Fall 2014

CSCI 420: **Computer Graphics**

6.1 Texture Mapping

Hao Li

(lecturer for 9/28: Justin Solomon)

http://cs420.hao-li.com

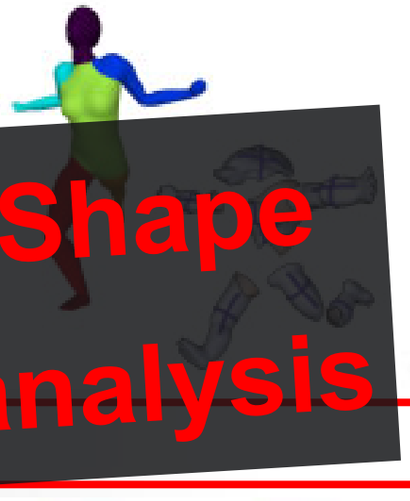
Quick Self-Introduction

Justin Solomon

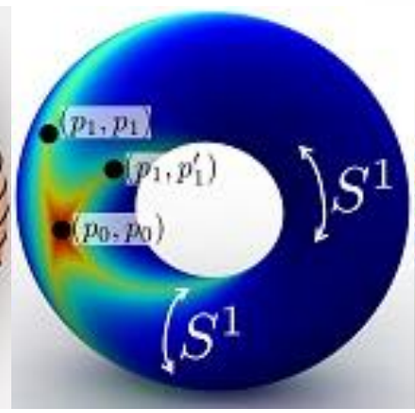
http://www.stanford.edu/~justso1

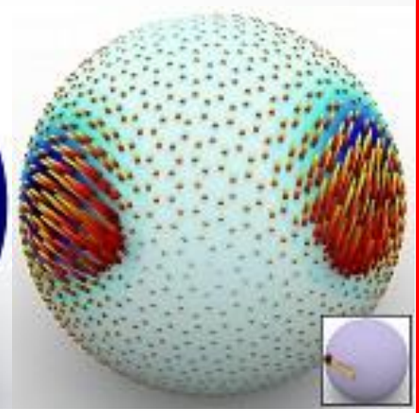
justin.solomon@stanford.edu

What I Work On

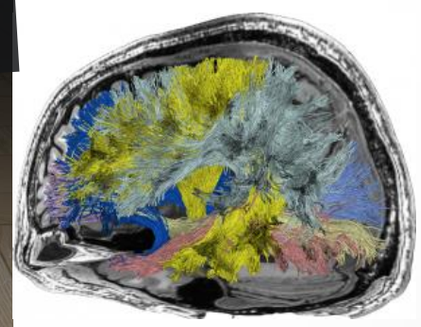


Shape analysis





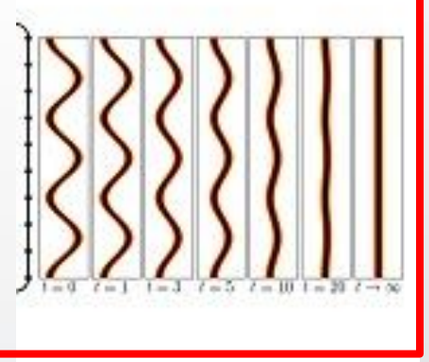
Applications



Modeling

Imaging

Learning

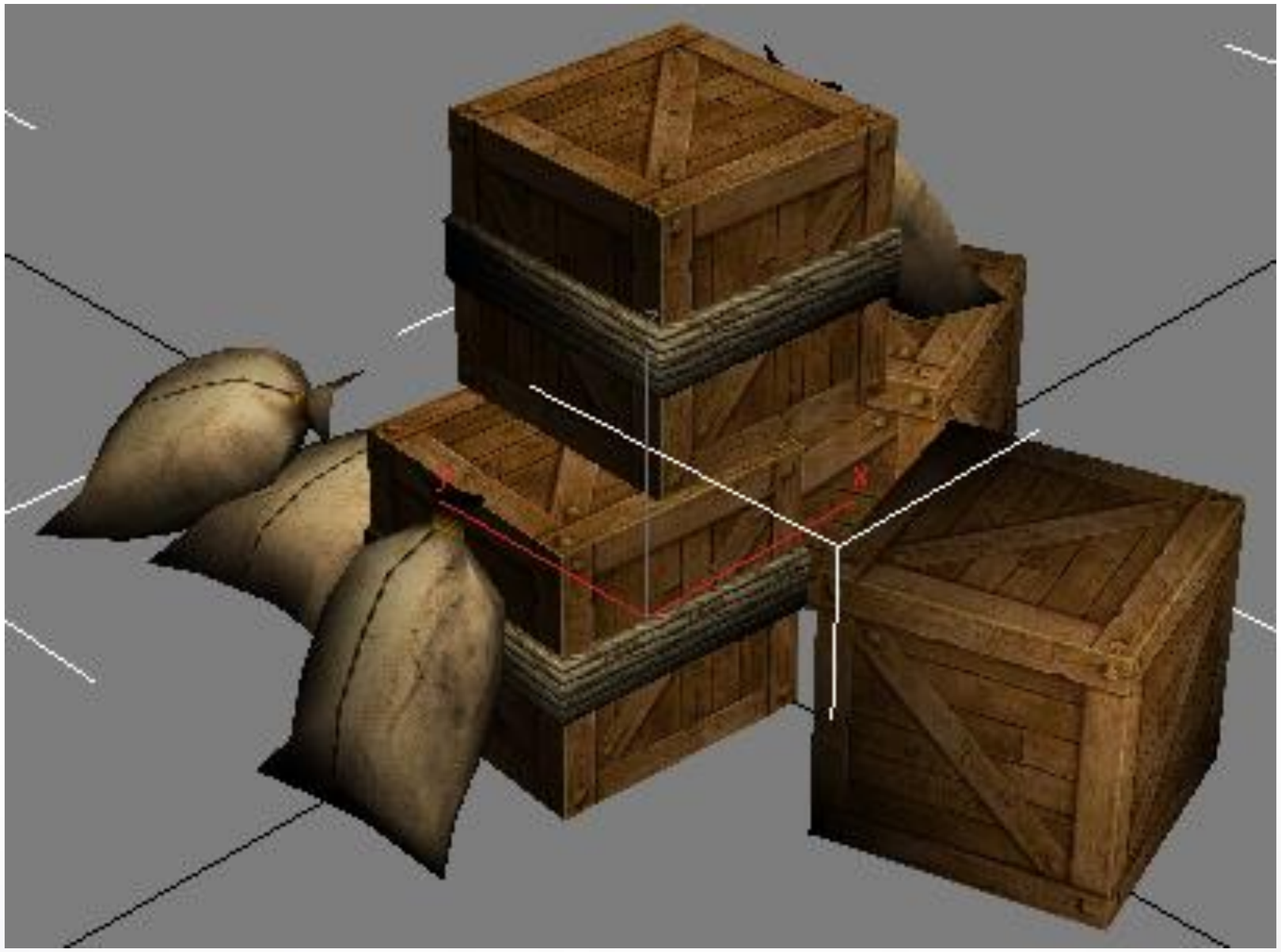


Outline

- Introduction
- Texture mapping in OpenGL
- Filtering and Mipmaps
- Example
- Non-color texture maps

How Do You Add Detail to a Cube?

http://dev.ryzom.com/projects/ryzom/wiki/ImportingMaxAssets



Six sides ⟹ six colors?!

Two Ideas for Adding Detail

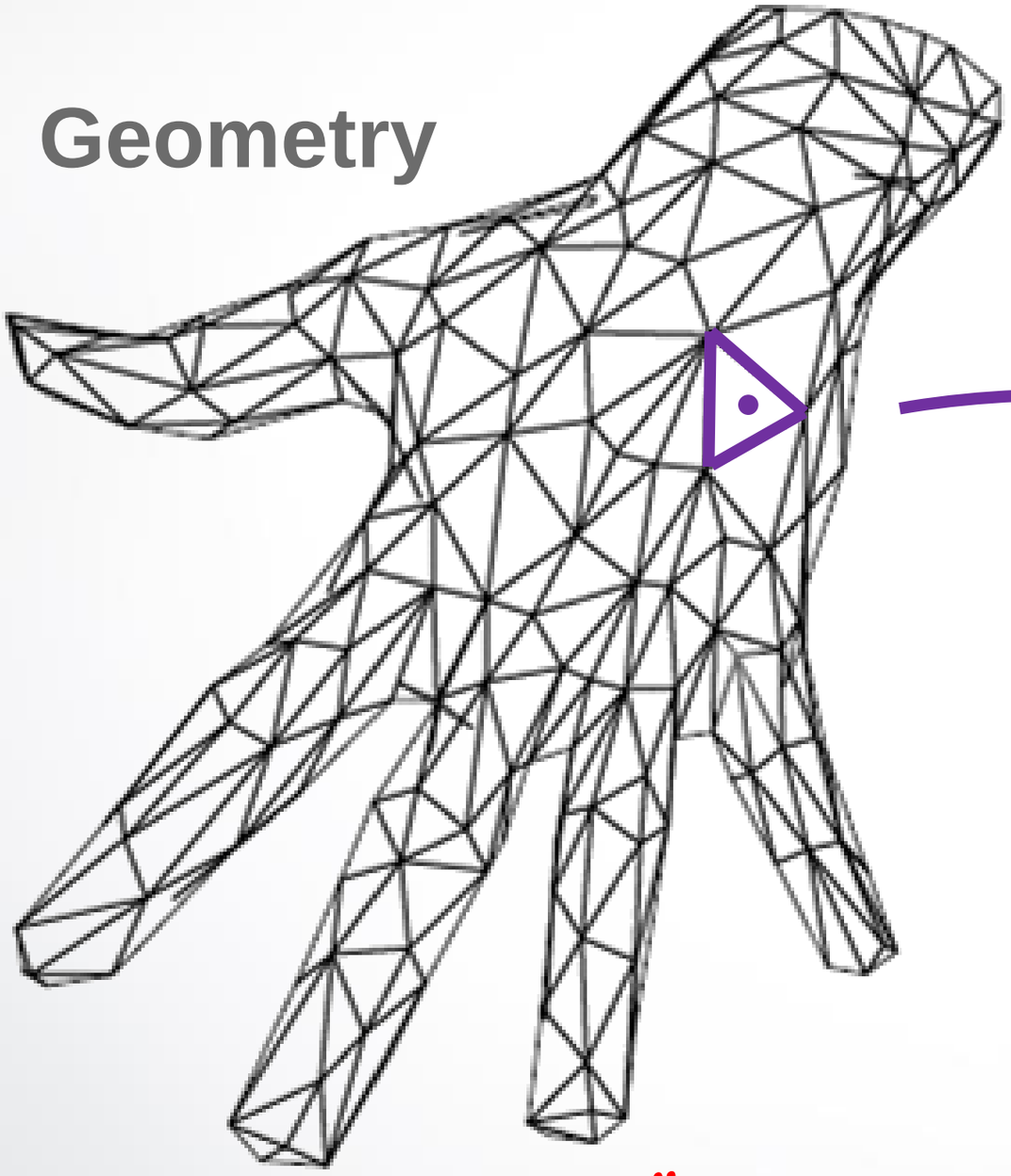
1. More polygons
   - Slow and hard to edit!
2. Map a texture from an image
   - Image size does not affect complexity
   - Built into hardware

Trompe L'Oeil ("Deceive the Eye")

- Extra 3D structure is **painted**
- Similar idea for texture mapping:
  Replace intricate geometry with an image!

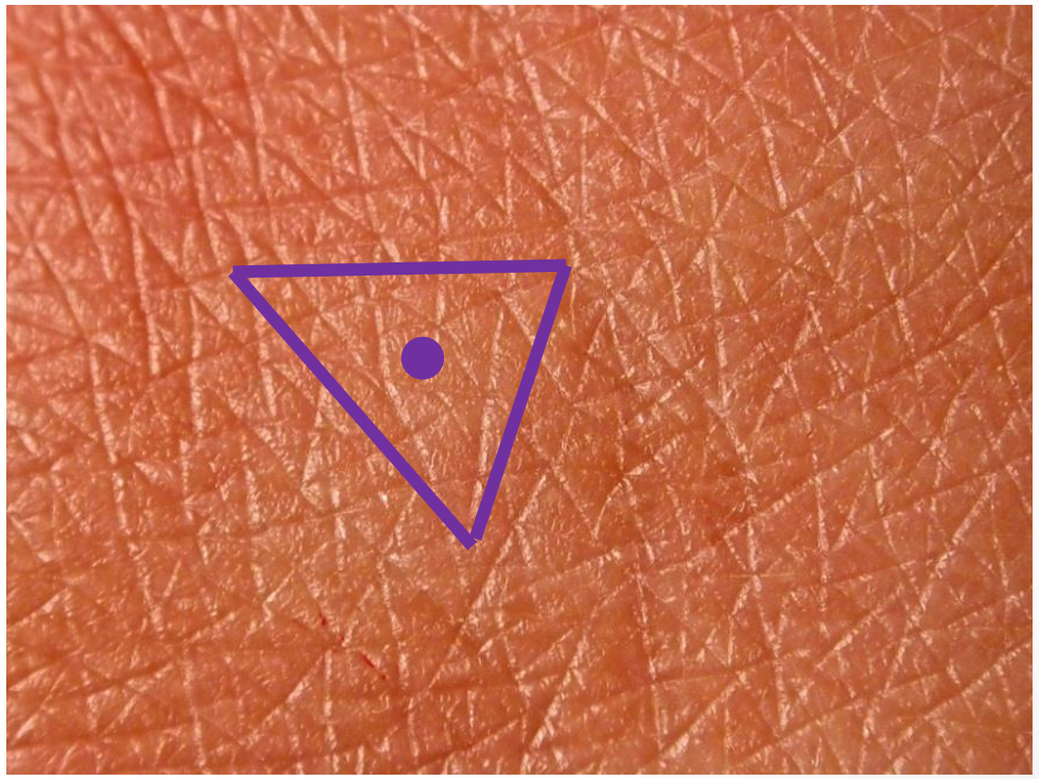
Texture Maps

Geometry



"3D coordinates"

Image



"Texture coordinates"

http://raweb.inria.fr/rapportsactivite/RA2006/iparla/uid33.html
http://runfatgirl.files.wordpress.com/2008/05/bigstockphoto_skin_texture_108750.jpg

How To Store a Texture

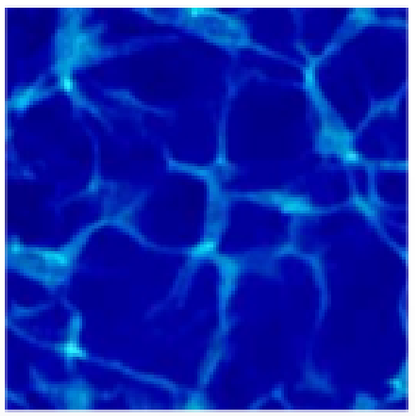
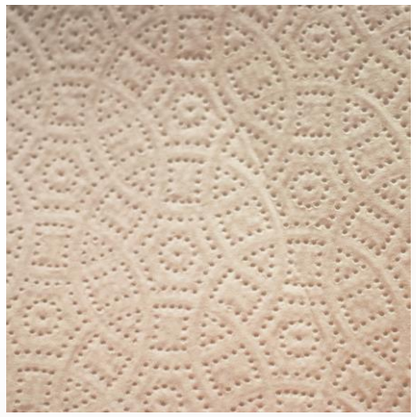
Bitmap image

Potential sources:

- Load using **image library**

- **Create** within the program



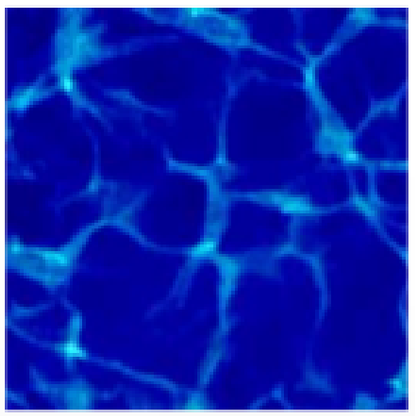
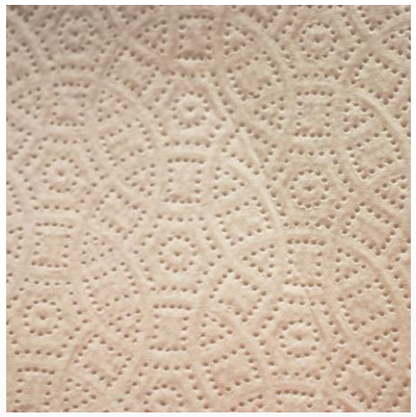

Bitmap image

Vocabulary:

- **Texels:** Pixels of texture

- **Texel coordinates:** Coordinates (s,t) scaled to [0,1]




In Code

- 2D array:

```
unsigned char texture[height][width][4]
```

- Unrolled into 1D array:

```
unsigned char texture[4*height*width]
```

In Code

- 2D array:

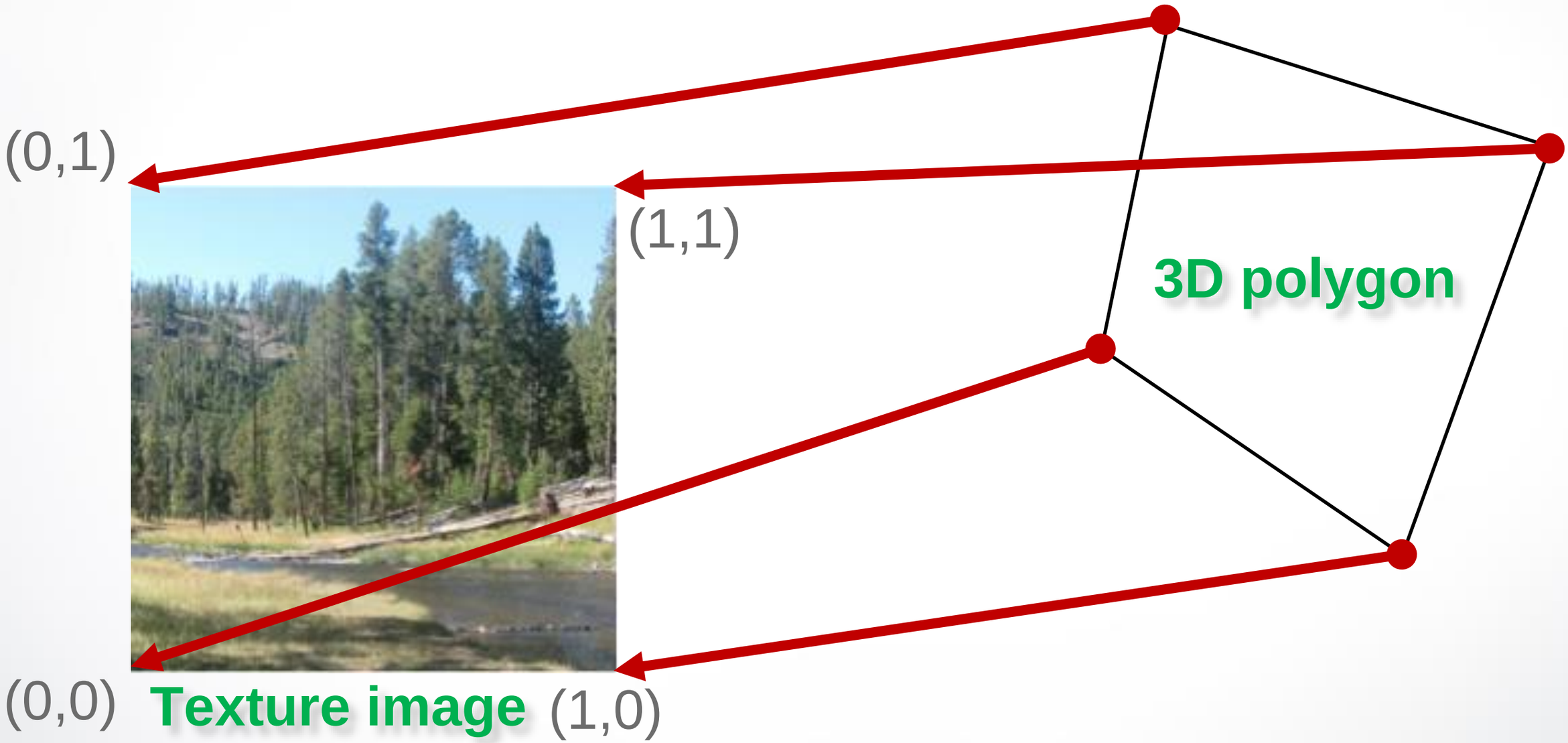
```
unsigned char texture[height][width][4]
```

Why?!

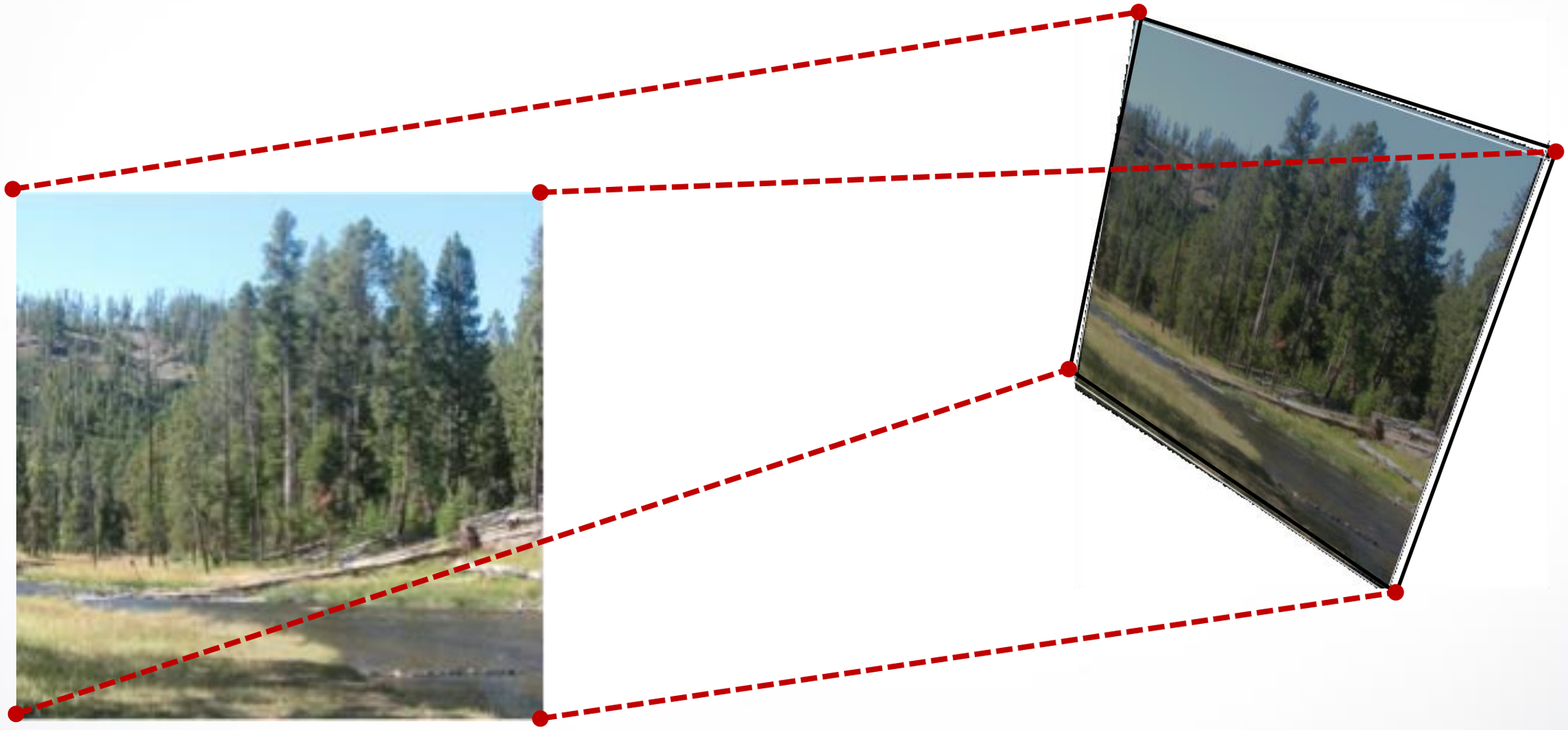
- Unrolled into 1D array:

```
unsigned char texture[4*height*width]
```

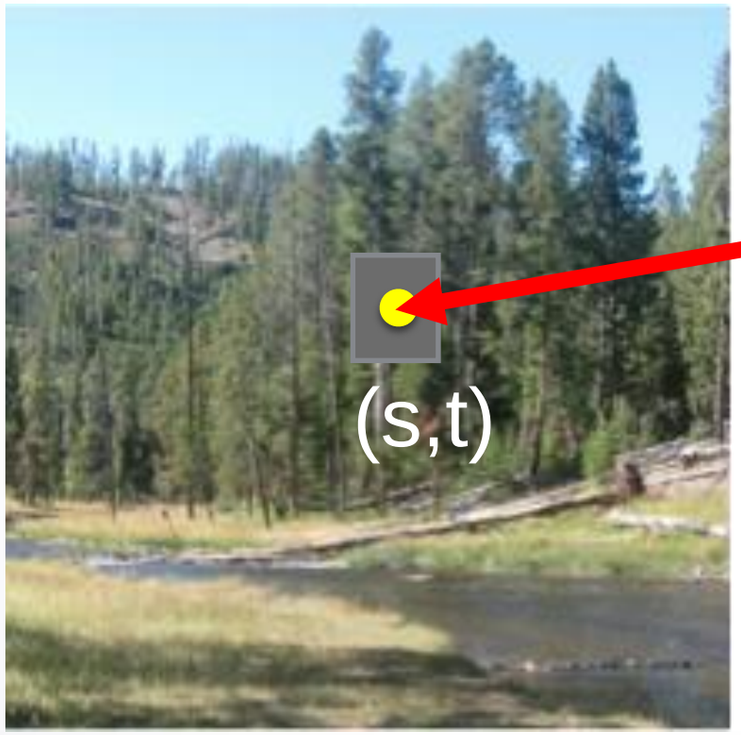
Texture Map



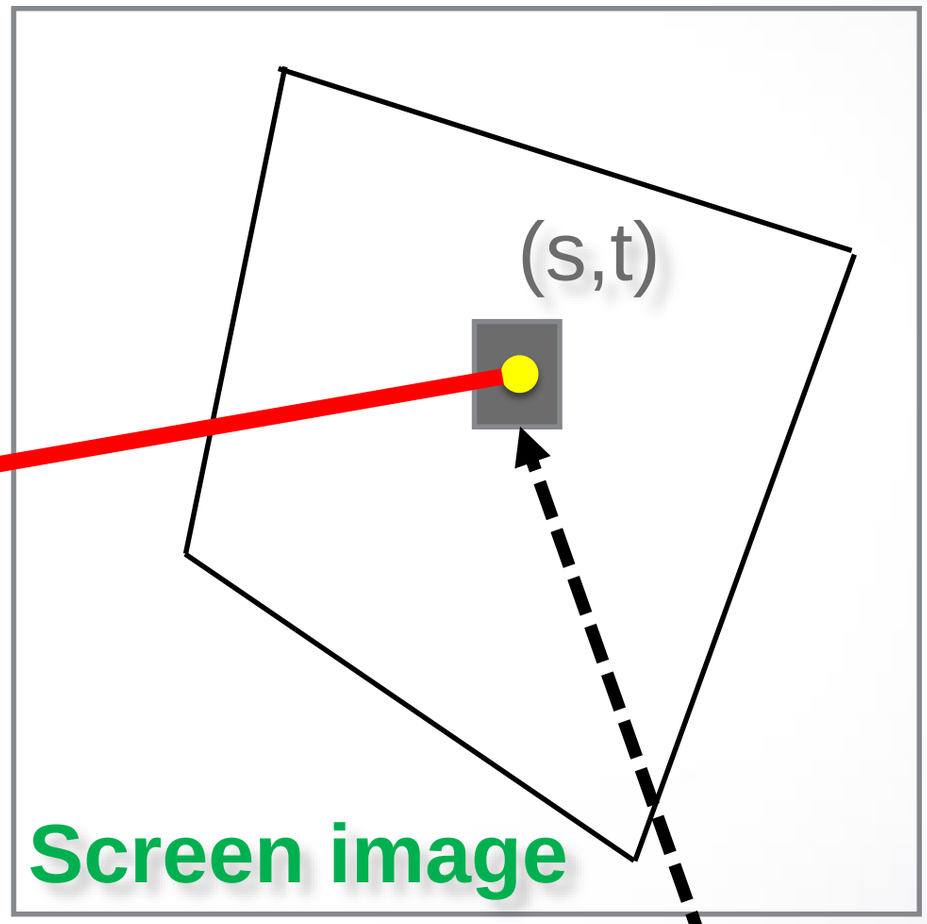
Texture Map



Texture Lookup



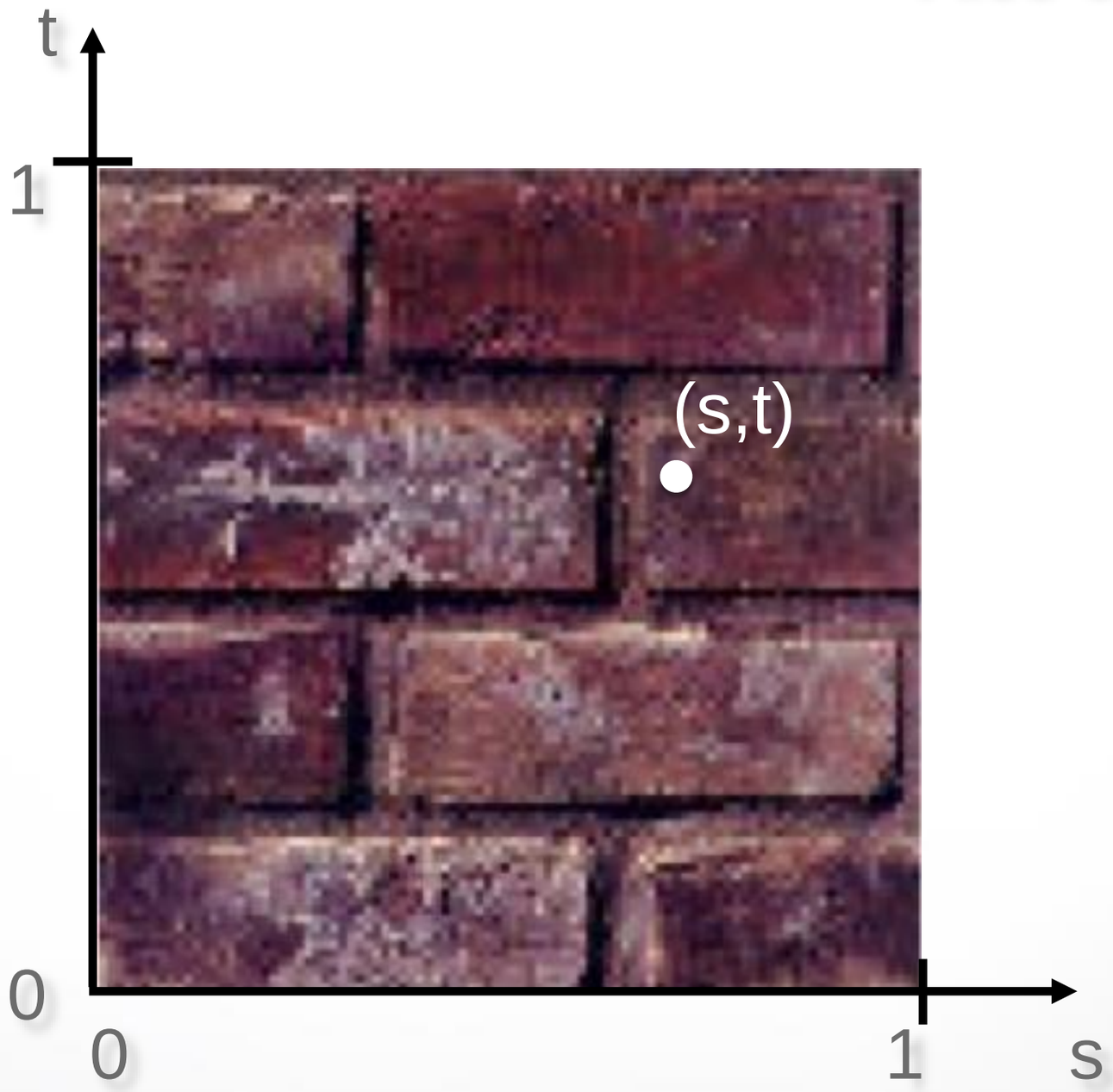
Texture image



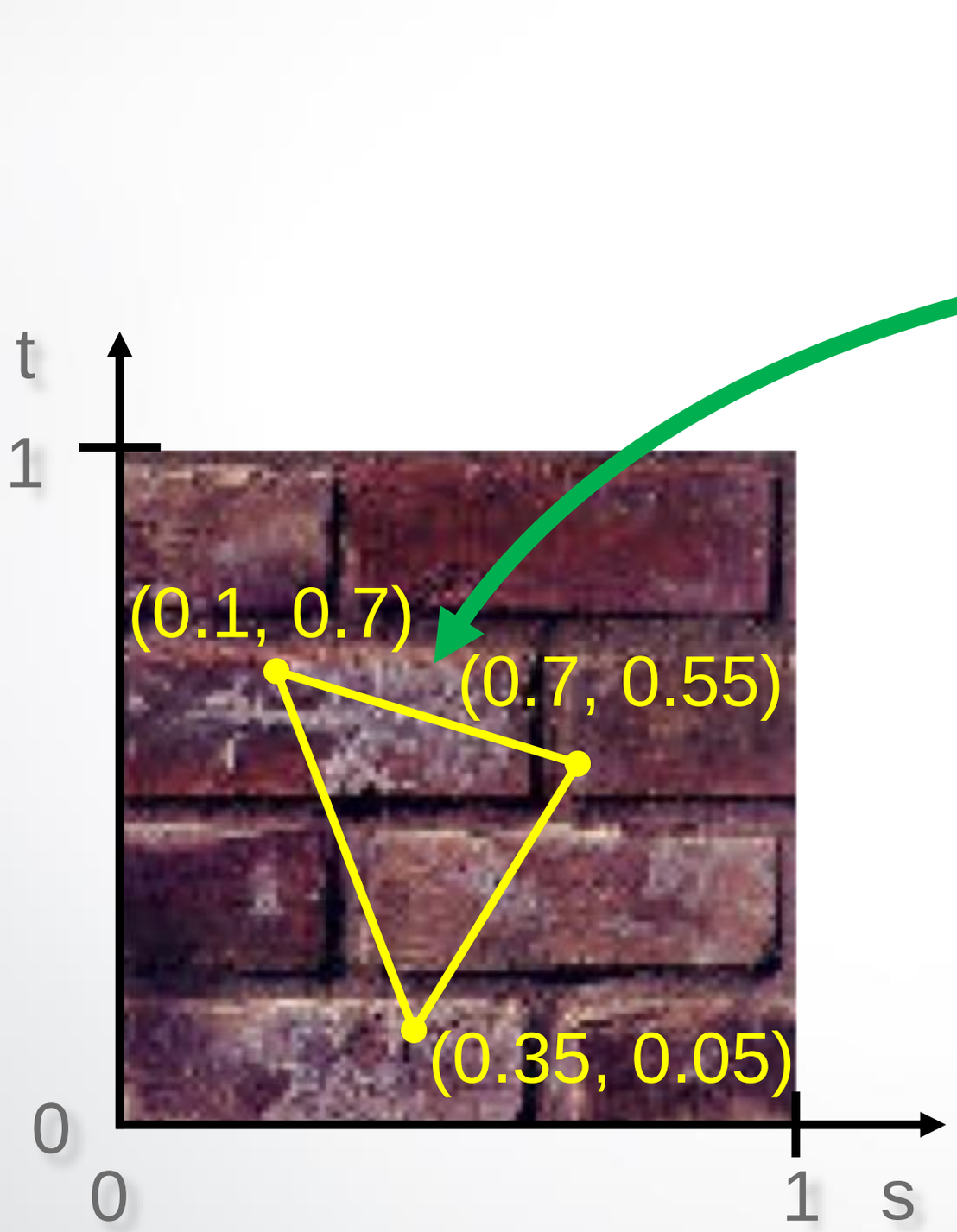
For each pixel, look into the texture to obtain color

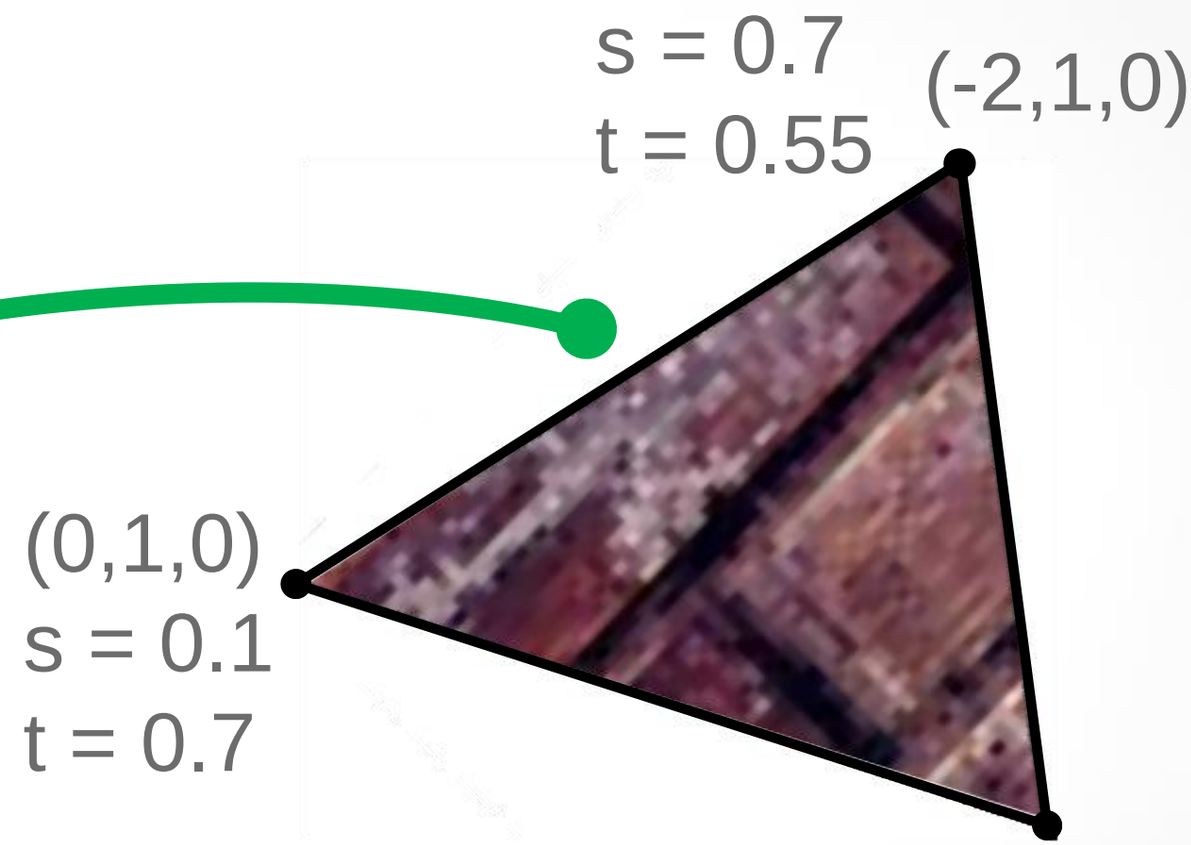
The “st” Coordinate System

Also called **uv space**



Texture Mapping: Key Slide

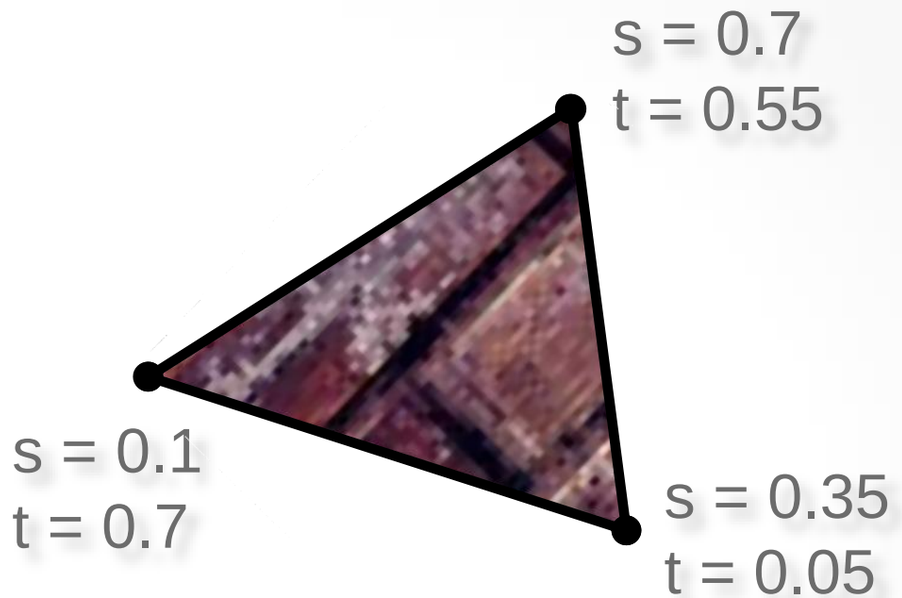




Specifying Texture Coordinates in OpenGL

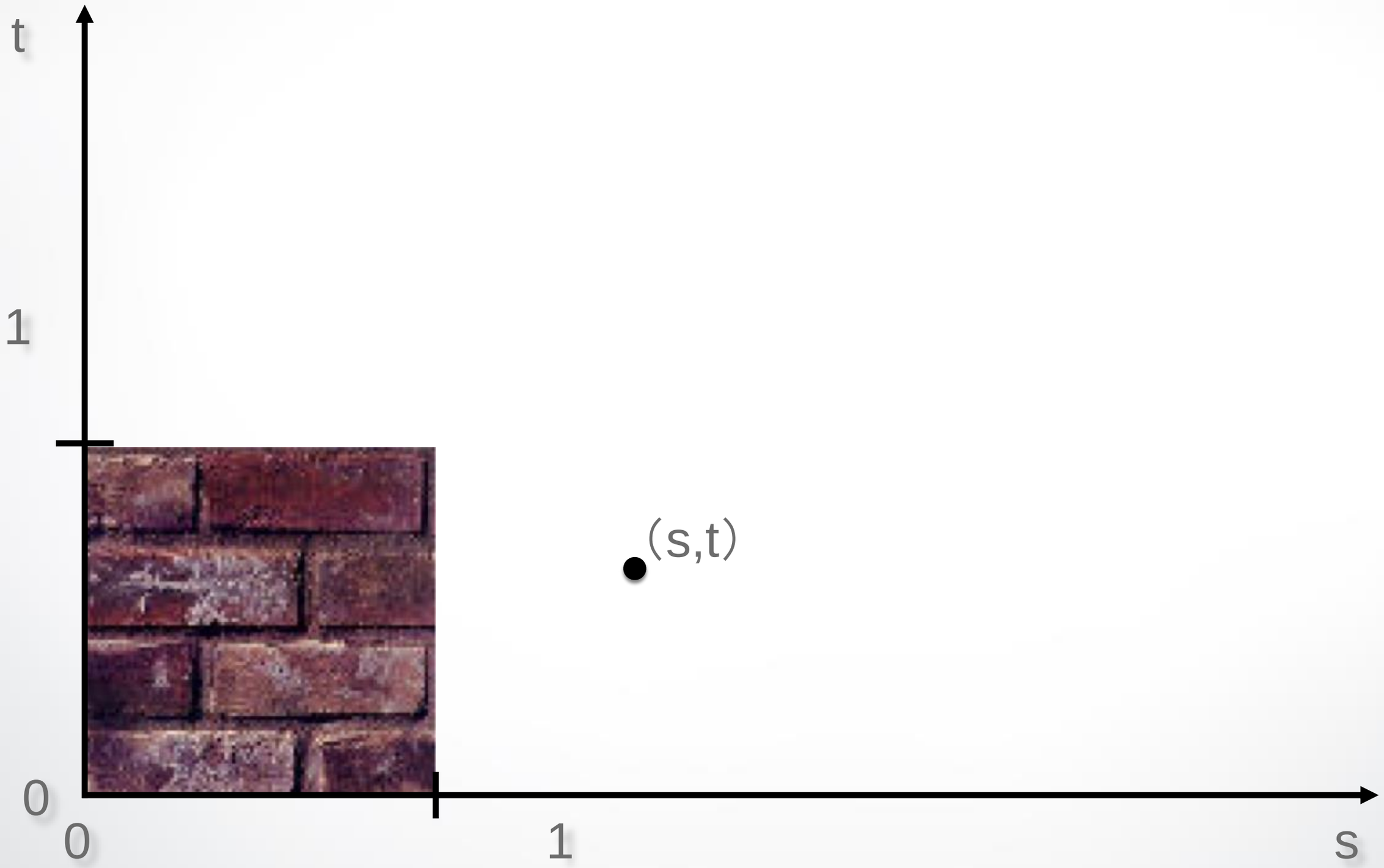
- Use **`glTexCoord2f(s,t)`**
- State machine: Texture coordinates

  remain valid until you change them
- Example (from previous slide) :



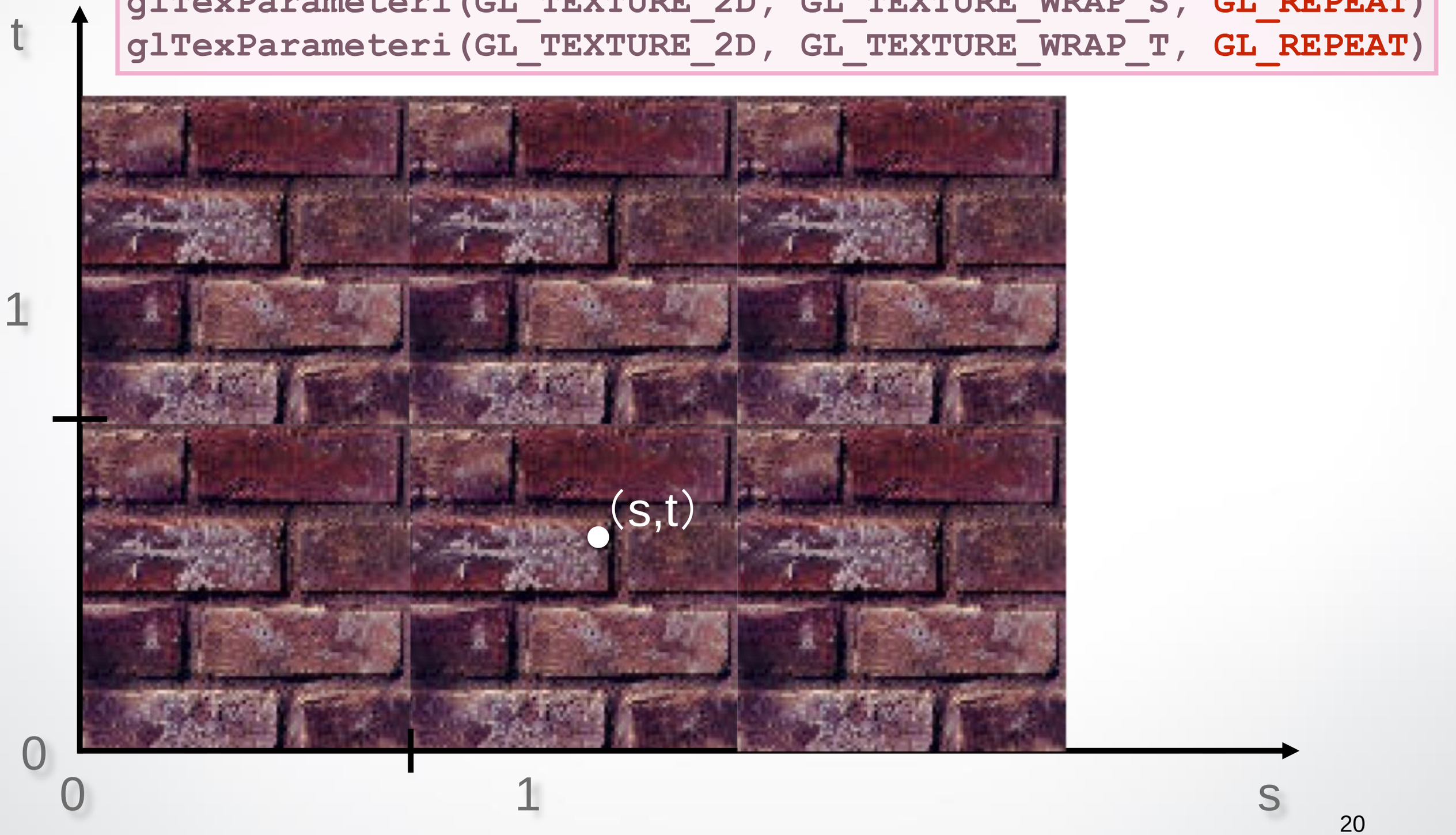

```
glEnable(GL_TEXTURE_2D); // turn texture mapping on
glBegin(GL_TRIANGLES);
   glTexCoord2f(0.35,0.05); glVertex3f(2.0,-1.0,0.0);
   glTexCoord2f(0.7,0.55); glVertex3f(-2.0,1.0,0.0);
   glTexCoord2f(0.1,0.7); glVertex3f(0.0,1.0,0.0);
glEnd();
glDisable(GL_TEXTURE_2D); // turn texture mapping off
```

What if texture coordinates are outside of [0,1] ?



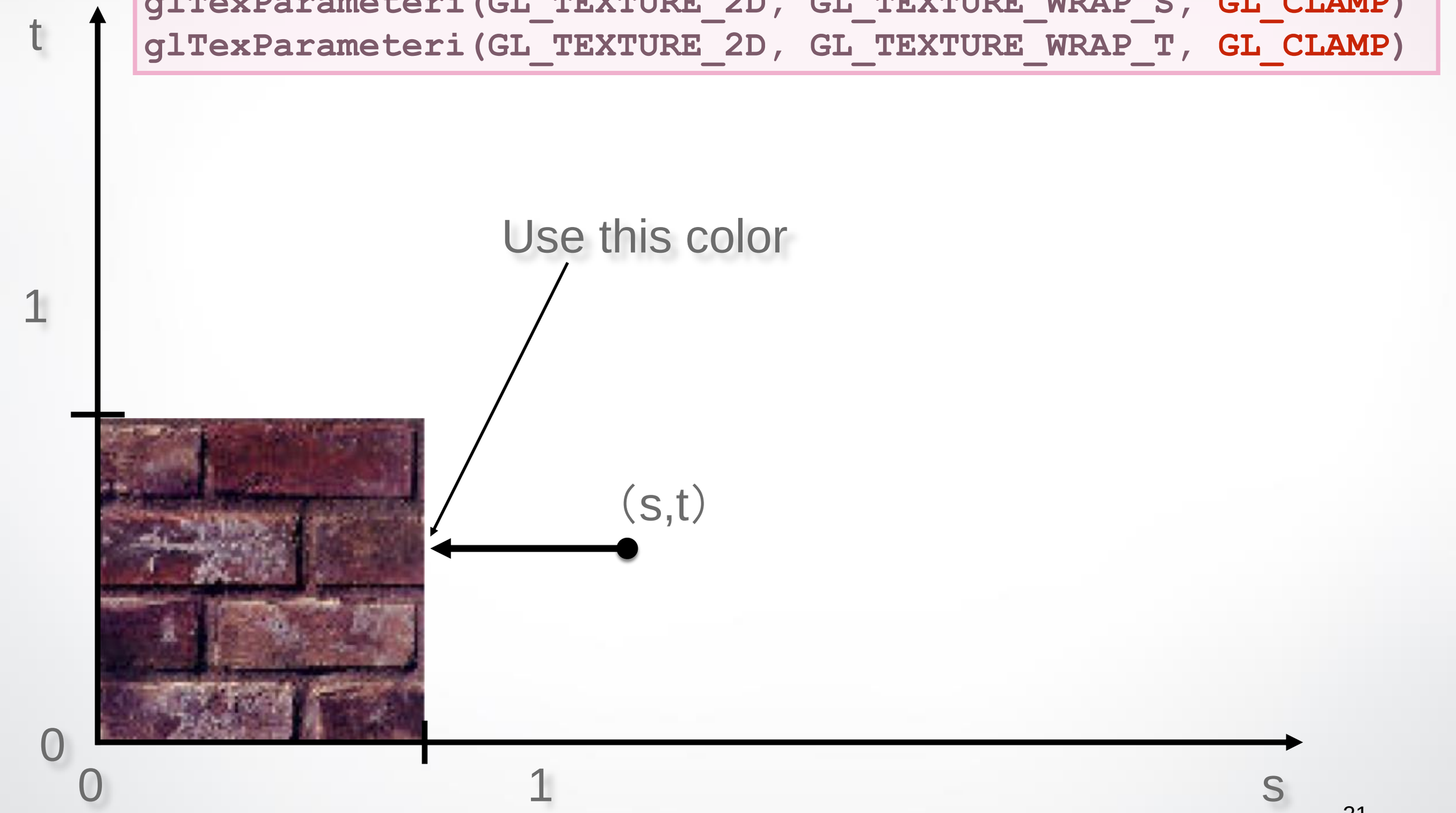
Solution 1: Repeat texture

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

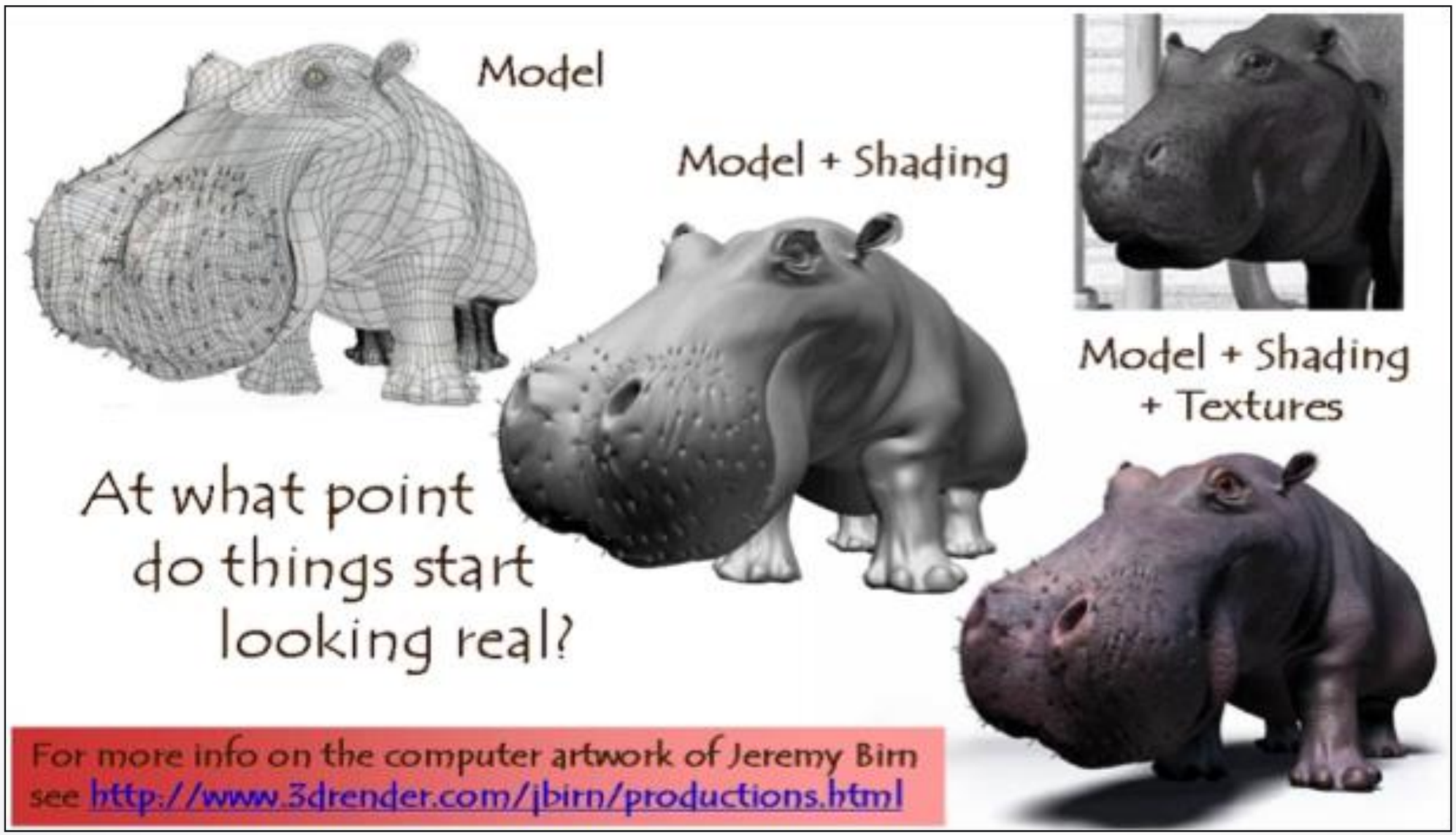


Solution 2: Clamp to [0,1]

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP)
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP)
```



Combining texture mapping and shading



Combining Texture Mapping and Shading

- Combine texture and OpenGL Phong shading
- **`GL_MODULATE`**: Multiply texture and Phong
- **`GL_BLEND`**: Linear combination of texture and Phong
- **`GL_REPLACE`**: Use texture color only (ignore Phong)

- Example:

```
glTexEnvf(GL_TEXTURE_ENV,
  GL_TEXTURE_ENV_MODE, GL_REPLACE);
```

Outline

- Introduction
- Texture mapping in OpenGL
- Filtering and Mipmaps
- Example
- Non-color texture maps

Texture Mapping in OpenGL

- **Initialization:**
  1. **Read texture image** from file into an array,
     or generate texture using your program
  2. Specify texture mapping **parameters:** Wrapping, filtering, etc.
  3. **Initialize** and **activate** the texture

- **In `display()`:**
  1. Enable OpenGL texture mapping
  2. Draw objects: Assign texture coordinates to vertices
  3. Disable OpenGL texture mapping

Initializing the Texture

- For each texture image call **`glTexImage2D`**
  [once during initialization]

- The dimensions of texture images **must be powers of 2**
  - if not, rescale image or pad with zero
  - or can use OpenGL extensions

- Can load textures dynamically if GPU memory is scarce

glTexImage2D

```
glTexImage2D(GL_TEXTURE_2D, level, internalFormat, width, height,
             border, format, type, data)
```

- **GL_TEXTURE_2D**: specifies that it is a 2D texture
- **level**: used for specifying levels of detail for mipmapping (default:0)
- **internalFormat**: Determines how texture is stored internally (often **GL_RGB** or **GL_RGBA**)
- **width**, **height**: The size of the texture must be powers of 2
- **border**: Often set to 0)
- **format**, **type**
  - Specifies what the input data is (**GL_RGB**, **GL_RGBA**, …)
  - Specifies the input data type (**GL_UNSIGNED_BYTE**, **GL_BYTE**, …)
  - Regardless of **format** and **type**, OpenGL converts the data to **internalFormat**
- **data**: pointer to the image buffer

Enabling and Disabling Texture Mode

- Before rendering primitives that are texture-mapped:

```
glEnable(GL_TEXTURE_2D)
glDisable(GL_TEXTURE_2D)
```

- Enable/disable texture mode to **switch** between drawing textured/non-textured polygons

- Changing textures:
  - Only **one texture is active** at any given time (with OpenGL extensions, more than one can be used called ***multitexturing***)
  - Use **`glBindTexture`** to select the active texture

Outline

- Introduction
- Texture mapping in OpenGL
- Filtering and Mipmaps
- Example
- Non-color texture maps

Texture Interpolation

This photo is too small.

Zooming

Consider a black and white image:

Task:

Blow up to poster size (zoom by a factor of 16)

Interpolation

Given: The values of a function f at a few locations.

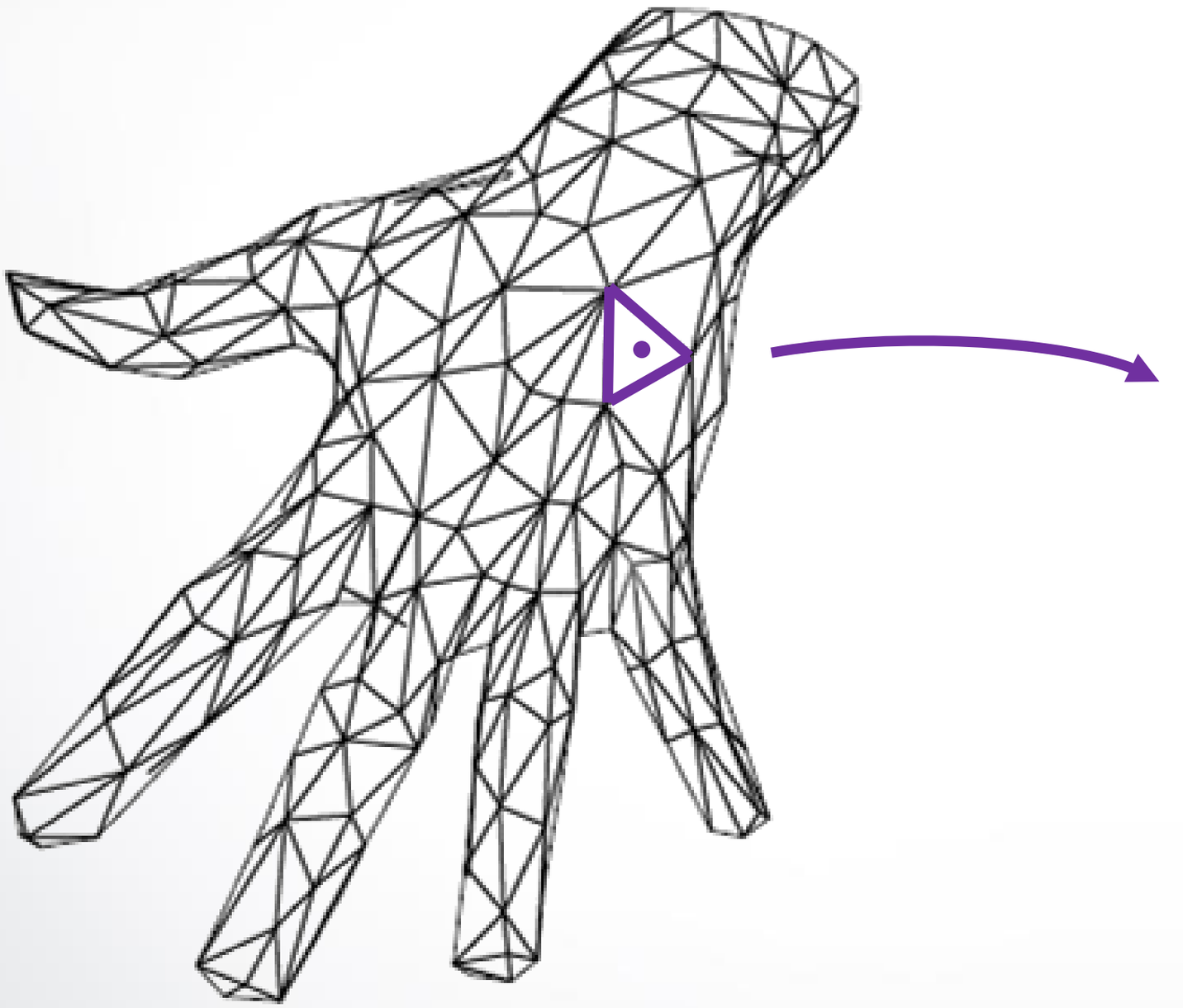
f(1), f(2), f(3), …

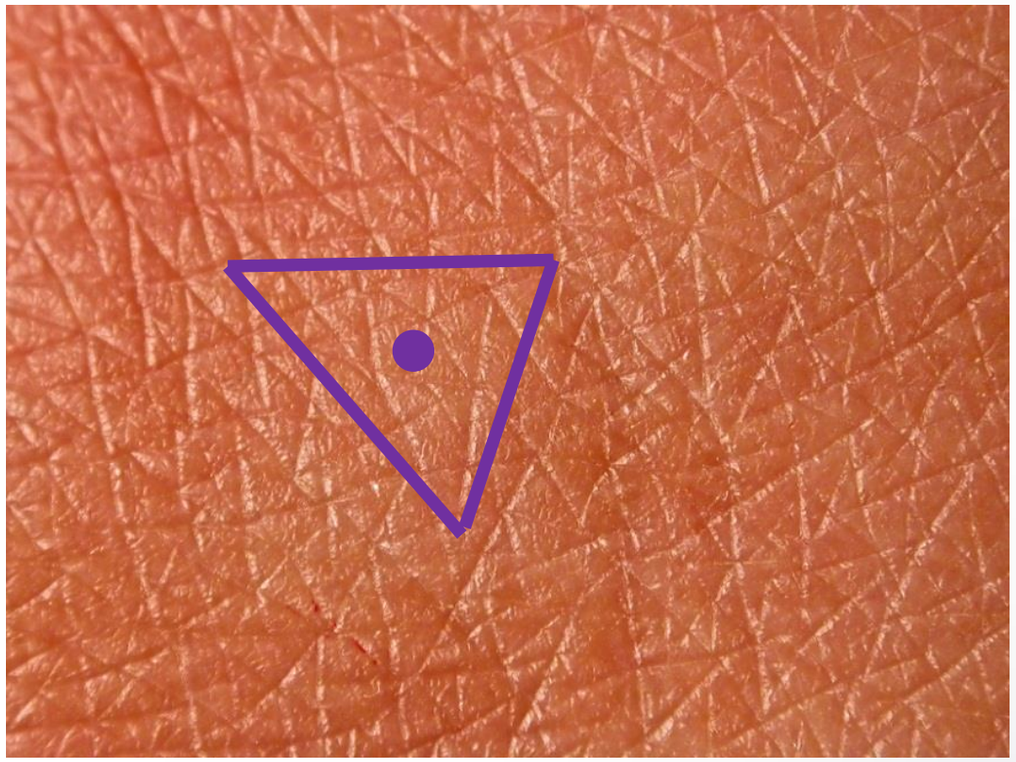
Compute: The values elsewhere

What is f(1.5)?

The challenge: Modeling how the function “should” behave.

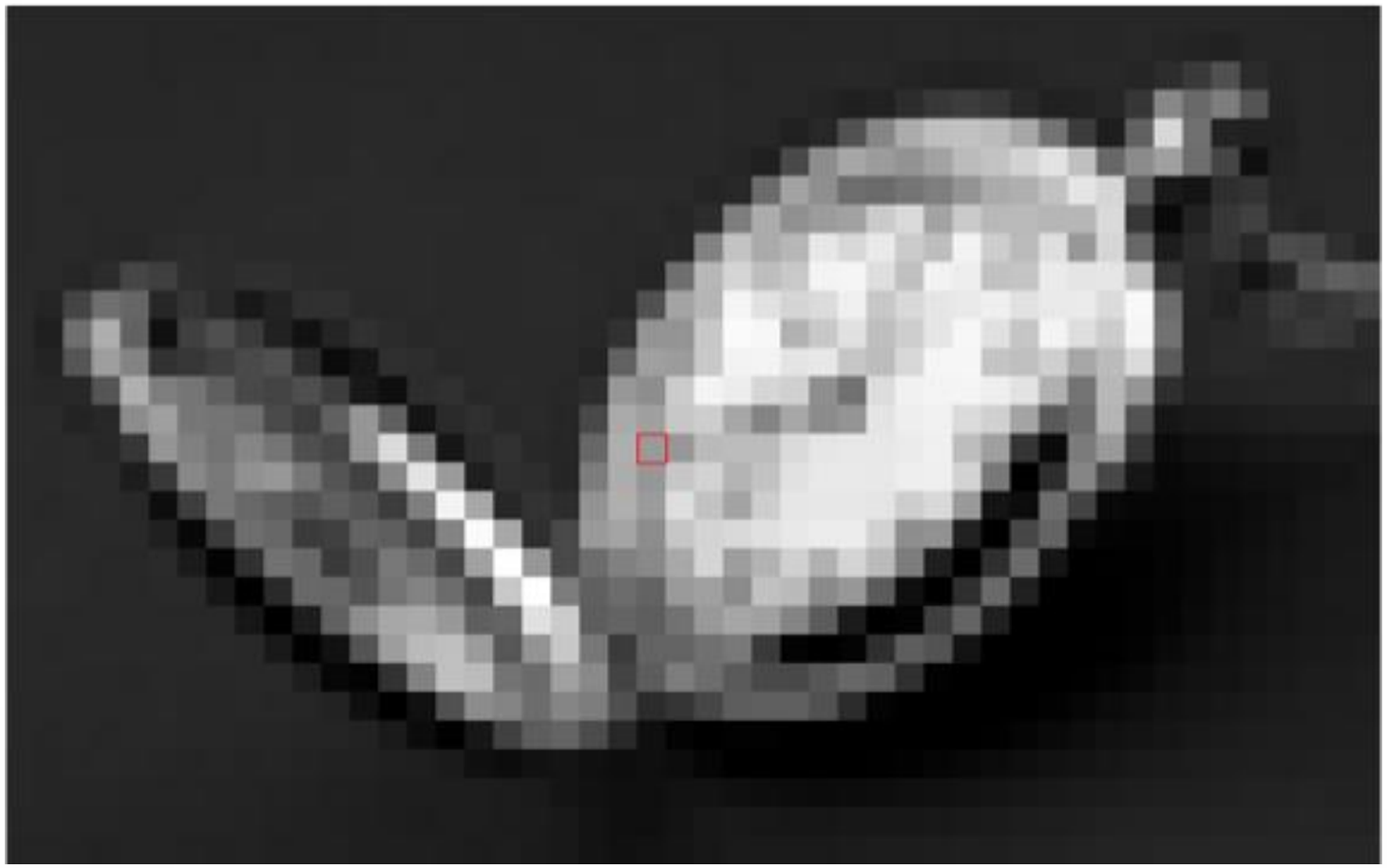
When Does Interpolation Happen?





(s,t) may not land at an integer location!

Nearest Neighbor Interpolation



First try:

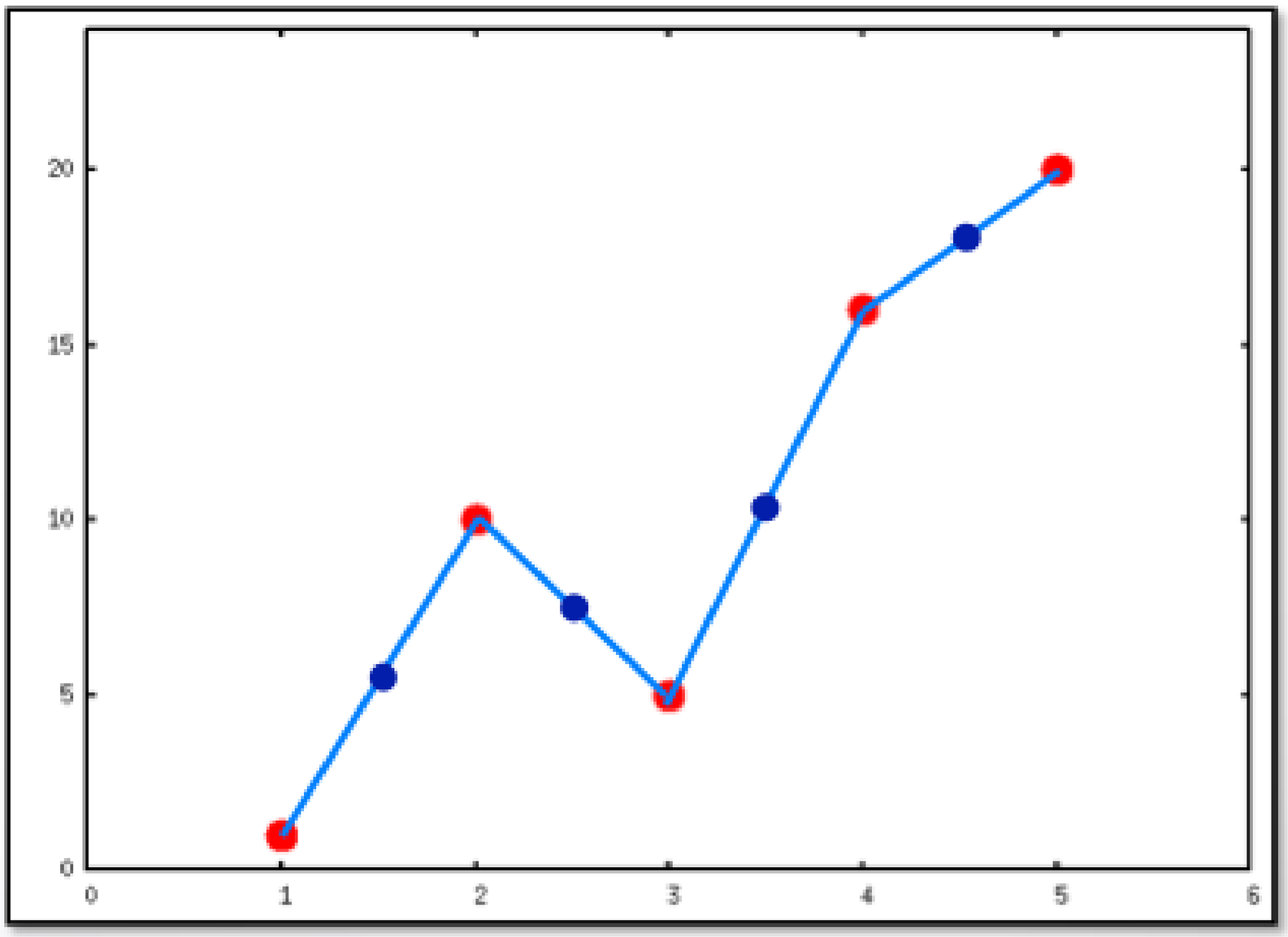
Repeat each row 16 times, then each column 16 times

Nearest Neighbor Interpolation

Discontinuous!

- We need a better way to find the in between values
- For now, we'll consider one **horizontal slice** through the image (one **scan line**)

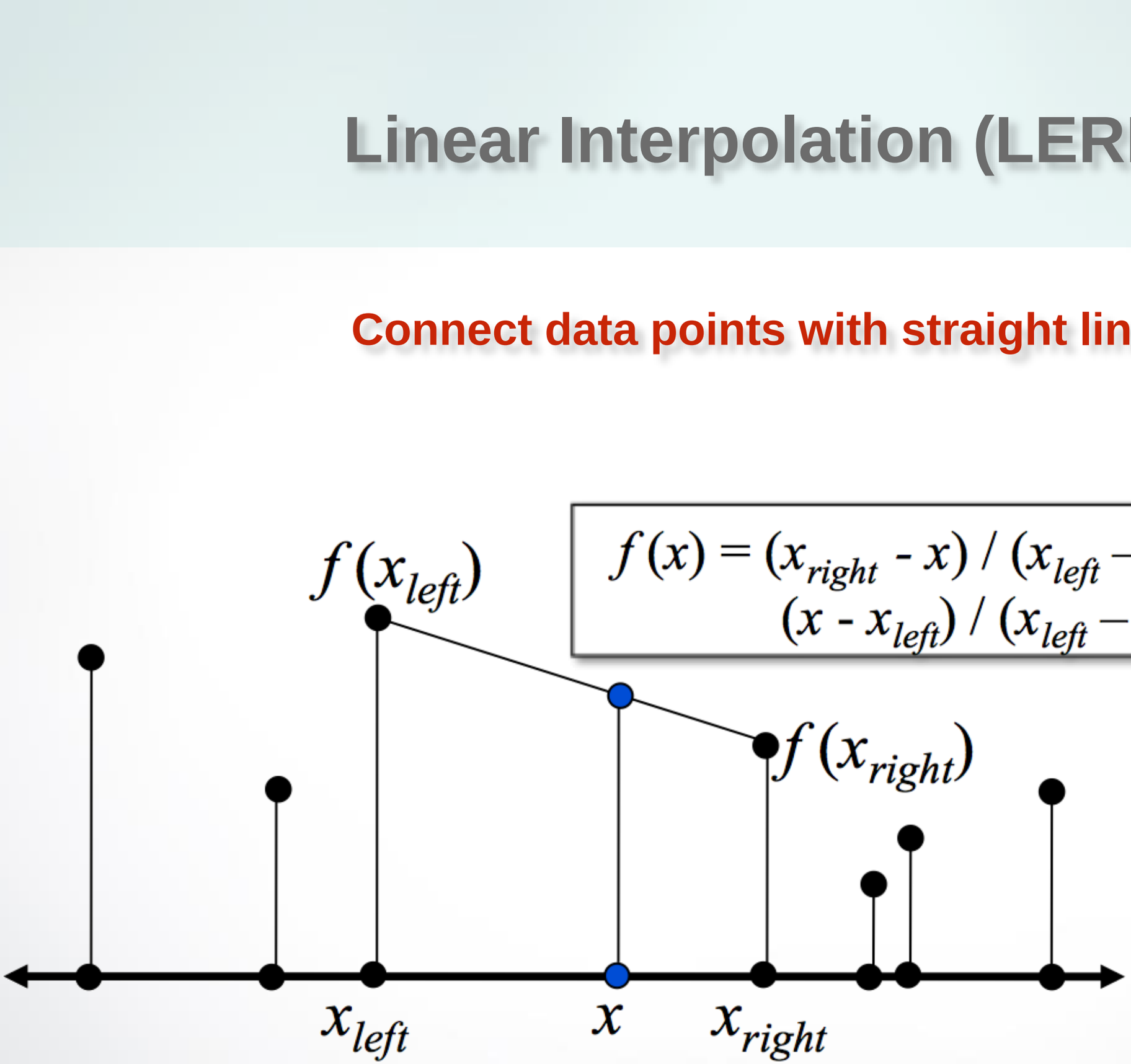
Linear Interpolation (LERP)



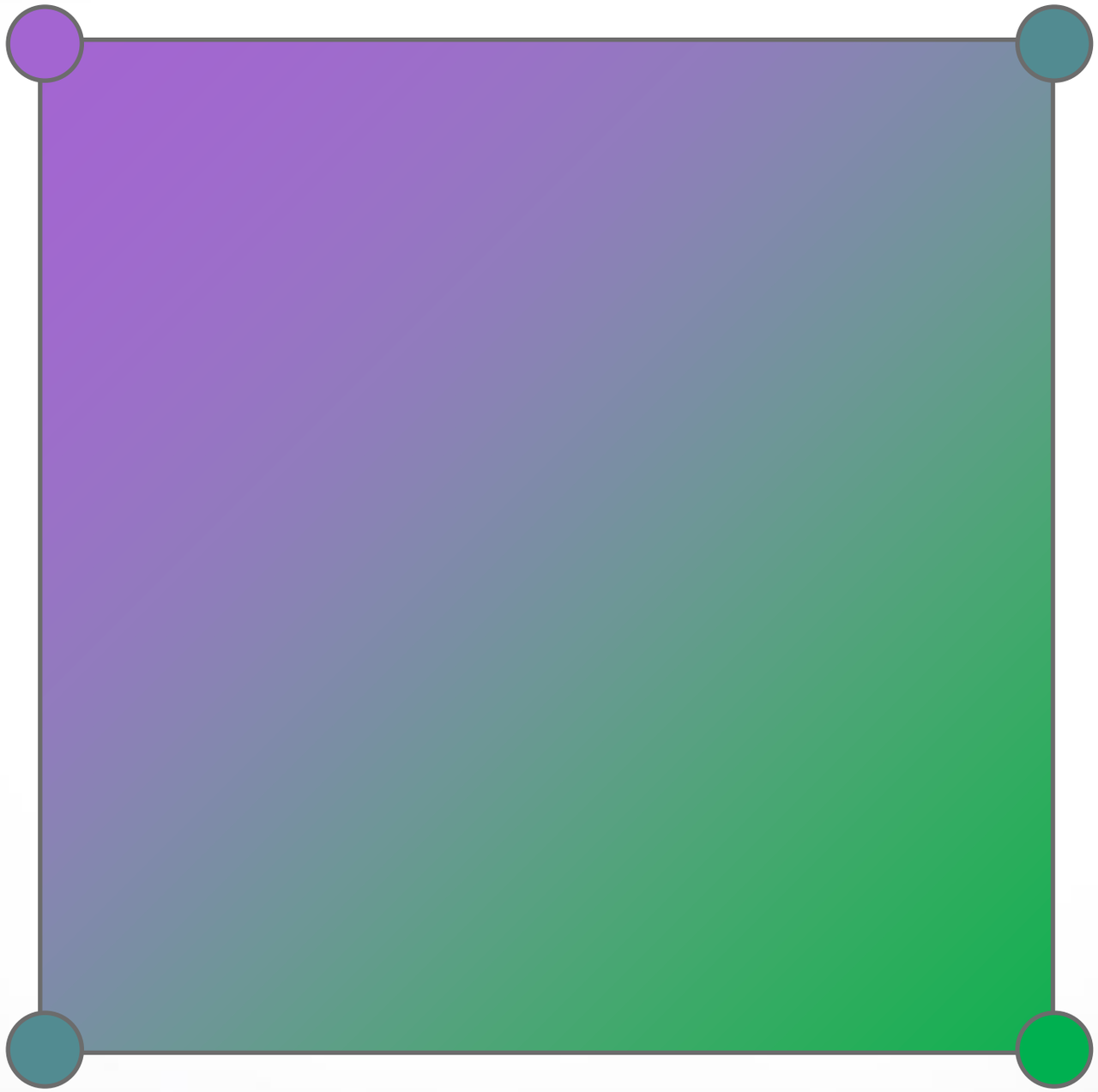
Linear Interpolation (LERP)

Connect data points with straight lines

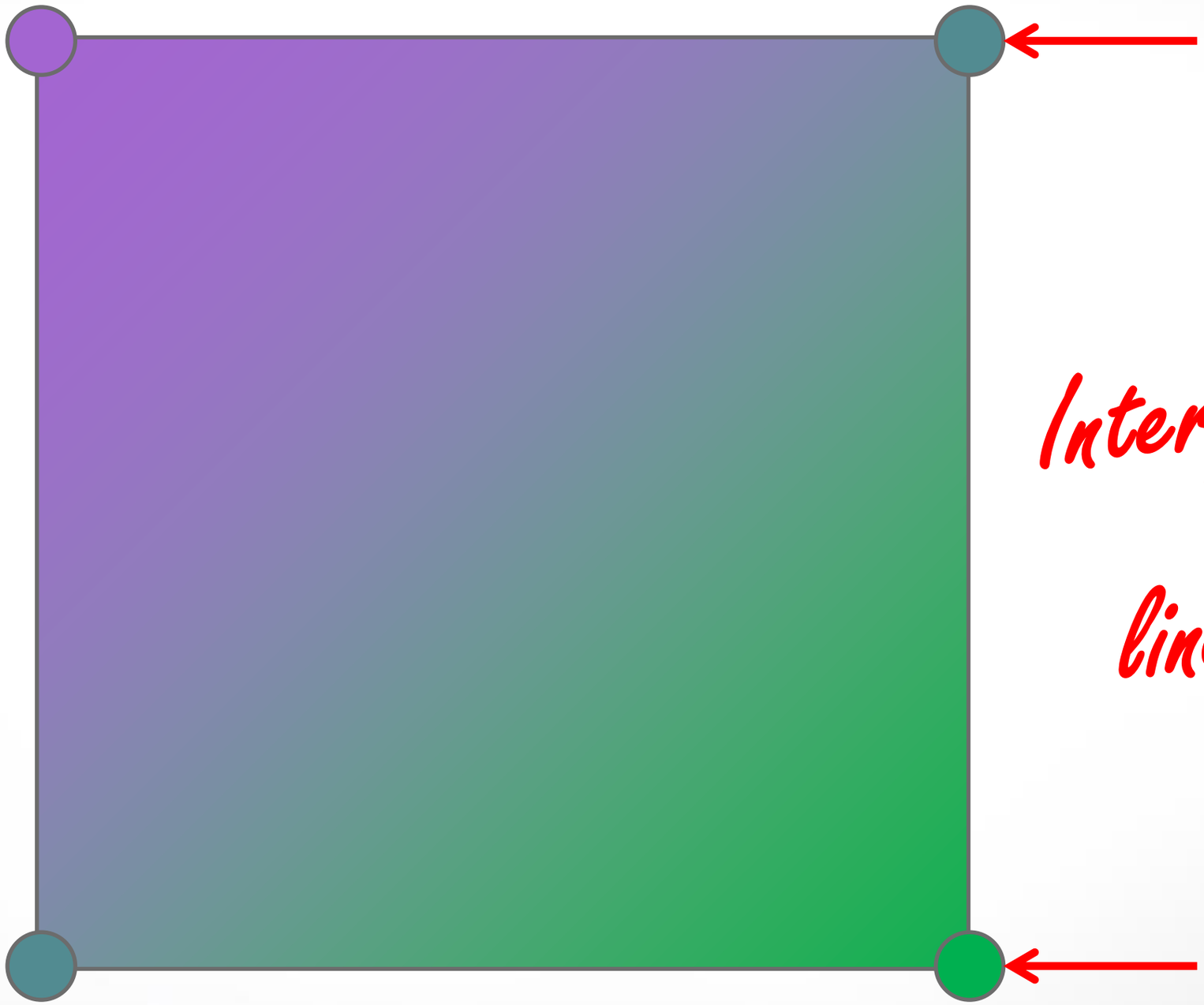
$$f(x) = (x_{right} - x) / (x_{left} - x_{right}) f(x_{left}) + (x - x_{left}) / (x_{left} - x_{right}) f(x_{right})$$



Bilinear Interpolation

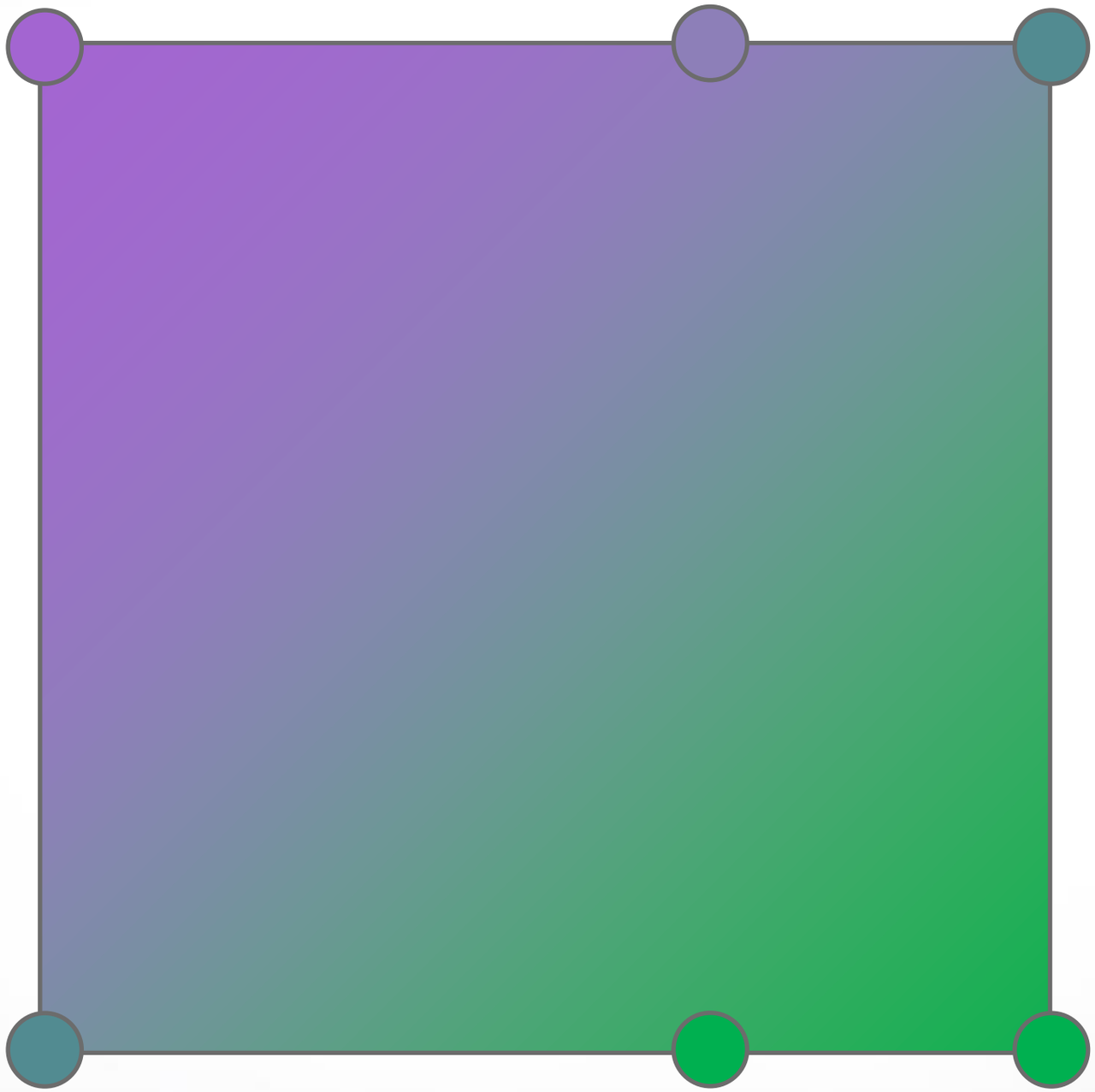


Bilinear Interpolation

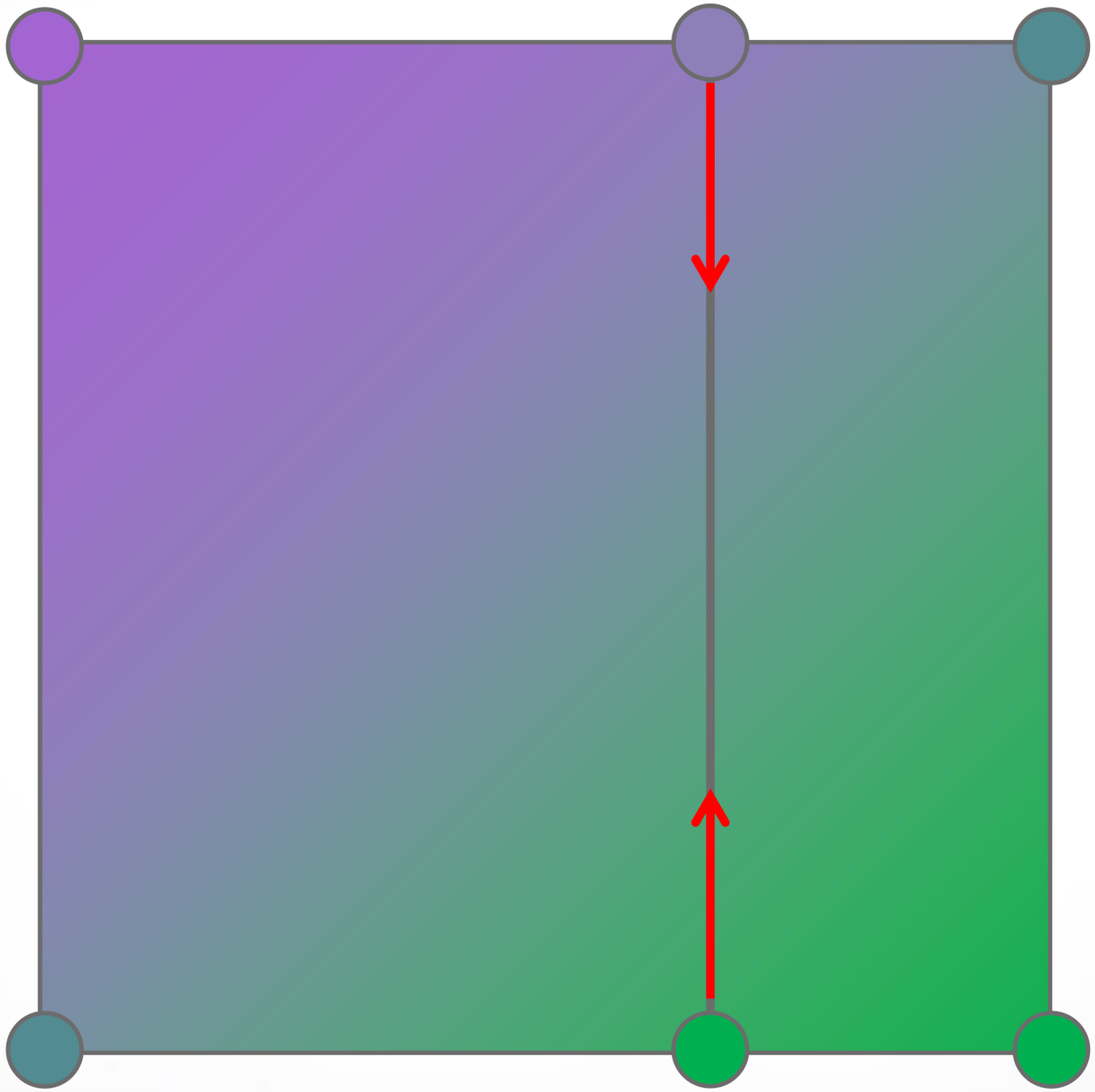


Interpolate linearly

Bilinear Interpolation



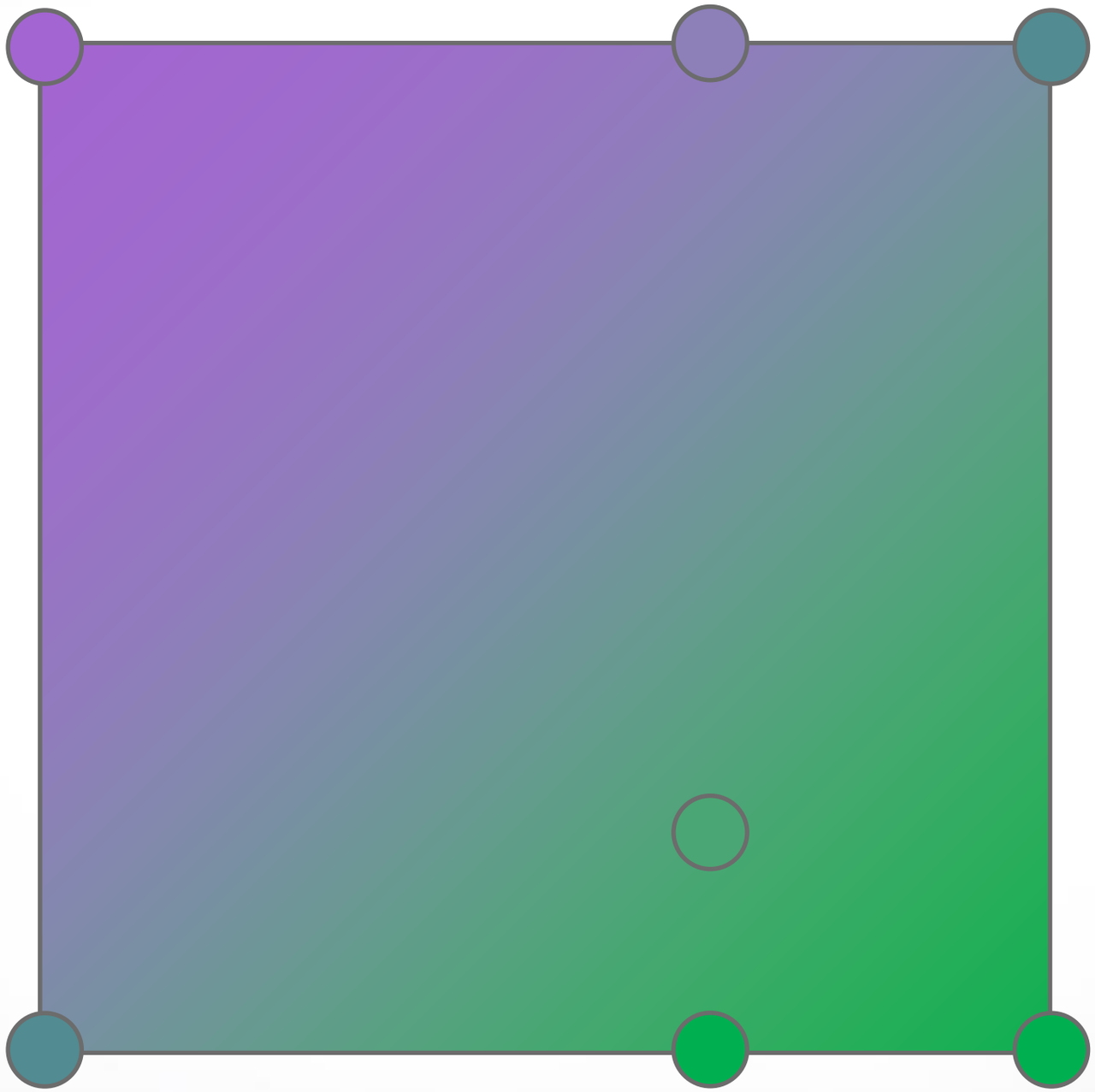
Bilinear Interpolation



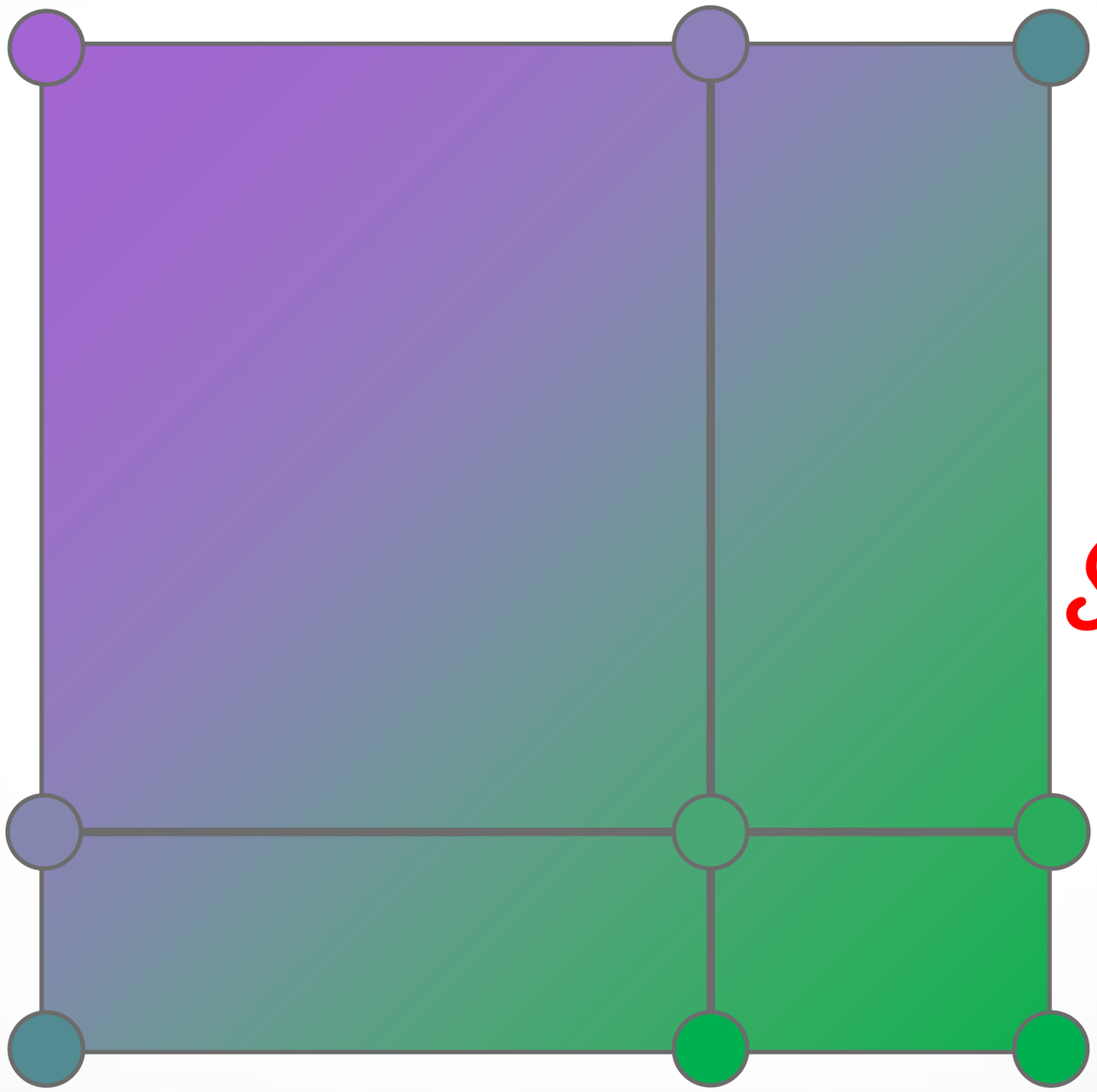
Interpolate

again

Bilinear Interpolation



Order Doesn't Matter

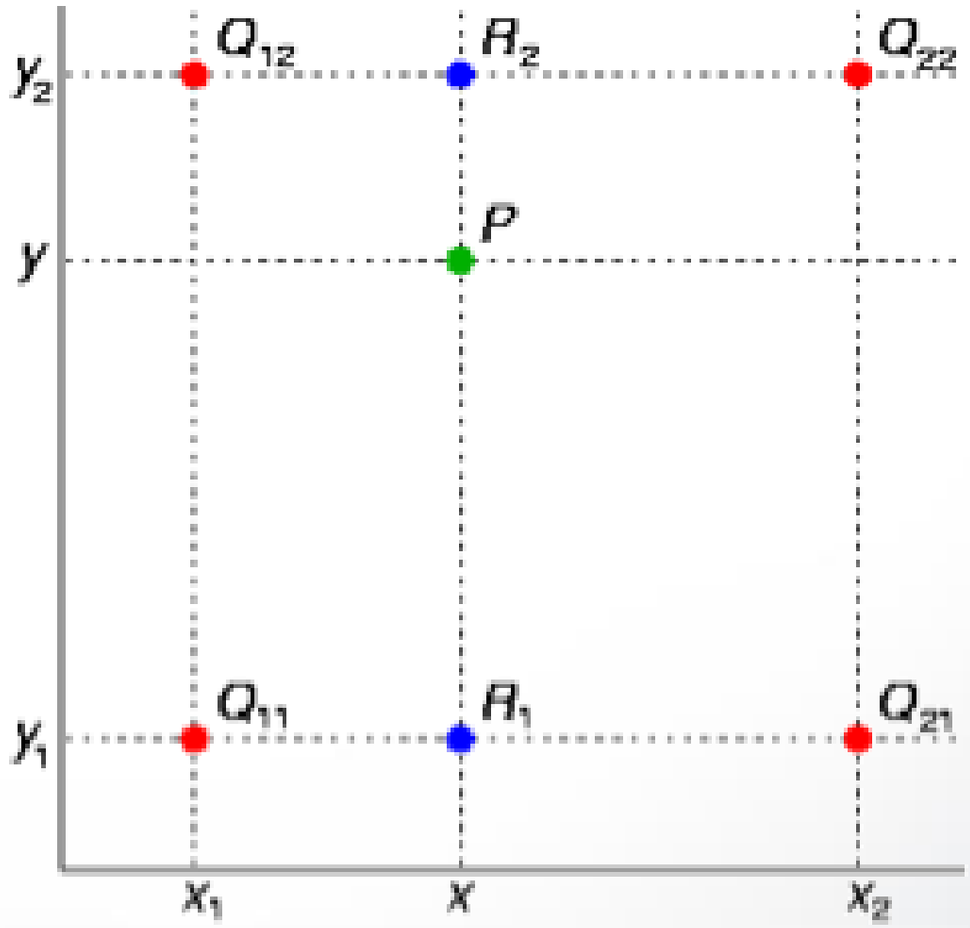


Same formula!

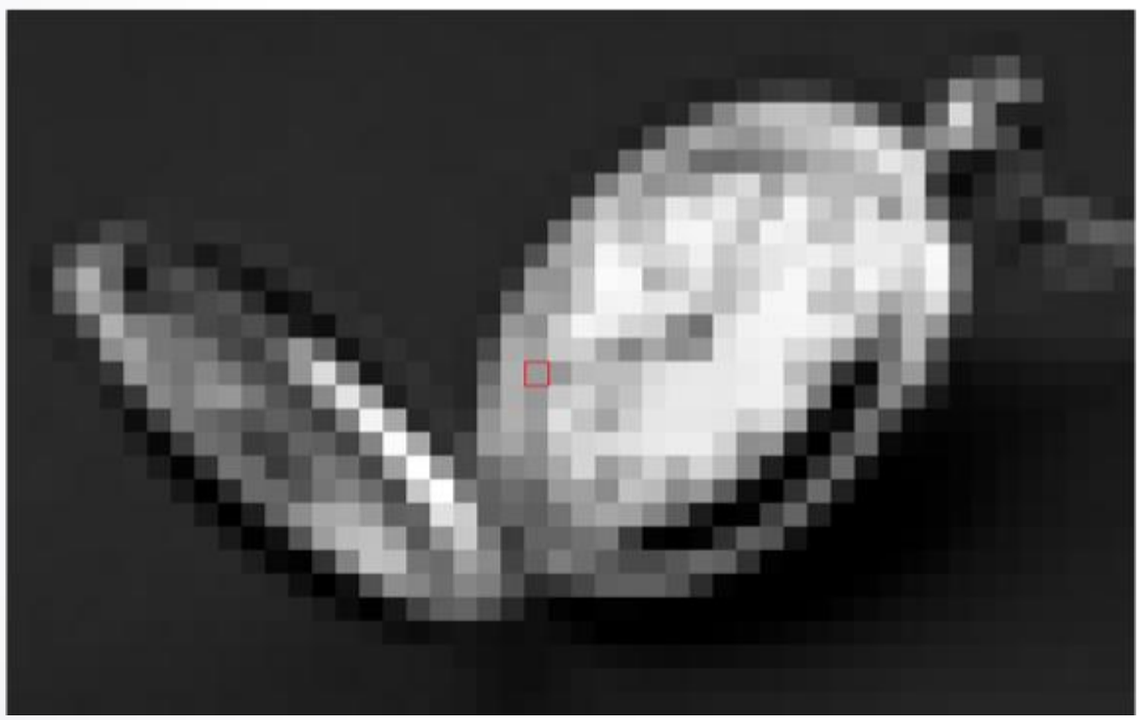
Bilinear Interpolation

Interpolate in x then in y (or vice versa!)

$$
\begin{aligned}
f(x, y) \approx & \frac{f(Q_{11})}{(x_2 - x_1)(y_2 - y_1)}(x_2 - x)(y_2 - y) \\
& + \frac{f(Q_{21})}{(x_2 - x_1)(y_2 - y_1)}(x - x_1)(y_2 - y) \\
& + \frac{f(Q_{12})}{(x_2 - x_1)(y_2 - y_1)}(x_2 - x)(y - y_1) \\
& + \frac{f(Q_{22})}{(x_2 - x_1)(y_2 - y_1)}(x - x_1)(y - y_1).
\end{aligned}
$$



Comparison



Nearest Neighbor

Bilinear

Texture Interpolation in OpenGL

Nearest-neighbor:

- Faster, but worse quality

```
glTexParameteri(GL_TEXTURE_2D,
                GL_TEXTURE_MIN_FILTER, GL_NEAREST)
```

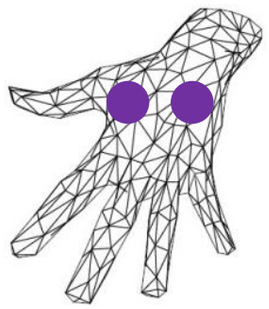
Linear:

- Incorporate colors of several neighbors to determine color
- Slower, better quality

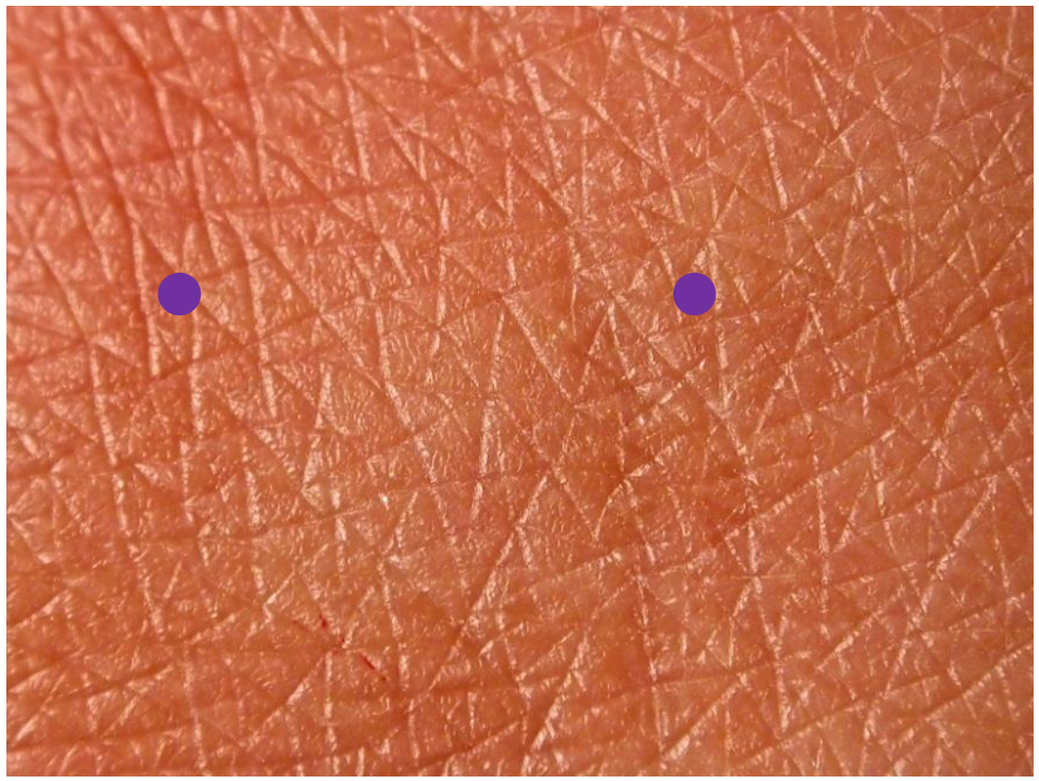
```
glTexParameteri(GL_TEXTURE_2D,
                GL_TEXTURE_MIN_FILTER, GL_LINEAR)
```

Opposite Issue: Minification

Small image



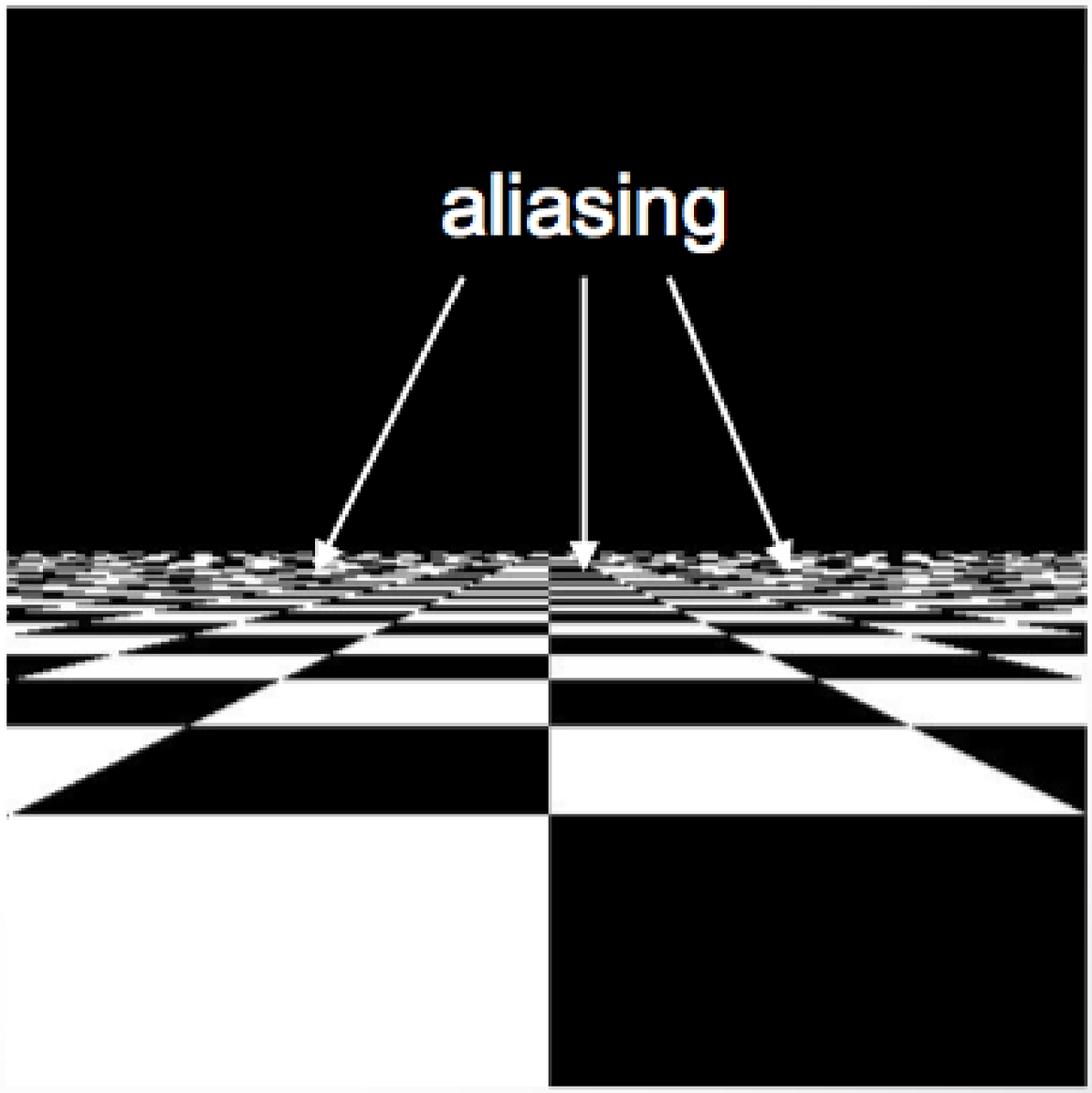
Large texture



Adjacent rendered pixels are far apart in texture

Why Do We Need to Filter?

- Texture image is **shrunk** in distant parts of the image
- This leads to **aliasing**
- Can be fixed with **filtering**
  - bilinear in space
  - trilinear in space and level of detail (mipmapping)

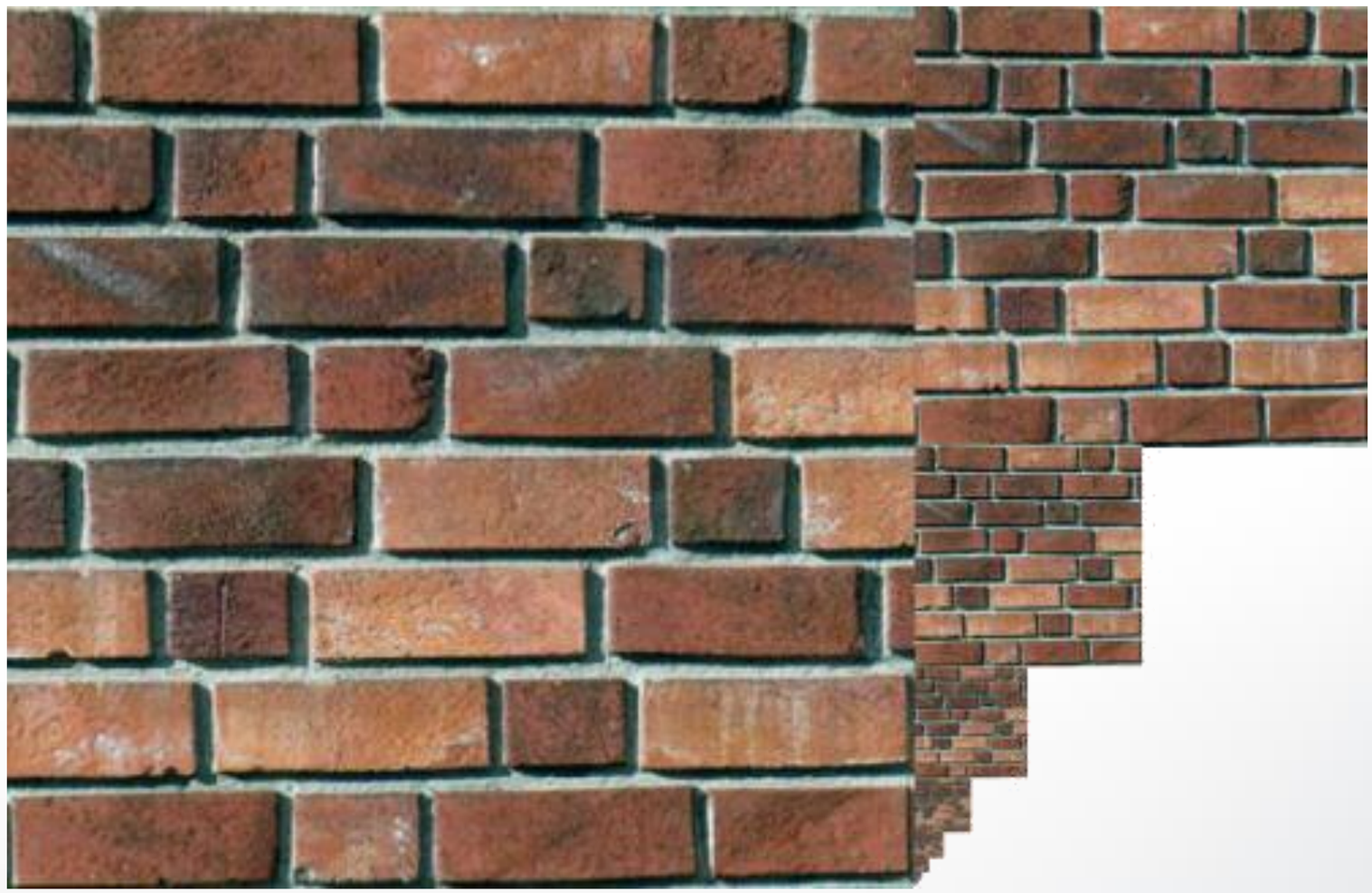


Mipmapping

- **Precompute** texture at different scales and use the appropriate texture at each distance
- When rendering, **choose scale** to avoid having to minify on the fly

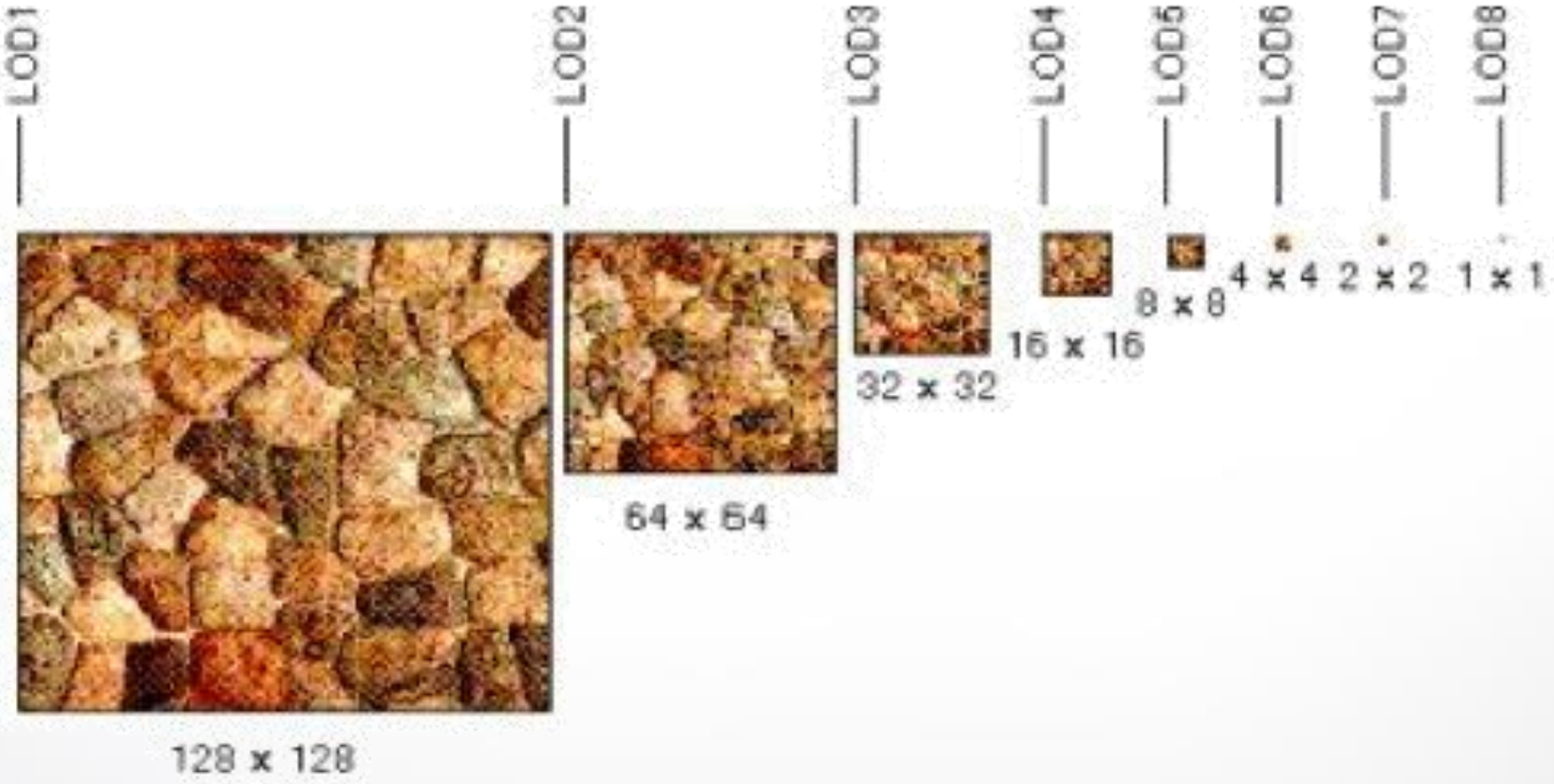
"Multum in Parvo"

("much in little")



Mipmapping

- Each piece represents one **level of detail** (LOD)
- Simplified by using **powers of two**



Mipmapping in OpenGL

Generate all the mipmaps automatically:

```
gluBuild2DMipmaps(GL_TEXTURE_2D, components,
        width, height, format, type, data)
```

Tell OpenGL to use the mipmaps for the texture:

```
glTexParameterf(GL_TEXTURE_2D,
                GL_TEXTURE_MIN_FILTER,
                GL_NEAREST_MIPMAP_NEAREST)
```

Outline

- Introduction
- Texture mapping in OpenGL
- Filtering and Mipmaps
- Example
- Non-color texture maps

Complete example

```
void initTexture()
{
  load image into memory;
  // Can use libjpeg, libtiff, or other image library.
  // Image should be stored as a sequence of bytes, usually 3 bytes per
     pixel (RGB), or 4 bytes (RGBA); image size is 4 * 256 * 256 bytes in
     this example.
  // We assume that the image data location is stored in pointer
  // "pointerToImage."

  // create placeholder for texture
  // must declare a global variable in program header: GLUint texName
glGenTextures(1, &texName);

  // make texture "texName" the currently active texture
glBindTexture(GL_TEXTURE_2D, texName);

  (continues on next page)
```

Complete example (part 2)

```
 // specify texture parameters (they affect whatever texture is active)
 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
// repeat pattern in s
 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
// repeat pattern in t

  // use linear filter both for magnification and minification
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
  GL_LINEAR);
  glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
  GL_LINEAR);

  // load image data stored at pointer "pointerToImage" into the currently
     active texture ("texName")
  glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, 256, 256, 0,
   GL_RGBA, GL_UNSIGNED_BYTE, pointerToImage);

} // end init()
```

Complete example (part 3)

```
void display()
{
  …
  // no modulation of texture color with lighting; use
texture color directly
  glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,
 GL_REPLACE);

  // turn on texture mapping (this disables standard
OpenGL lighting, unless in GL_MODULATE mode)
  glEnable(GL_TEXTURE_2D);

  (continues on next page)
```

Complete example (part 4)

```
glBegin(GL_QUADS); // draw a textured quad
  glTexCoord2f(0.0,0.0); glVertex3f(-2.0,-1.0,0.0);
  glTexCoord2f(0.0,1.0); glVertex3f(-2.0,1.0,0.0);
  glTexCoord2f(1.0,0.0); glVertex3f(0.0,1.0,0.0);
  glTexCoord2f(1.0,1.0); glVertex3f(0.0,-1.0,0.0);
glEnd();

  // turn off texture mapping
  glDisable(GL_TEXTURE_2D);

  // draw some non-texture mapped objects
(standard OpenGL lighting will be used if it is enabled)
  …
  // switch back to texture mode, etc.
  …
} // end display()
```

Outline

- Introduction
- Texture mapping in OpenGL
- Filtering and Mipmaps
- Example
- Non-color texture maps

Textures do not have to represent color

- Specularity (patches of shininess)
- Transparency (patches of clearness)
- Normal vector changes (bump maps)
- Reflected light (environment maps)
- Shadows
- Changes in surface height (displacement maps)

Bump mapping

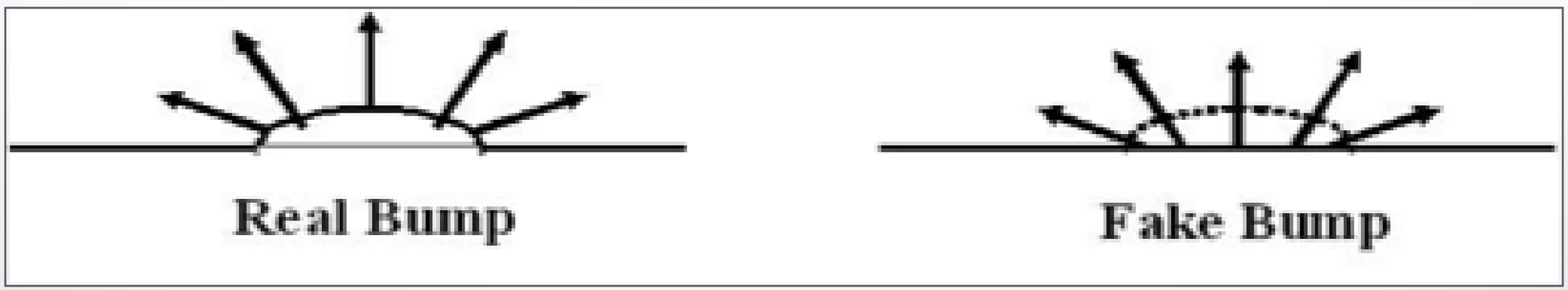
Bump mapping

- How do you make a surface look ***rough***?

  Option 1: model the surface with tiny polygons

  Option 2: perturb normal vectors before shading

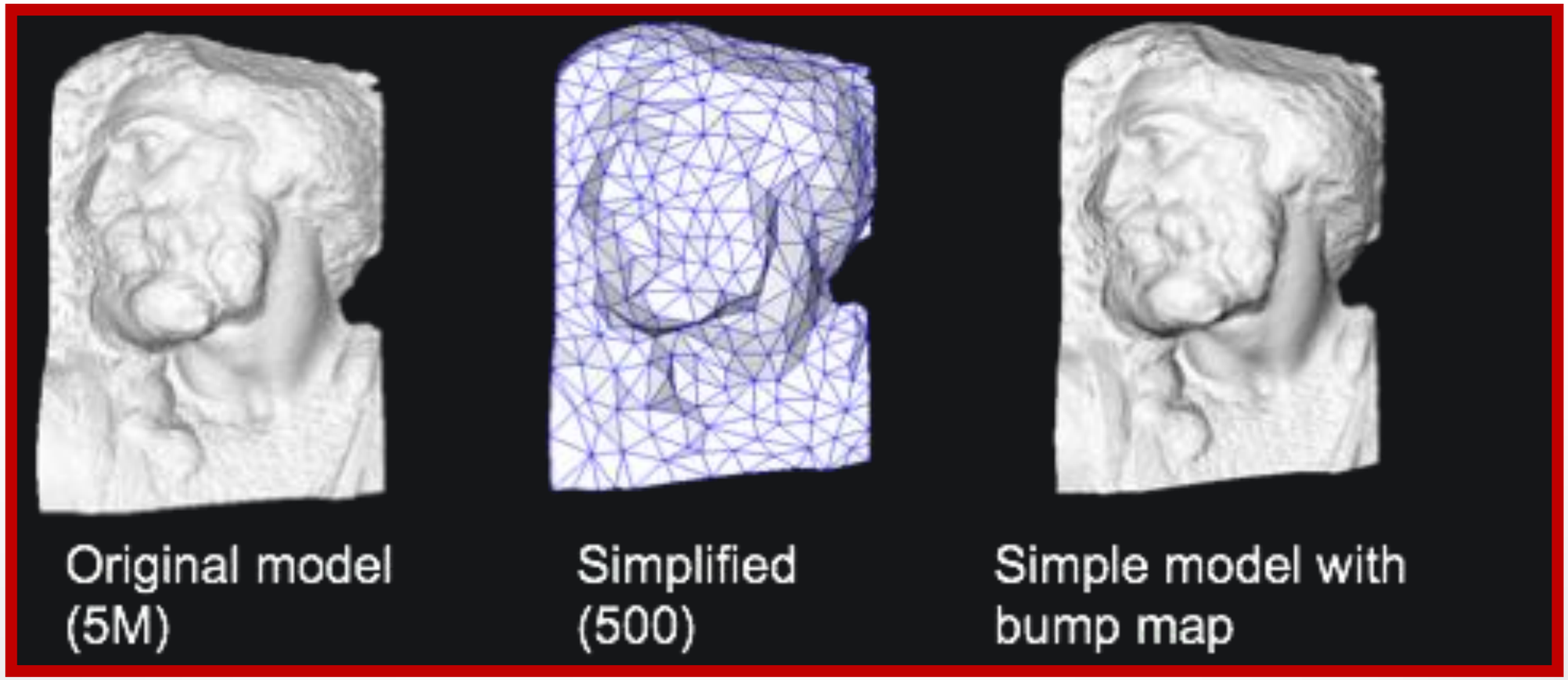
  - Fakes small displacements above or below the true surface
  - The surface doesn't actually change,
    but shading makes it look like there are irregularities!
  - A texture stores information about the "fake" height of the surface



Bump mapping

- Perturb normal **without** making any actual change to the shape
  - This illusion can be seen through:

How?



Light Mapping

- *Quake* used *light maps* in addition to texture maps
  - Texture maps add detail to surfaces
  - Light maps store pre-computed illumination
- Multiplied at runtime, cached for efficiency

Texture Map Only

Light Map

Texture + Light Map

Summary

- Introduction
- Texture mapping in OpenGL
- Filtering and Mipmaps
- Example
- Non-color texture maps

http://cs420.hao-li.com

Thanks!