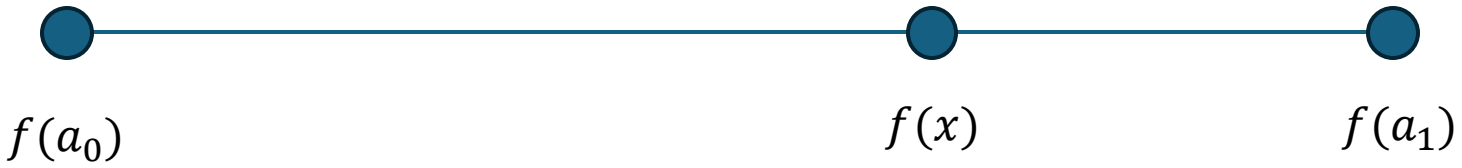
https://www.ntu.edu.sg/home/ehchua/programming/opengl/CG_BasicsTheory.html

Texture Filtering

GL_NEAREST GL_LINEAR

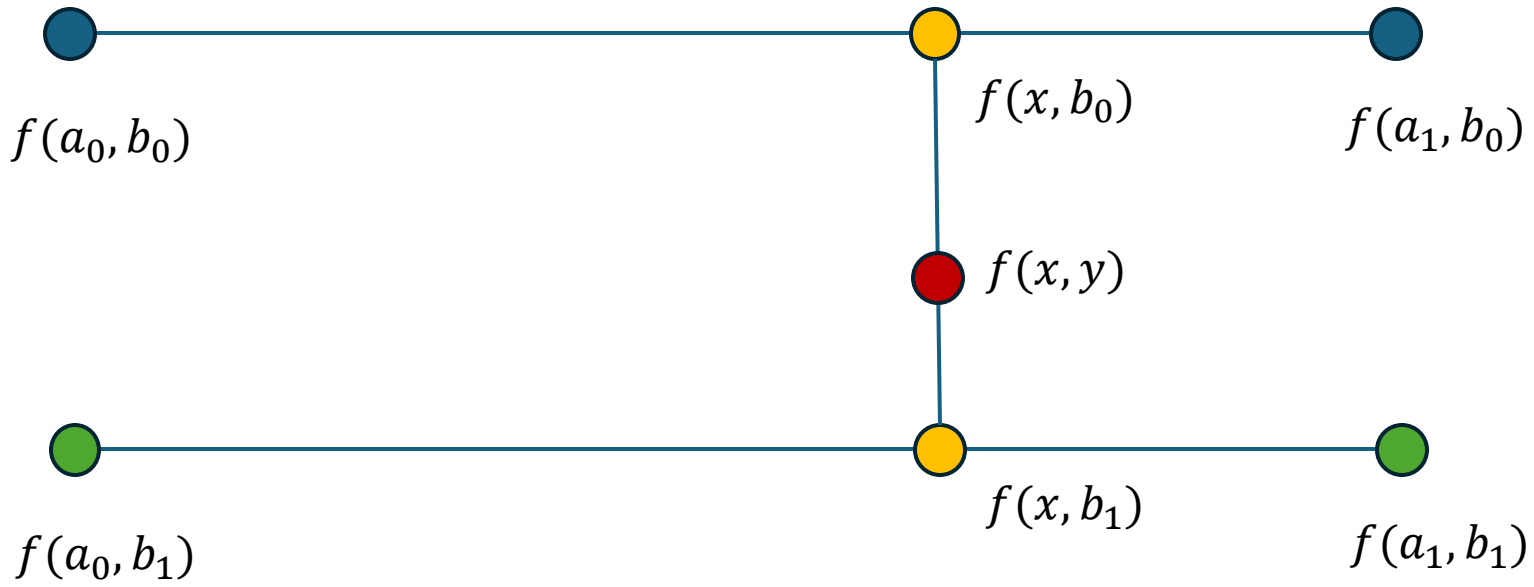
```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
```

Linear Interpolation

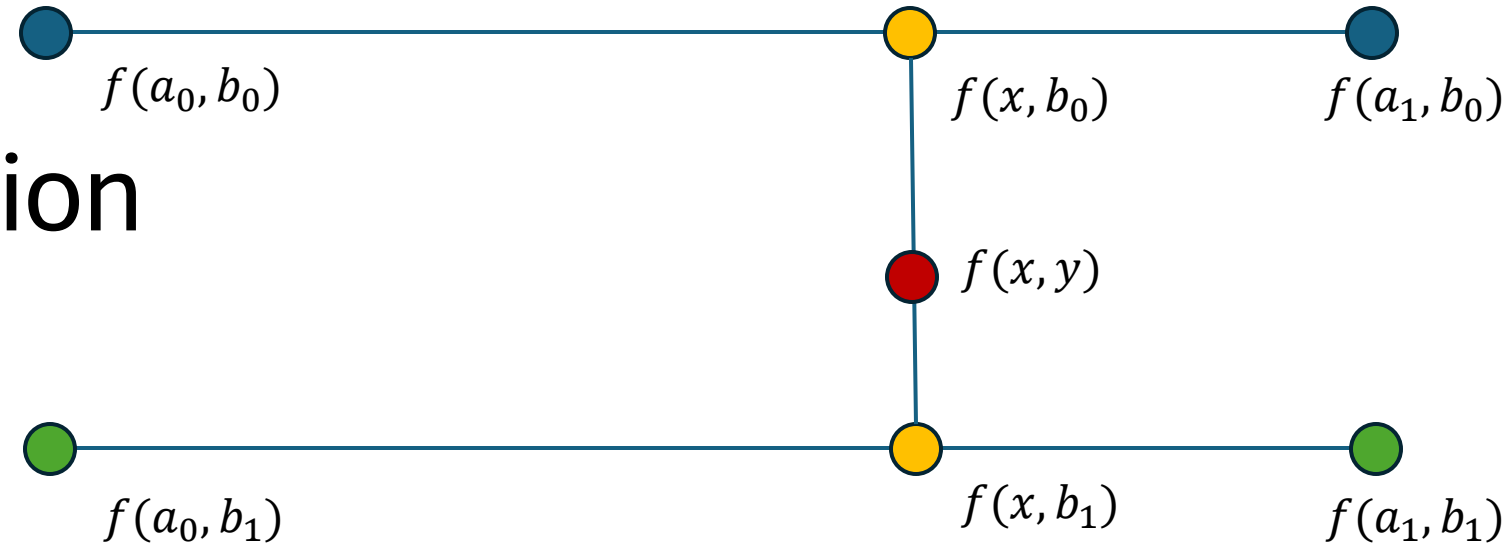


$$f(x) = f(a_0) + \frac{x - a_0}{a_1 - a_0}\big(f(a_1) - f(a_0)\big)$$

Bilinear Interpolation



Bilinear Interpolation



$$f(x, b_0) = f(a_0, b_0) + \frac{x - a_0}{a_1 - a_0}\left(f(a_1, b_0) - f(a_0, b_0)\right)$$

$$f(x, b_1) = f(a_0, b_1) + \frac{x - a_0}{a_1 - a_0}\left(f(a_1, b_1) - f(a_0, b_1)\right)$$

$$f(x, y) = f(x, b_0) + \frac{y - b_0}{b_1 - b_0}\left(f(x, b_1) - f(x, b_0)\right)$$

Magnification Aliasing

Magnification aliasing – walls are lower resolution than on-screen pixels

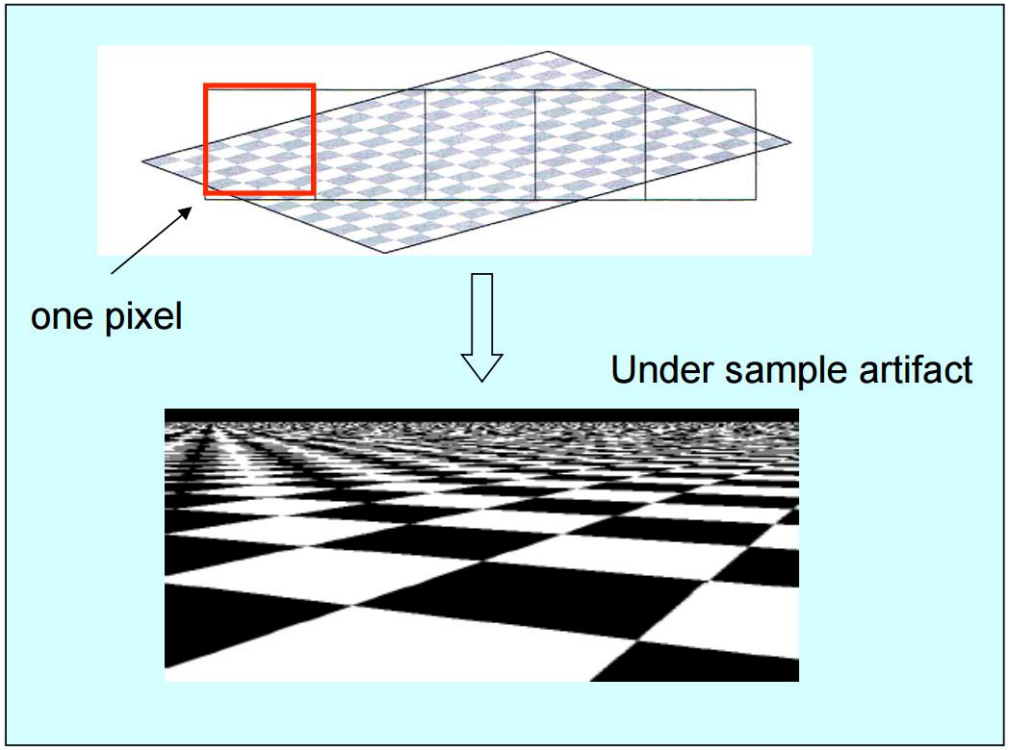
(Tomb Raider, Eidos Interactive, 1996)

Minification Aliasing

Visible flicker when camera moves

Minification aliasing – trees are higher resolution than on-screen pixels (*Combat Mission, Battlefront.com, 1999)*

Aliasing



- One pixel corresponds to many texels.
- This causes aliasing due to under sampling.

Aliasing

Sufficiently sampled, no aliasing

Insufficiently sampled, aliasing

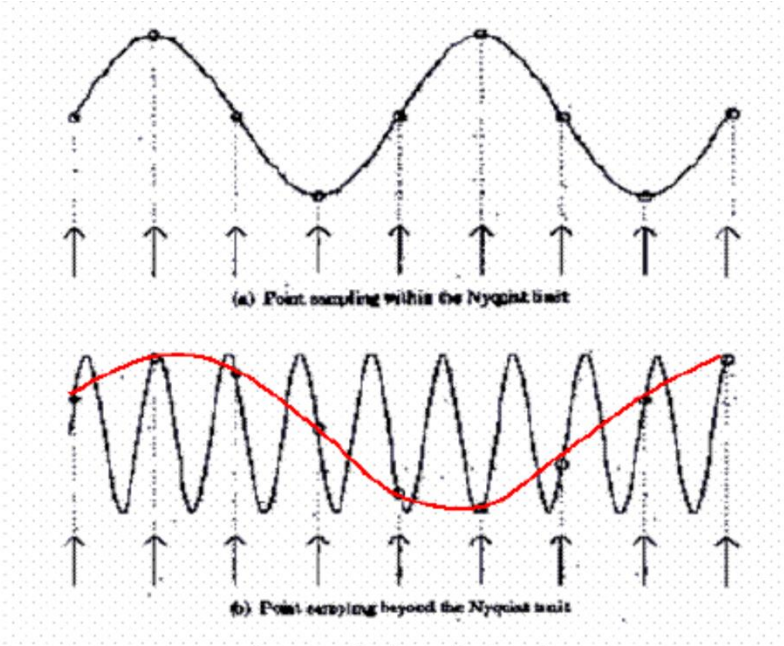


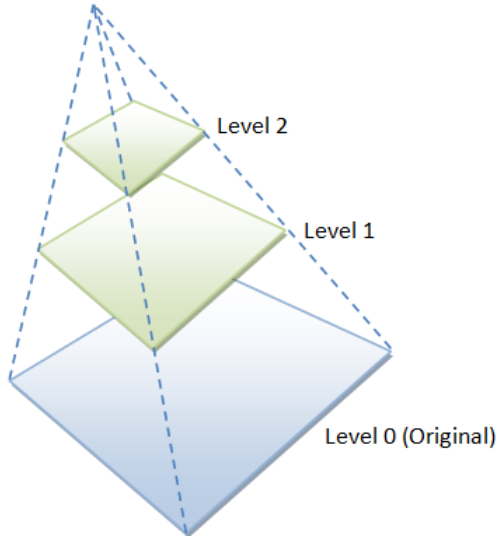
Image: Robert L. Cook

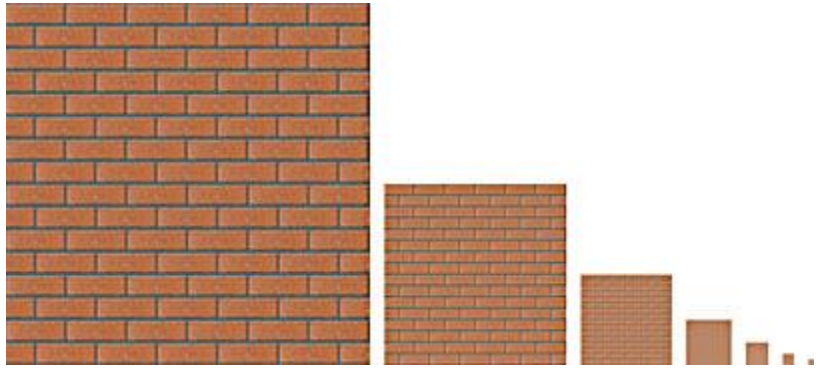
High frequencies in the input data can appear as lower frequencies in the sampled signal

- Anti-aliasing: Either increase sampling rate or reduce the texture frequency by texture filtering

MIP maps

- Practical solution to aliasing.
- Pre-calculated filtered textures at multiple resolutions.
- Reduce resolution by twice iteratively.
- Supported by modern hardware and APIs

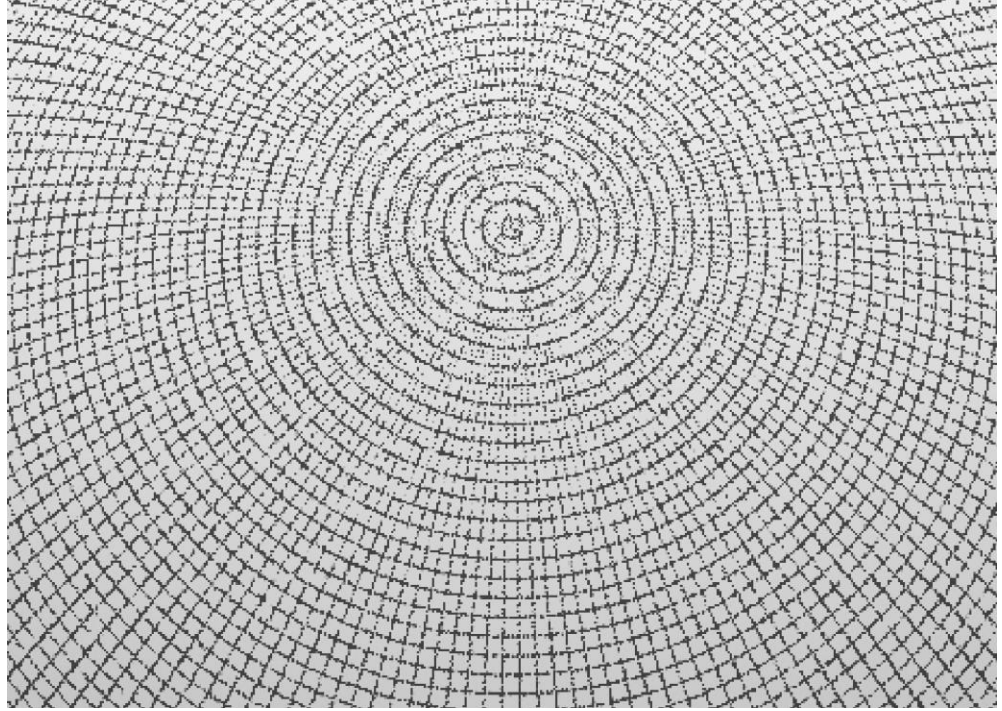


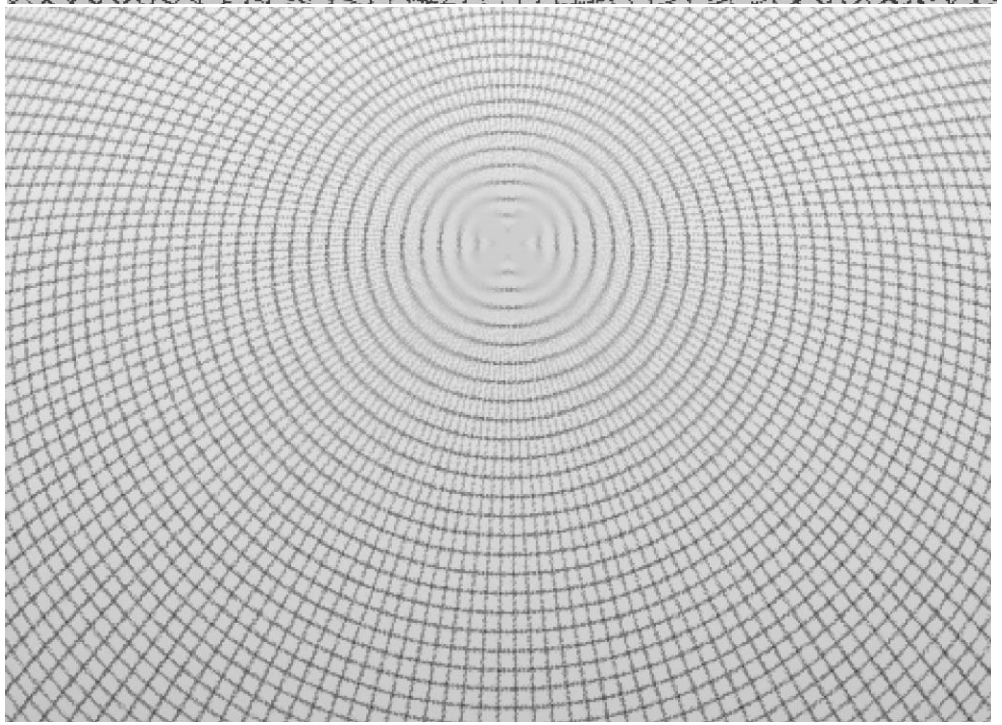


MIP – *multum in parvo*
“many things in a small place”

Rendering with MIP maps

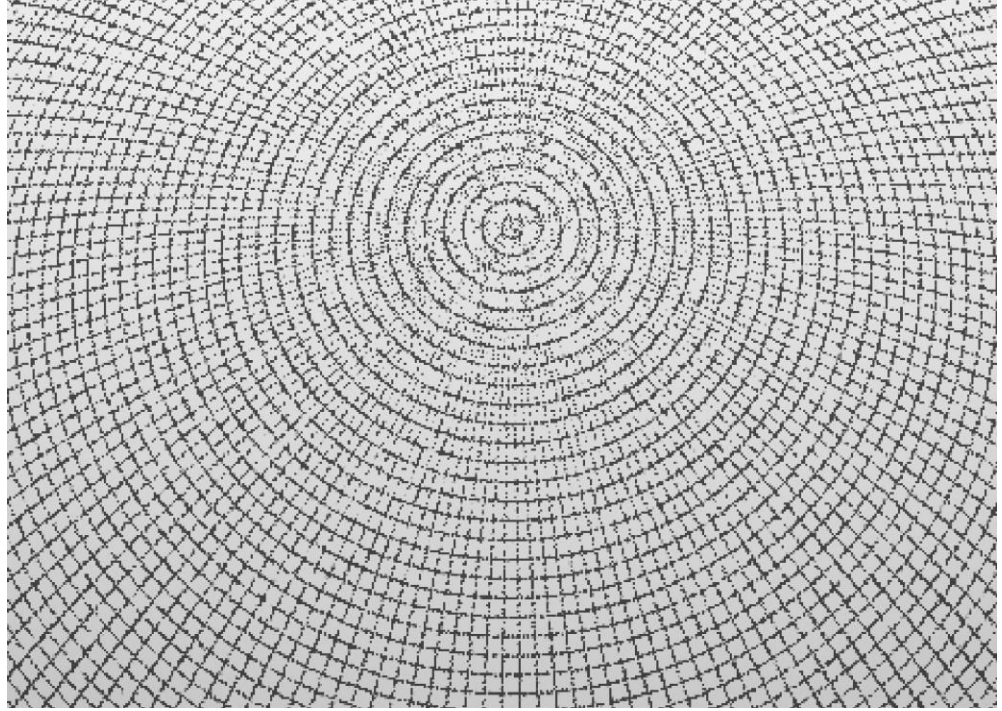
- Estimate pixel size in texture space.
- Choose two appropriate MIP map levels based on the pixel size.
- **Trilinear** interpolation: **bilinear** interpolation in both MIP maps, and then **linearly** interpolate the results.

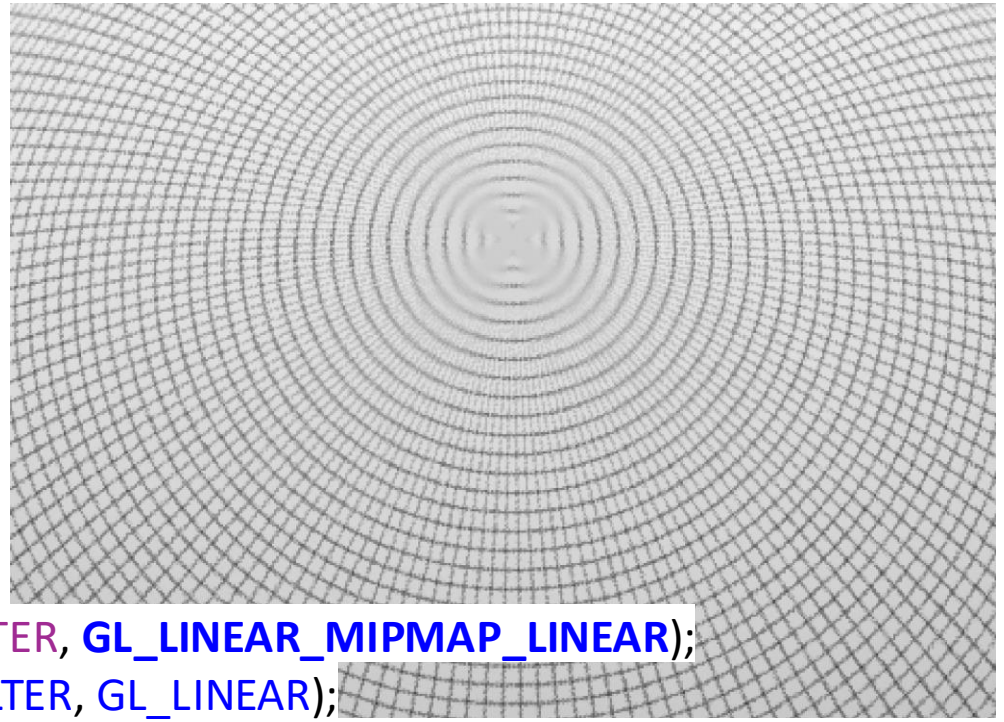




Rendering with MIP maps

- Estimate pixel size in texture space.
- Choose two appropriate MIP map levels based on the pixel size.
- **Trilinear** interpolation: **bilinear** interpolation in both MIP maps, and then **linearly** interpolate the results.





```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
```

Texturing with MIP maps in OpenGL

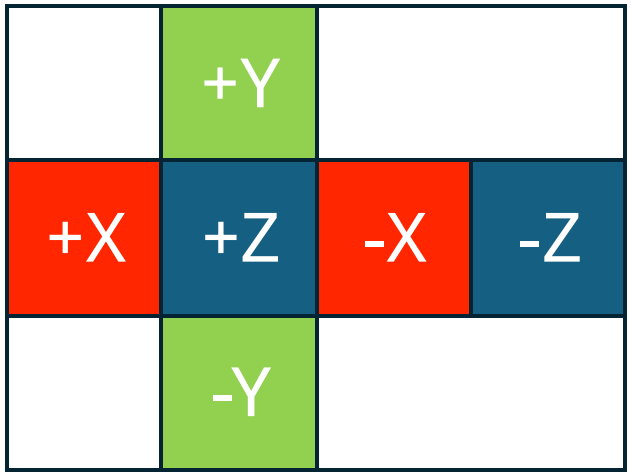
```
GLuint texture;
glGenTextures(1, &texture);
glBindTexture(GL_TEXTURE_2D, texture);

// Texture parameters
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);

// Load img (image buffer) into texture and generate MIP maps
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, w, h, 0, GL_RGB, GL_UNSIGNED_BYTE, img);
glGenerateMipmap(GL_TEXTURE_2D);
```

More on possible values for GL_TEXTURE_MIN_FILTER:

```
GL_NEAREST, GL_LINEAR,
GL_NEAREST_MIPMAP_NEAREST, GL_LINEAR_MIPMAP_NEAREST,
GL_NEAREST_MIPMAP_LINEAR, GL_LINEAR_MIPMAP_LINEAR
```

https://registry.khronos.org/OpenGL-Refpages/gl4/html/glTexParameter.xhtml

Environment Modeling



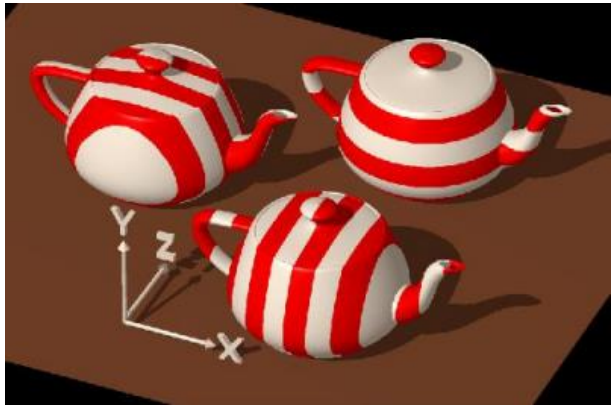
- Use textures to model surrounding world
- Skybox (also known as a cubemap)

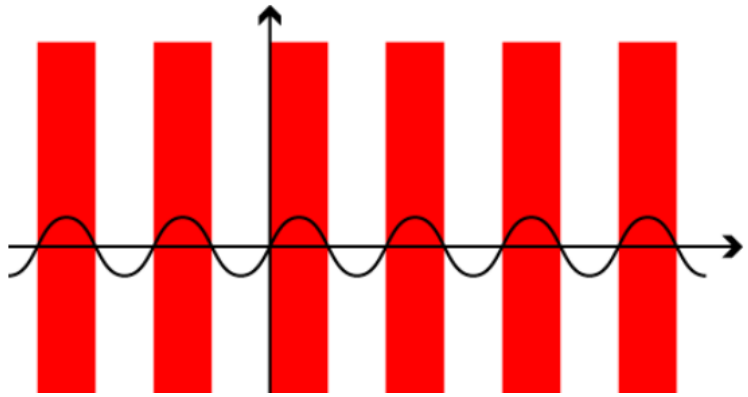
Procedural textures

- We can also use a mathematical procedure to create a 3D texture

- Example: use the coordinates of each point in our 3D model to calculate the appropriate colour value, e.g., stripes along X-axis

```
stripe( x_p, y_p, z_p )
{
  if ( sin x_p > 0 )
    return color0;
  else
    return color1;
  }
}
```





Extra Reading

- Multi-texturing in OpenGL
- Frame-buffer object (FBO) in OpenGL
- Environment Mapping: http://www.pauldebevec.com/Probes/
- Real-time Rendering, 3rd Edition, Akenine-Moller
- Physically Based Rendering: From Theory to Implementation, Matt Pharr, Wenzel Jakob, and Greg Humphreys. Chapter 10.