Textures

Joaquim Madeira

April 2024

Overview

- Motivation
- Textures
- Texture Mapping
- Textures in Three.js

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 ?

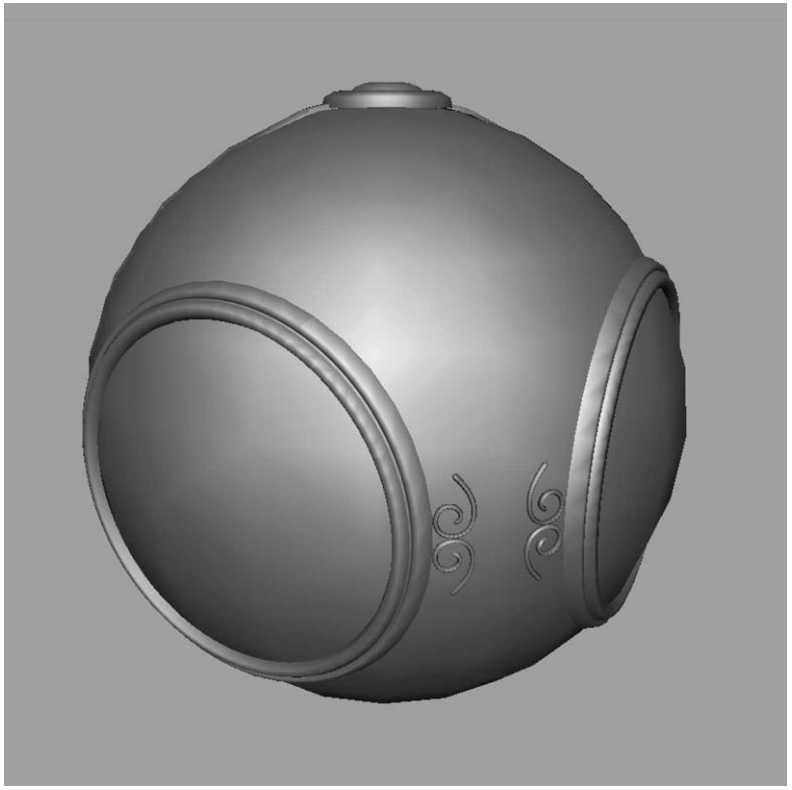
- An orange colored sphere ?
  - Too simple !
- A more complex shape to convey details ?
  - How to represent surface features ?
  - Takes too many triangles to model all the dimples…

How to model / render an orange ?

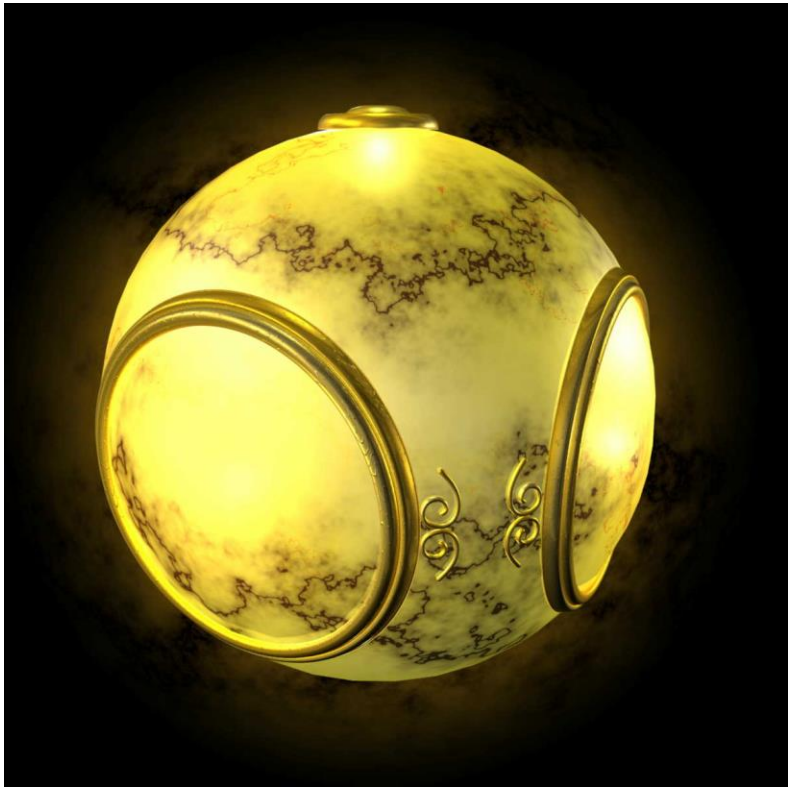
- Simple geometric model + Texture
  - Take a picture of a real orange
  - Scan and “paste” it onto model
  - Texture mapping
- Might not be sufficient: surface will be smooth
- How to “change” local shape ?
  - Bump mapping

TEXTURES

Texture Mapping



geometric model



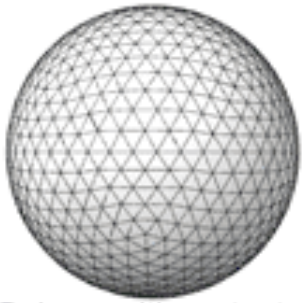
texture mapped

[Ed Angel]

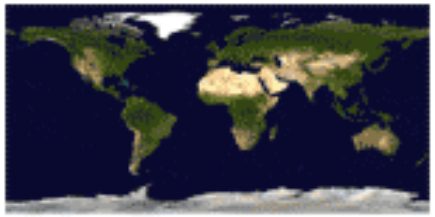
Texture Mapping

- 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

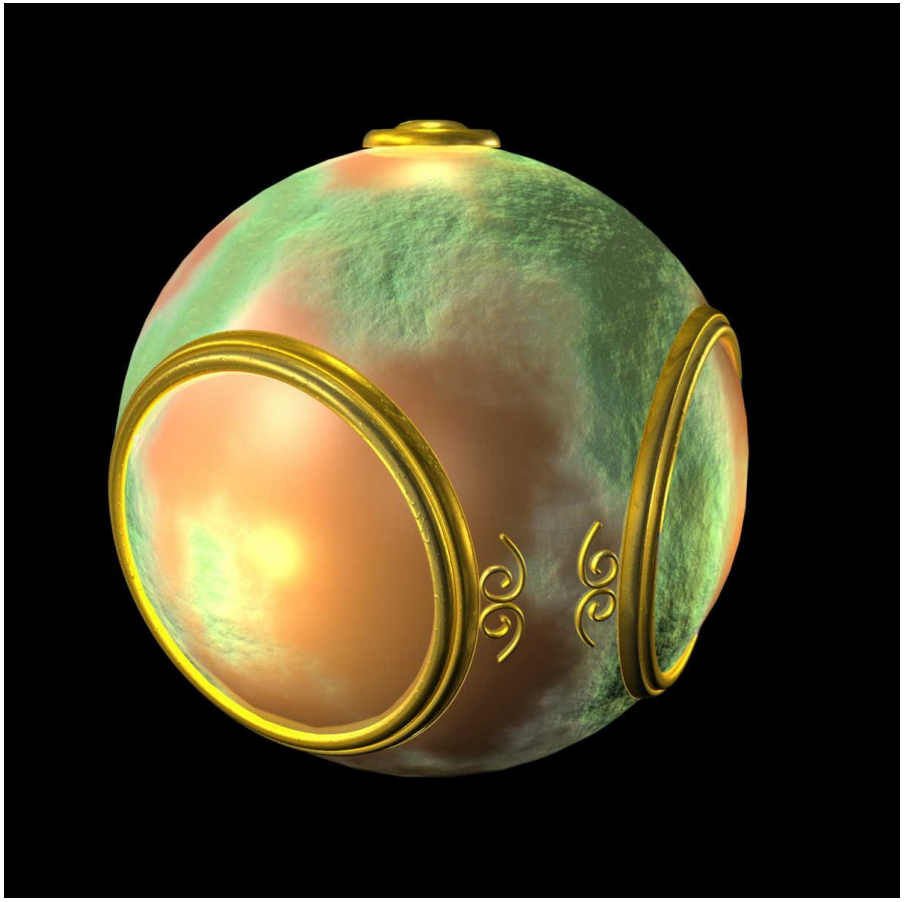


Texture image

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

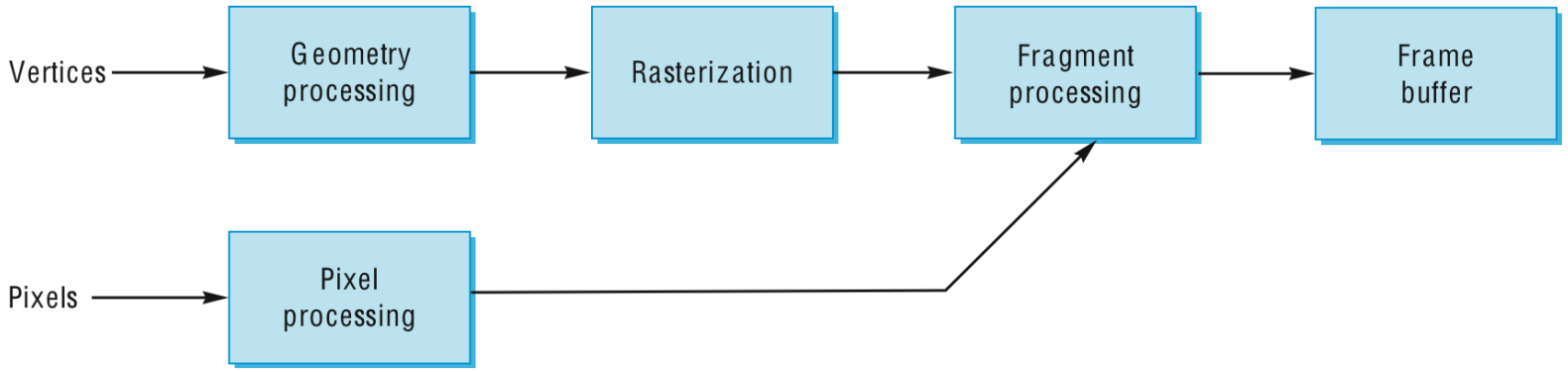
TEXTURE MAPPING

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 ?

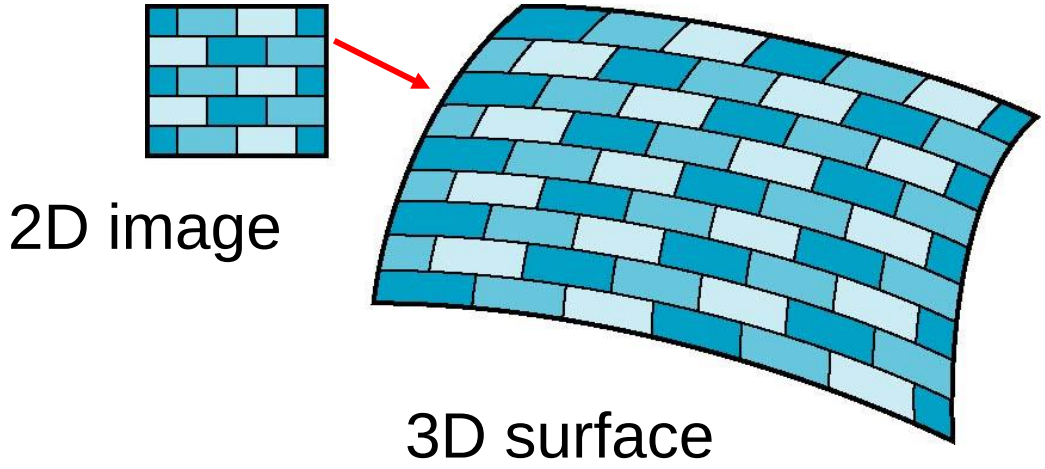
- Mapping techniques are implemented at the end of the rendering pipeline
  - Very efficient because few polygons make it past the clipper



[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

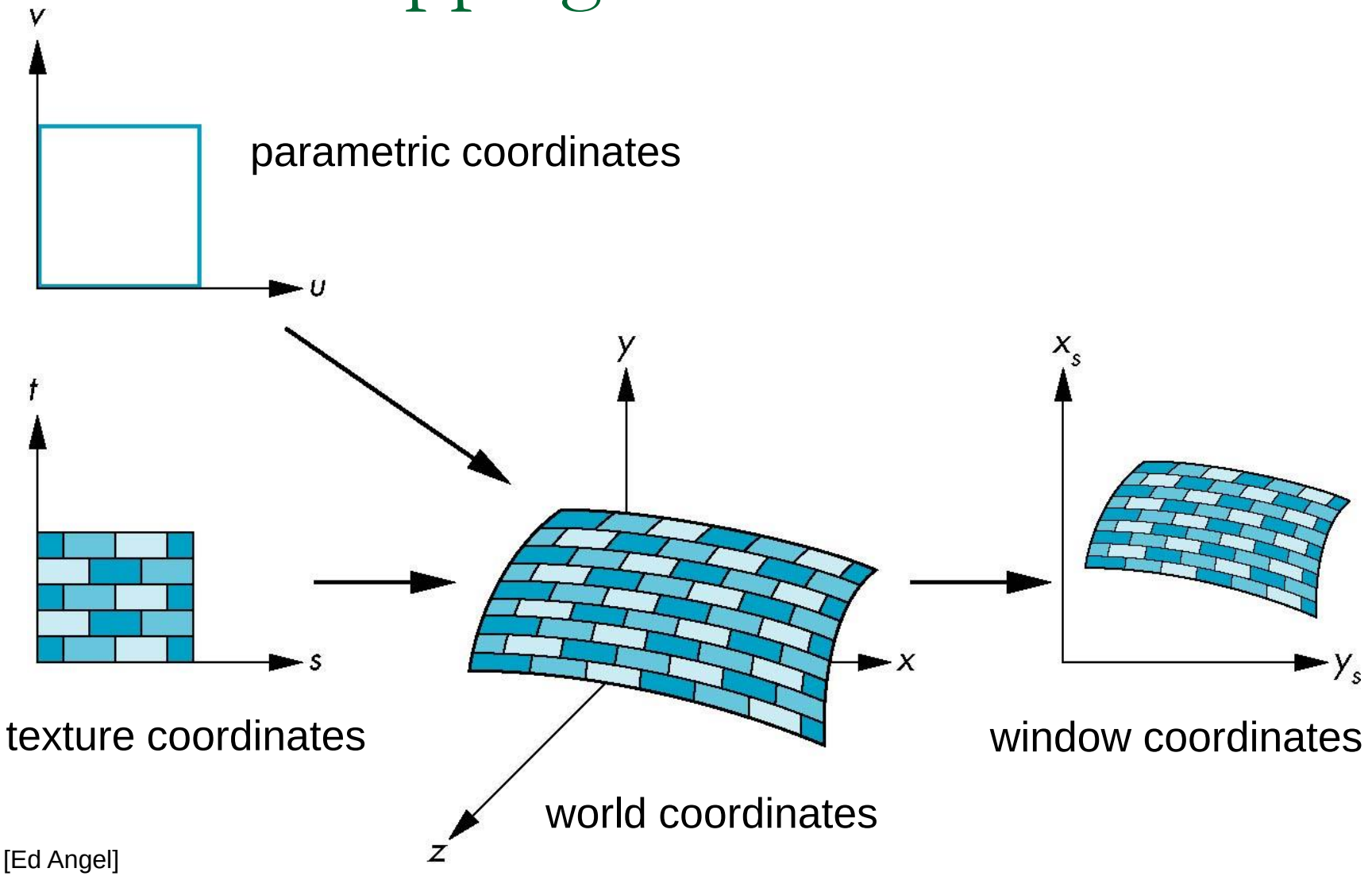


[Ed Angel]

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



[Ed Angel]

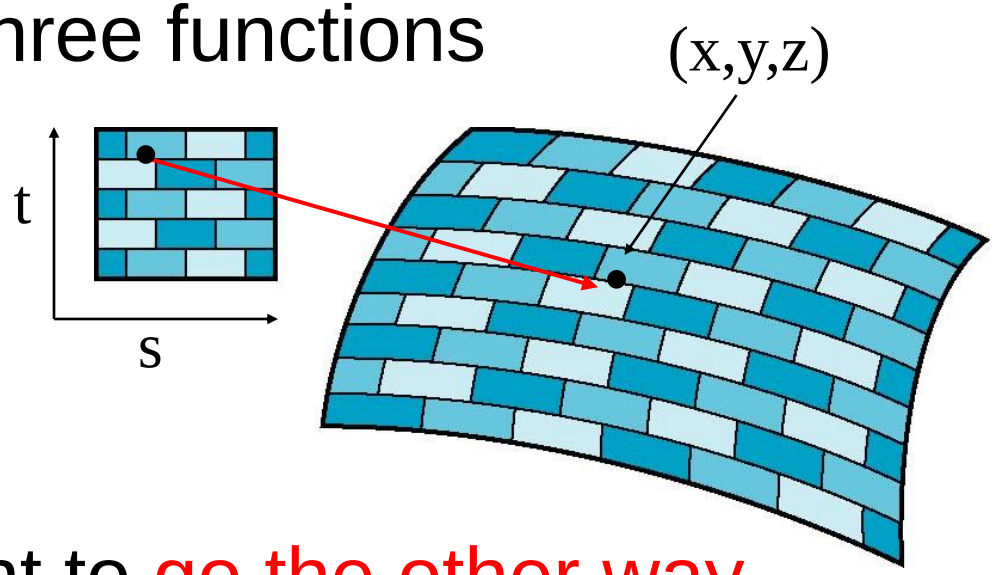
Mapping Functions

- 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

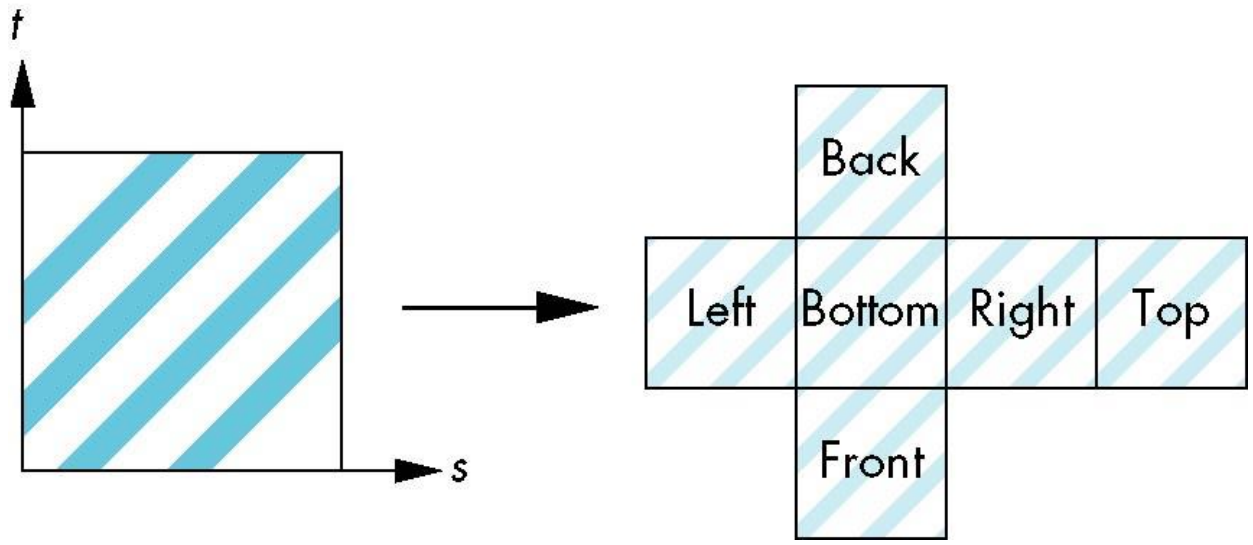
  $s = s(x,y,z)$

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

- Such functions are difficult to find in general

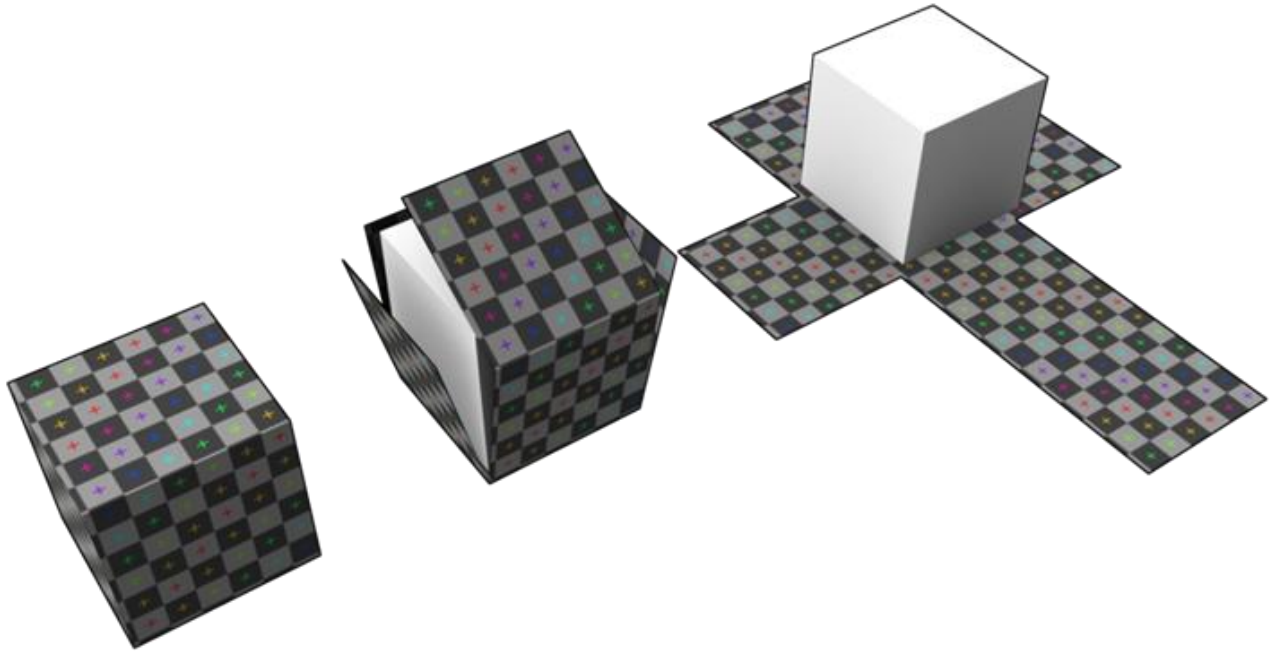
Box Mapping

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



[Ed Angel]

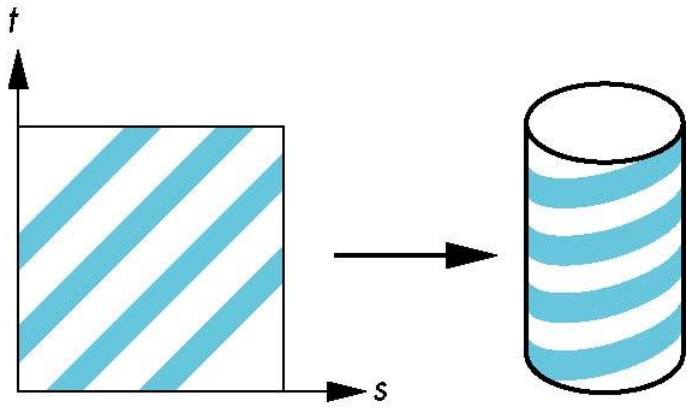
Example



[Andy Van Dam]

Two-part mapping

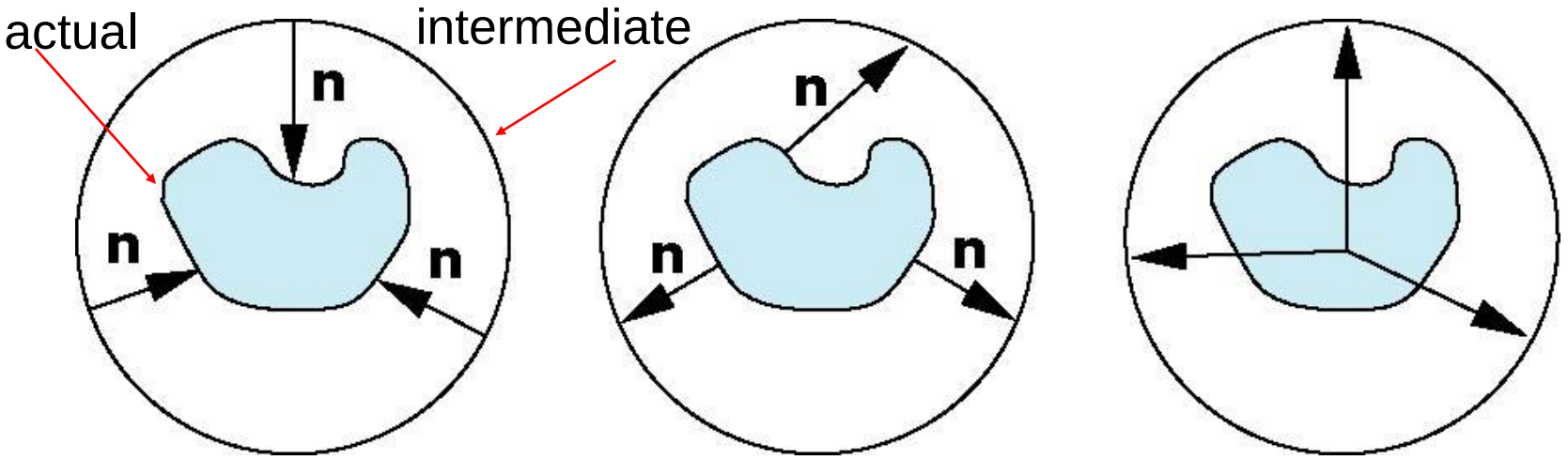
- One solution to the mapping problem is to first map the texture to a simple intermediate surface
- Example: map to cylinder



[Ed Angel]

Second Mapping

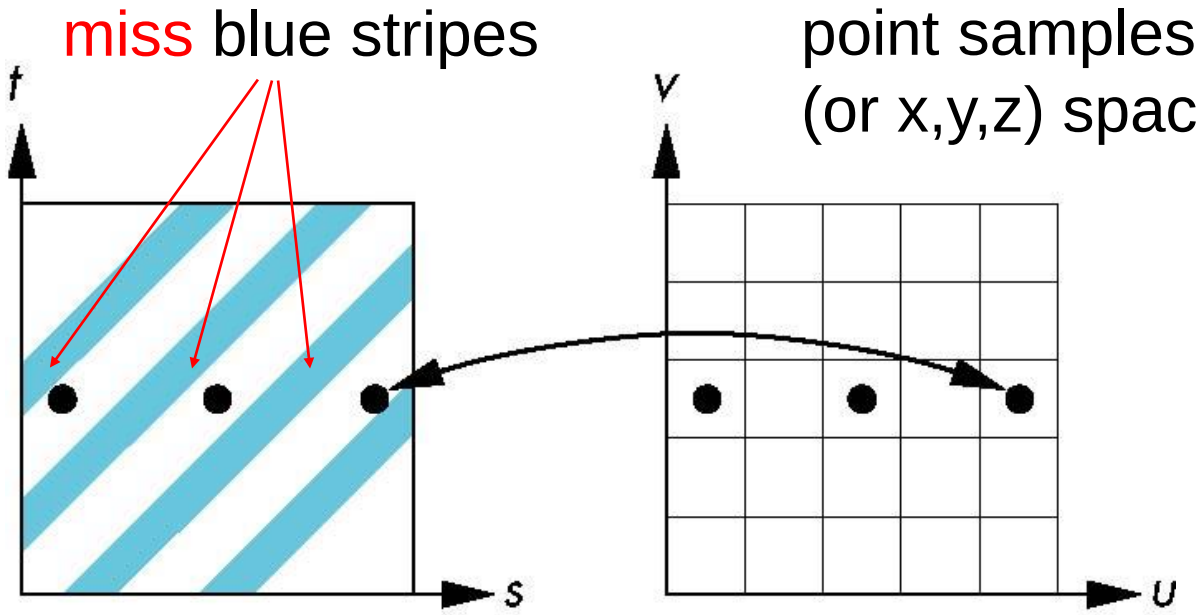
- Map from intermediate object to actual object
  - Normals from intermediate to actual
  - Normals from actual to intermediate
  - Vectors from center of intermediate



[Ed Angel]

Aliasing

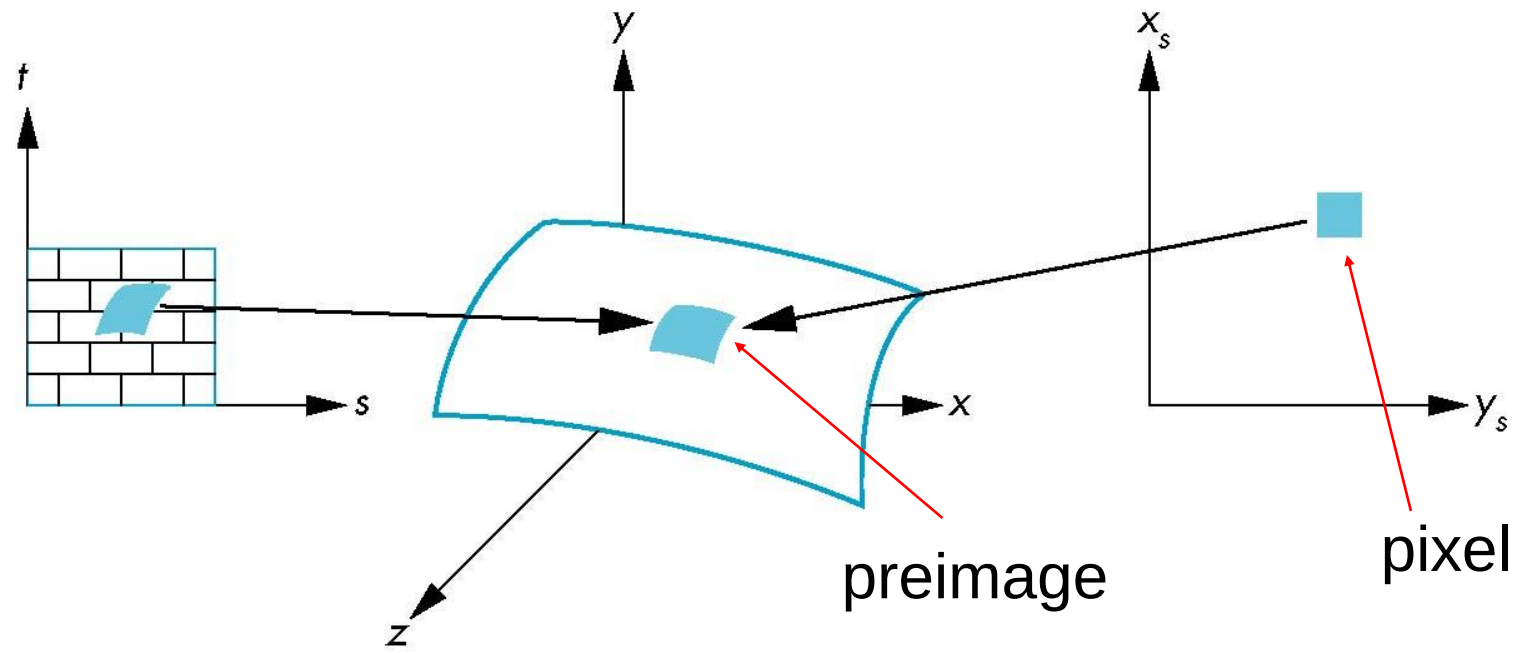
- Point sampling of the texture can lead to aliasing errors



[Ed Angel]

Area Averaging

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

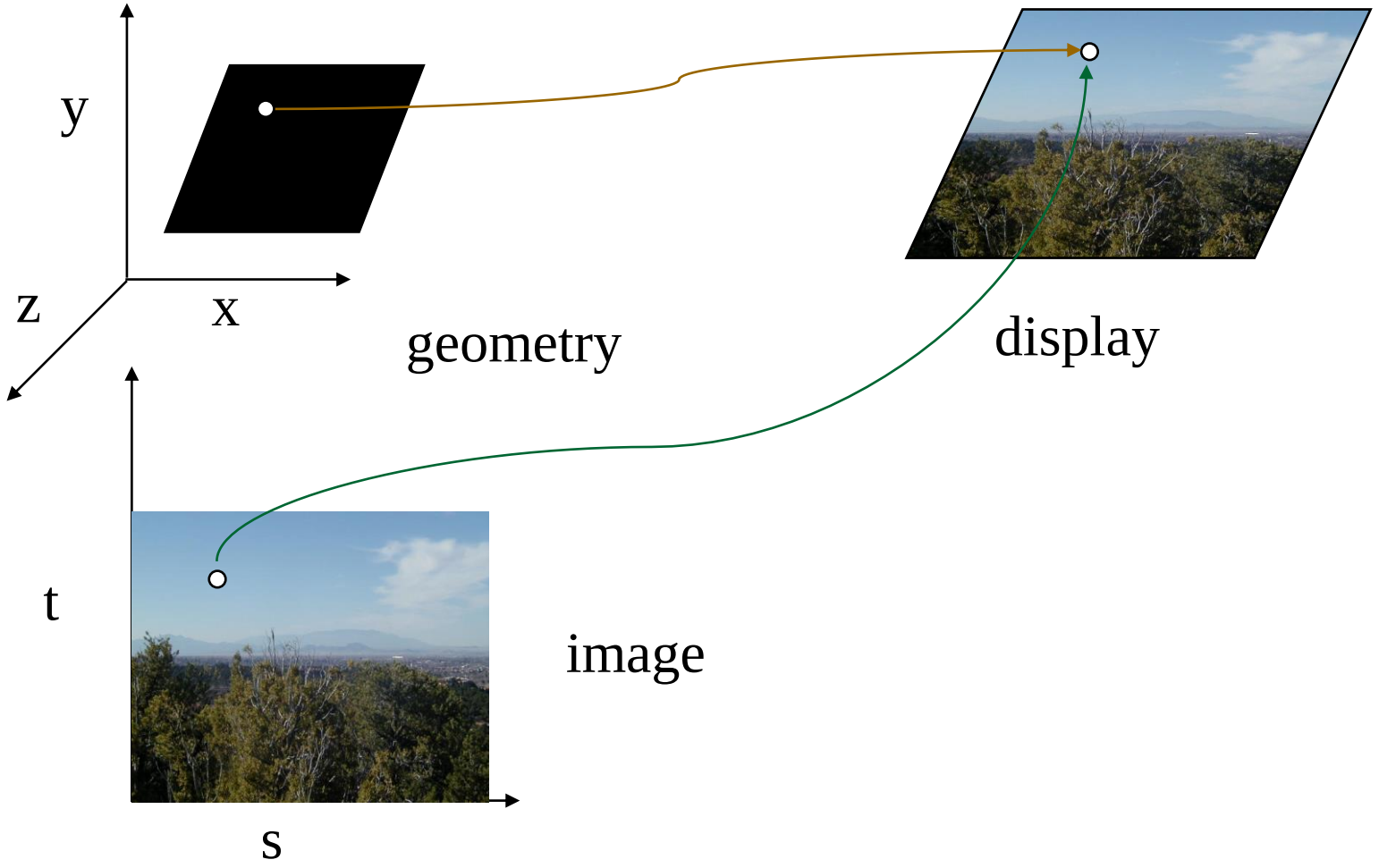


[Ed Angel]

Textures – Basic Strategy

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

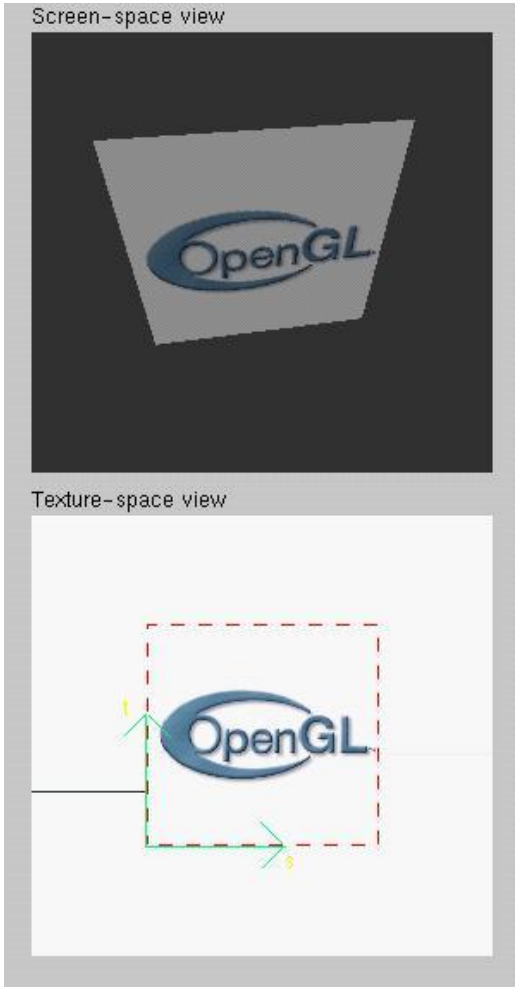
Texture Mapping



[Ed Angel]

Texture Example

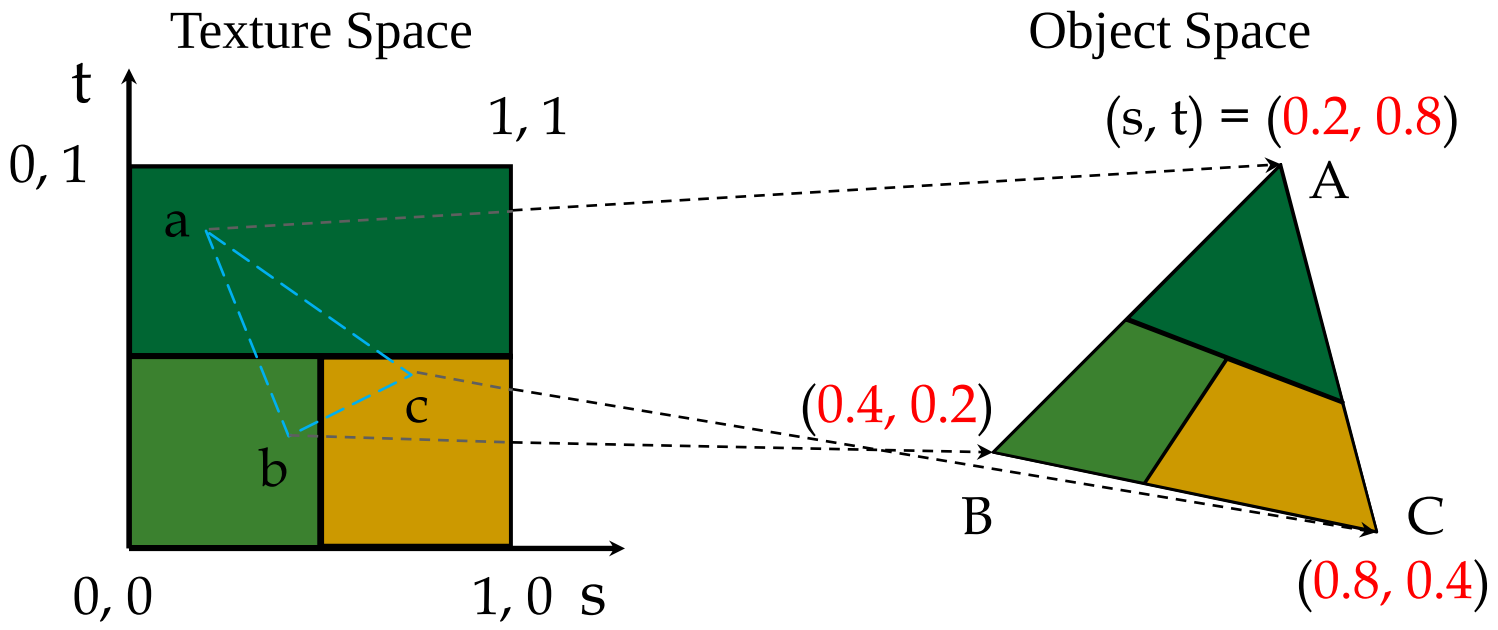
- The texture (below) is a 256 x 256 image
- It has been mapped to a rectangular polygon which is viewed in perspective



[Ed Angel]

Mapping a Texture

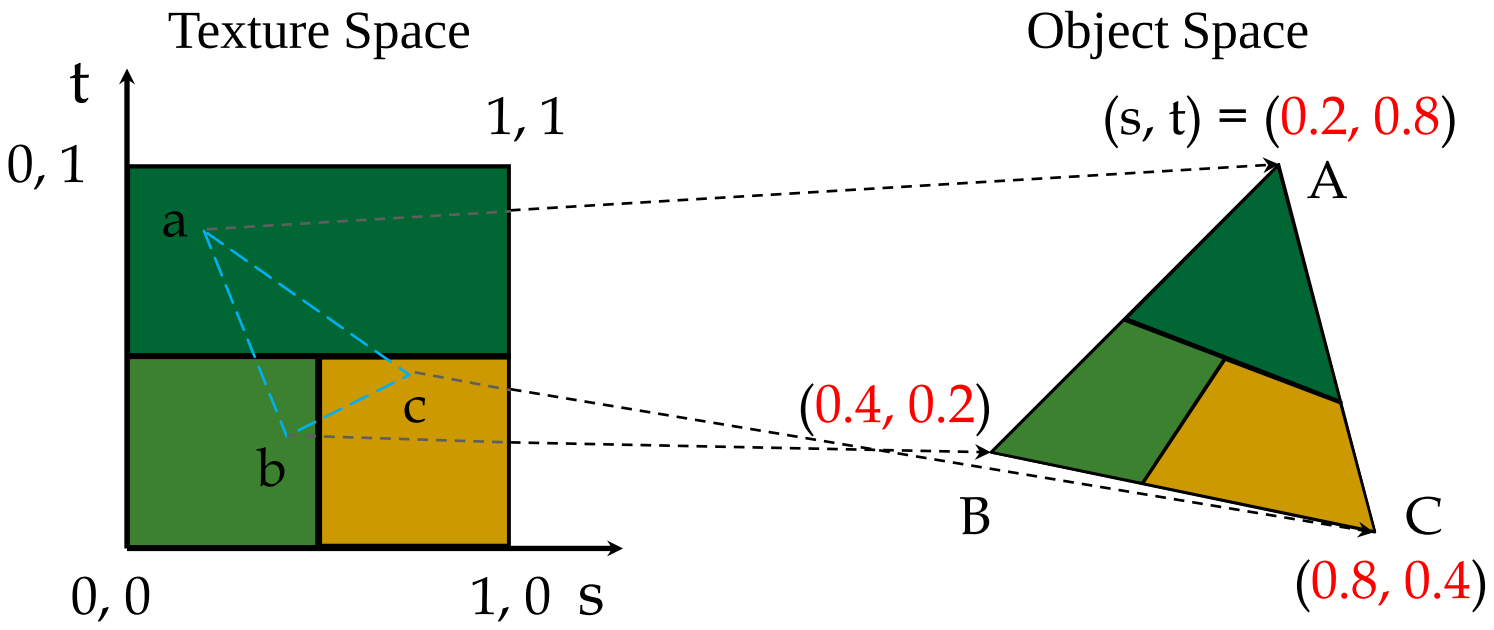
- Specify texture coordinates as a 2D vertex attribute
- Same vertex may have different texture coordinates for different triangles



[Ed Angel]

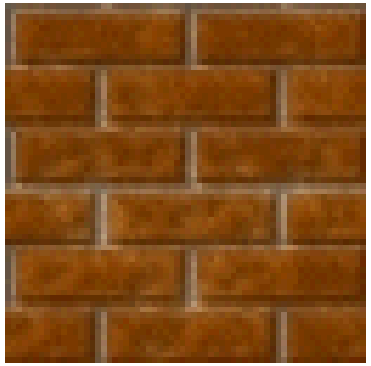
Mapping a Texture

- Texture coordinates are linearly interpolated across triangles

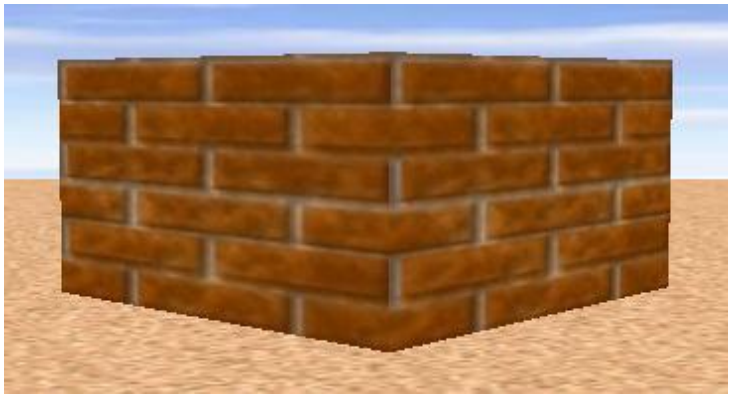


[Ed Angel]

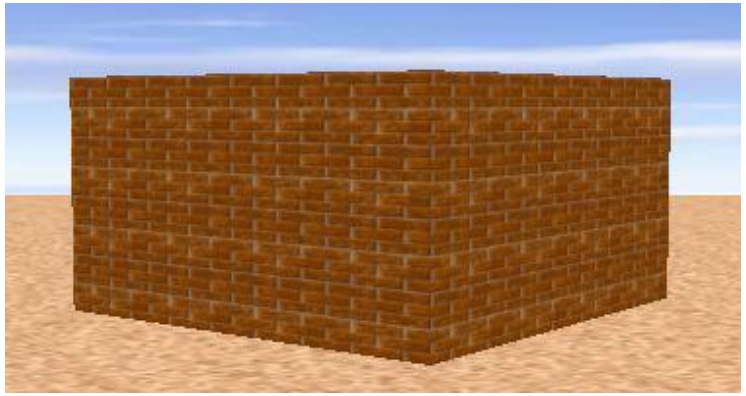
Texture Mapping Style – Tiling



Texture



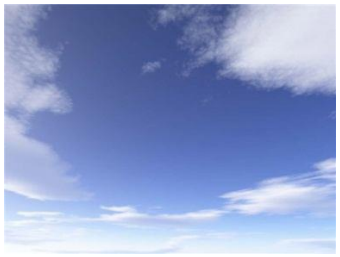
Without Tiling



With Tiling

[Andy Van Dam]

Texture Mapping Style – Stretching



Texture

Applied with stretching

[Andy Van Dam]

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

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

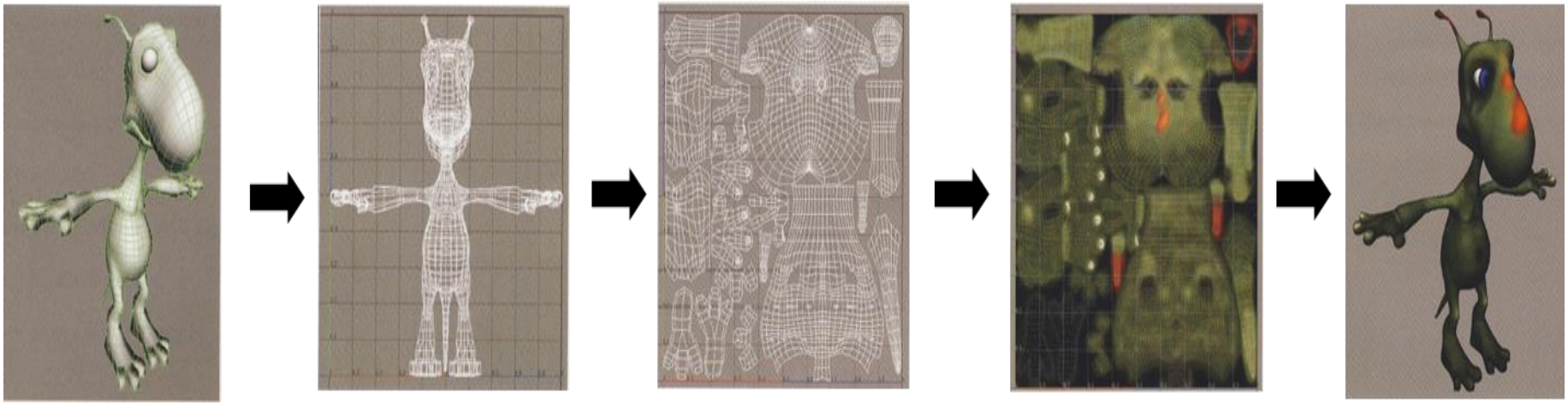
Applying Textures

- 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

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]

TEXTURES IN THREE.JS

What can we do ?

- Use a texture to define the colors of individual mesh pixels

[Dirksen]

Loading and applying a texture

```
function createMesh(geom, imageFile) {

    var texture = THREE.ImageUtils.loadTexture(
                "../assets/textures/general/" + imageFile );

    var mat = new THREE.MeshPhongMaterial();
    mat.map = texture;

    var mesh = new THREE.Mesh(geom, mat);
    return mesh;
}
```

Loading and applying a texture

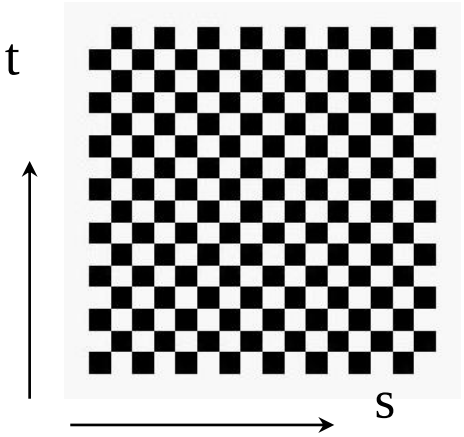
- Textures are correctly applied
- Almost any image can be used as a texture
- Best results are obtained with square images of size 2^k x 2^k
- Loaders for common formats

[Dirksen]

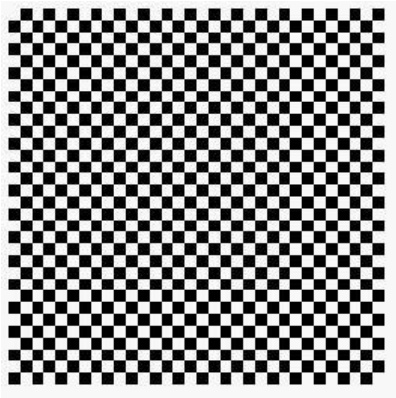
Loading and applying a texture

- Textures are correctly applied
  - For a cube, each side will show the complete image
  - For a sphere, the complete texture is wrapped around the sphere
- What if we want the texture to repeat itself ?

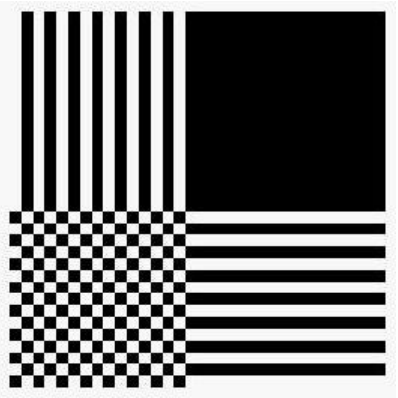
Wrapping Mode



texture



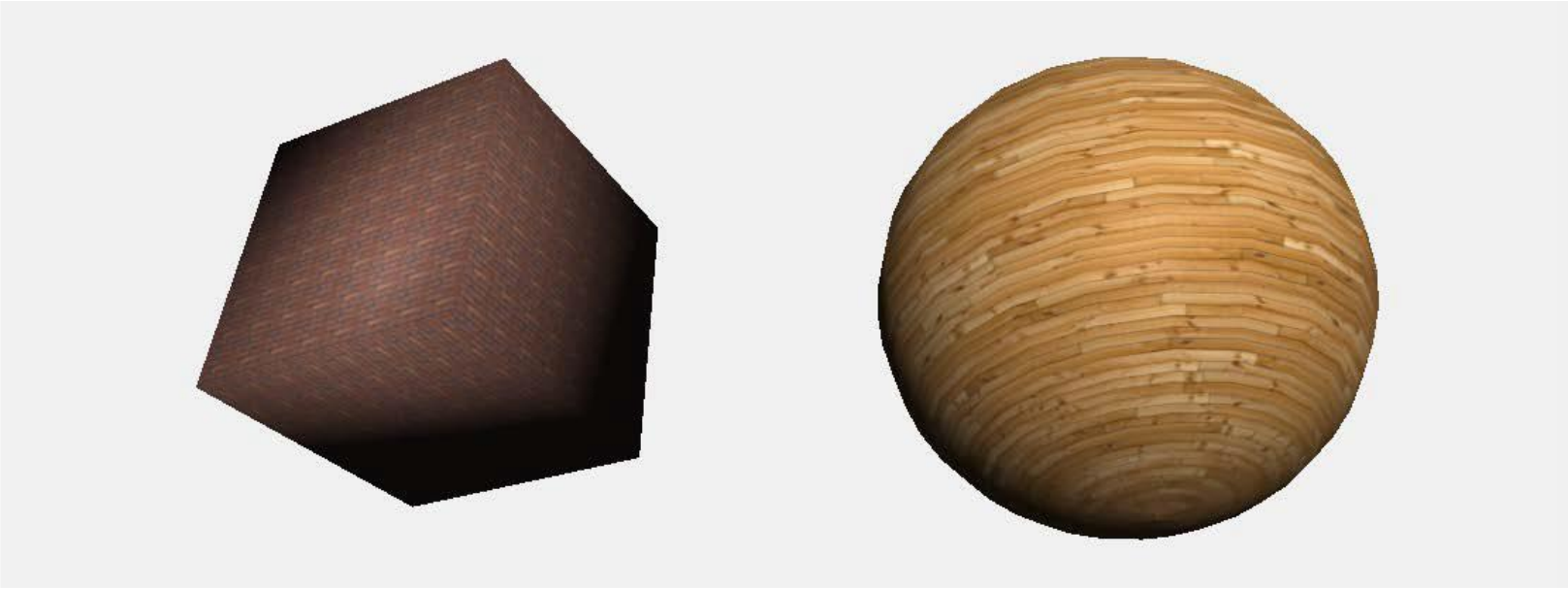
REPEAT
wrapping



CLAMP
wrapping

[Ed Angel]

Repeat Wrapping



```
cube.material.map.wrapS = THREE.RepeatWrapping;
cube.material.map.wrapT = THREE.RepeatWrapping;
```

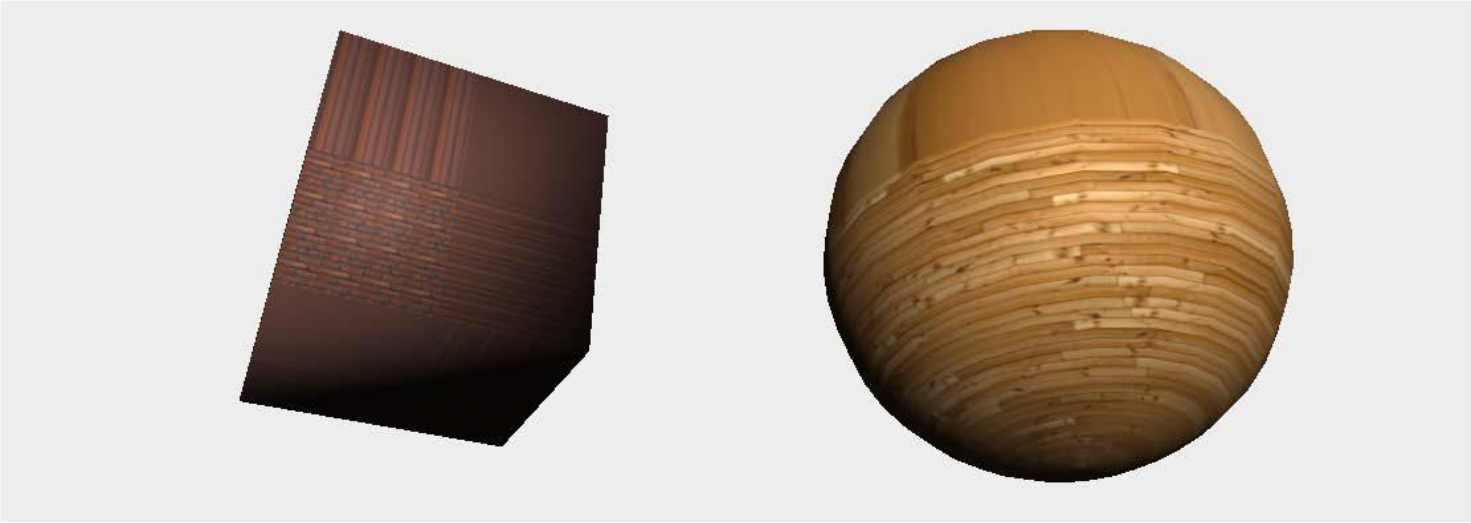
Repeat Wrapping

cube.material.map.repeat.set(repeatX, repeatY);

- repeatX : how the texture is repeated along XX
- repeatY : the same for the YY axis

- If set to 1, the texture won't repeat itself
- If set to a higher value, the texture will start repeating
- Values less than 1 : zoom in on the texture
- A negative value : the texture will be mirrored

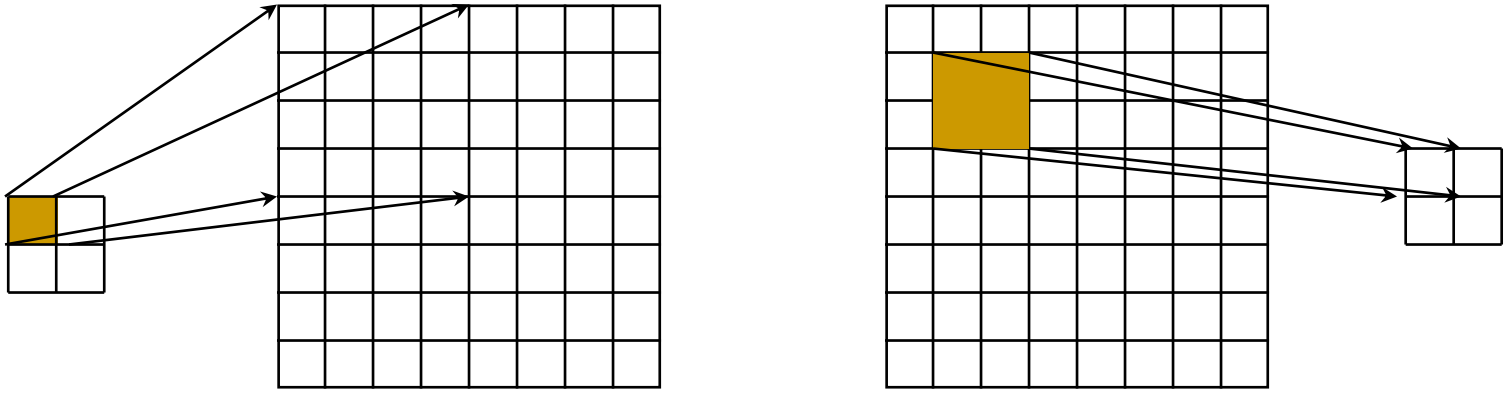
Clamping



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



Magnification Minification

[Ed Angel]

Magnification and Minification

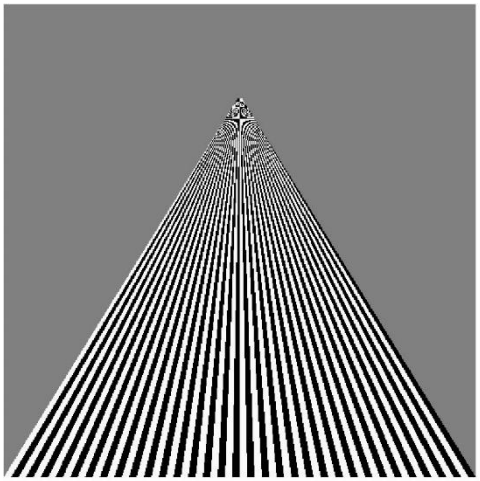
Name	Description
`THREE.NearestFilter`	This filter uses the color of the nearest texel that it can find. When used for magnification, this will result in blockiness, and when used for minification, the result will lose much detail.
`THREE.LinearFilter`	This filter is more advanced and uses the color value of the four neighboring texels to determine the correct color. You'll still lose much detail in minification, but the magnification will be much smoother and less blocky.

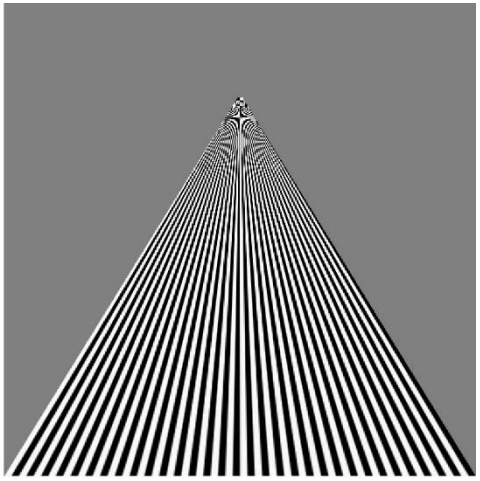
Mipmapped Textures

- *Mipmapping* allows for prefiltered texture maps of decreasing resolutions
- Set of texture images, each half the size of the previous one
  - Created when the texture is loaded
  - Allows for smoother filtering
- Lessens interpolation errors for smaller textured objects

Example

point sampling

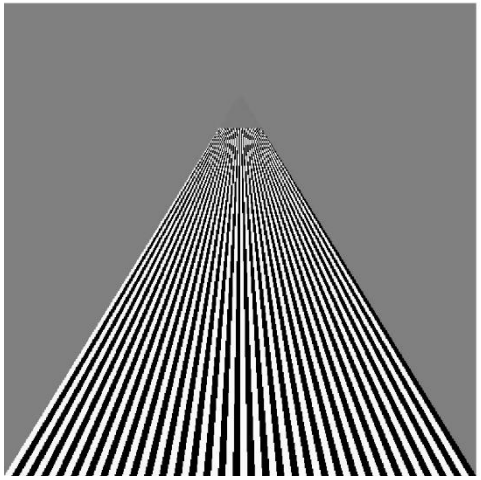


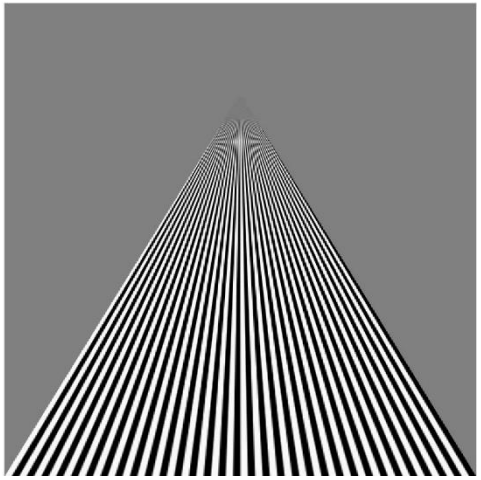


linear filtering

[Ed Angel]

mipmapped point sampling





mipmapped linear filtering

Mipmapped Textures

Name	Description
`THREE.NearestMipMapNearestFilter`	This property selects the mipmap that best maps the required resolution and applies the nearest filter principle that we discussed in the previous table. Magnification is still blocky, but minification looks much better.
`THREE.NearestMipMapLinearFilter`	This property selects not just a single mipmap but the two nearest mipmap levels. On both these levels, a nearest filter is applied to get two intermediate results. These two results are passed through a linear filter to get the final result.
`THREE.LinearMipMapNearestFilter`	This property selects the mipmap that best maps the required resolution and applies the linear filter principle we discussed in the previous table.

Mipmapped Textures

Name	Description
`THREE.LinearMipMapLinearFilter`	This property selects not a single mipmap but the two nearest mipmap levels. On both these levels, a linear filter is applied to get two intermediate results. These two results are passed through a linear filter to get the final result.

What can we do ?

- Use a bump map to create bumps / wrinkles

[Dirksen]

Loading and applying a bump map

```
function createMesh(geom, imageFile, bump) {
    var texture =THREE.ImageUtils.loadTexture(imageFile);
    var mat = new THREE.MeshPhongMaterial();
    mat.map = texture;
    if (bump) {
       var bump =THREE.ImageUtils.loadTexture(bump);
       mat.bumpMap = bump;
       mat.bumpScale = 0.2;
    }
    var mesh = new THREE.Mesh(geom, mat);
    return mesh;
}
```

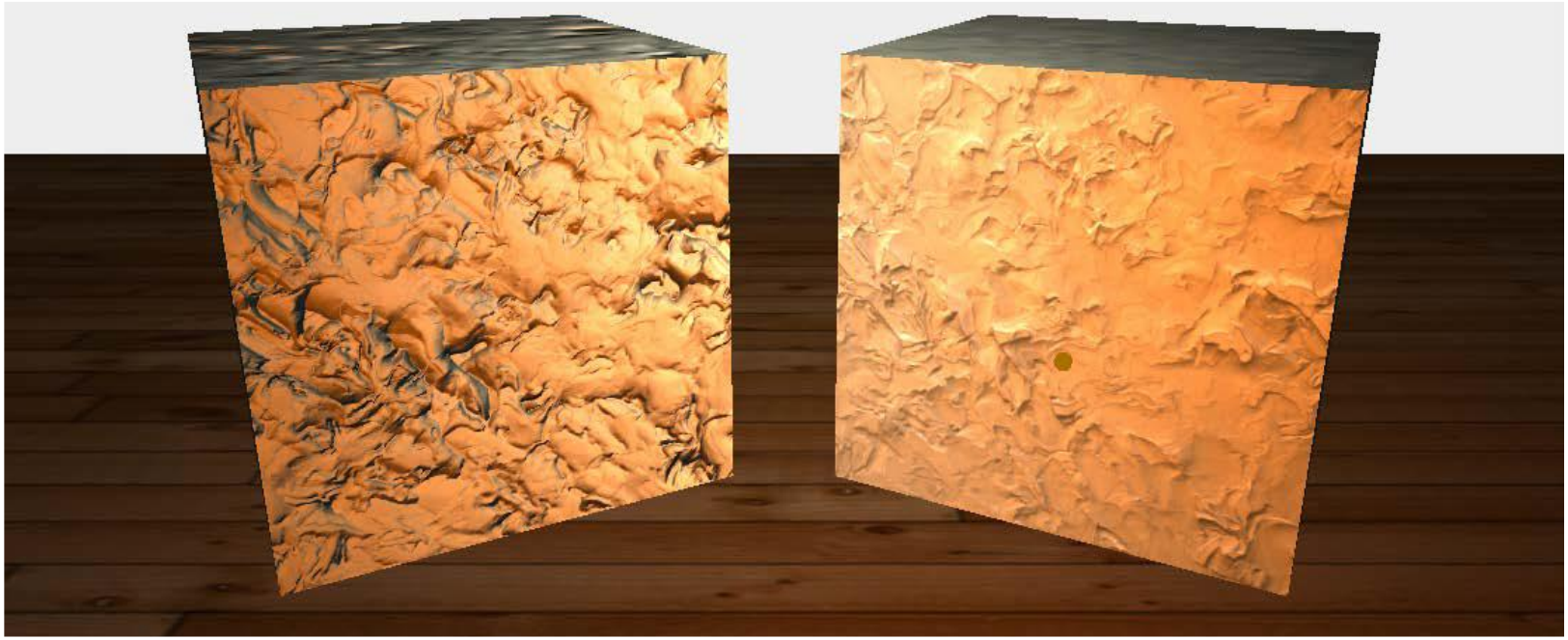
Texture and Bump Map

The Bump Map

- Grayscale or color image
- Intensity of each pixel defines the height of the bump
- Only contains the relative height of a pixel
  - No direction / slope information
- Limited level of detail and depth perception
- For finer detail, use a normal map

What can we do ?

- Use a normal map to achieve more detail



[Dirksen]

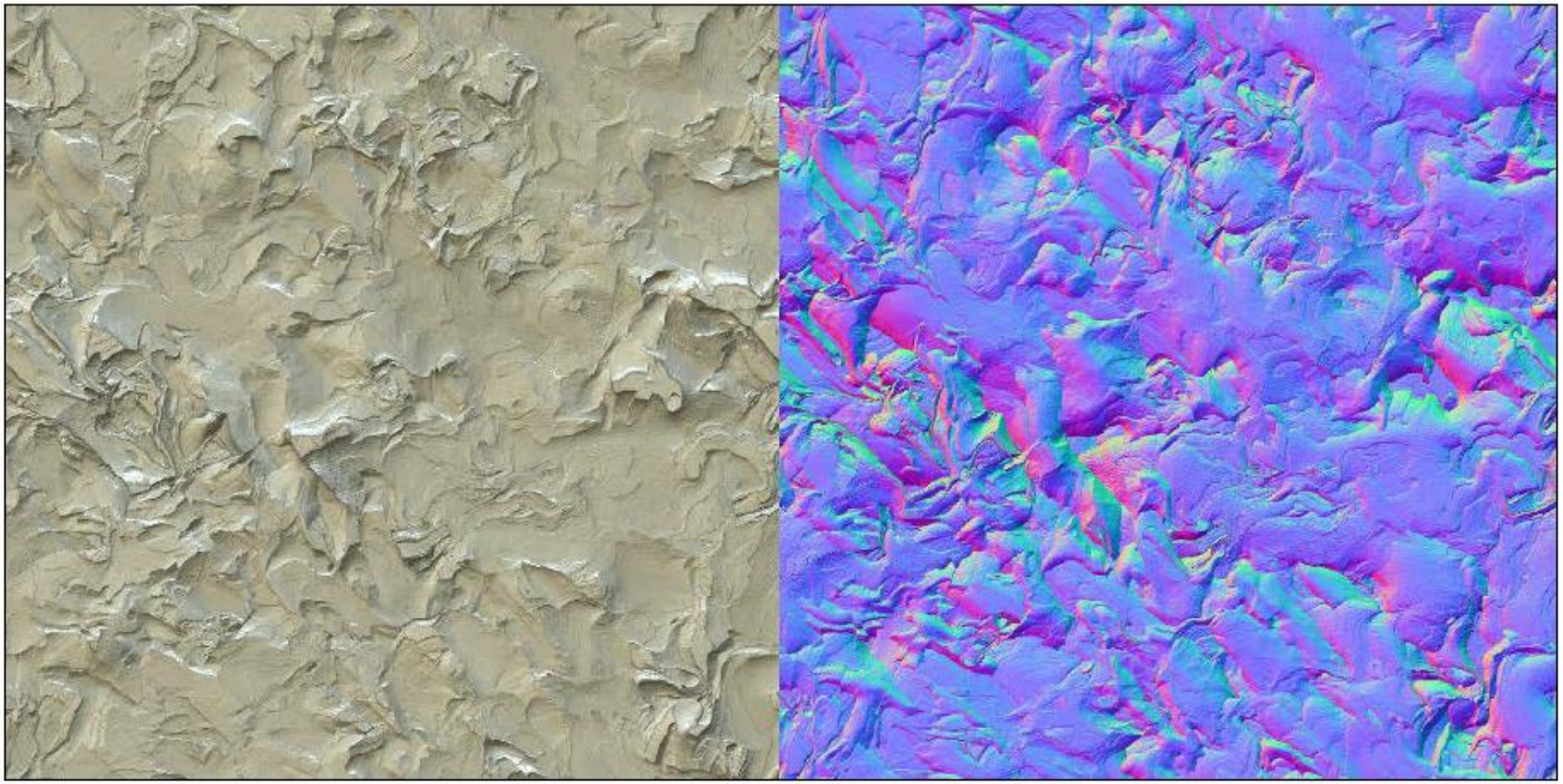
Loading and applying a normal map

```
function createMesh(geom, imageFile, normal) {
   var t = THREE.ImageUtils.loadTexture(imageFile);
   var m = THREE.ImageUtils.loadTexture(normal);

   var mat = new THREE.MeshPhongMaterial();
   mat.map = t;
   mat.normalMap = m;

   var mesh = new THREE.Mesh(geom, mat);
   return mesh;
}
```

Texture and Normal Map

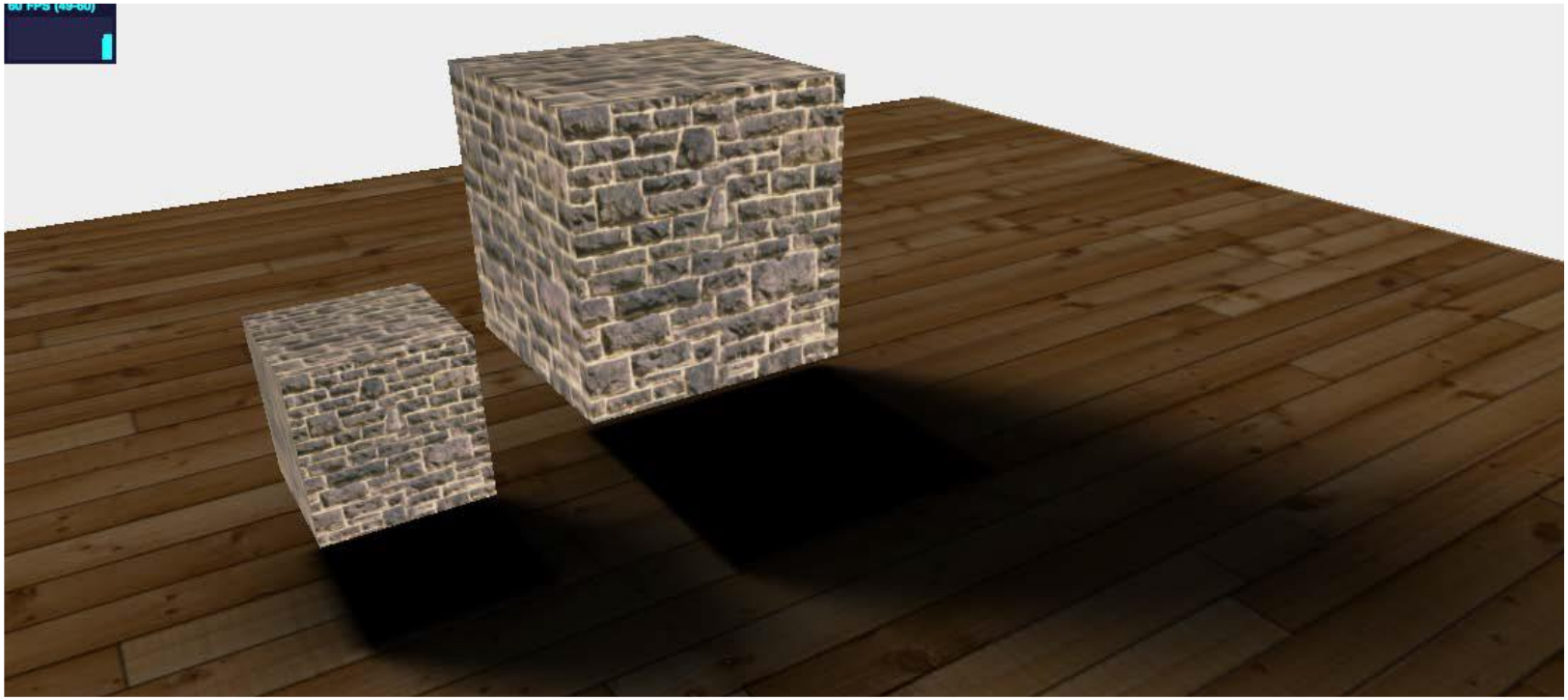


The Normal Map

- Not easy to create !
- Use specialized tools, such as Blender or Photoshop
  - Create normal maps from high-resolution renderings or textures
- Three.js provides a way to do this during runtime
  - getNormalMap() in THREE.ImageUtils
  - Take a JavaScript/DOM image as input and convert it into a normal map

What can we do ?

- Create fake shadows using a light map



[Dirksen]

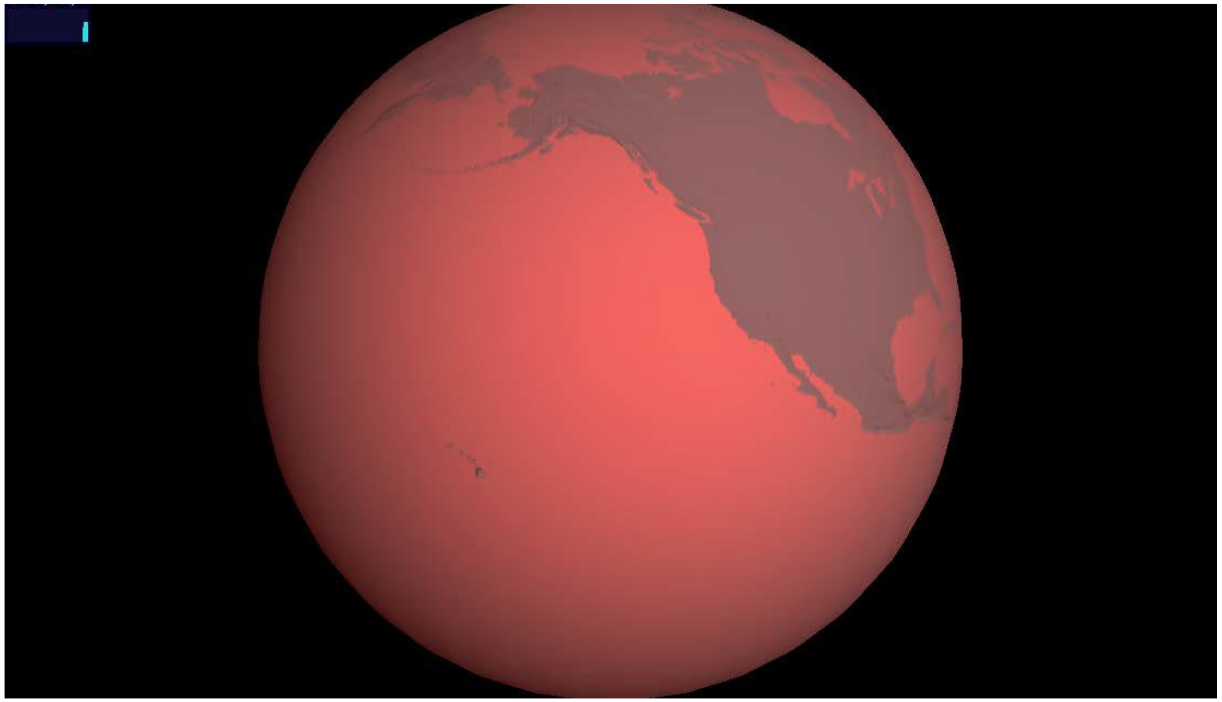
What can we do ?

- Create fake reflections using an environment map

[Dirksen]

What can we do ?

- Specify the shininess and the highlight color of a material with a specular map



[Dirksen]

threejs.org – Examples

- Check the examples !!

- https://threejs.org/examples/?q=texture

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.
- As well as from the Three.js book by Jos Dirksen
- Thanks !