Fall 2017

CSCI 420: **Computer Graphics**

6.1 Texture Mapping

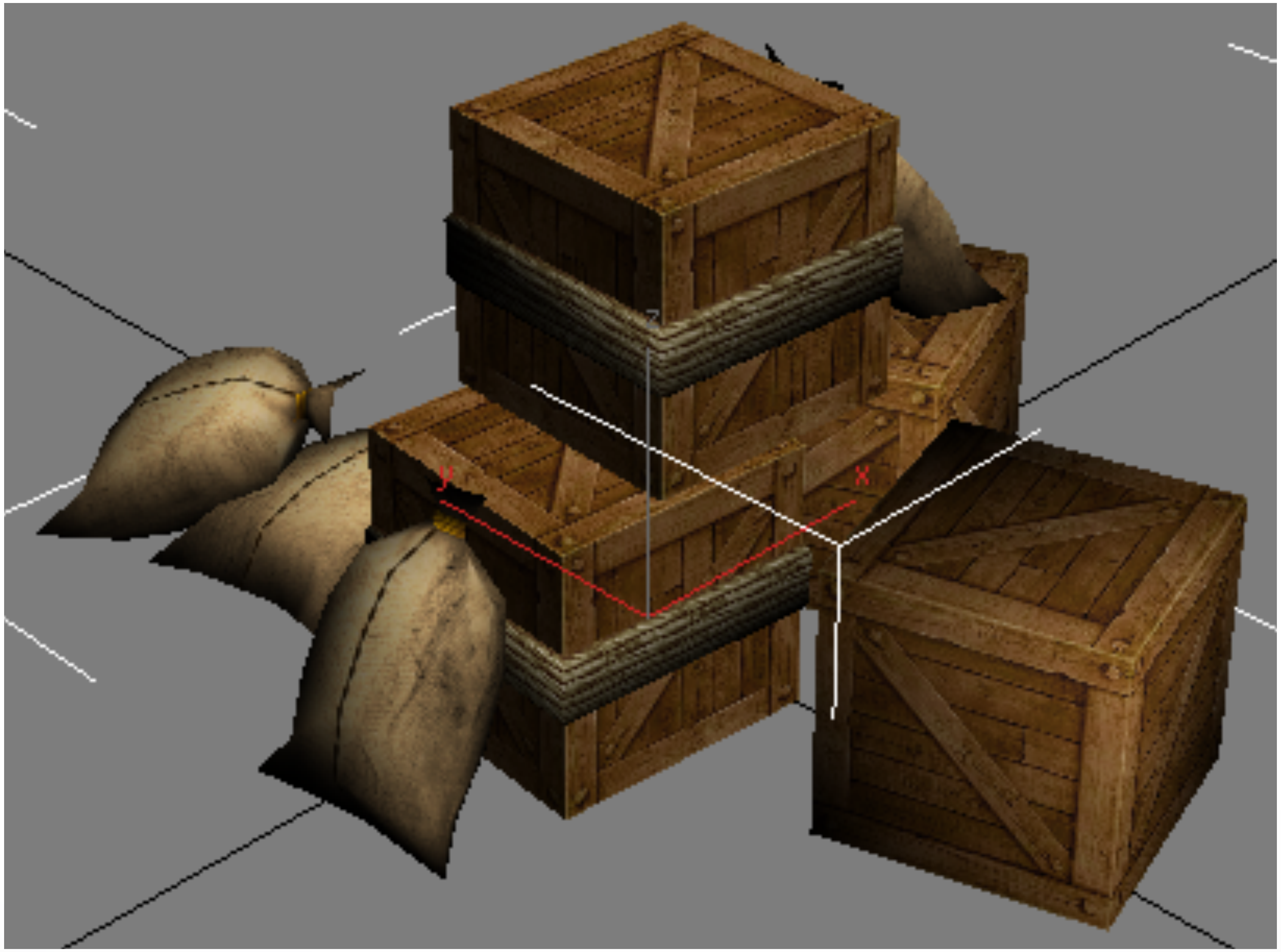
Hao Li

http://cs420.hao-li.com

Outline

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

How Do You Add Detail to a Cube?



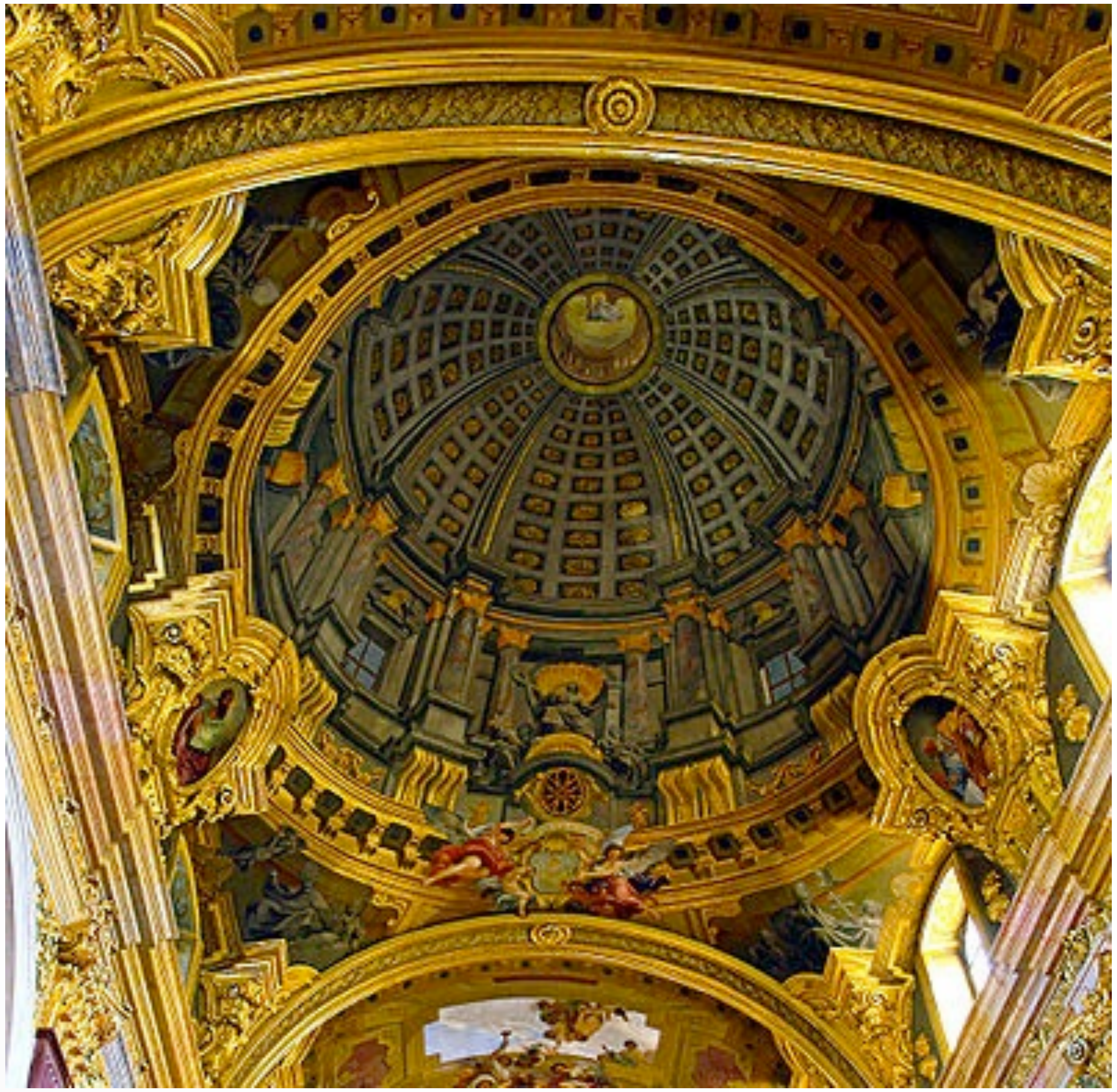
six sides - six colors?

Texture Mapping

- A way of adding surface details
- Two ways can achieve the goal:
  - Model the surface with more polygons
    - Slows down rendering speed
    - Hard to model fine features
  - Map a texture to the surface
    - This lecture
    - **Image complexity does not affect complexity of processing**
- Efficiently supported in hardware

I live
in my own
little world
but that's
all right,
they
know me
there.

Trompe L’Oeil (“Deceive the Eye”)



Jesuit Church, Vienna, Austria

- Windows and columns in the dome are painted, not a real 3D object
- Similar idea with texture mapping:

  Rather than modeling the intricate 3D geometry, replace it with an image !

Map textures to surfaces

an image

texture map

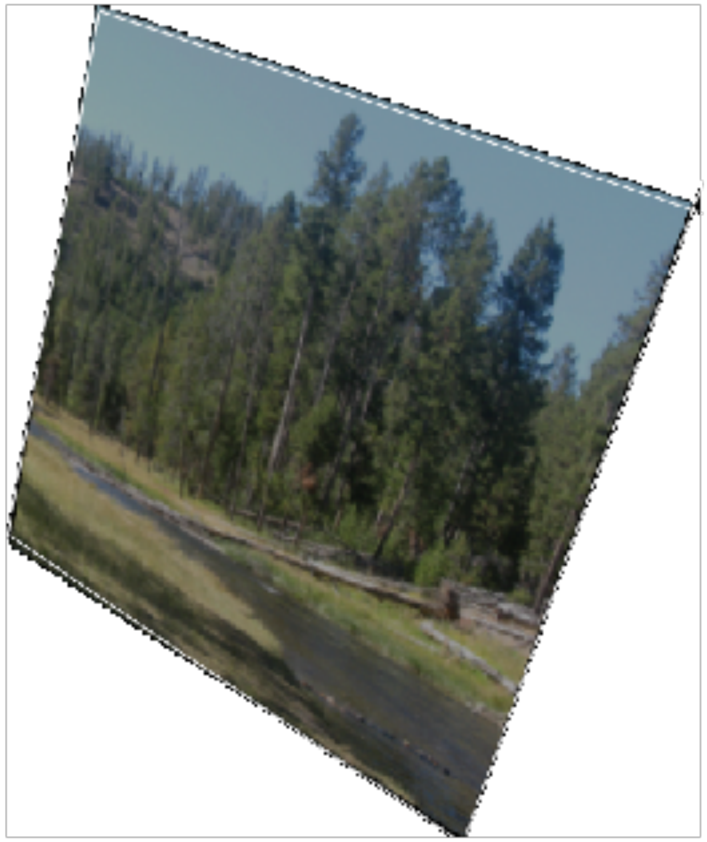
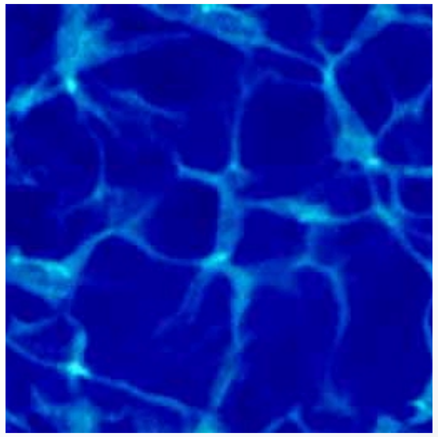


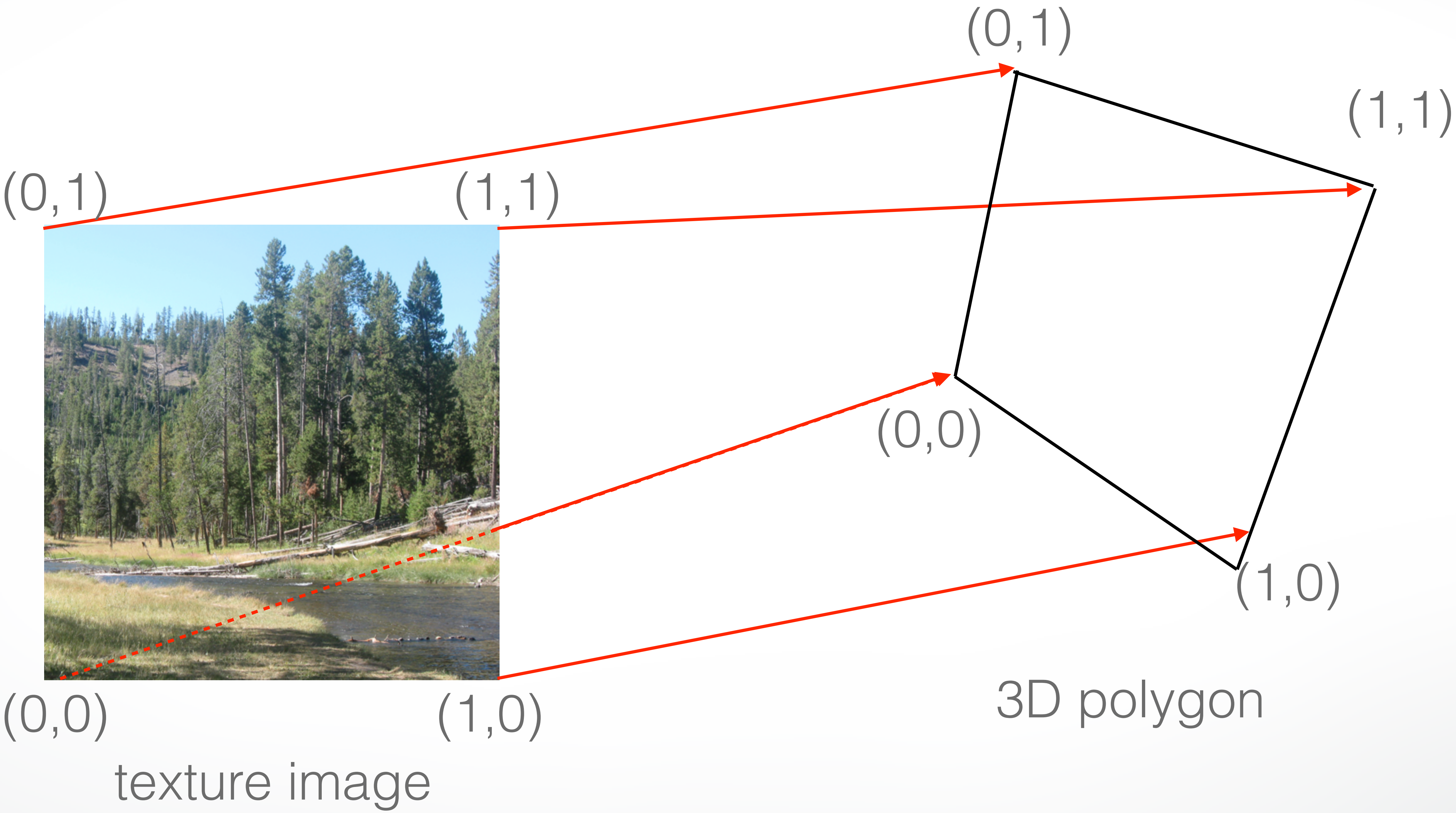
image mapped
to a 3D polygon
The polygon can
have arbitrary size,
shape and 3D position

The Texture

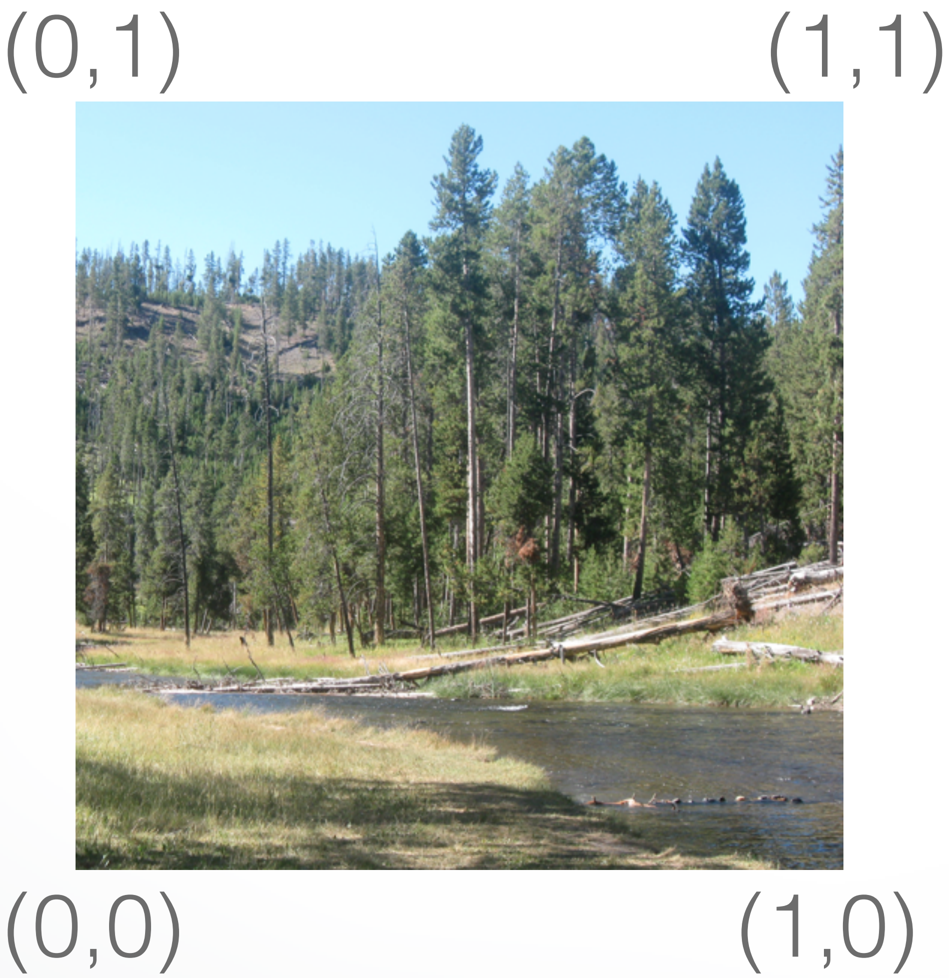
- Texture is a bitmap image
  - Can use an image library to load image into memory
  - Or can create images yourself within the program
- 2D array:
  unsigned char texture[height][width][4]
- Or unrolled into 1D array:
  unsigned char texture[4*height*width]
- Pixels of the texture are called *texels*
- Texel coordinates (s,t) scaled to [0,1] range



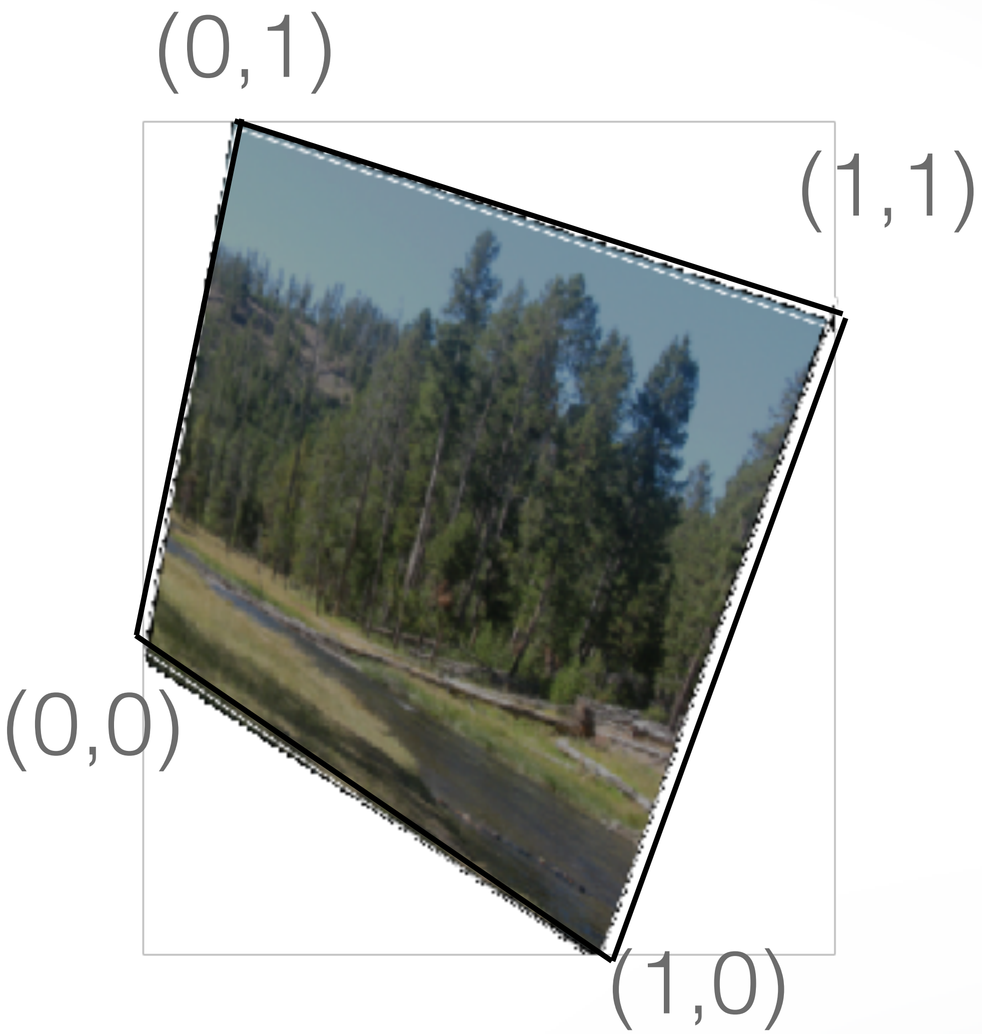
Texture map



Texture map

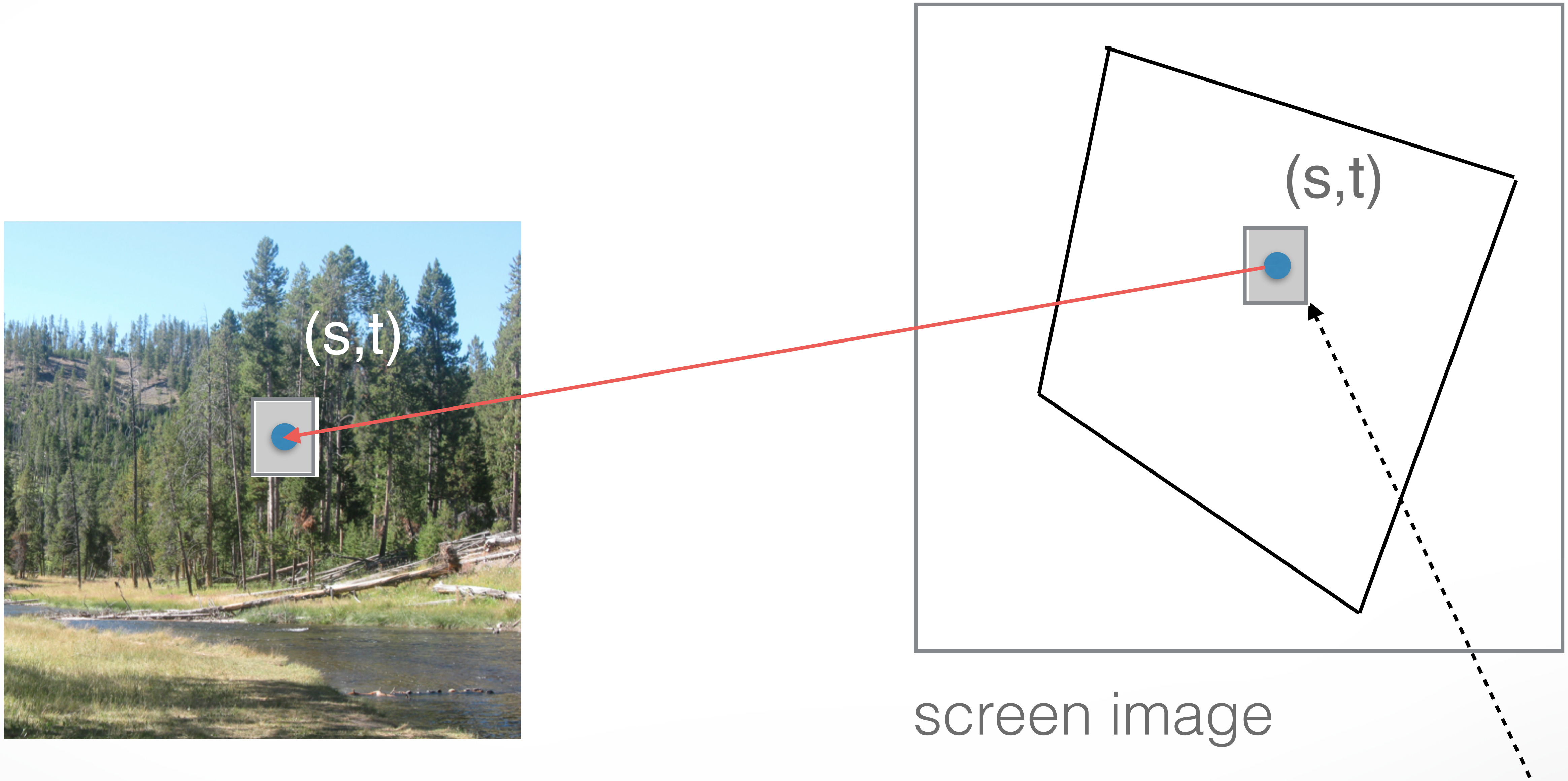


texture image



3D polygon

Inverse texture map

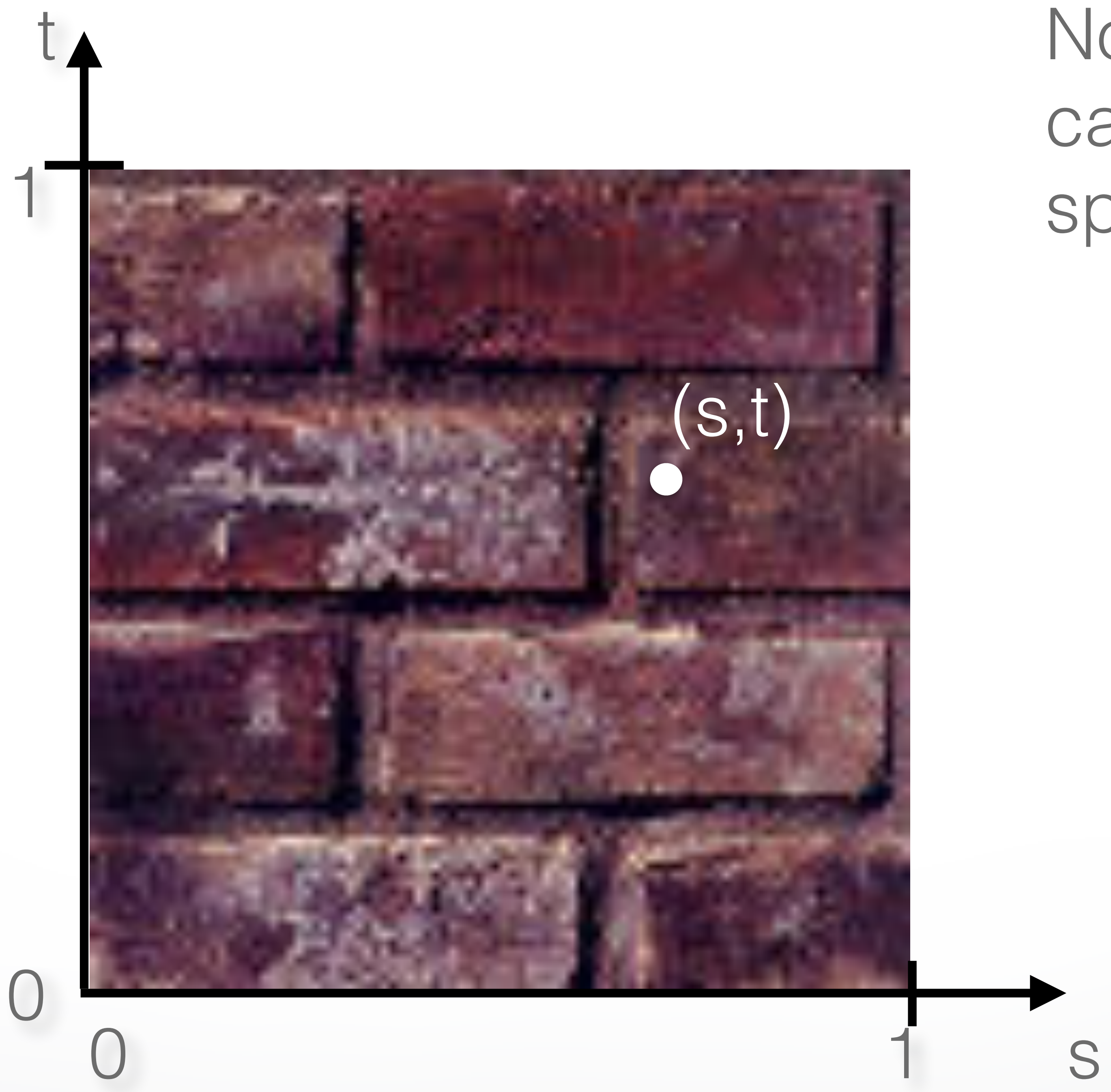


screen image

texture image

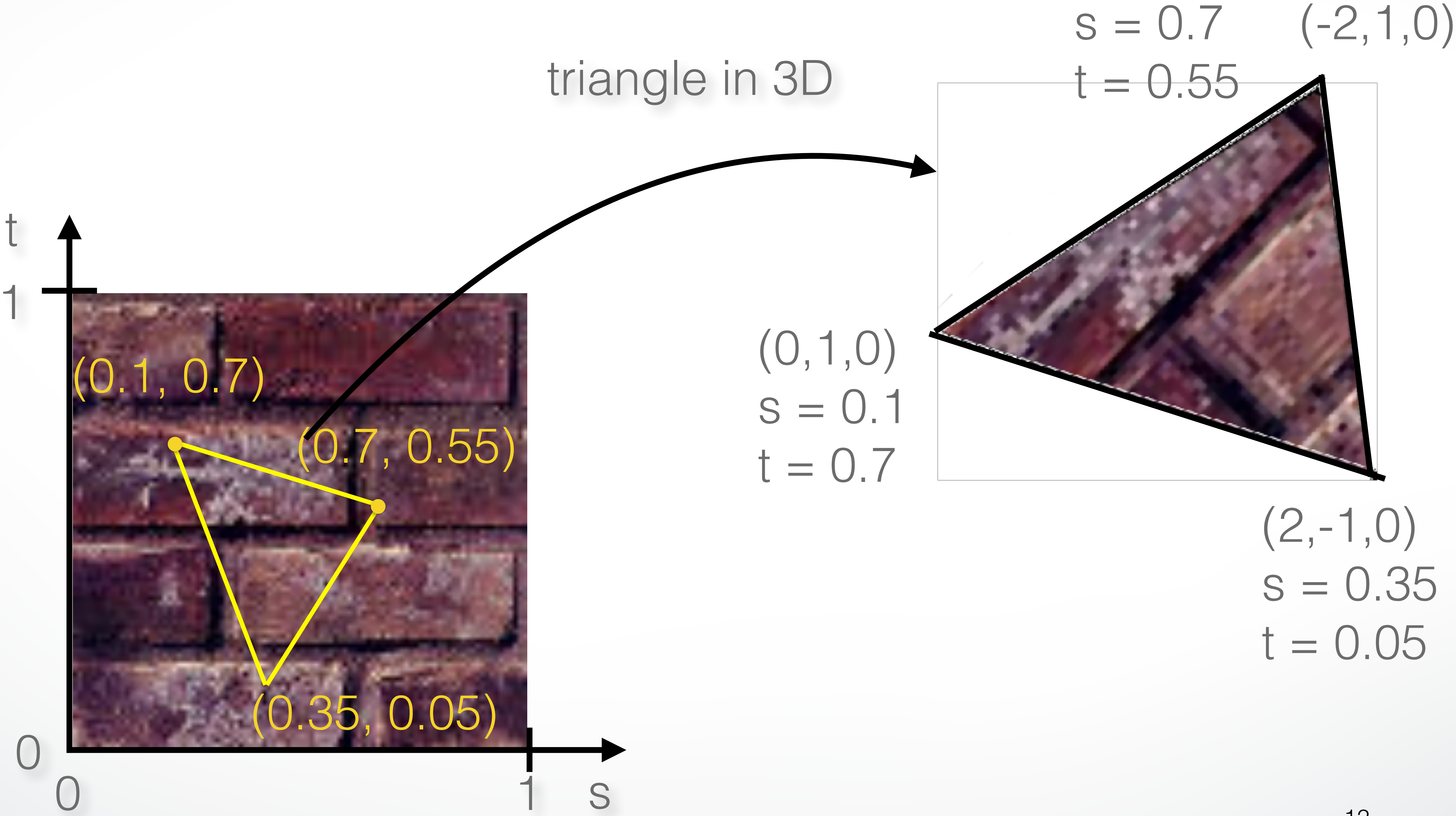
For each pixel, lookup into the texture image to obtain color

The “st” coordinate system



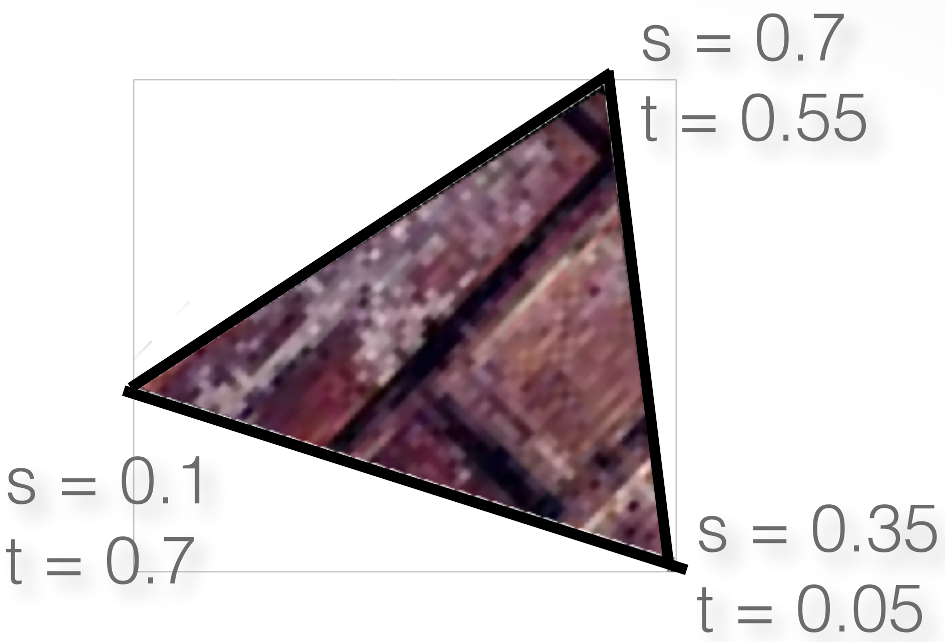
Note: also called “uv” space

Texture mapping: key slide



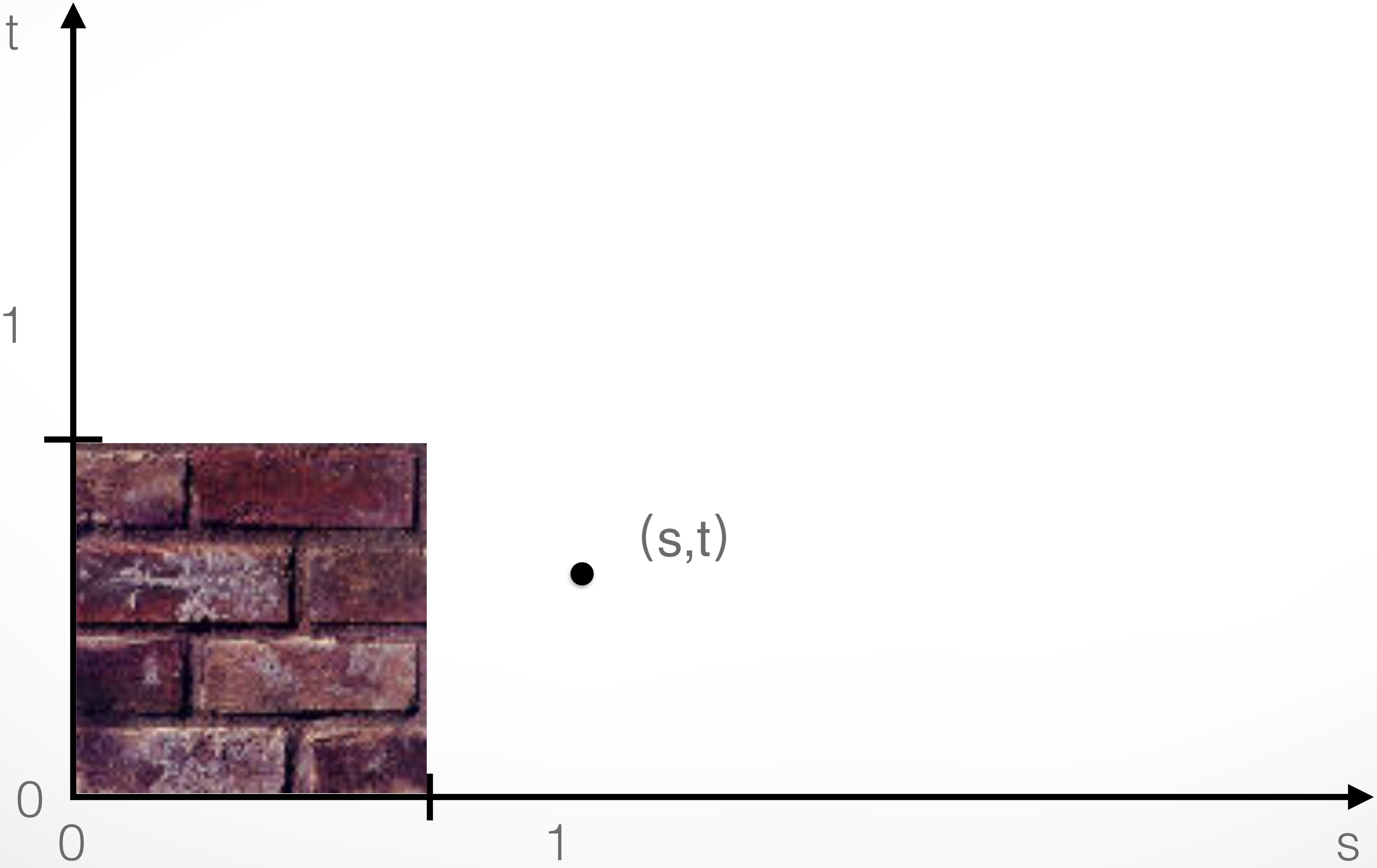
Specifying texture coordinates in OpenGL

- Use glTexCoord2f(s,t)
- State machine: Texture coordinates remain valid until you change them
- Example (from previous slide) :



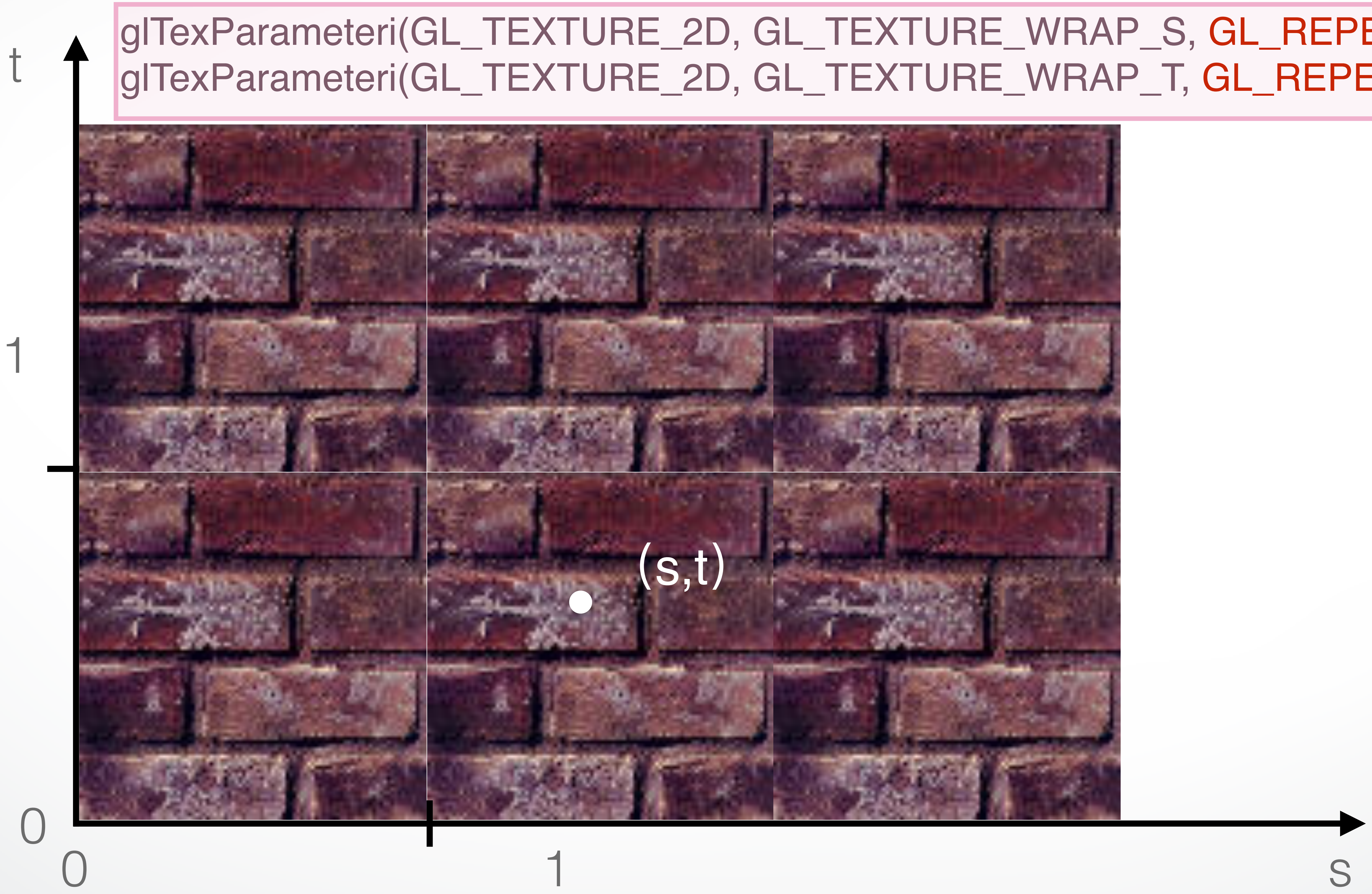
```
glEnable(GL_TEXTURE_2D); // turn texture mapping on
glBegin(GL_TRIANGLES);
  glTexCoord2f(0.35,0.05); glVertex3f(2.0,-1.0,0.0);
  glTexCoord2f(0.7,0.55); glVertex3f(-2.0,1.0,0.0);
  glTexCoord2f(0.1,0.7); glVertex3f(0.0,1.0,0.0);
glEnd();
glDisable(GL_TEXTURE_2D); // turn texture mapping off
```

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



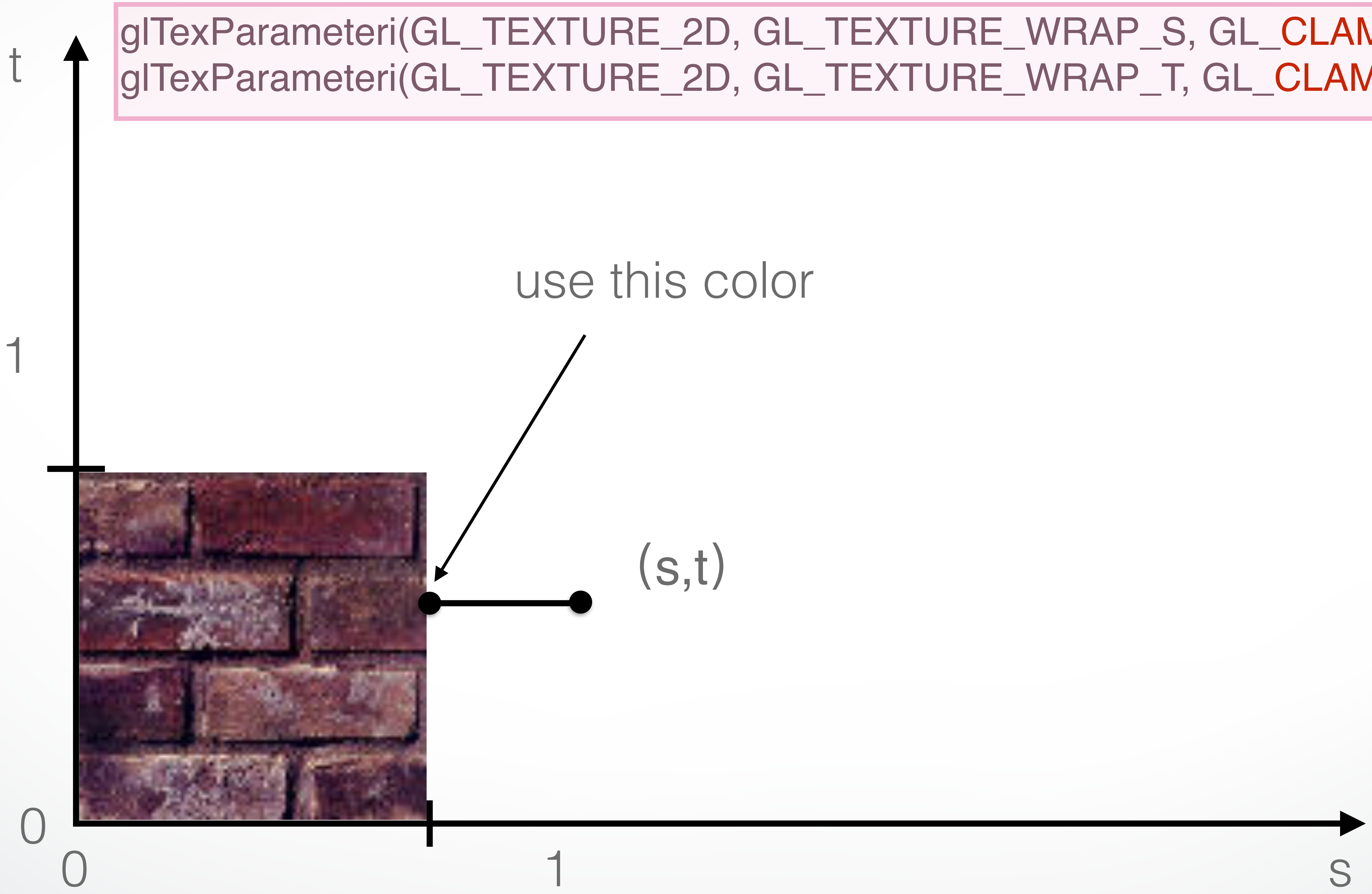
Solution 1: Repeat texture

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

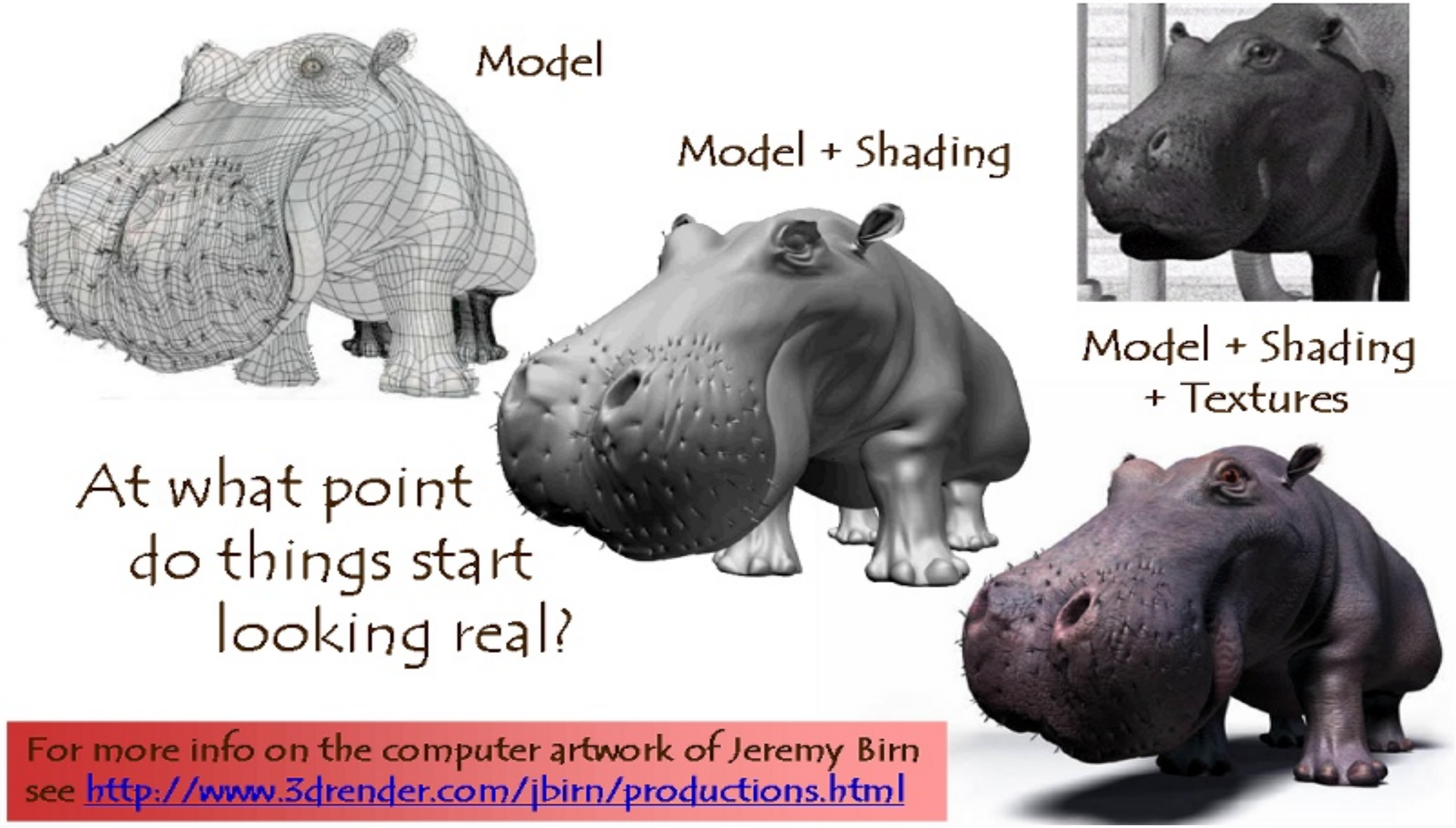


Solution 2: Clamp to [0,1]

```
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



At what point do things start looking real?

For more info on the computer artwork of Jeremy Birn see http://www.3drender.com/jbirn/productions.html

Combining texture mapping and shading

- Final pixel color = a combination of texture color and color under standard OpenGL Phong lighting
- GL_MODULATE:
  multiply texture and Phong lighting color
- GL_BLEND:
  linear combination of texture and Phong lighting color
- GL_REPLACE:
  use texture color only (ignore Phong lighting)
- 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

- **During your initialization:**

  1. Read texture image from file into an array in memory,

     or generate the image 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

- Do once during initialization, for each texture image in the scene, by calling glTexImage2D
- 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
  - Often: GL_RGB or GL_RGBA
  - Determines how the texture is stored internally
- 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

Enable/disable texture mode

- Must be done before rendering any primitives that are to be texture-mapped

  glEnable(GL_TEXTURE_2D)

  glDisable(GL_TEXTURE_2D)

- Successively 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 simultaneously; this is 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

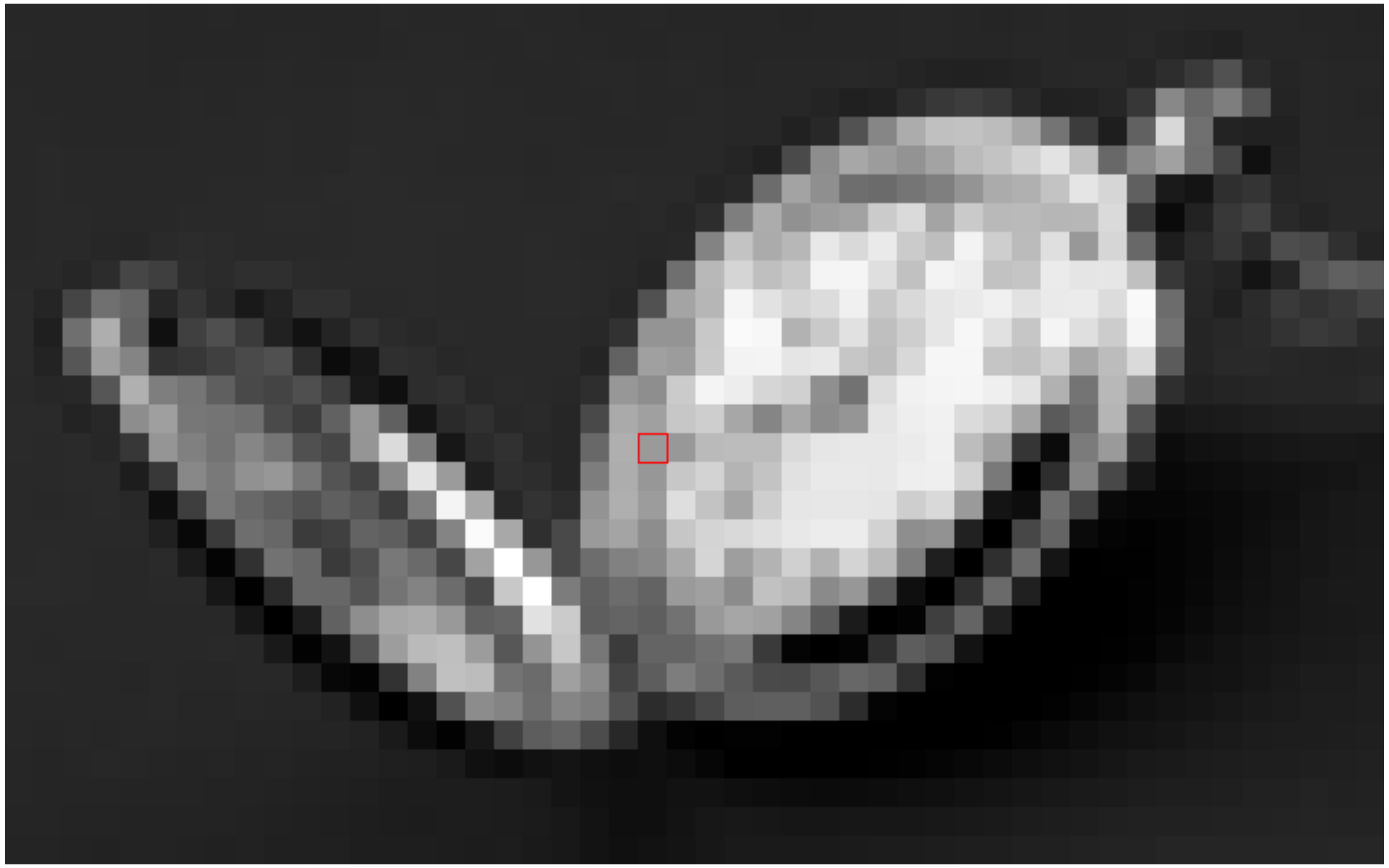
- This photo is too small

Zooming

- First consider a black and white image

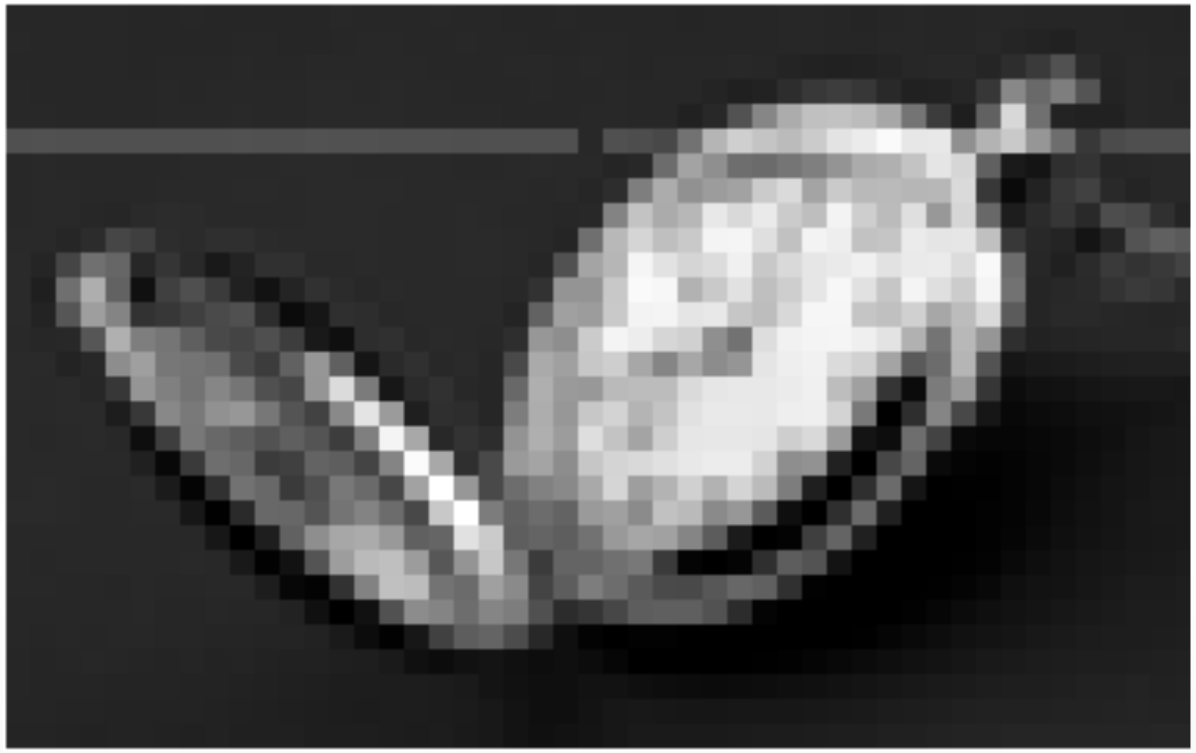
- We want to blow it up to poster size (zoom by a factor of 16)
- Firs try: repeat each row 16 times, then each column 16 times

Zooming: Nearest Neighbor Interpolation



Zooming: First Attempt

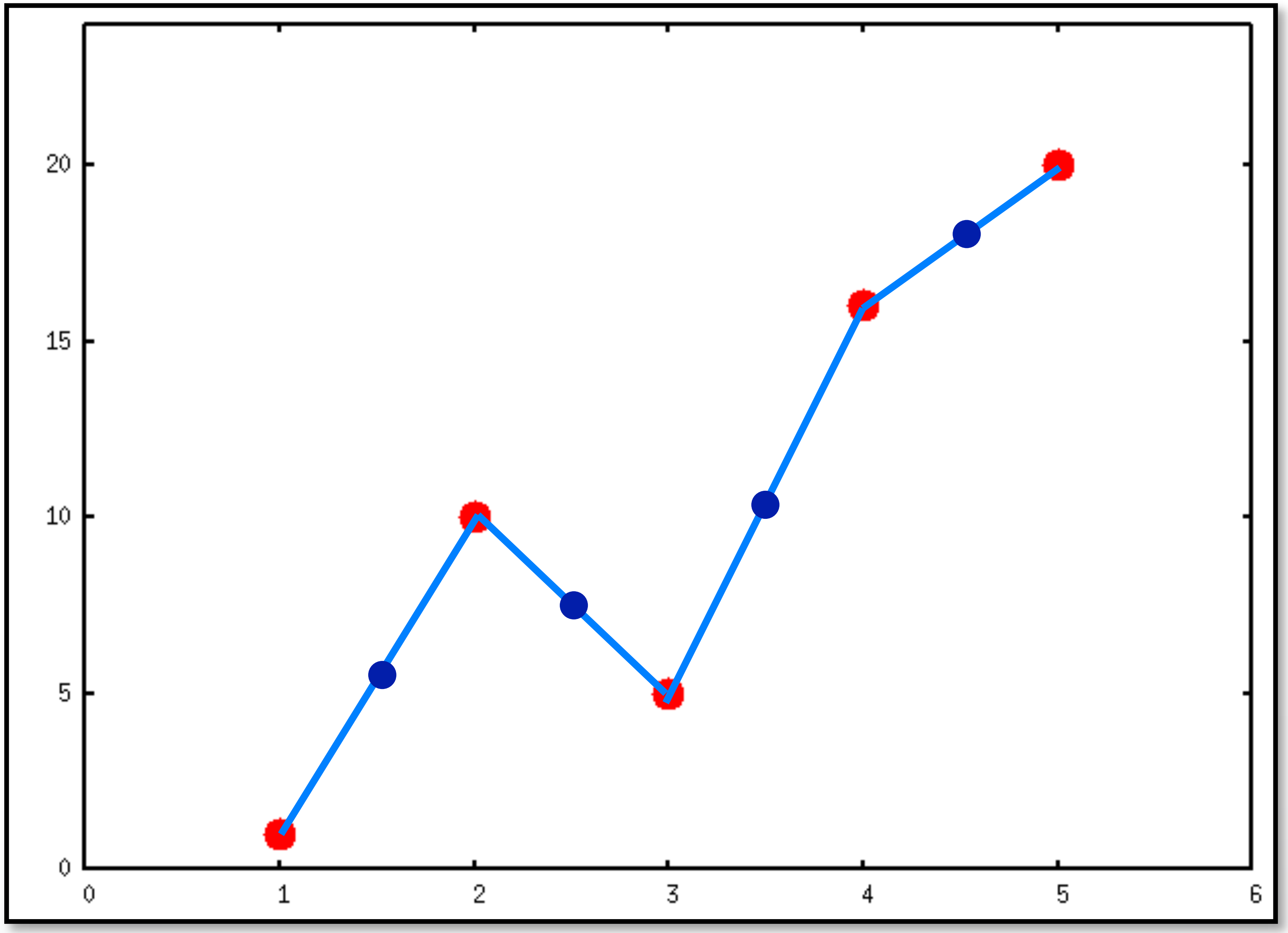
- That didn’t work so well
- We need a better way to find the in between values
- Let’s consider one horizontal slice through the image (one scanline)



Interpolation

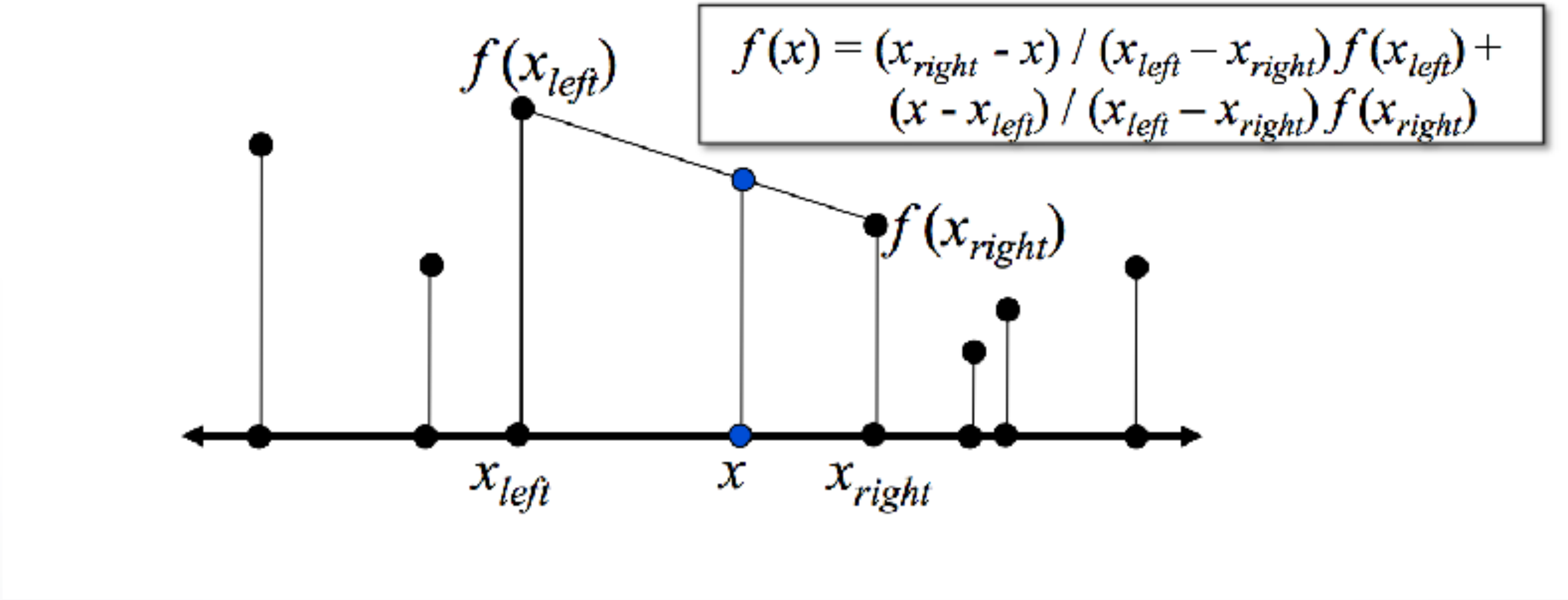
- Problem statement:
  - Given the values of a function f at a few locations, e.g. f(1), f(2), f(3), …
  - Find the rest of the values: what is f(1.5)?
- This is called **Interpolation**
- We need some models that predicts how the function behaves

Linear Interpolation (LERP)



Linear Interpolation (LERP)

- To compute f(x), find the two points x_{left} and x_{right} that x lies between

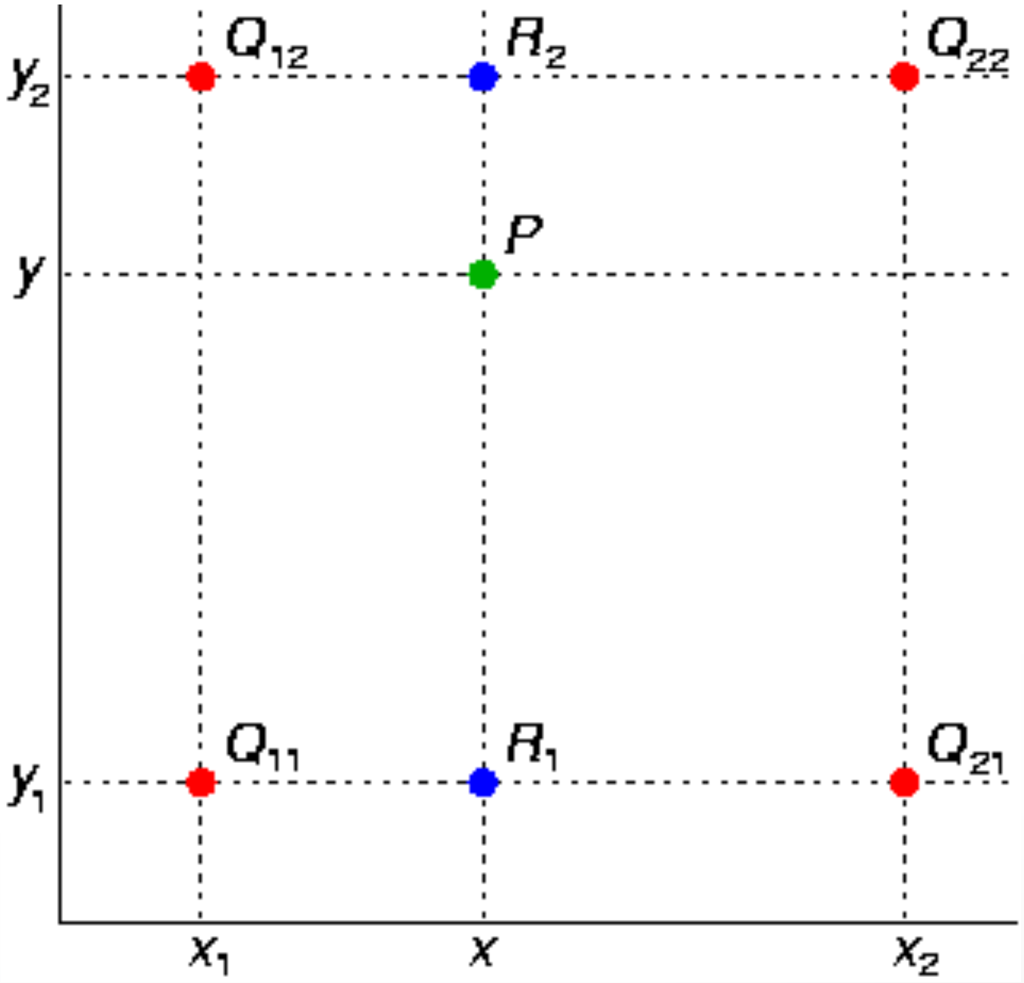


$$f(x) = (x_{right} - x) / (x_{left} - x_{right}) f(x_{left}) + (x - x_{left}) / (x_{left} - x_{right}) f(x_{right})$$

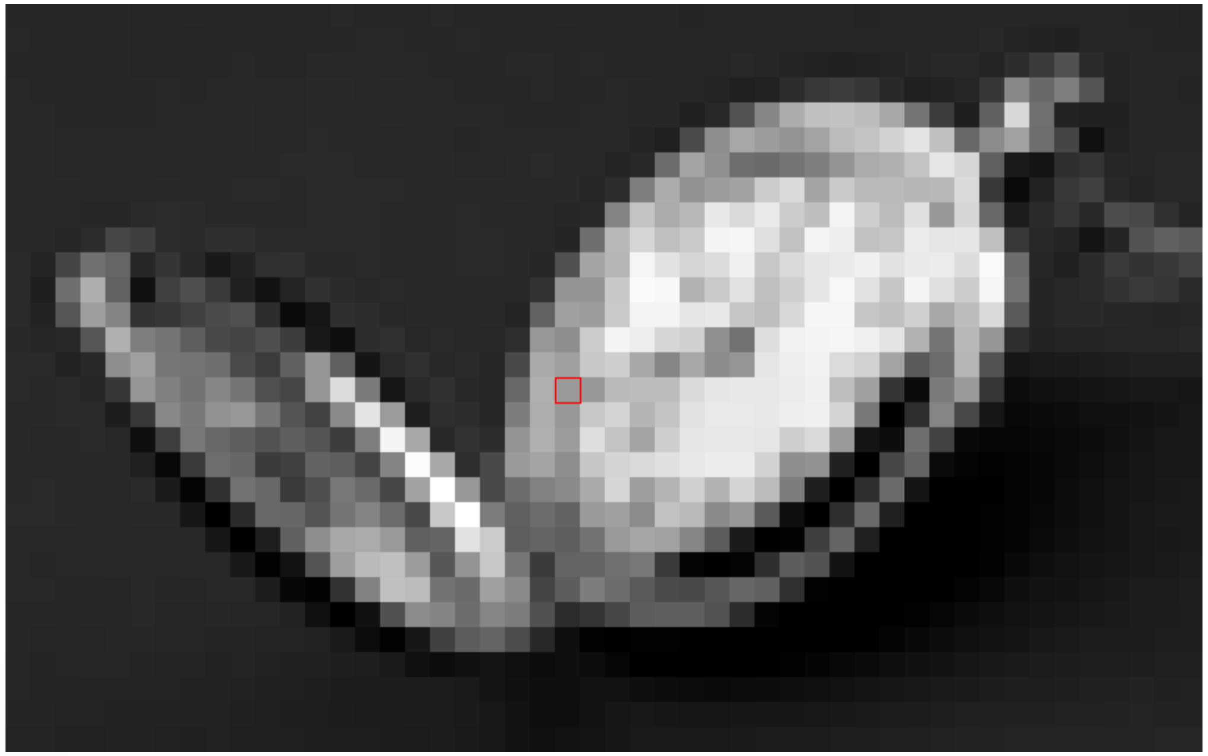
Bilinear Interpolation (in 2D)

- Interpolate in x then in y

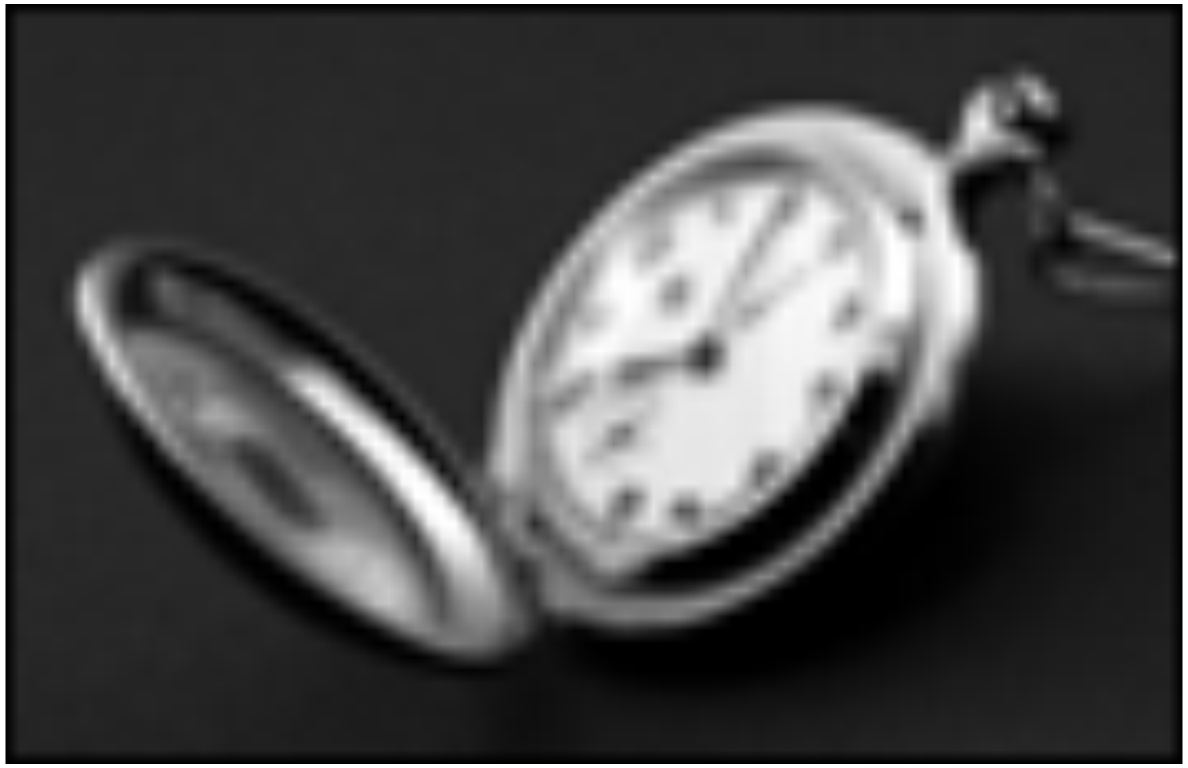
$$\begin{aligned} f(x,y) \approx\ & \frac{f(Q_{11})}{(x_2-x_1)(y_2-y_1)}(x_2-x)(y_2-y) \\ &+ \frac{f(Q_{21})}{(x_2-x_1)(y_2-y_1)}(x-x_1)(y_2-y) \\ &+ \frac{f(Q_{12})}{(x_2-x_1)(y_2-y_1)}(x_2-x)(y-y_1) \\ &+ \frac{f(Q_{22})}{(x_2-x_1)(y_2-y_1)}(x-x_1)(y-y_1). \end{aligned}$$



Comparison



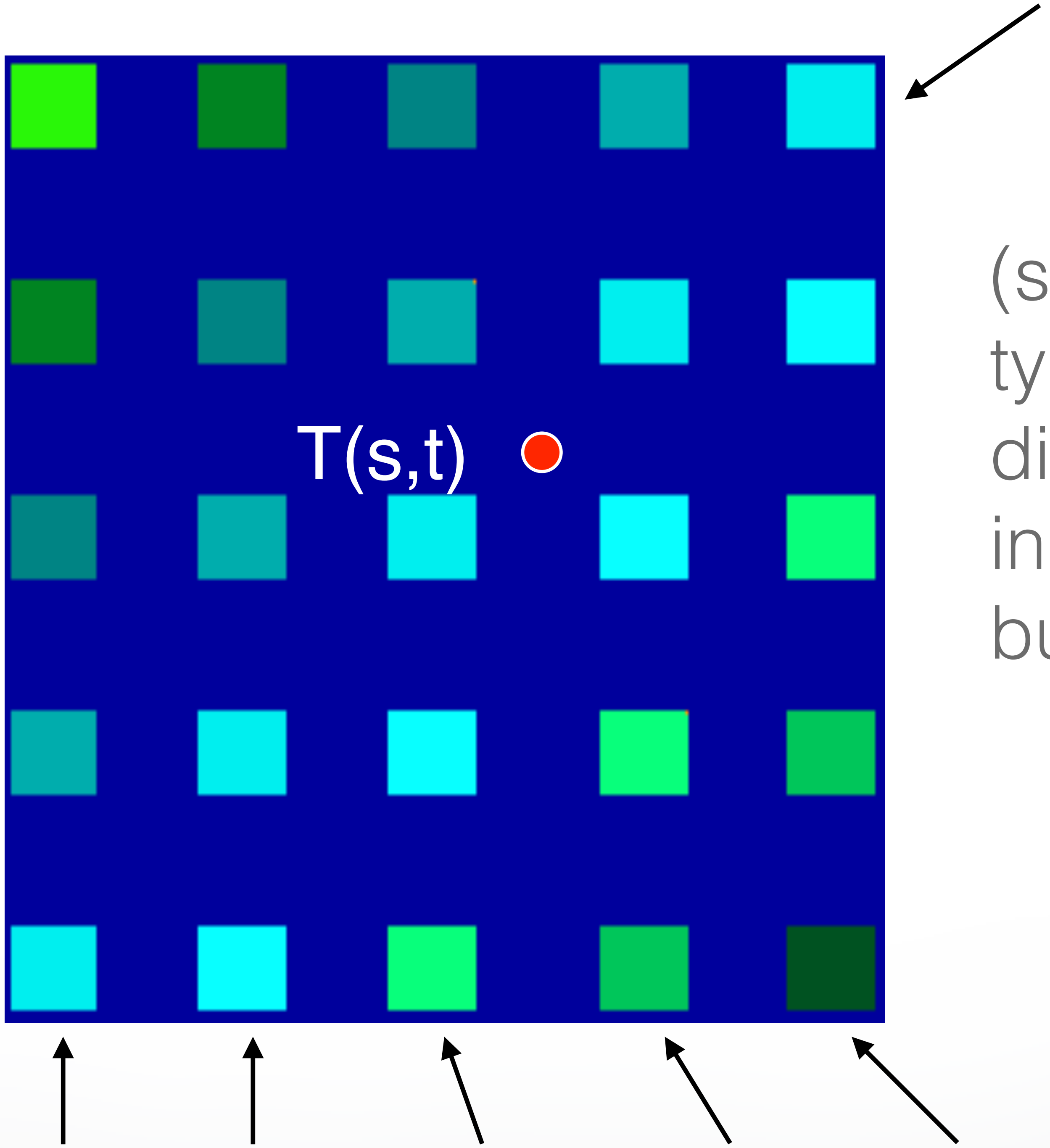
Nearest Neighbor



Bilinear

Texture interpolation

5 x 5 texture



(s,t) coordinates typically not directly at pixel in the texture, but in between

Texture Interpolation in OpenGL

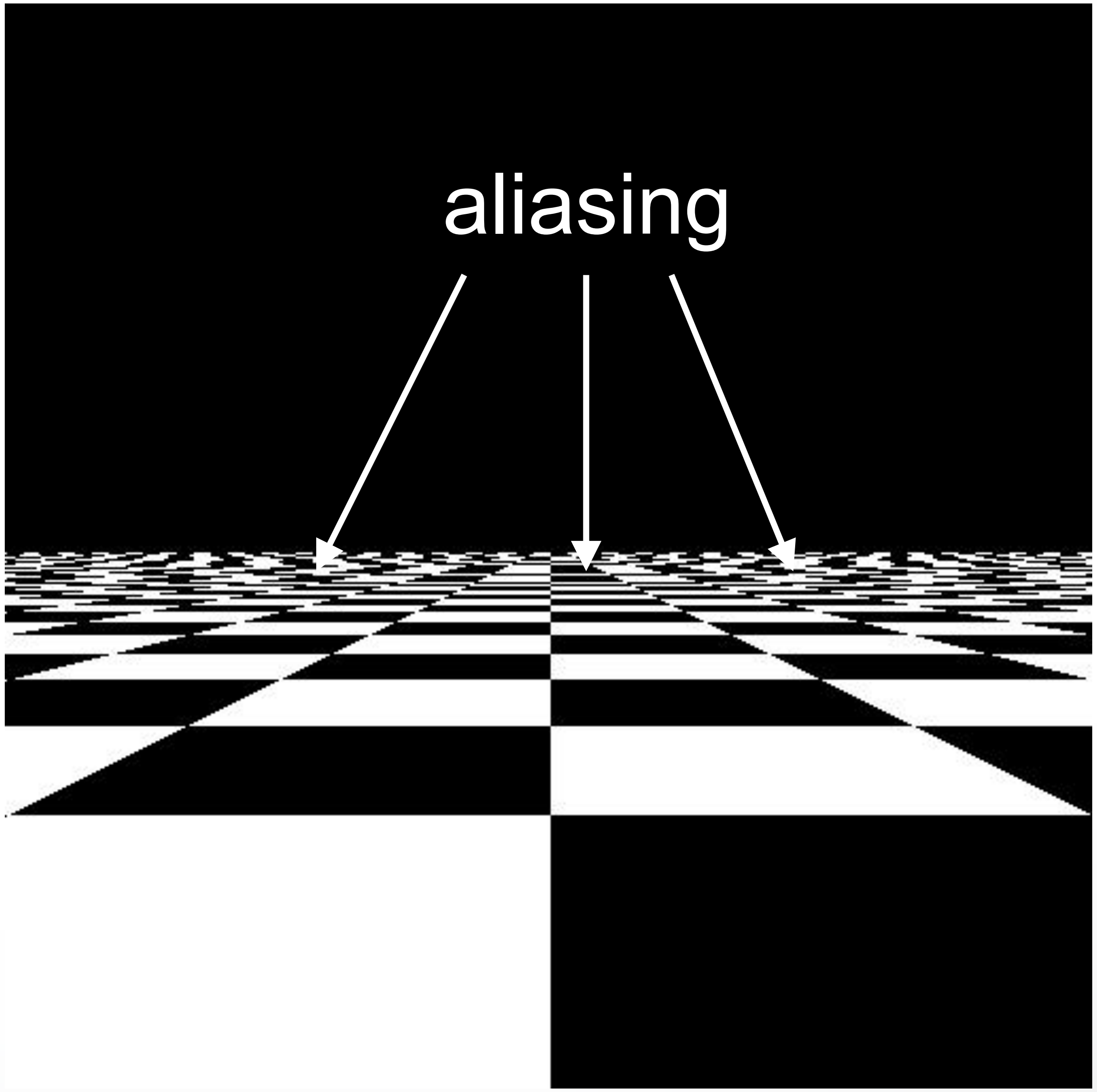
- (s,t) coordinates typically not directly at pixel in the texture, but in between
- Solutions:
  - Use the nearest neighbor to determine color
    - Faster, but worse quality

      ```
      glTexParameteri(GL_TEXTURE_2D,
                            GL_TEXTURE_MIN_FILTER, GL_NEAREST)
      ```
  - Linear interpolation
    - Incorporate colors of several neighbors to determine color
    - Slower, better quality

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

Filtering

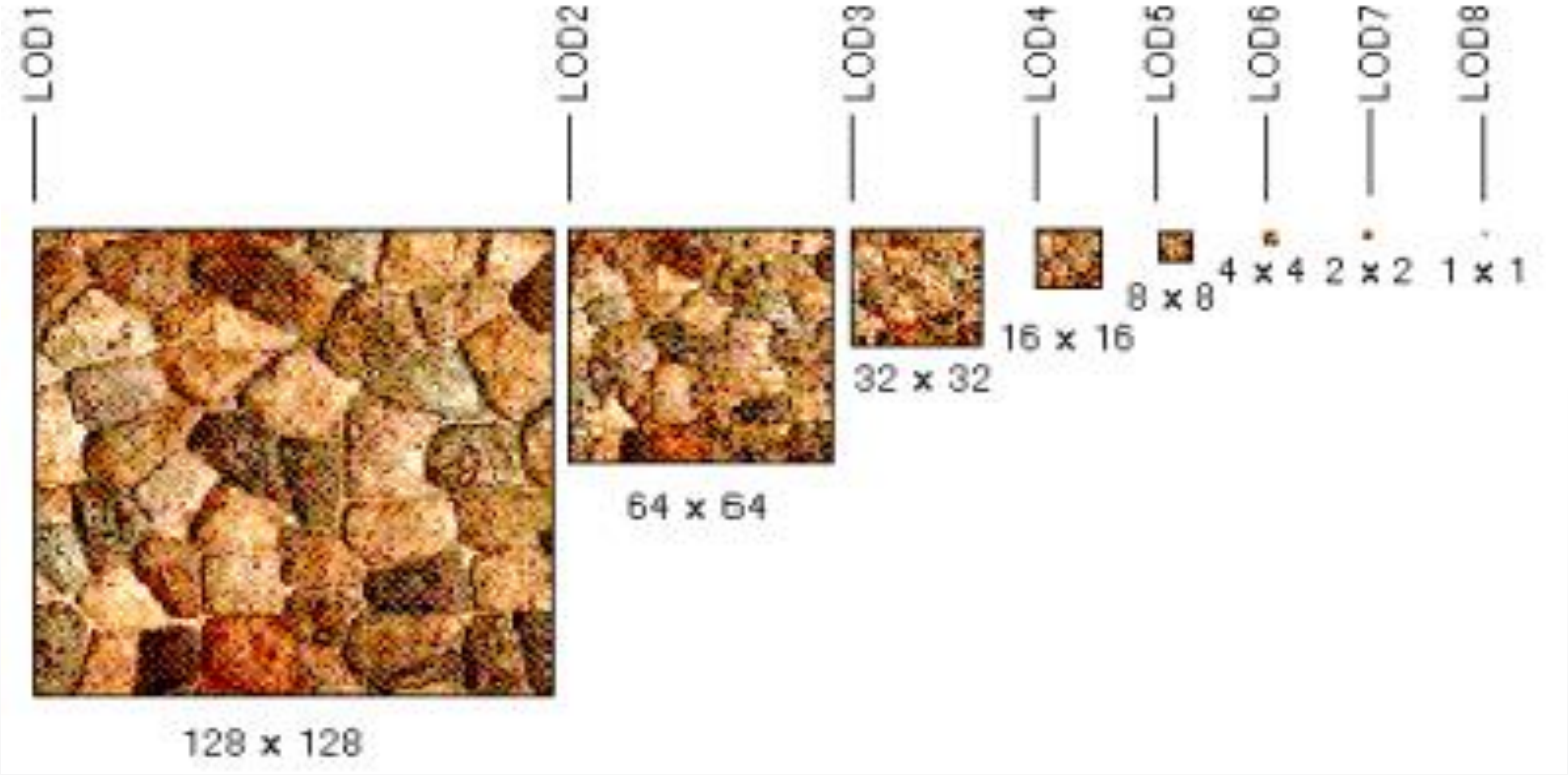
- Texture image is shrunk in distant parts of the image
- This leads to aliasing
- Can be fixed with *filtering*
  - bilinear in space
  - trilinear in space and level of detail (mipmapping)



Mipmapping

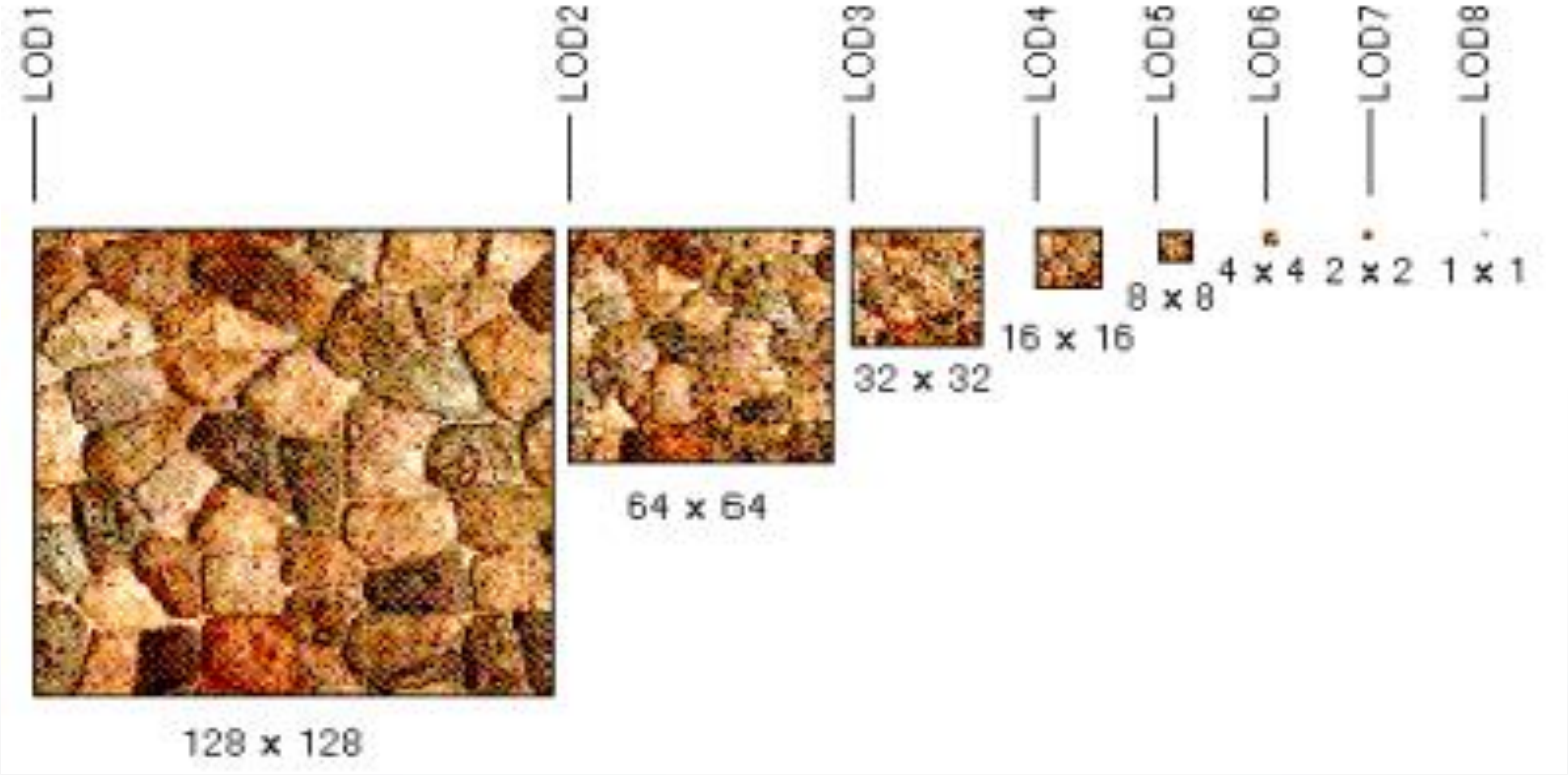
- Pre-calculate how the texture should look at various distances, then use the appropriate texture at each distance

- Reduces / fixes the aliasing problem



Mipmapping

- Each mipmap (each image below) represents a level of depth (LOD).
- Powers of 2 make things much easier.



Mipmapping in OpenGL

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

  - This will generate all the mipmaps automatically

- glTexParameterf(GL_TEXTURE_2D,
                                GL_TEXTURE_MIN_FILTER,
                                GL_NEAREST_MIPMAP_NEAREST)

  - This will tell GL to use the mipmaps for the texture

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
  glGenTextures(1, &texName); // must declare a global variable in
 program header: GLUint texName
  glBindTexture(GL_TEXTURE_2D, texName); // make texture
 "texName" the currently active texture
```

(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()
```

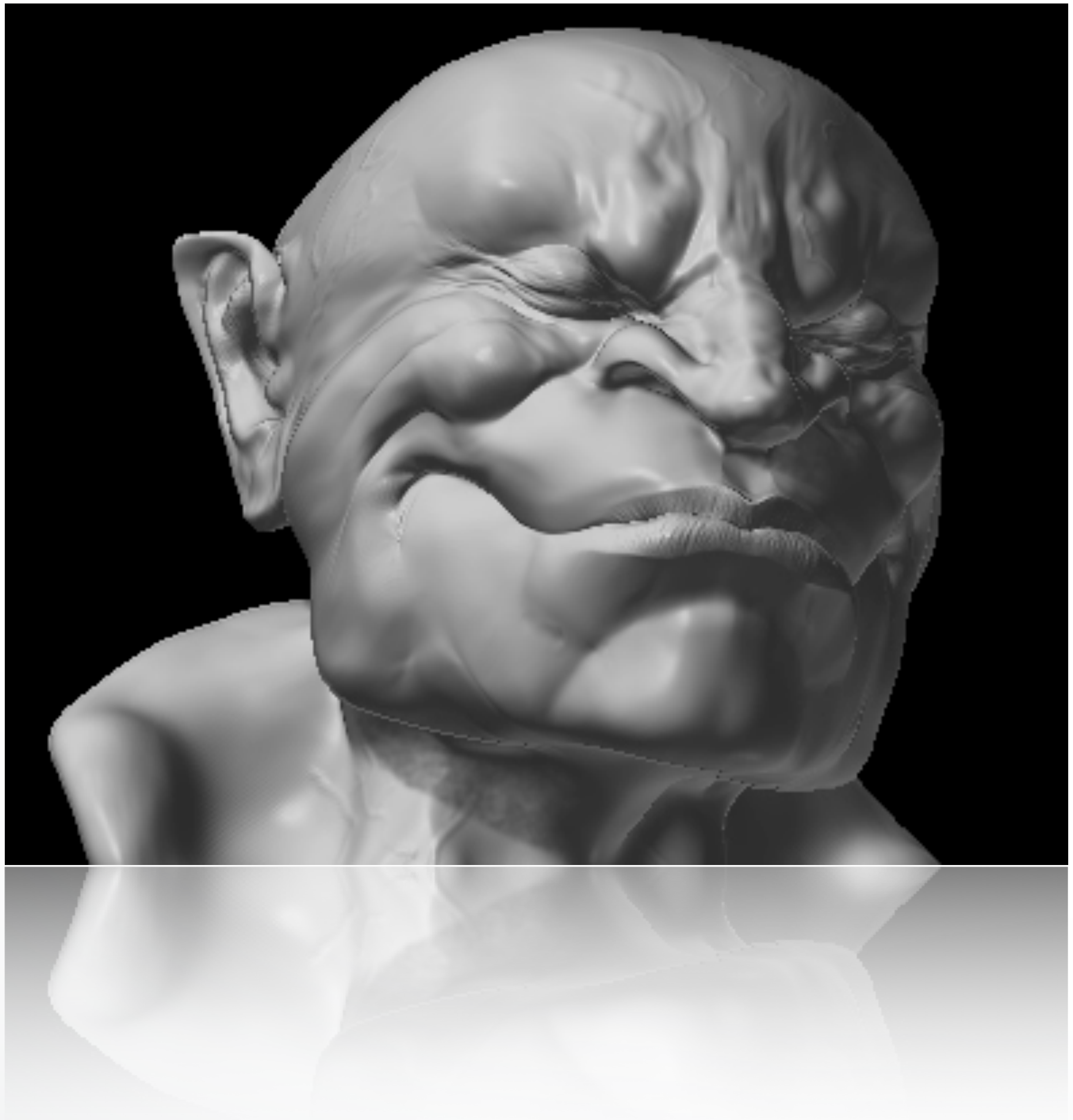
Outline

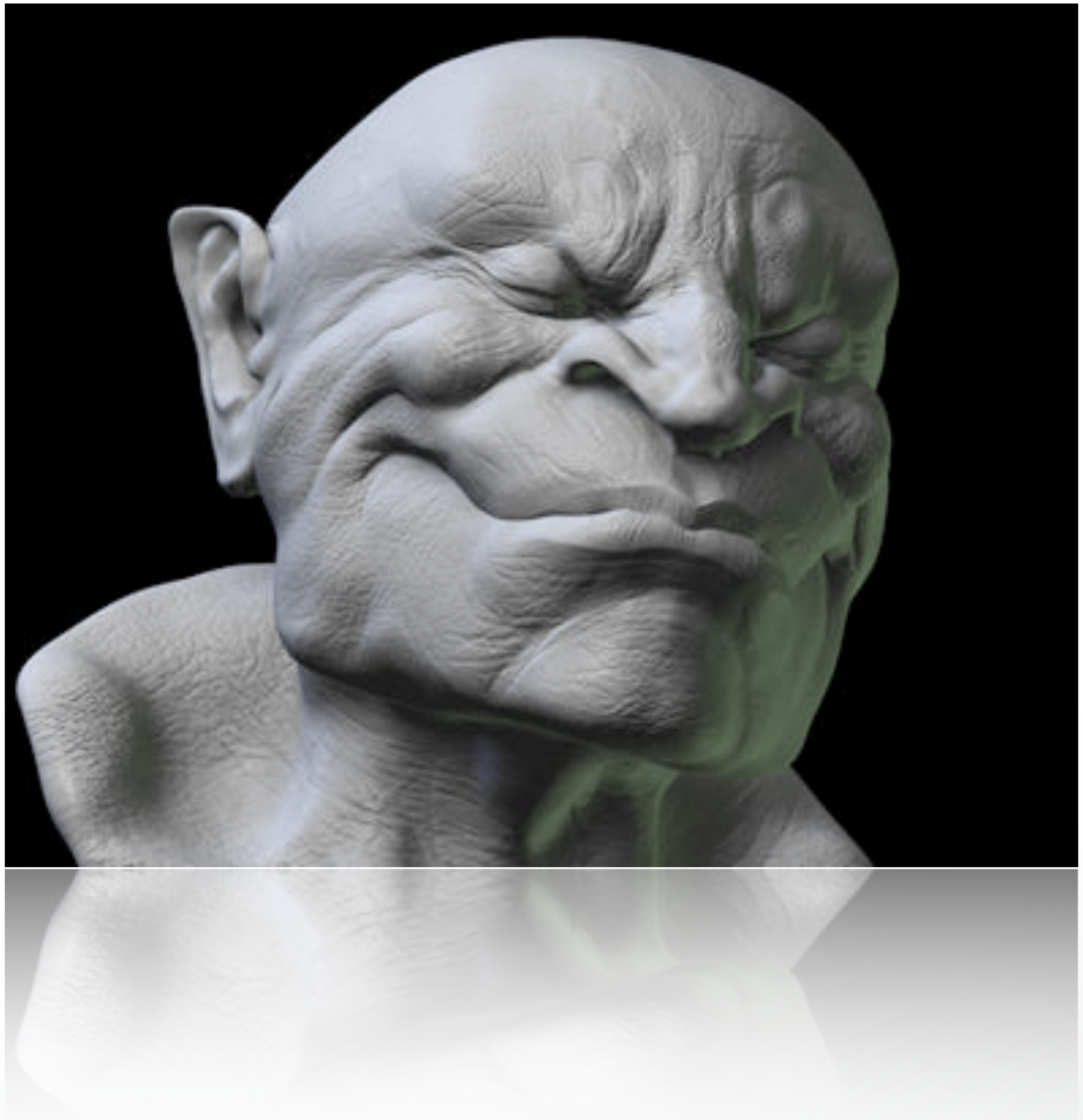
- Introduction
- Texture mapping in OpenGL
- Filtering and Mipmaps
- Example
- Non-color texture maps

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)

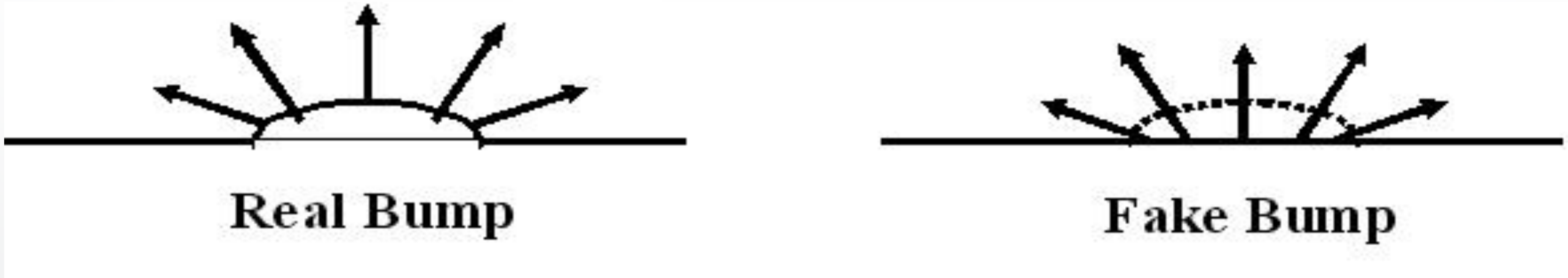
Bump mapping





Bump mapping

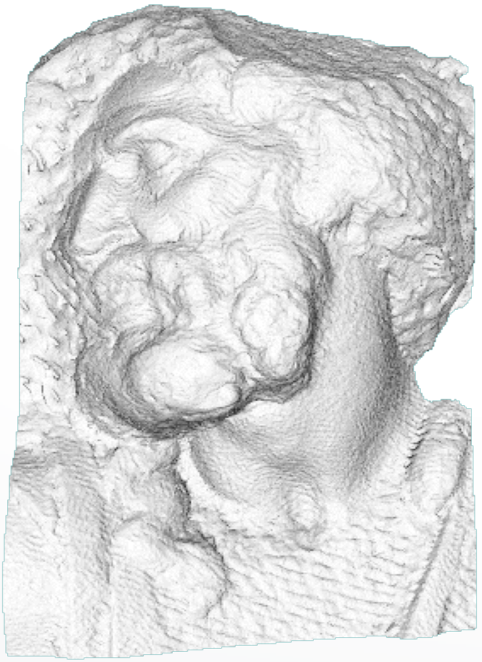
- How do you make a surface look *rough*?
  - Option 1: model the surface with many small polygons
  - Option 2: perturb the normal vectors before the shading calculation
    - Fakes small displacements above or below the true surface
    - The surface doesn't actually change, but shading makes it look like there are irregularities!
    - A texture stores information about the "fake" height of the surface



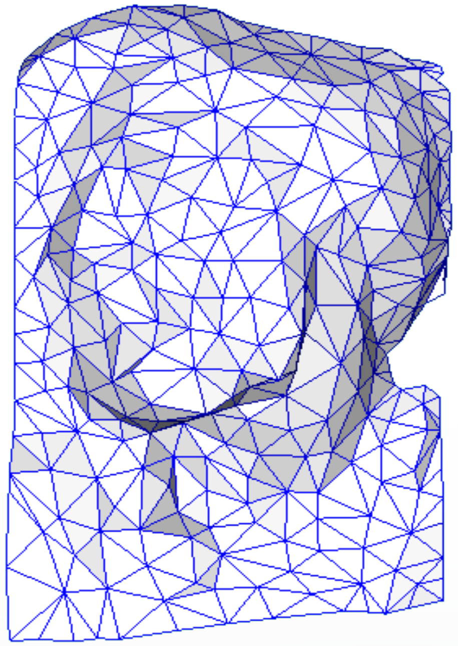
Bump mapping

- We can perturb the normal vector without having to make any actual change to the shape.

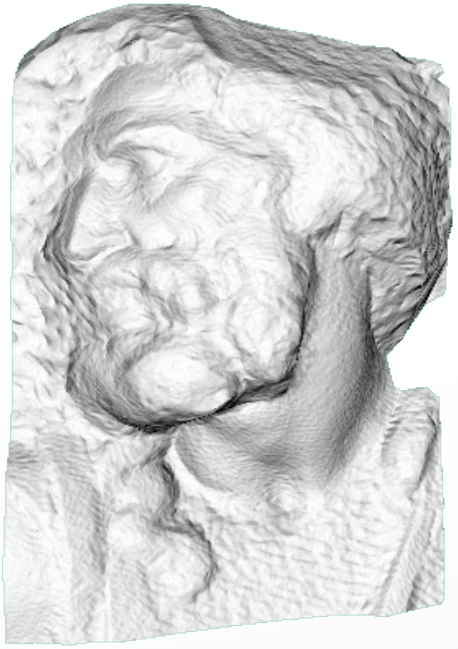
- This illusion can be seen through—how?



Original model (5M)



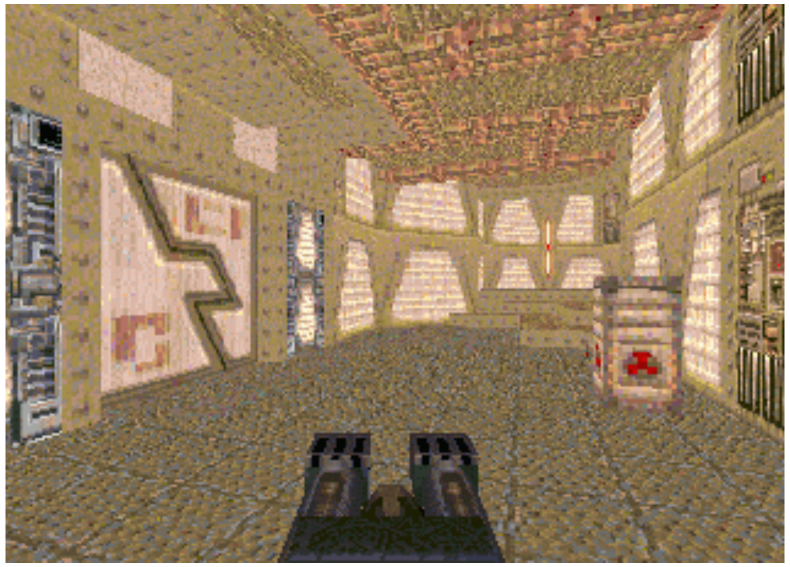
Simplified (500)



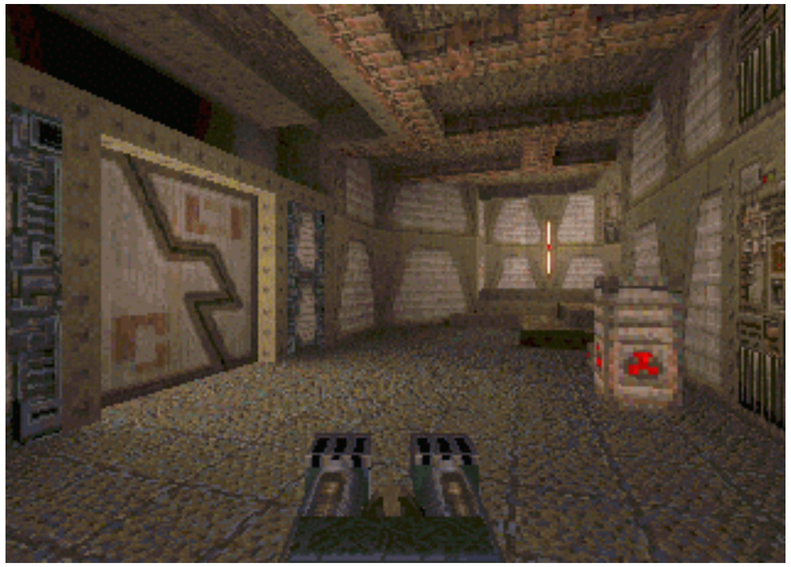
Simple model with bump map

Light Mapping

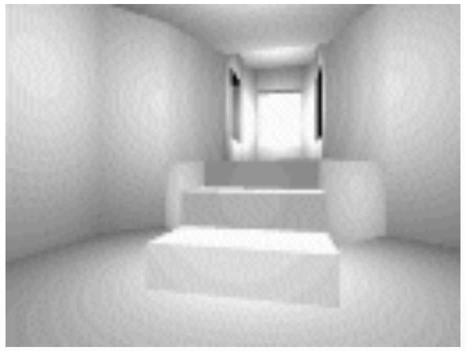
- *Quake* uses *light maps* in addition to texture maps. Texture maps are used to add detail to surfaces, and light maps are used to store pre-computed illumination. The two are multiplied together at run-time, and cached for efficiency.



Texture Map Only



Texture + Light Map



Light Map

Summary

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

http://cs420.hao-li.com

Thanks!