Textures

Joaquim Madeira

November 2020

Overview

- Motivation
- Textures
- Texture Mapping
- Texture Features
- Applying Textures in WebGL

MOTIVATION

Geometric Modeling – Limits

- Graphics cards can render millions of triangles per second
- BUT, that might not be sufficient…
- Skin / Terrain / Grass / Clouds / …

How to model / render an orange ?

- An orange colored sphere ?
  - Too simple !

- A more complex shape to convey details ?
  - How to represent surface features ?
  - Takes too many triangles to model all the dimples…

How to model / render an orange ?

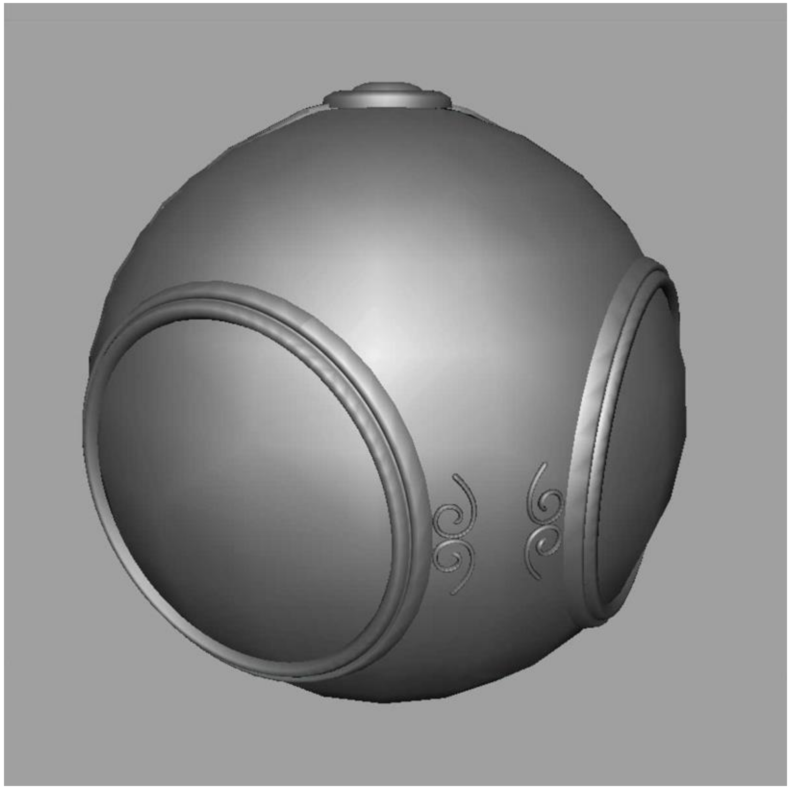
- Simple geometric model + Texture
  - Take a picture of a real orange
  - Scan and “paste” it onto model
  - Texture mapping

- Might not be sufficient: surface will be smooth

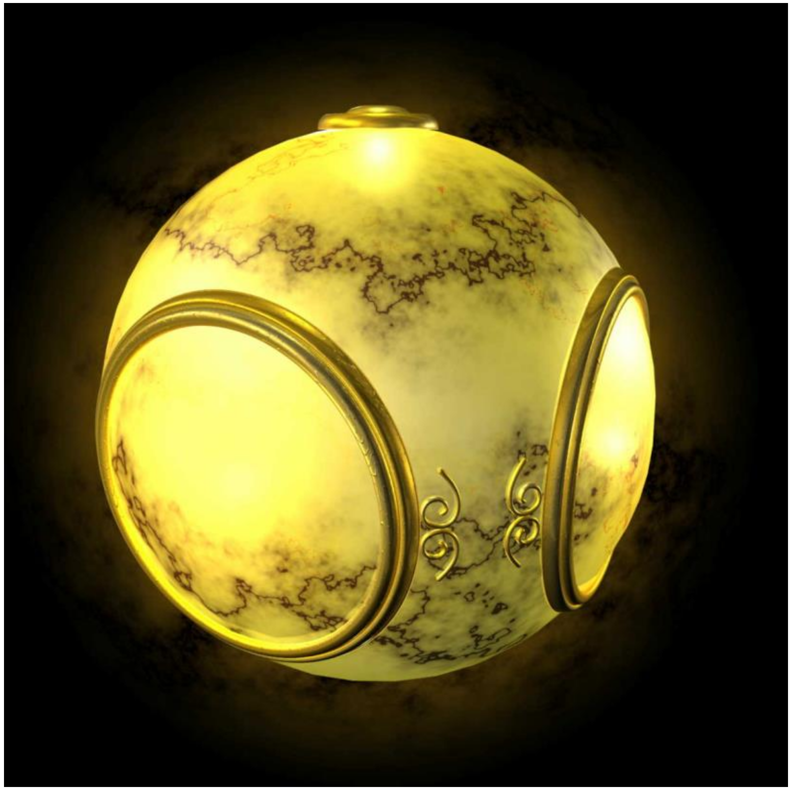
- How to “change” local shape ?
  - Bump mapping

TEXTURES

Texture Mapping



geometric model



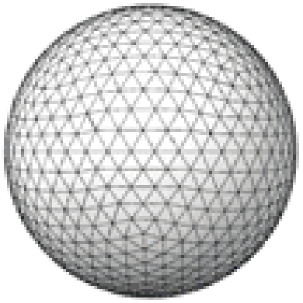
texture mapped

[Ed Angel]

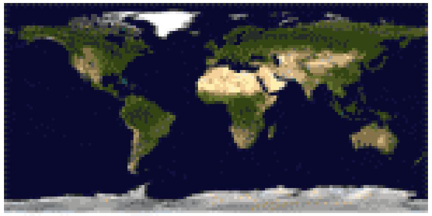
Texture Mapping

- Implemented in hardware on every GPU
- Simplest surface detail hack

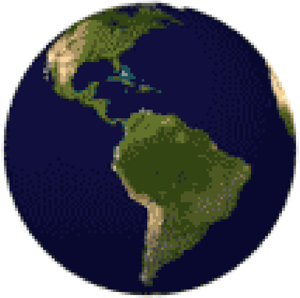
- Paste the texture on a surface to add detail without adding more triangles
  - Get surface color or alter computed surface color



Sphere with no texture



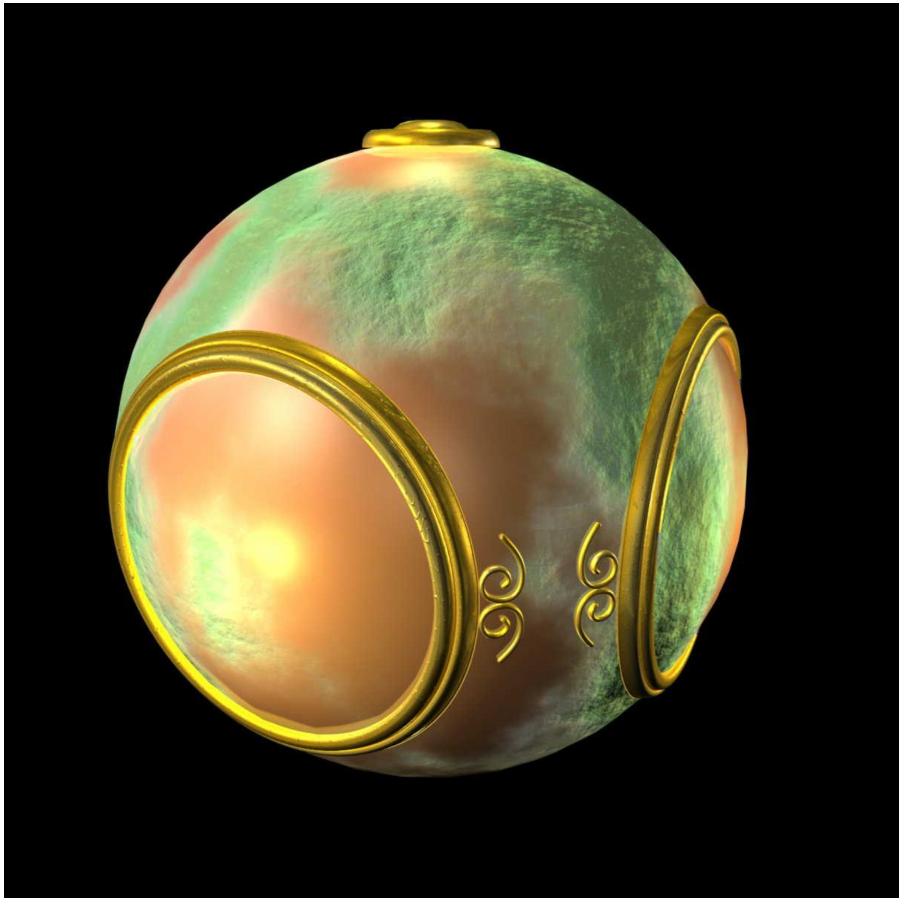
Texture image



Sphere with texture

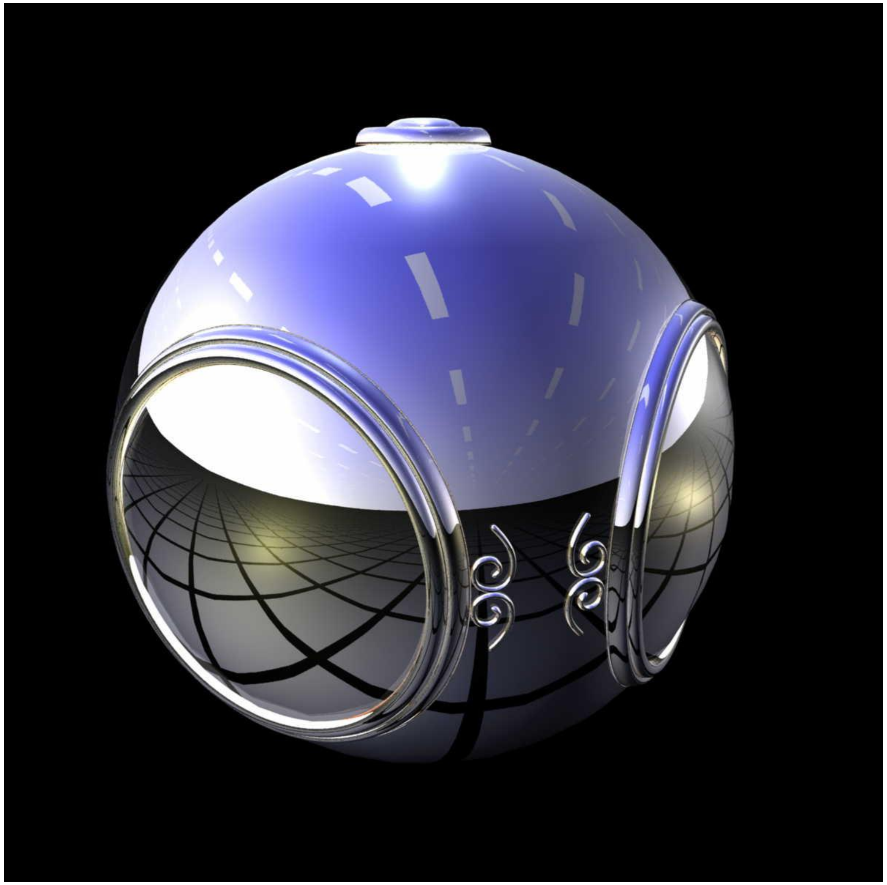
[Andy Van Dam]

Bump Mapping



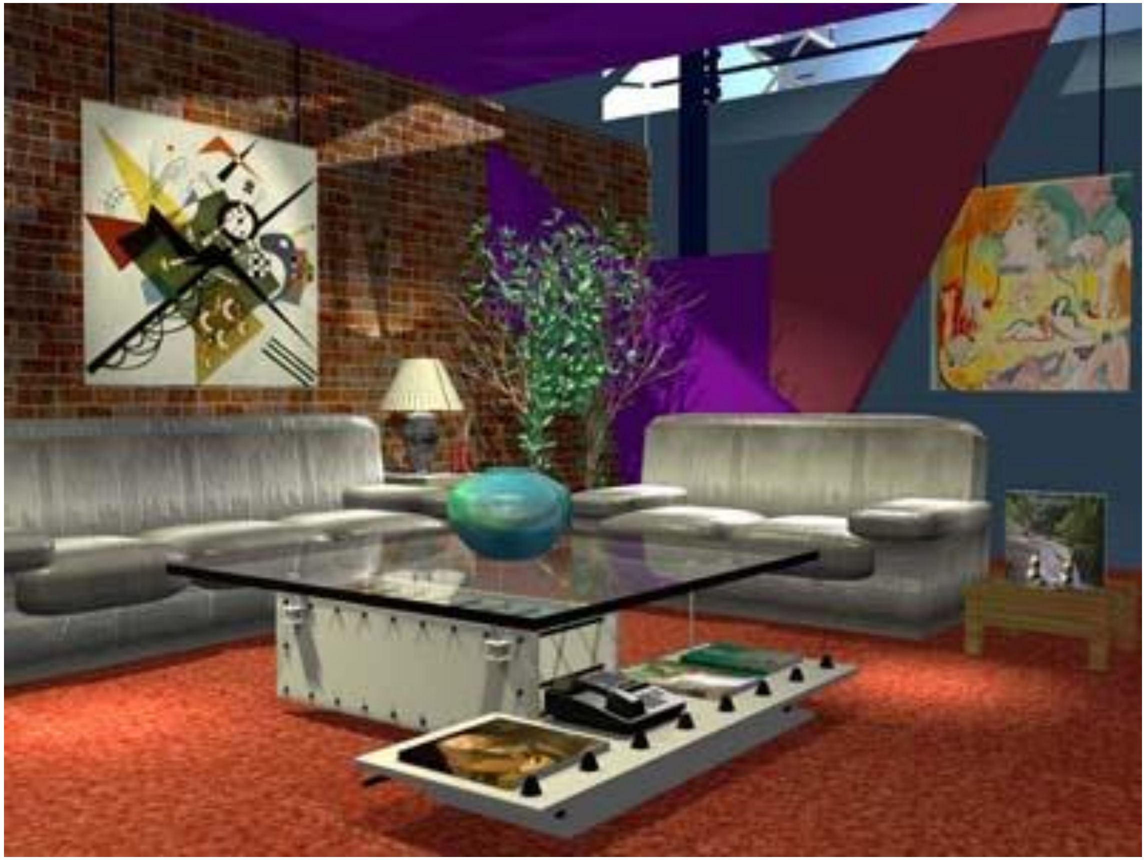
[Ed Angel]

Environment Mapping



[Ed Angel]

Textures – Simulating Ray-Tracing



[http://www.okino.com]

- Increased realism !!
  - 11 light sources + 25 texture maps

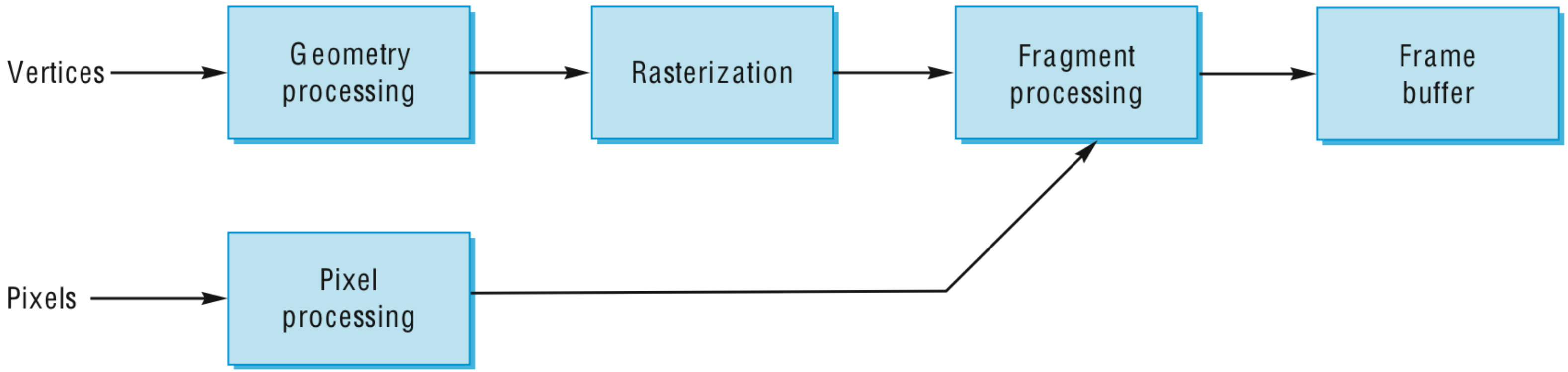
TEXTURE MAPPING

Mapping

- Texture Mapping
  - Uses images to fill inside of triangles
- Bump mapping
  - Emulates altering normal vectors during the rendering process
- Environment (reflection mapping)
  - Uses a picture of the environment for texture maps
  - Allows simulation of highly specular surfaces

Where does it take place ?

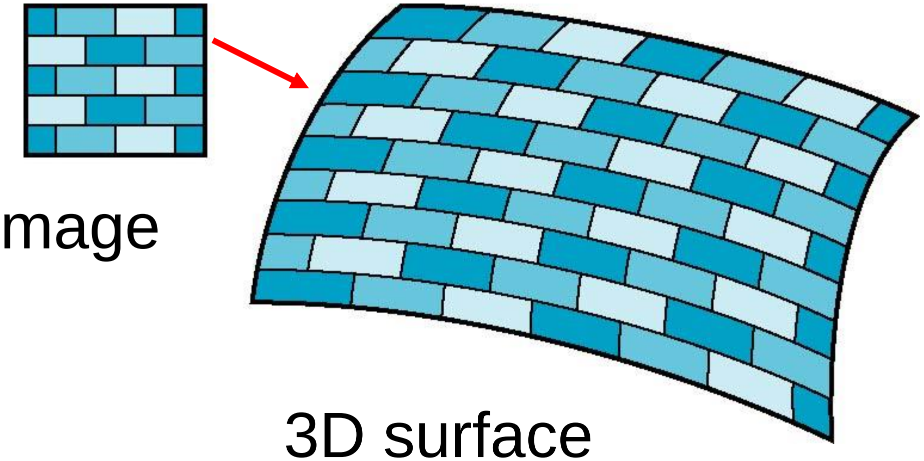
- Mapping techniques are implemented at the end of the rendering pipeline
  - Very efficient because few polygons make it past the clipper



[Ed Angel]

Mapping – Is it simple ?

- Although the idea is simple – map an image to a surface – there are 3 or 4 coordinate systems involved

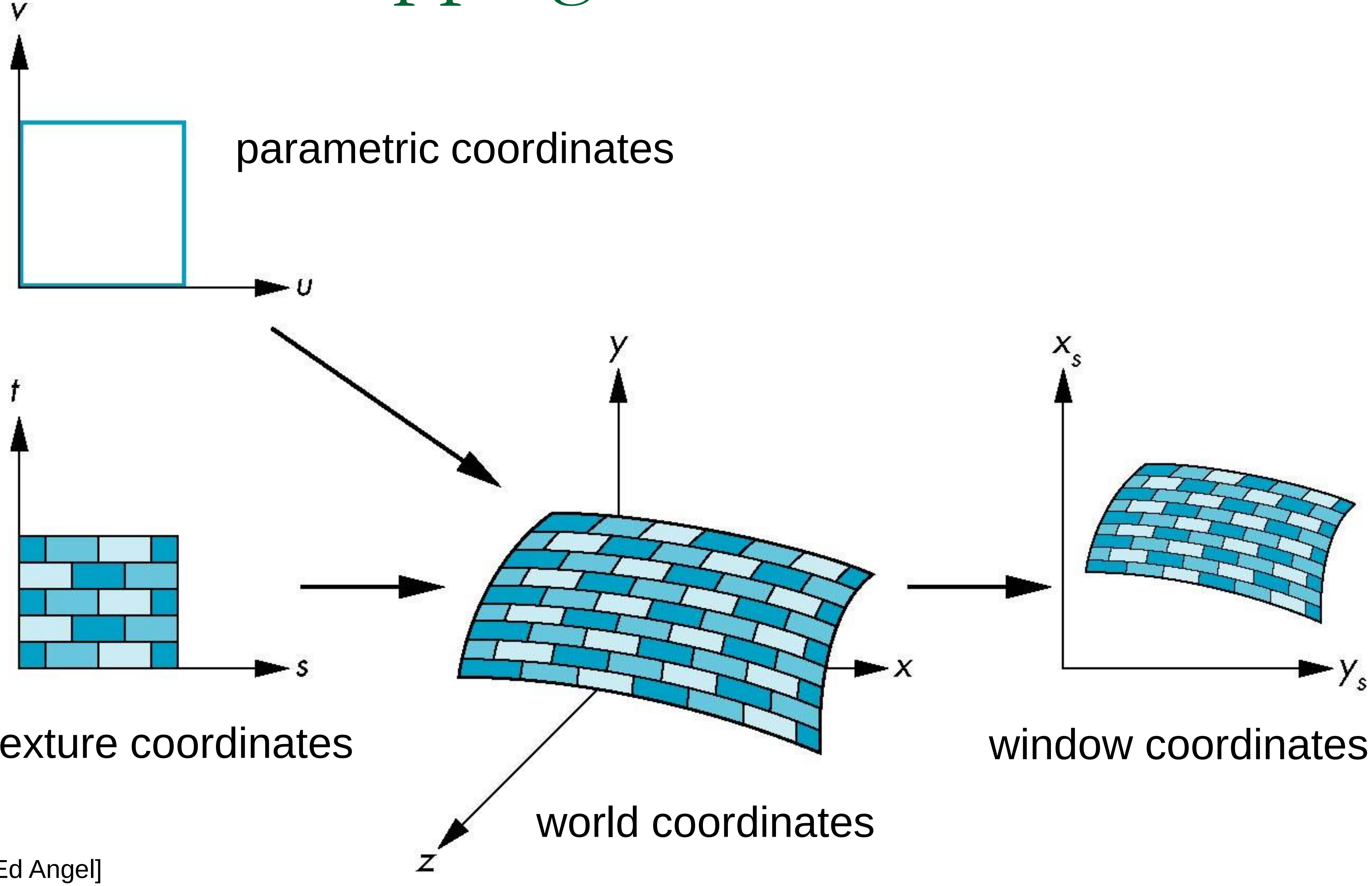


[Ed Angel]

Coordinate Systems

- Parametric coordinates
  - May be used to model surfaces
- Texture coordinates
  - Used to identify points in the image to be mapped
- Object or World Coordinates
  - Conceptually, where the mapping takes place
- Window Coordinates
  - Where the final image is really produced

Texture Mapping



parametric coordinates

texture coordinates

world coordinates

window coordinates

[Ed Angel]

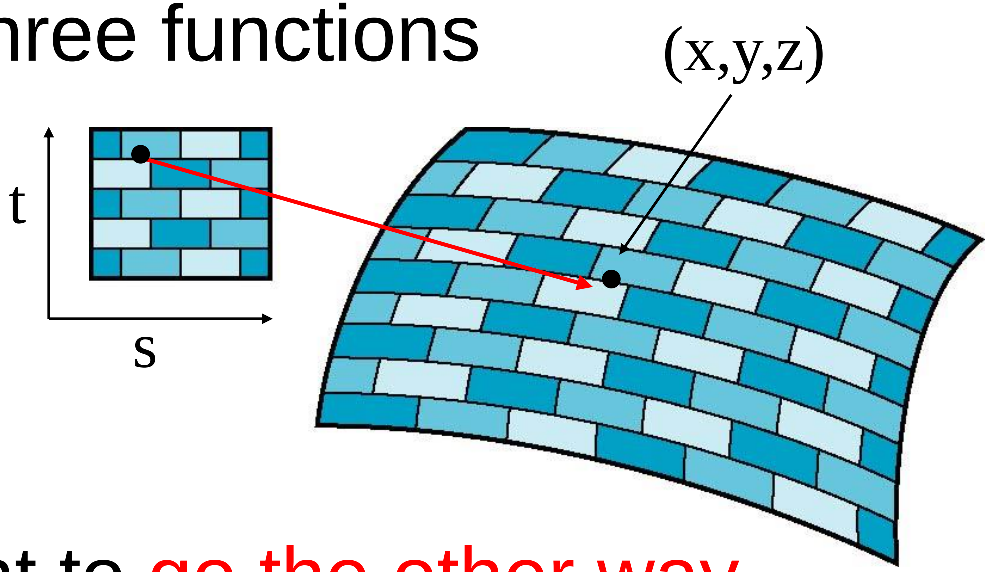
Mapping Functions

- Mapping from texture coordinates to a point on a surface
- Appear to need three functions

$x = x(s,t)$

$y = y(s,t)$

$z = z(s,t)$



- But we really want to go the other way

[Ed Angel]

Backward Mapping

- Given a pixel, we want to know to which point on an object it corresponds

- Given a point on an object, we want to know to which point in the texture it corresponds

- Need a map of the form

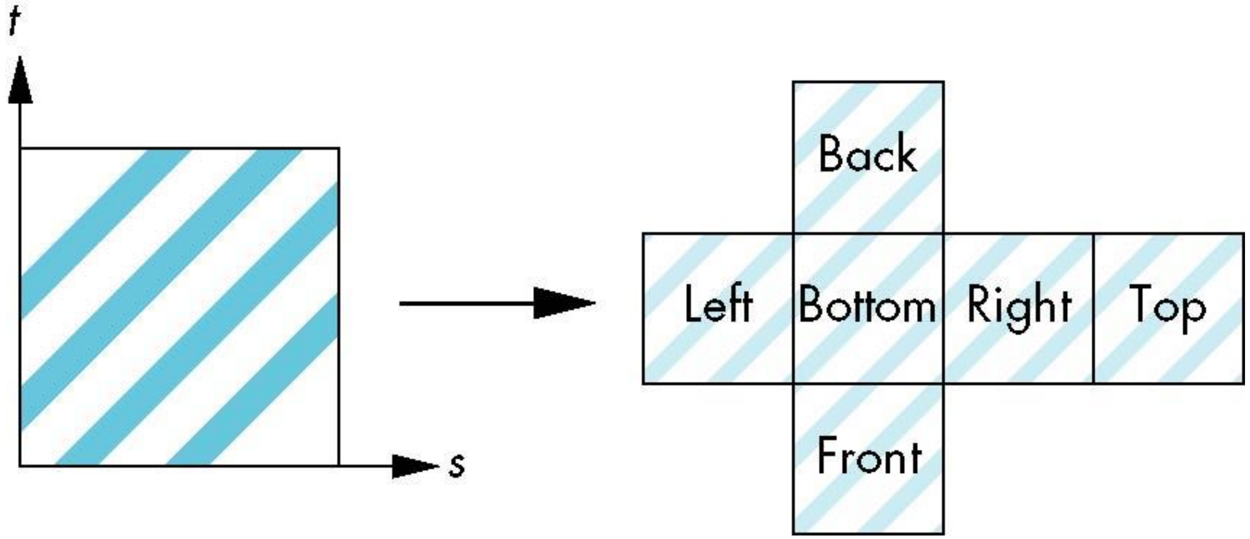
  $s = s(x,y,z)$

  $t = t(x,y,z)$

- Such functions are difficult to find in general

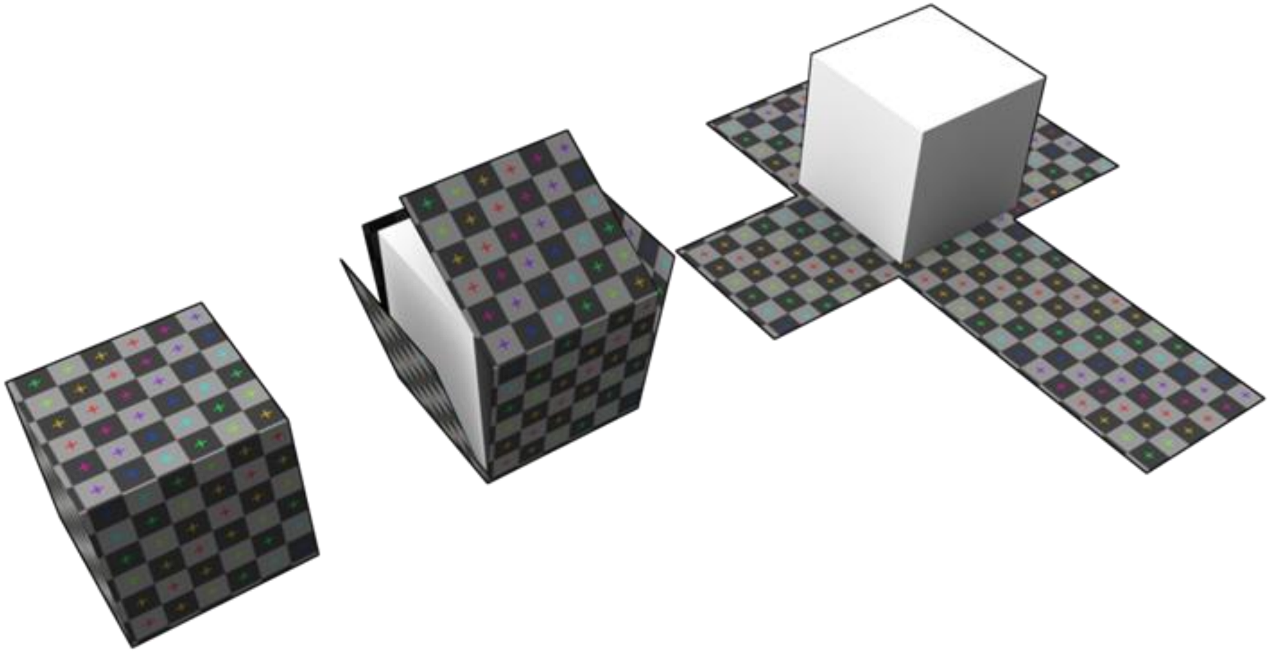
Box Mapping

- Easy to use with simple orthographic projection
- Also used in environment maps



[Ed Angel]

Example

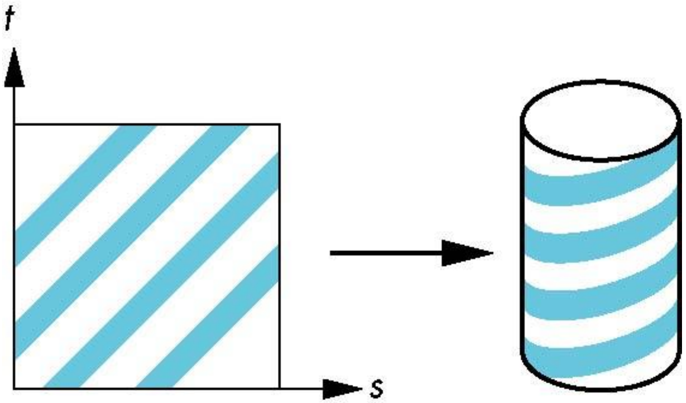


[Andy Van Dam]

Two-part mapping

- One solution to the mapping problem is to first map the texture to a simple intermediate surface

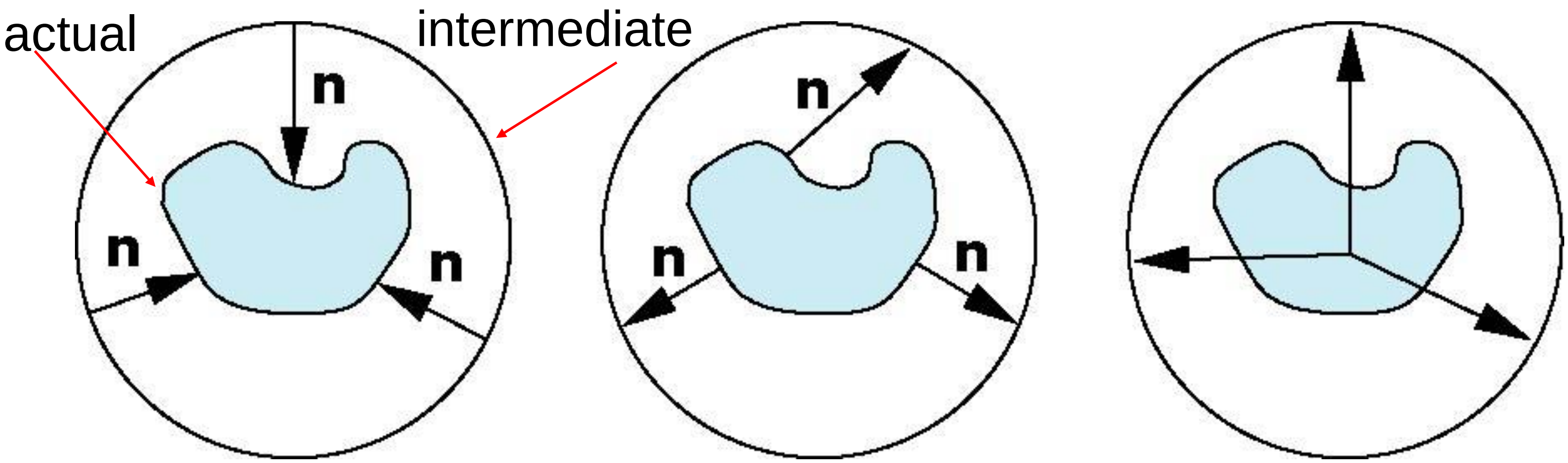
- Example: map to cylinder



[Ed Angel]

Second Mapping

- Map from intermediate object to actual object
  - Normals from intermediate to actual
  - Normals from actual to intermediate
  - Vectors from center of intermediate



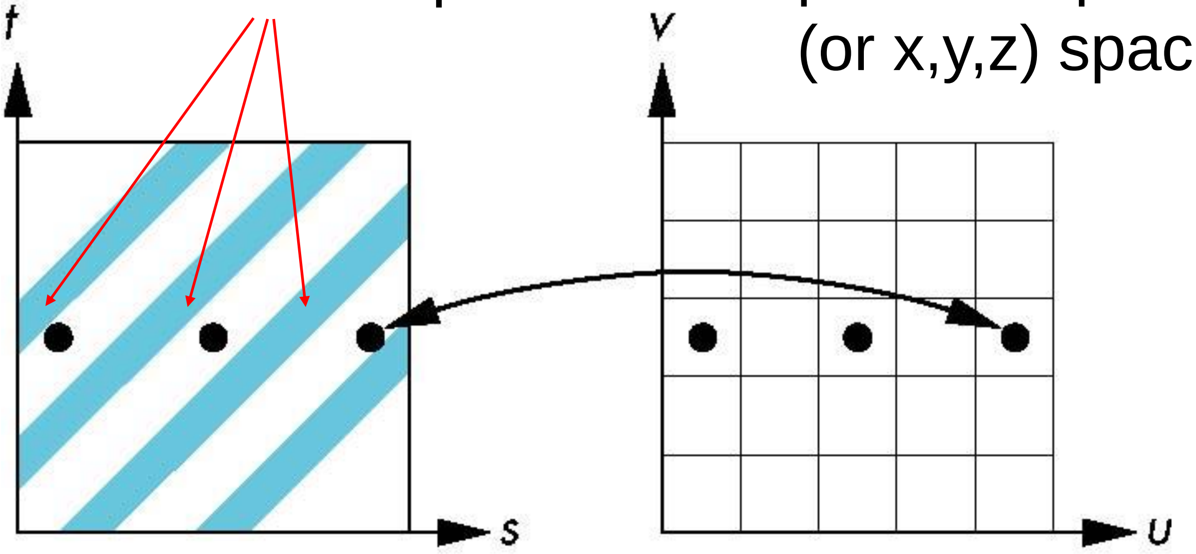
[Ed Angel]

Aliasing

- Point sampling of the texture can lead to aliasing errors

miss blue stripes

point samples in u,v
(or x,y,z) space

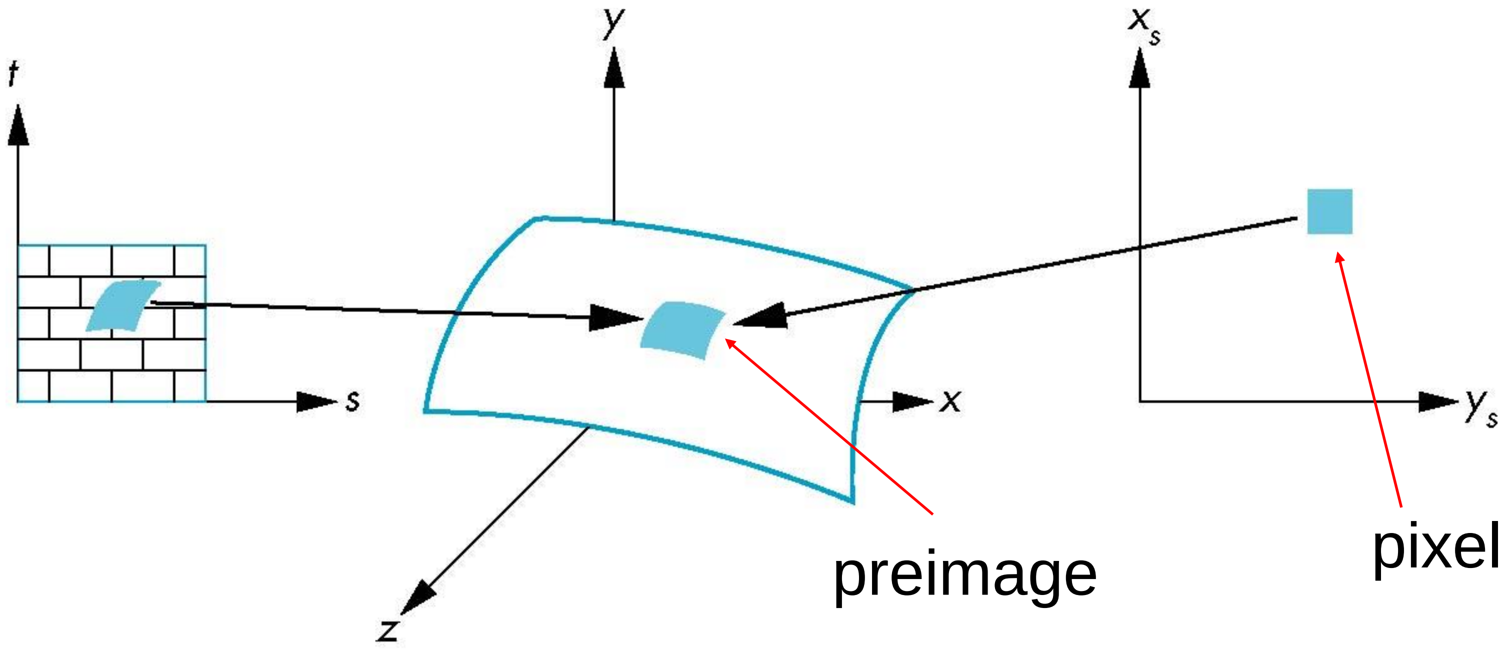


point samples in texture space

[Ed Angel]

Area Averaging

- A better but slower option is to use *area averaging*



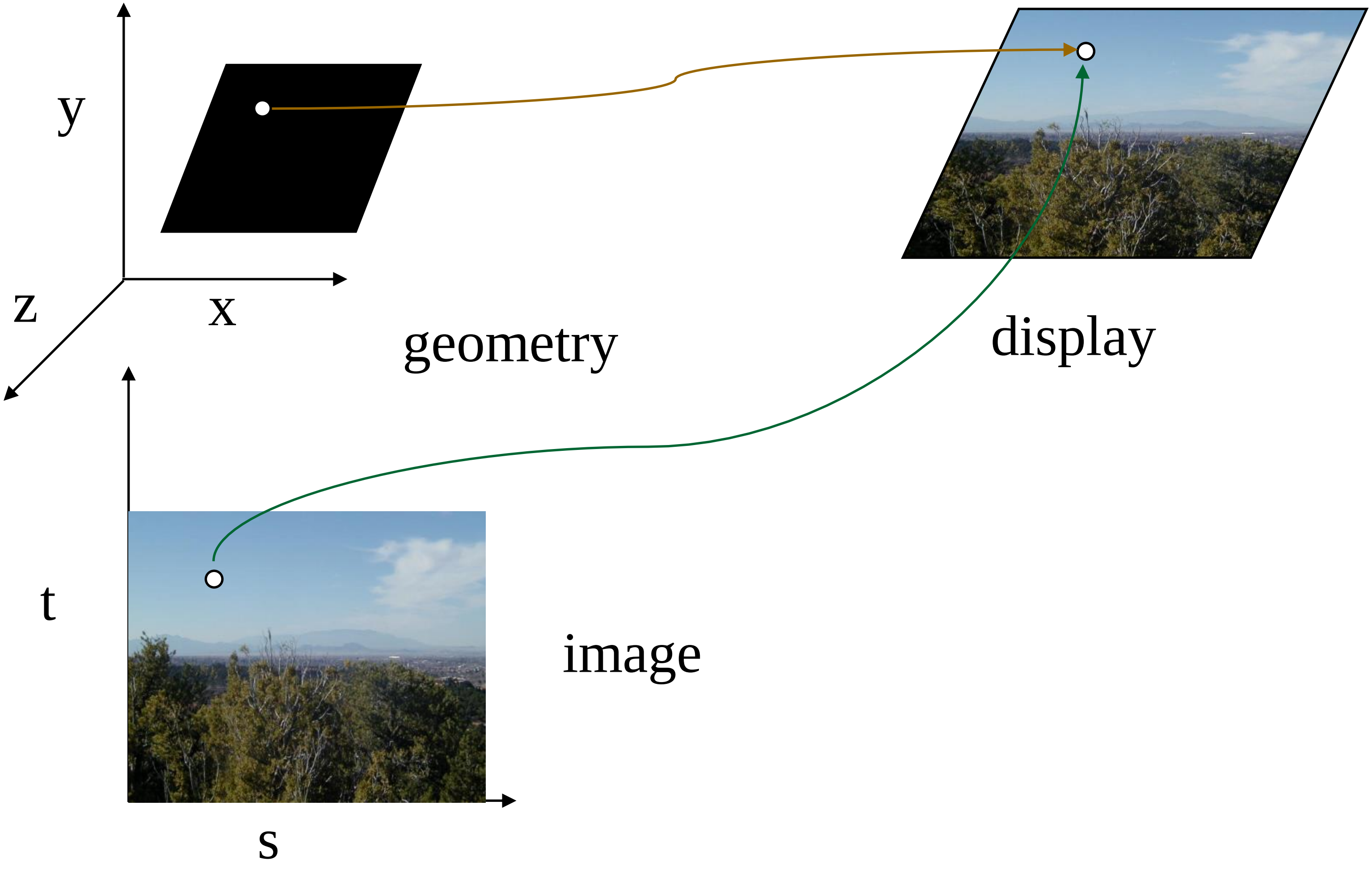
[Ed Angel]

WEBGL
– APPLYING TEXTURES

Textures – Basic Strategy

- Three steps to applying a texture
  1. specify the texture
     - read or generate image
     - assign to texture
     - enable texturing
  2. assign texture coordinates to vertices
     - Proper mapping function is left to application
  3. specify texture parameters
     - wrapping, filtering

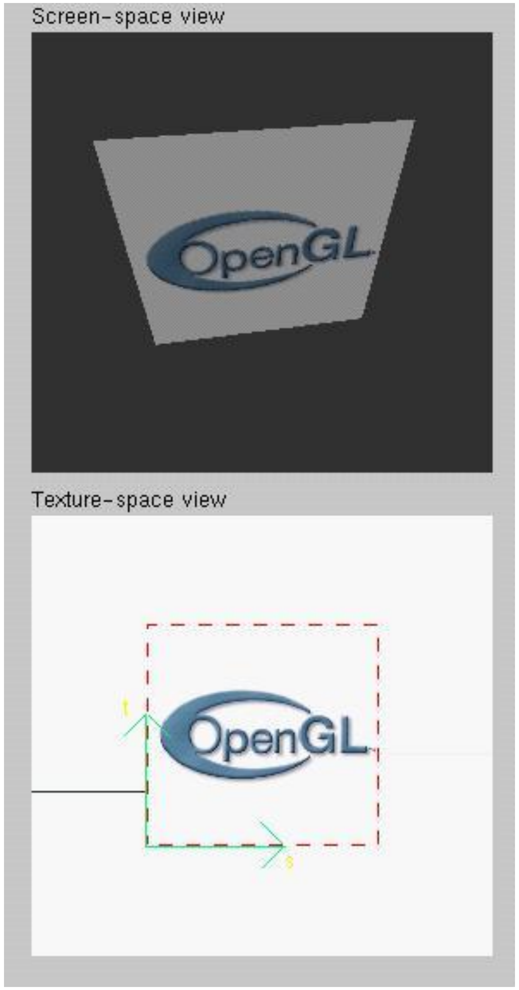
Texture Mapping



[Ed Angel]

Texture Example

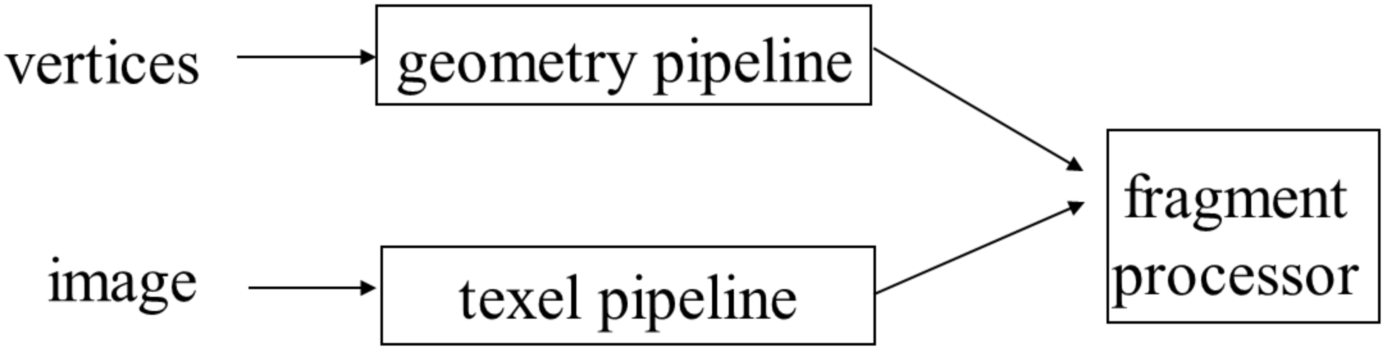
- The texture (below) is a 256 x 256 image
- It has been mapped to a rectangular polygon which is viewed in perspective



[Ed Angel]

The WebGL pipeline

- Geometry and images flow through separate pipelines
- “Complex” textures do not affect “geometric complexity”



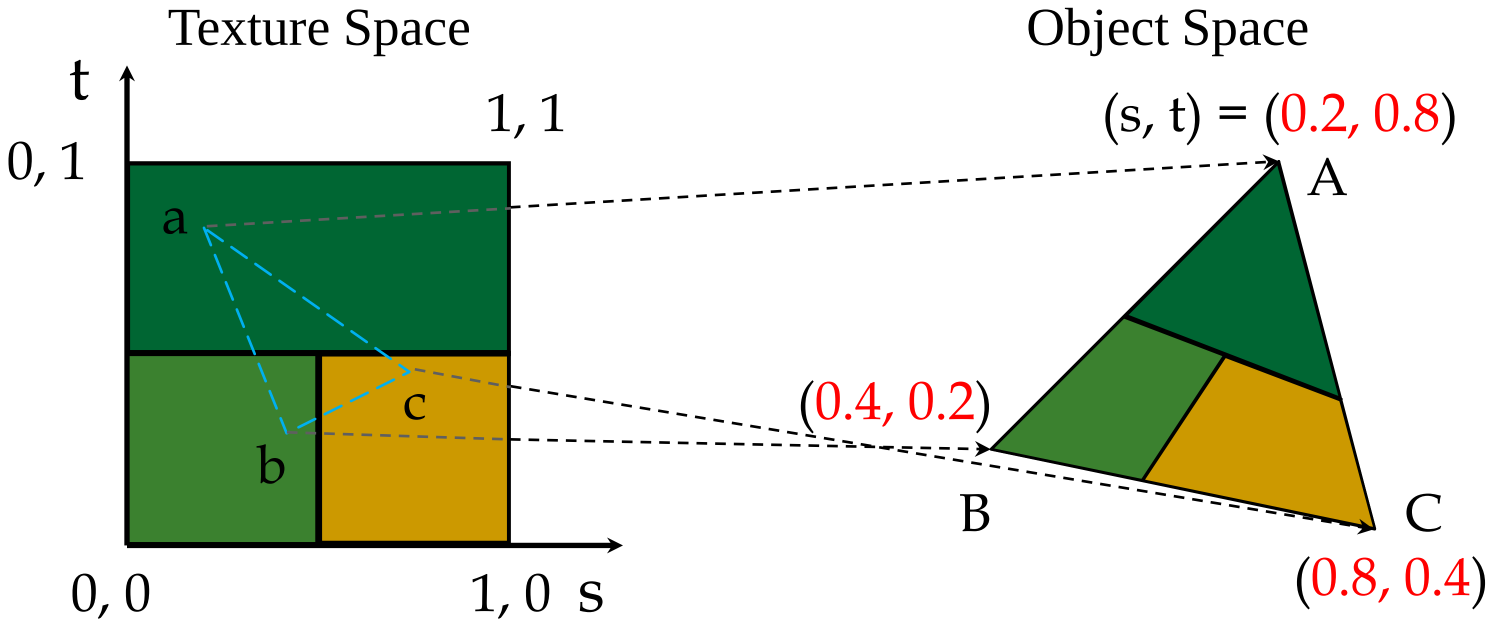
[Ed Angel]

Specifying a texture image

- Define a texture image from an array of *texels* in CPU memory
- Use an image in a standard format such as JPEG
  - Scanned image
  - Generate by application code
- WebGL supports only 2 dimensional texture maps
  - OpenGL supports 1-4 dimensional texture maps

Mapping a Texture

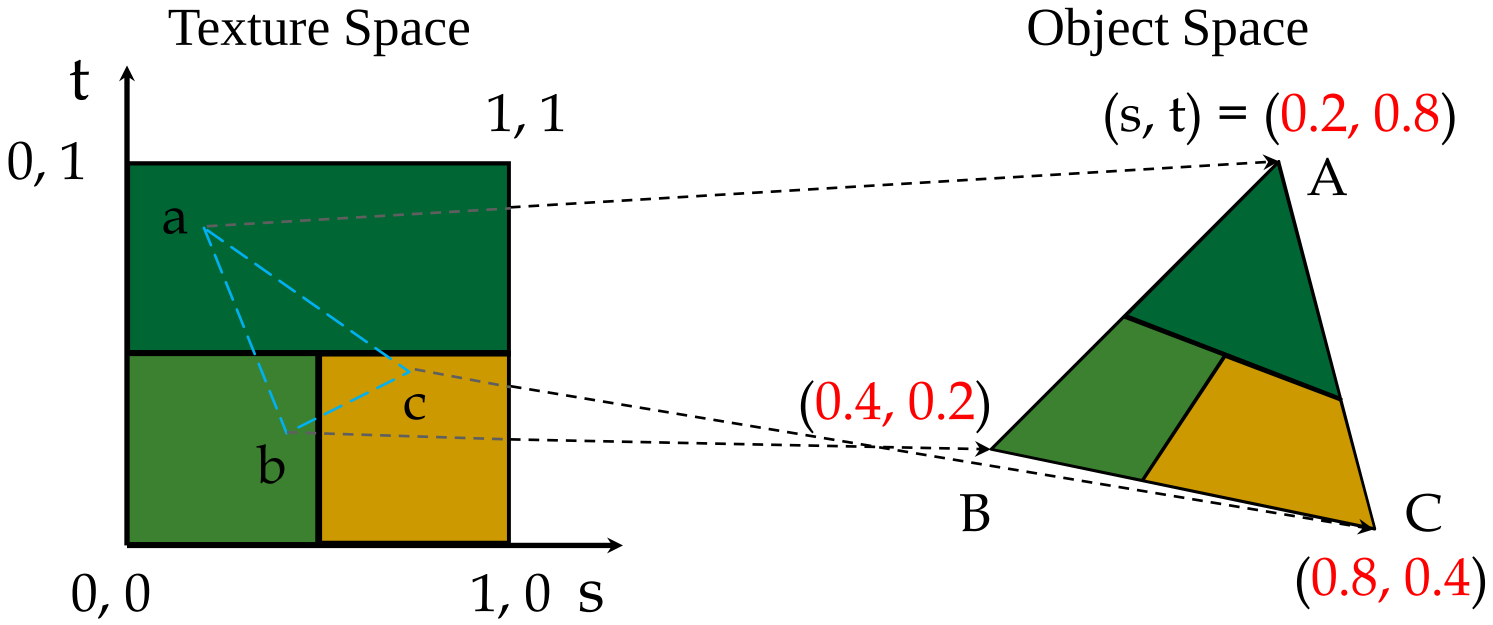
- Specify texture coordinates as a 2D vertex attribute
- Same vertex may have different texture coordinates for different triangles



[Ed Angel]

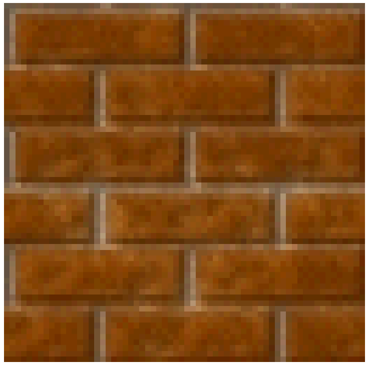
Mapping a Texture

- Texture coordinates are linearly interpolated across triangles

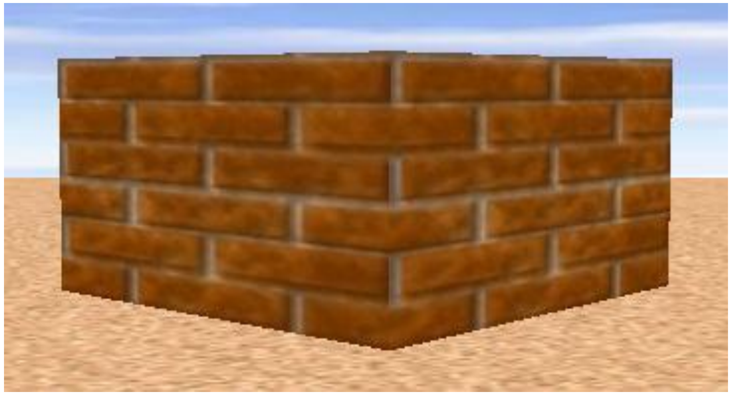


[Ed Angel]

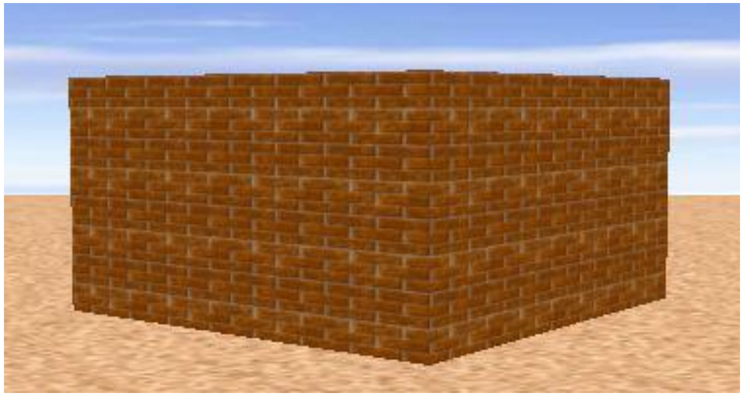
Texture Mapping Style – Tiling



Texture



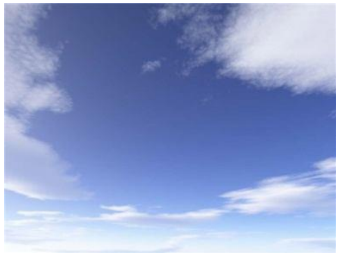
Without Tiling



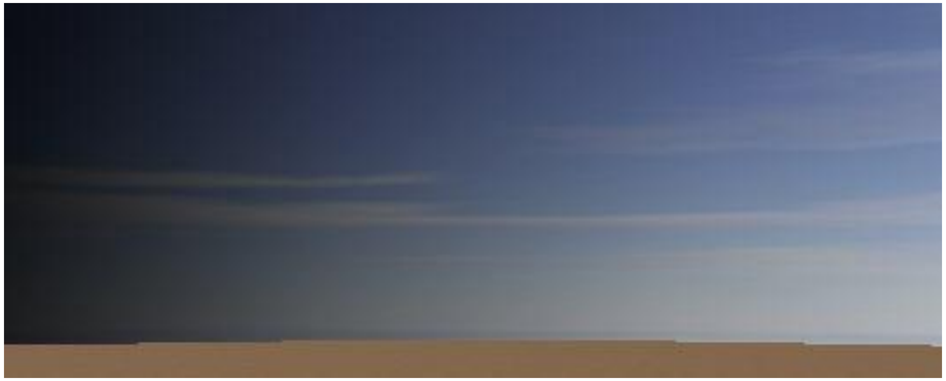
With Tiling

[Andy Van Dam]

Texture Mapping Style – Stretching



Texture



Applied with stretching

[Andy Van Dam]

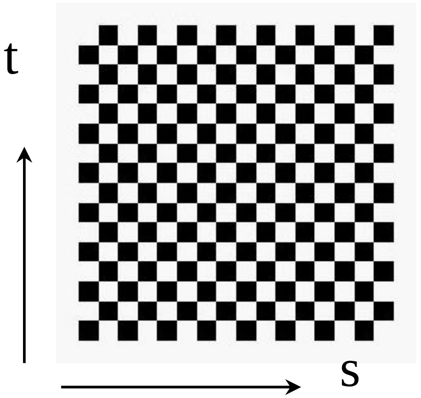
WebGL – Using texture objects

- Specify textures in texture objects
- Set texture filter
- Set texture function
- Set texture wrap mode
- Set optional perspective correction hint
- Bind texture object
- Enable texturing
- Supply texture coordinates for vertex
  - Coordinates can also be generated

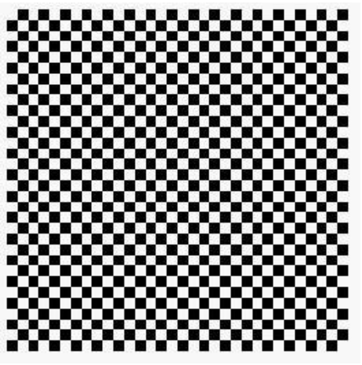
Texture Parameters

- How is a texture applied ?
  - Wrapping parameters determine what happens if s and t are outside the (0,1) range
  - Filter modes allow us to use area averaging instead of point samples
  - Mipmapping allows us to use textures at multiple resolutions
  - Environment parameters determine how texture mapping interacts with shading

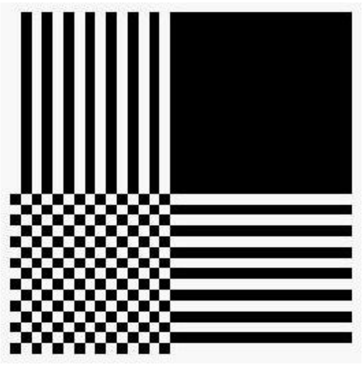
Wrapping Mode



texture



REPEAT
wrapping



CLAMP
wrapping

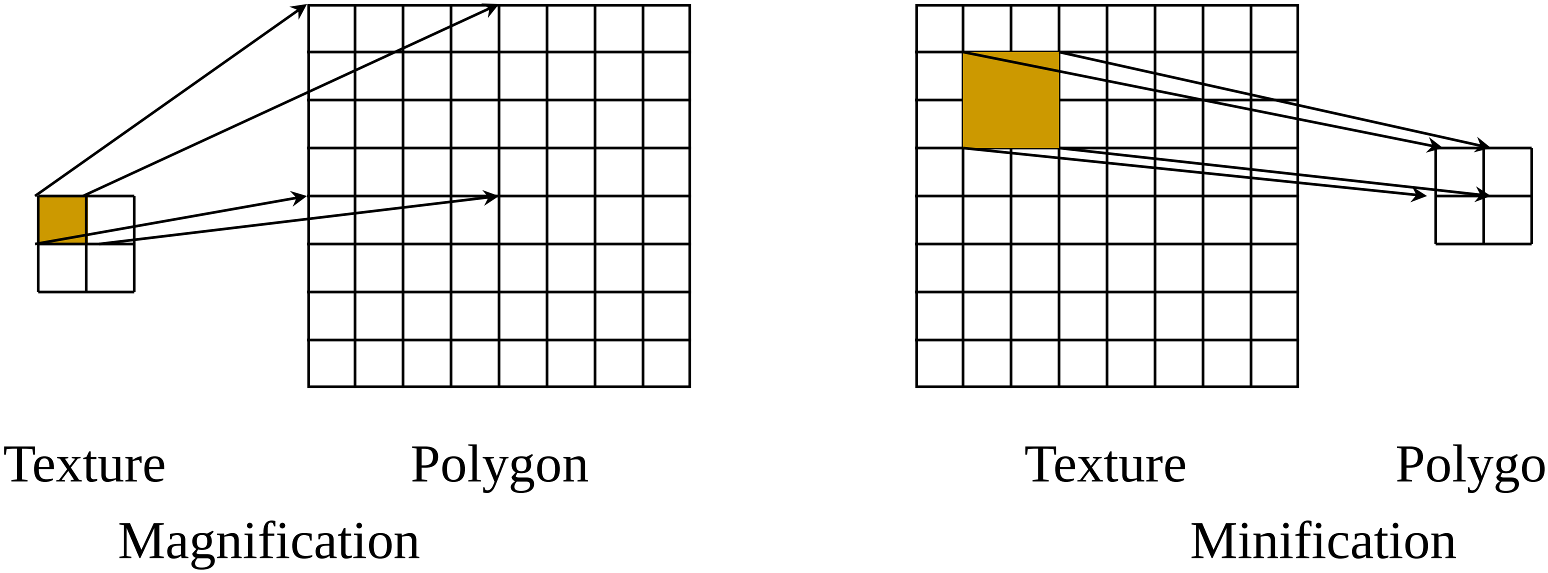
```
gl.texParameteri(gl.TEXTURE_2D,
     gl.TEXTURE_WRAP_S, gl.CLAMP )
gl.texParameteri( gl.TEXTURE_2D,
     gl.TEXTURE_WRAP_T, gl.REPEAT )
```

[Ed Angel]

Magnification and Minification

- *Magnification* : more than one pixel can cover a texel
- *Minification* : more than one texel can cover a pixel

- Can use point sampling (nearest texel) or linear filtering (2 x 2 filter) to obtain texture values



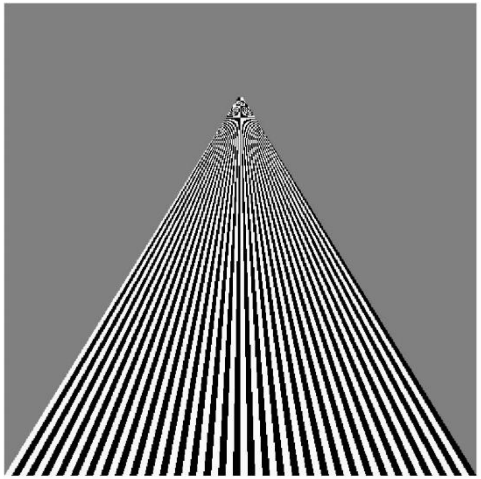
[Ed Angel]

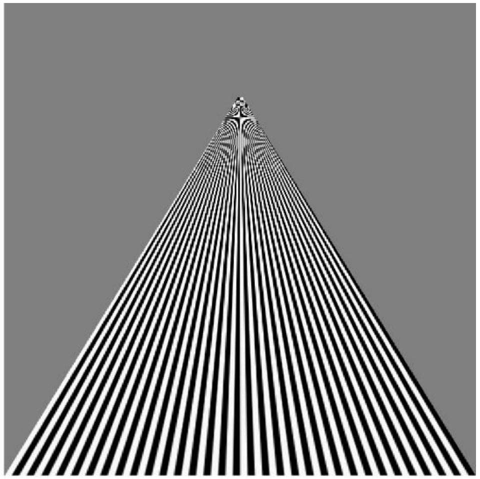
Mipmapped Textures

- *Mipmapping* allows for prefiltered texture maps of decreasing resolutions
- Lessens interpolation errors for smaller textured objects

Example

point sampling

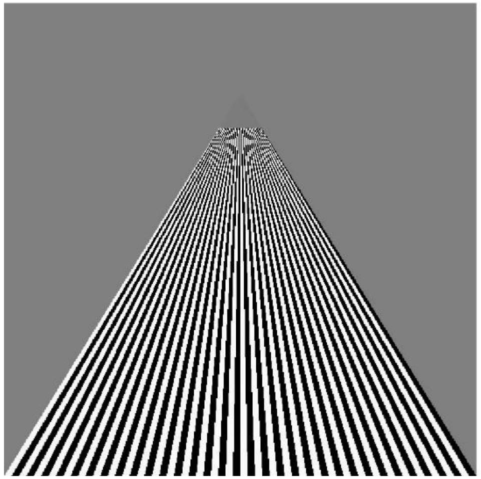


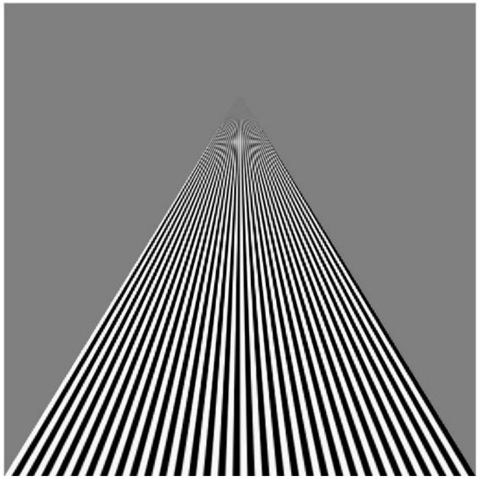


linear filtering

[Ed Angel]

mipmapped point sampling





mipmapped linear filtering

Other Texture Features

- Environment Maps
  - Start with image of environment through a wide-angle lens
    - Can be either a real scanned image
  - Use this texture to generate a spherical map
  - Alternative is to use a cube map

- Multitexturing
  - Apply a sequence of textures through cascaded texture units

Applying Textures

- Textures can be applied in many ways
- A texture fully determines color
- A texture is modulated with a computed color
- A texture is blended with an environmental color

WebGL – Applying textures

- Textures are applied during fragment shading by a sampler
- Samplers return a texture color from a texture object

```
varying vec4 color;  //color from rasterizer
varying vec2 texCoord; //texture coordinate from rasterizer
uniform sampler2D texture; //texture object from application

void main()  {
    gl_FragColor = color * texture2D( texture, texCoord );
}
```

[Ed Angel]

WebGL – Vertex-shader

- The vertex-shader computes
  - Vertex positions
  - Vertex colors, if needed
- Usually, it will also output texture coordinates

```
attribute vec4 vPosition; //vertex position in object coordinates
attribute vec4 vColor;  //vertex color from application
attribute vec2 vTexCoord; //texture coordinate from application

varying vec4 color; //output color to be interpolated
varying vec2 texCoord; //output tex coordinate to be interpolated
```

[Ed Angel]

WebGL – Link with shaders

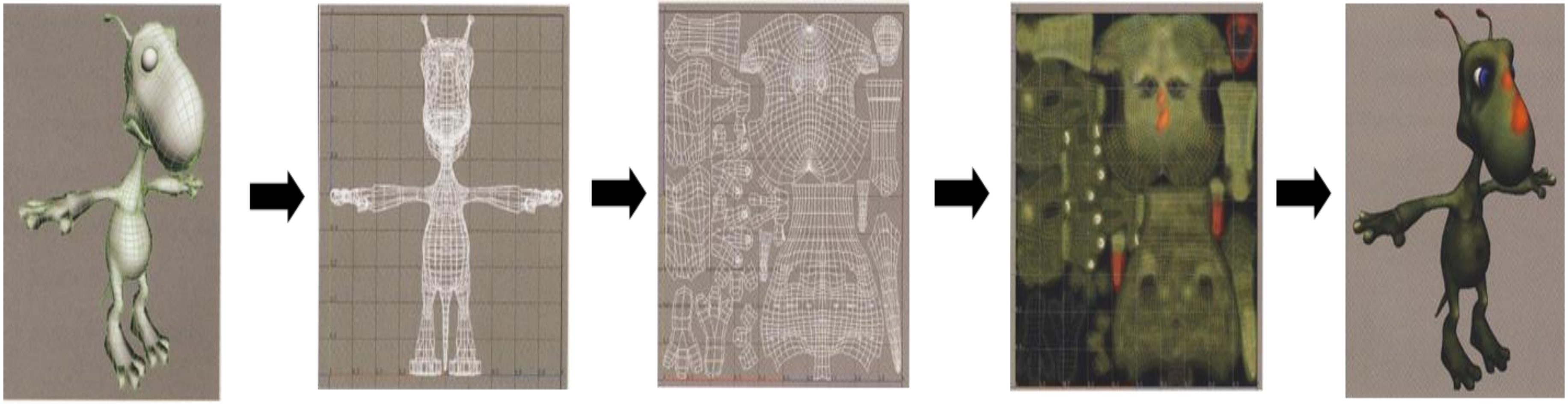
```
var vTexCoord = gl.getAttribLocation( program, "vTexCoord" );
gl.enableVertexAttribArray( vTexCoord );
gl.vertexAttribPointer( vTexCoord, 2, gl.FLOAT, false, 0, 0);

// Set the value of the fragment shader texture sampler variable
//   ("texture") to the the appropriate texture unit. In this case,
//   zero for GL_TEXTURE0 which was previously set by calling
//   gl.activeTexture().

gl.uniform1i( glGetUniformLocation(program, "texture"), 0 );
```

[Ed Angel]

Complex Geometry/Real Applications

- Texture mapping of complicated objects, not simple primitives
- Need precise control over how the texture map looks on the object
- Use 3D modeling programs
  - E.g., Maya, Zbrush, Blender, …

Complex Geometry/Real Applications



[Andy Van Dam]

Acknowledgments

- Some ideas and figures have been taken from slides of other CG courses.
- In particular, from the slides made available by Ed Angel and Andy van Dam.
- Thanks!