

**-=Catmull's Texturing=-**  
**1974**  
*but with shaders*

**Part I of Texturing**

Anton Gerdelan

# Textures

- **Edwin Catmull's** PhD thesis "*Computer display of curved surfaces*", 1974 – **U.Utah**
  - Also invented the z-buffer / depth buffer
  - [http://pixar.wikia.com/Ed\\_Catmull](http://pixar.wikia.com/Ed_Catmull)
- **J.F. Blinn & M.E.Newell** "*Texture and reflection in computer-generated images*" Communications ACM, 1976 (U.Utah guys again)
- James Blinn – NASA Jet Propulsion Lab - Voyager approach graphics
- Martin Newell – owner of the teapot, Xerox, Adobe, **Newell's algorithm** (depth-sorting related)
- Didn't have hardware for real-time rendering textures until the 1990s
- Algorithm:
  - Load surface colours from image file
  - **Map** colours to surface area
- First adopted in 1990 by...

(anyone know?)

# Ultima Underworld

1992, Blue Sky Productions



But beaten to market by...

# Catacomb 3-D

1991, Id Software

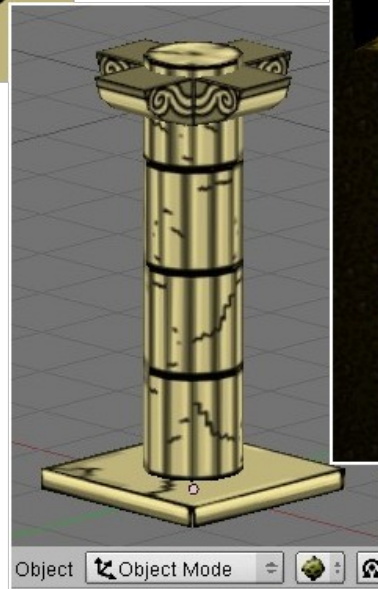


No hardware graphics acceleration yet - slower PCs - limited to walls

# Basic Texture-Application Process



Source image



Set up **texture coordinates** to map mesh to image

**Sample** texture at run-time in fragment shader



# Problem 1: Create Source Image

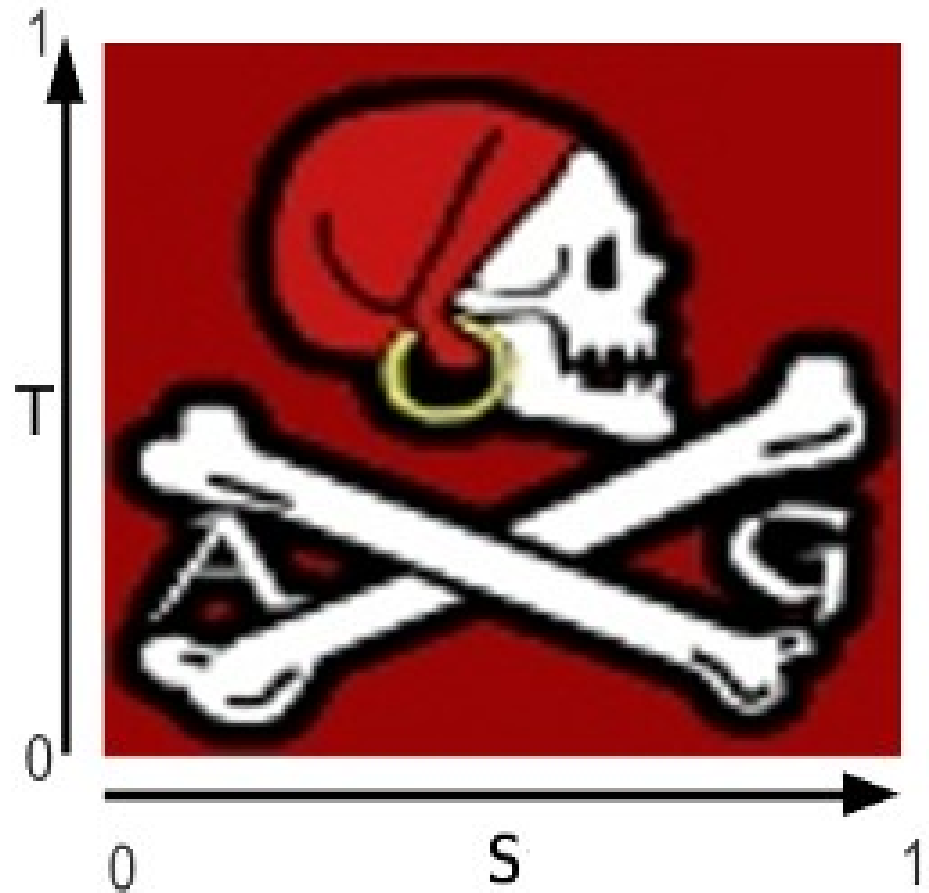
- How many colour channels?
- How many bits per colour channel?
- What file format?
- What resolution?
  - Older hardware/drivers only allowed **power-of-two** sizes

# Problem 2: Create Texture Coordinates

- Usually done in modelling software (Blender/Maya etc.)
- Can also be manually specified (for simple meshes)
- One **pair** of texture coordinates **per-vertex**
  - $[u,v]$  or  $[s,t]$
  - floats
  - `vec2`

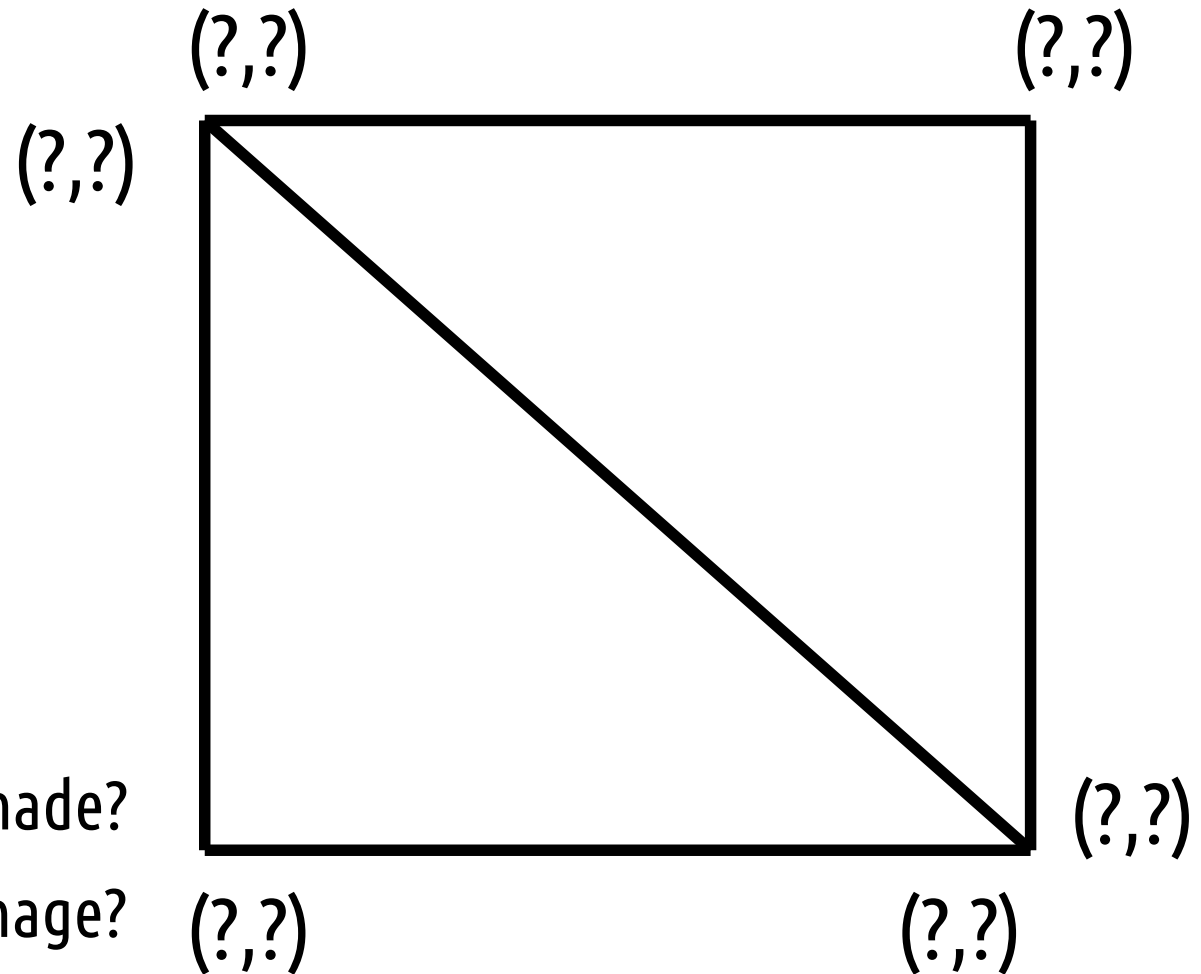
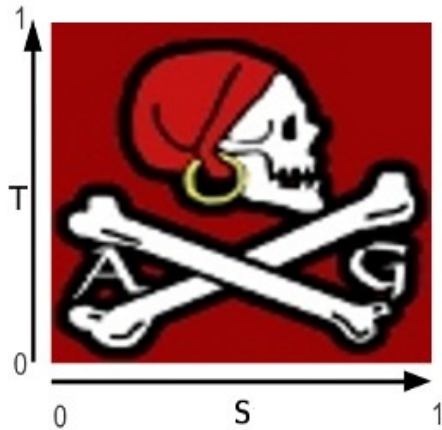
# Texture Coordinates

- Each vertex must map to a particular point on the image
- I recall Direct3D uses U,V and V is 0 at top, 1 at bottom
- **So for each vertex...**
  - Drag onto position on texture (Blender)  
**or**
  - Manually create s,t (C code)
  - Texcoords go in vertex buffer object like colours did
- **Q. Main difference to vertex colours is?**





# Texture Coordinates



- Q. What coords?
- Q. Why are some repeated?
- Q. What mistakes could be made?
- Q. Why is that a good test image?

# Main Problem (not part of algorithm)

- Preparing “texture” is a bit of work
  - Load image file with **image loader library**
  - Decoded image RGB[A] bytes are in main memory
  - **Create** OpenGL texture object
  - **Copy** image bytes into texture (goes to graphics memory)
  - Set up texture **filtering** properties of texture
  - Set up texture **wrapping** properties of texture
  - **Bind** texture into active texture slot number 0

# Image-Loader Library

- Sean Barrett's **stb\_image.h** is awesome (and simple and small and doesn't need linking)  
<https://github.com/nothings/stb>
- **libPNG** is kind of complicated to use, but okay
- **SOIL** loads and creates GL textures too, but a bit OOD
- You can write your own – RAW formats and TGA are pretty simple to load manually
  - It will help a lot to know how to read/write raw binary image files

# Image-Loader Library

- Library **decodes** image from one or more **compressed** image formats
- You get a block of RGBARGBARGBARGBA or RGBRGB...
- And hopefully image **width** and **height** dimensions
- You need to know how to copy memory using **pointers** and **addresses**
- = Pretty much the same as copying points data into a VBO
- You may need to flip the texture upside-down

# Bonus Question

- What does this code do?

```
if ((x & (x - 1)) != 0 || (y & (y - 1)) != 0) {
```

# Relevant GL Functions to Load Texture

- `glGenTextures`
- `glActiveTexture (GL_TEXTURE0);` ← Always call before binding a texture
- `glBindTexture`
- `glTexImage2D` ← Copies actual image data into texture
  - `GL_RGBA` or `GL_RGB`
- `glTexParameteri` ← Must call 4 times and set these 4 parameters or may not work at all
  - `GL_TEXTURE_WRAP_S`
  - `GL_TEXTURE_WRAP_T`
  - `GL_TEXTURE_MAG_FILTER`
  - `GL_TEXTURE_MIN_FILTER`

# Texture Parameters: Wrapping

- Defines what happens when using a texture coordinate outside range of 0 to 1
- Set parameter separately for S and T direction
  - GL\_CLAMP\_TO\_EDGE
  - GL\_CLAMP\_TO\_BORDER
  - GL\_MIRRORED\_REPEAT
  - GL\_REPEAT
  - GL\_MIRROR\_CLAMP\_TO\_EDGE
- Let's just try it (**demo time**)

# Texture Parameters: Min and Mag Filters



**Magnification aliasing** – walls are lower resolution than on-screen pixels (Tomb Raider, Eidos Interactive, 1996)



# Texture Parameters: Min and Mag Filters

Visible flicker  
when  
camera moves



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

# Texture Parameters: Min and Mag Filters

**Nearest-neighbour  
algorithm**

- sometimes leaf
- sometimes gap



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

# Texture Parameters: Min and Mag Filters

- Built-in **anti-aliasing** filters
- Mag. Filters:
  - Nearest neighbour
  - Bi-linear interpolation
- Min. Filters:
  - Nearest neighbour
  - Bi-linear
  - Various MiP-Maps (look at these later)
- **Q. Why would you WANT to disable these anti-aliasing filters?**

# Problem 3: Sampling a Texture

- Built-in functionality to sample a texture in a fragment/pixel shader
- We must get texture coordinates to the fragment shader somehow.
- **Q. How?**

# Sampling a Texture (almost)

- In to the vertex shader with other per-vertex data
- Out of the vertex shader where it is...

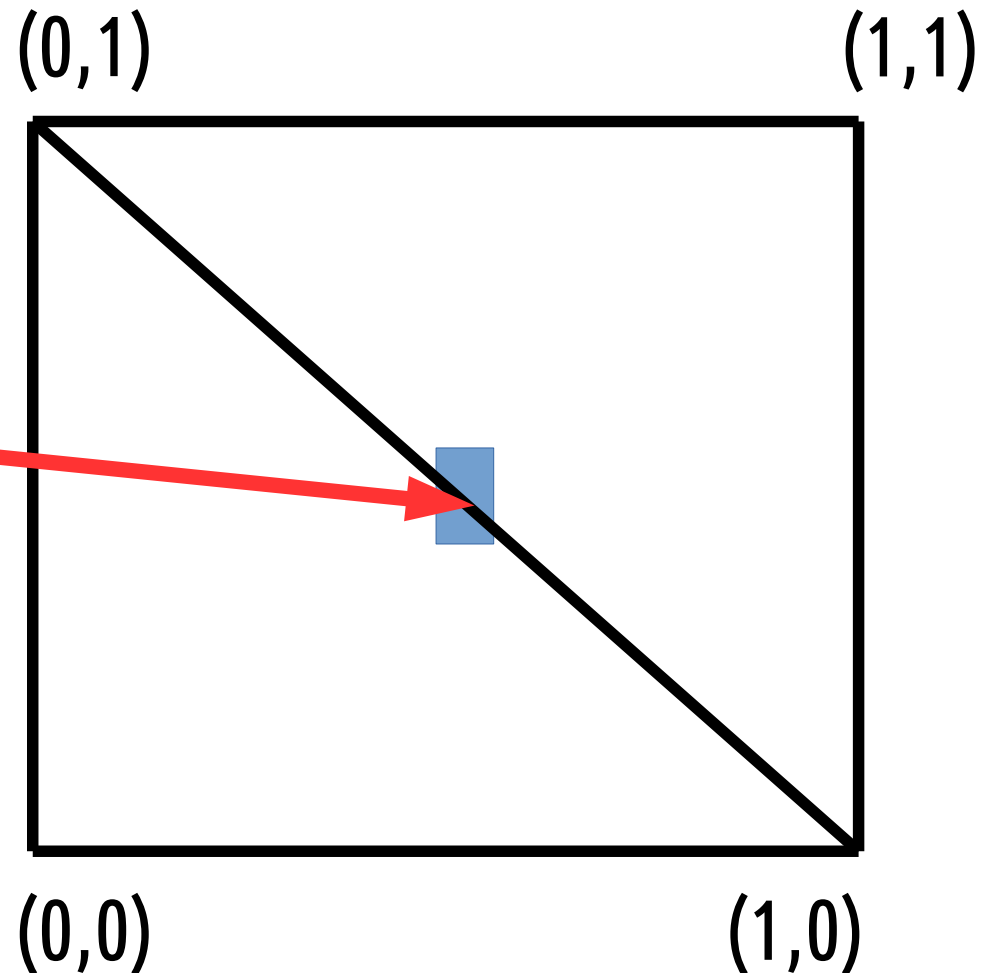
## Q. What happens to it here?

- ...available as an input to the fragment shaders


```
test_vs.gsl x
1 #version 400
2
3 layout (location = 0) in vec3 vertex_position;
4 layout (location = 1) in vec2 vt; // per-vertex texture co-ords
5
6 uniform mat4 view, proj;
7
8 out vec2 texture_coordinates;
9
10 void main() {
11     texture_coordinates = vt;
12     gl_Position = proj * view * vec4(vertex_position, 1.0);
13 }
```

# Sampling a Texture

- Q. What texcoords should this fragment's shader get?
- Q. What is this process called again?



# Sampling a Texture in a Fragment Shader

- Add a “`sampler2D`” uniform – knows texture params
- Is actually an integer
- Its value is **active texture #** (slot) to use
- We bound into `glActiveTexture (GL_TEXTURE0);` 
- Uniforms are = 0 by default
- There are max. ~8-32 “active” textures in total (at one time)
  - we don't need the others yet

# Sampling a texture

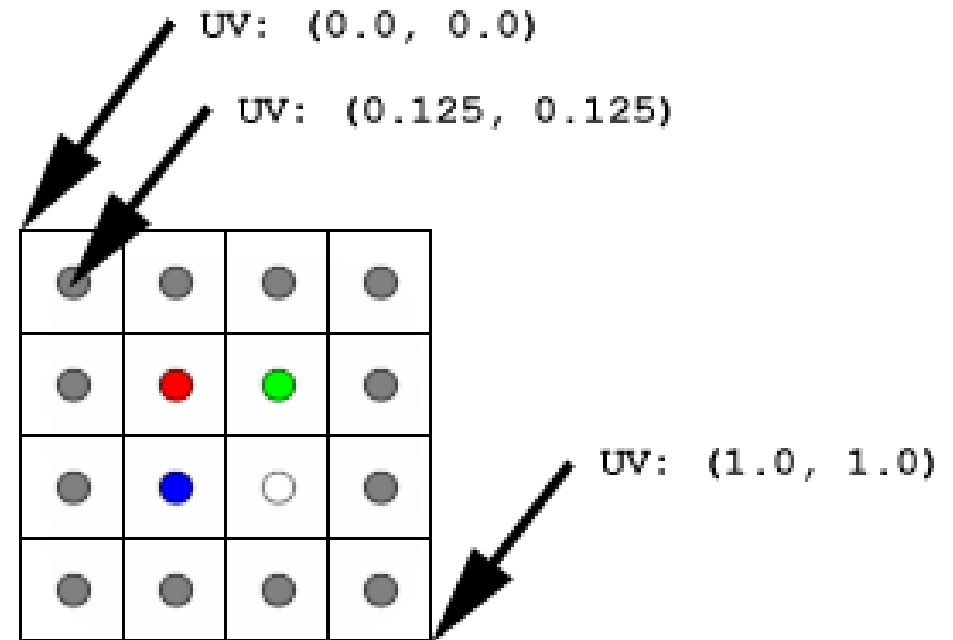
- `texture()` function samples texture
  - `texture2D()` on older versions of GL
- Returns a **texel**.
- **Q. Why is this not just a “pixel”?**

```
test_fs.glsl x
1 #version 400
2
3 in vec2 texture_coordinates;
4 uniform sampler2D basic_texture;
5 out vec4 frag_colour;
6
7 void main() {
8     → vec4 texel = texture(basic_texture, texture_coordinates);
9     → frag_colour = texel;
10 }
```



# Texel = “Texture Element”

- Without filtering = closest pixel in image
- With bi-linear the texel is actually a weighted blend of surrounding pixels
- **Q. What RGB colour do you expect at UV (0.5, 0.5) with bi-linear filtering?**



**Figure 7b**

*Texels can be thought of as off-screen equivalents to pixels, where each texel is defined at the exact center of a grid cell.*

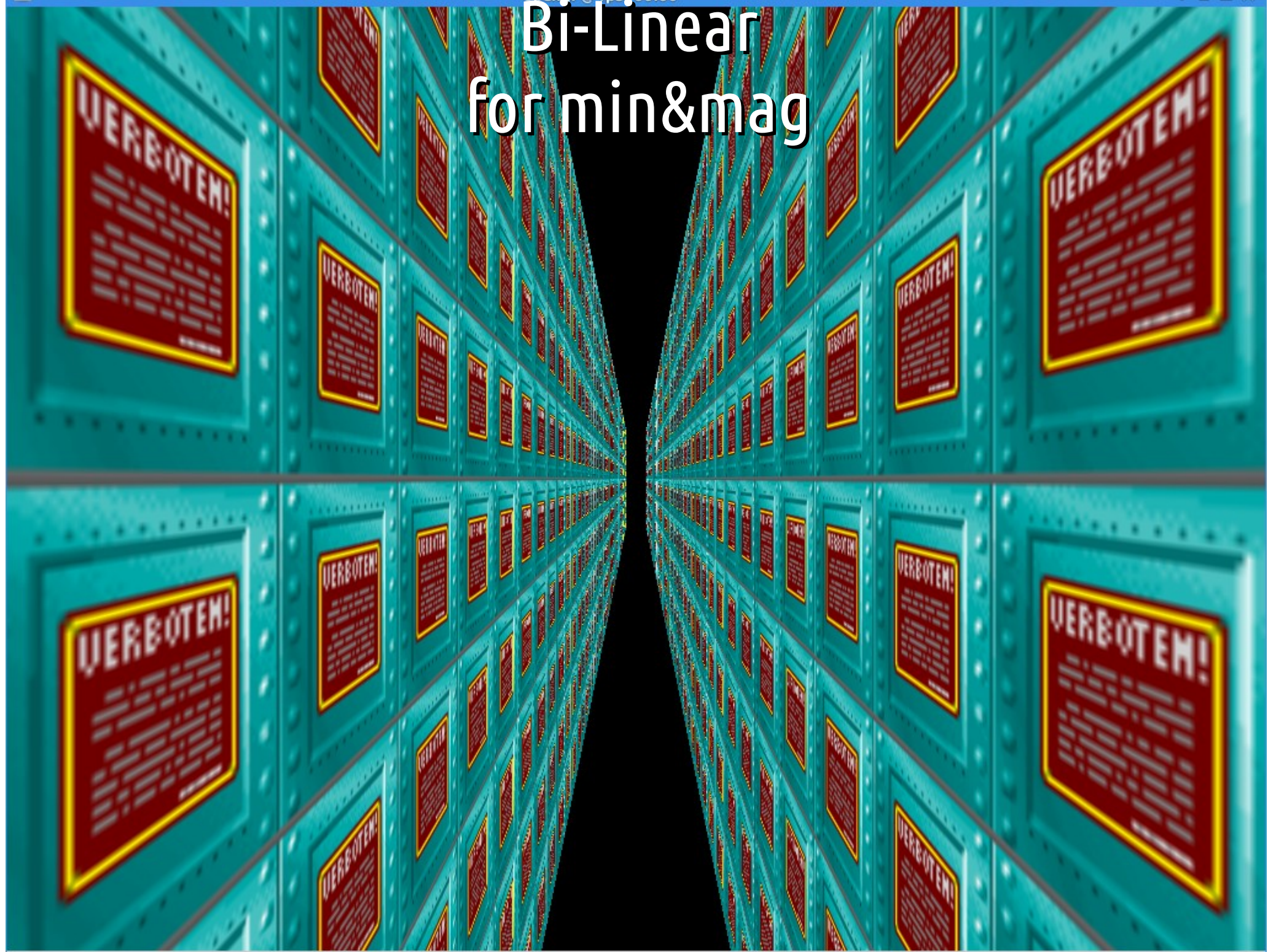
Image Source: [msdn.microsoft.com](http://msdn.microsoft.com)

# Nearest Neighbour for min&mag





# Bi-Linear for min&mag



# And in Part II

- MiP-Maps
- Multi-Texturing
- Environment Maps
- \*More advanced texture stuff to come after lighting

*Prepare yourself for*

# Texturing Part 2

advanced texture-mapping algorithms



# Multim im Parvo [Maps]

- Lance Williams,  
“*Pyramidal parametrics*”,  
1983
- **Q. Guess where  
Williams studied?**
- **Mipmapping** is a texture  
filtering algorithm for  
minification filtering



—————> `glGenerateMipmap (GL_TEXTURE_2D);`

# Multim im Parvo [Maps]

- Pre-calculate a set of images
- 256x256, 128x128...1x1
- Sum of these is 1/3 of the original
- Generating Mipmaps for a texture
  - Pre-calculated
  - Super-fast hardware extension
  - DDS
- Sampler chooses next-smallest sub-image to fit actual size



# Enabling Basic Mipmaps

```
GLTexParameteri (  
    GL_TEXTURE_2D,  
    GL_TEXTURE_MIN_FILTER,  
    GL_NEAREST_MIPMAP_NEAREST  
);
```

- Basic MM with NN min filtering

```
GLTexParameteri (  
    GL_TEXTURE_2D,  
    GL_TEXTURE_MIN_FILTER,  
    GL_LINEAR_MIPMAP_NEAREST  
);
```

- Basic MM with bi-linear filtering



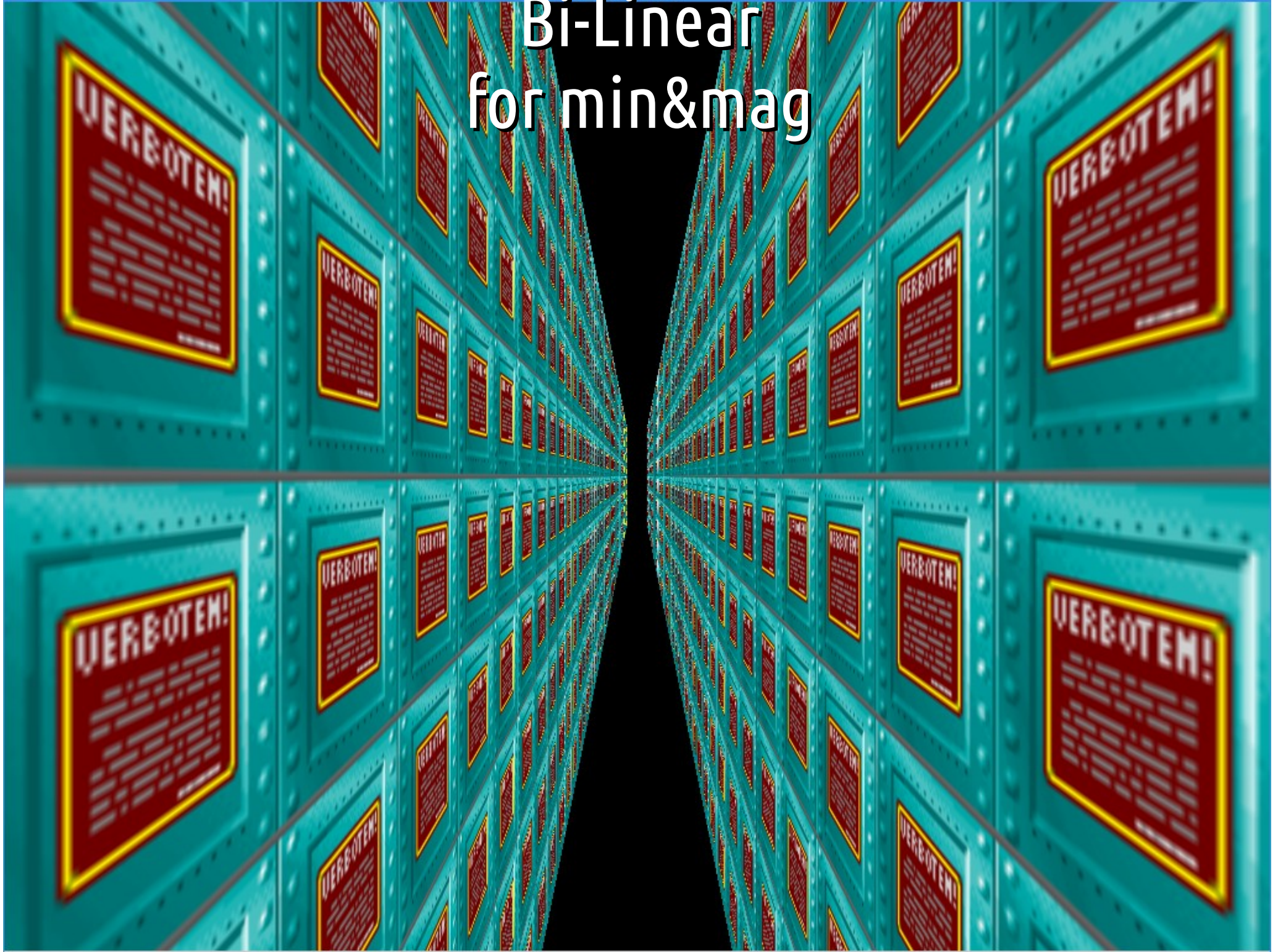
# Multim im Parvo [Maps]

- As textured geometry is smaller/farther away...
- Rendering is faster
- Higher quality – no more aliasing due to minification + nearest neighbour
  - smaller images are already **anti-aliased**
- Uses 1/3 more memory
- **Q. Why 1/3?**



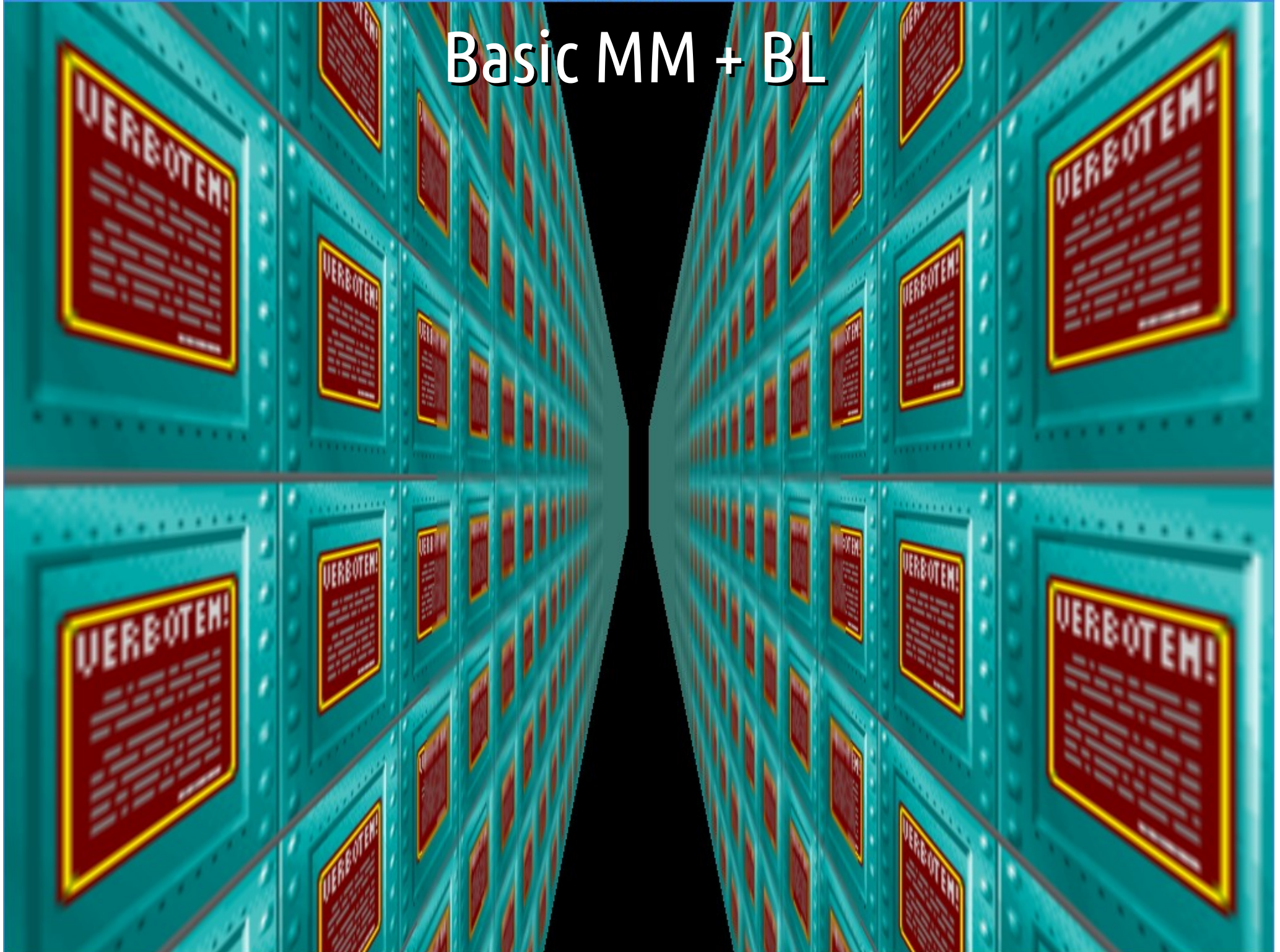


# Bi-Linear for min&mag





# Basic MM + BL



# Tri-Linear Filtering

- Instead of having the sampler choose the next-biggest mipmap...
- Have them choose the next-smaller **AND** next-biggest and linearly blend between the two

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



# Trilinear Filtering



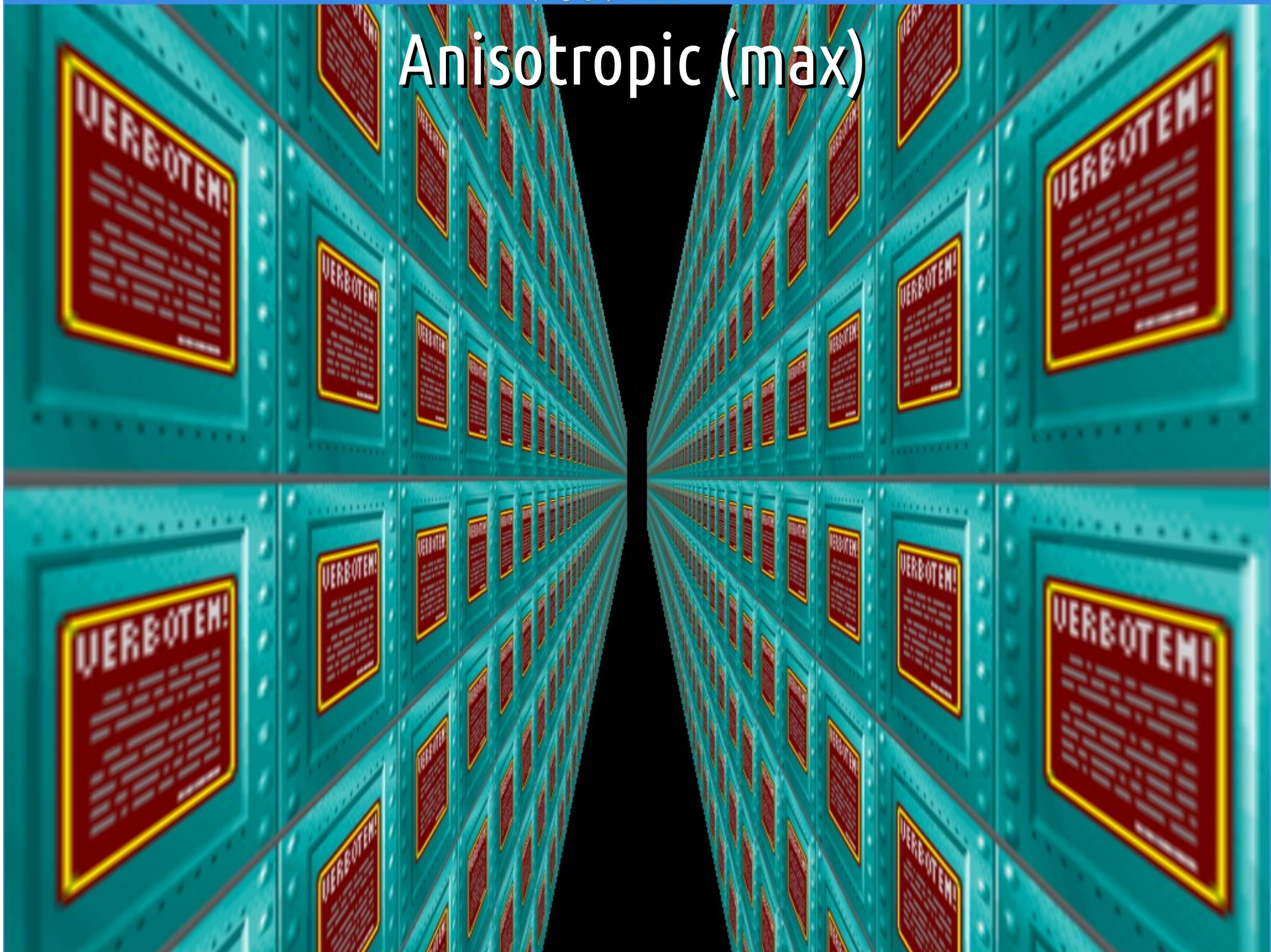
# Anisotropy and MiP-Maps

- Main problem with our example is that textures are at an acute angle to the viewer
- Isotropy = not direction dependent, an-isotropy is
- Area requiring texture is high but very narrow
- Next smallest mipmap is the narrow size x narrow size
- Solution – generate rectangular sizes too  
256x128 128x256, 128x64, 64x128...

```
GLfloat max_aniso = 0.0f;  
GLGetFloatv (GL_MAX_TEXTURE_MAX_ANISOTROPY_EXT, &max_aniso);  
// set the maximum!  
glTexParameterf (GL_TEXTURE_2D,  
GL_TEXTURE_MAX_ANISOTROPY_EXT, max_aniso);
```



# Anisotropic (max)



# Further Reading

- Multi-Texturing
  - Use more than 1 texture sampler in frag shader
  - Blending edges together smoothly
  - GUI controls / progress bars / health meters etc.
- Cube maps
  - Sky boxes w/ parallax effect
  - Reflection (fake/static or dynamic)
  - Refraction (water, fluids etc.) - fake usually
- Texture projection
  - Very fancy lights
  - Shadows



# Further Reading

- Real Time Rendering book
  - Advanced shading chapter
  - Texture mapping chapter
- Fundamentals
  - Texturing mapping
- Tutorials for the previous slide!
  - Worth doing for extra points in project