Objectives

- Introduce the OpenGL texture functions and options
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

Geometry

Screen

Image
The texture (below) is a 256 x 256 image that has been mapped to a rectangular polygon which is viewed in perspective.
Texture Mapping and the OpenGL Pipeline

- Images and geometry flow through separate pipelines that join at the rasterizer
  - “complex” textures do not affect geometric complexity
Specify Texture Image

- Define a texture image from an array of *texels* (texture elements) 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
  - no need to enable as in desktop OpenGL
  - desktop OpenGL supports 1-4 dimensional texture maps
Define Image as a Texture

```c
glTexImage2D( target, level, components, w, h, border, format, type, texels );
```

target: type of texture, e.g. `GL_TEXTURE_2D`
level: used for mipmapping (discussed later)
components: elements per texel
w, h: width and height of texels in pixels
border: used for smoothing (discussed later)
format and type: describe texels
texels: pointer to texel array

```c
glTexImage2D(GL_TEXTURE_2D, 0, 3, 512, 512, 0, GL_RGB, GL_UNSIGNED_BYTE, my_texels);
```
Based on parametric texture coordinates

`glTexCoord*()` specified at each vertex

A

B

C

Texture Space

Object Space

(s, t) = (0.2, 0.8)

(0.4, 0.2)

(0.8, 0.4)
Interpolation

OpenGL uses **bilinear** interpolation to find proper texels from specified texture coordinates.

Can be distortions:
- Good selection of tex coordinates
- Poor selection of tex coordinates

Texture stretched over trapezoid showing effects of bilinear interpolation.
Texture Parameters

- **WebGL**: a variety of parameters that determine how texture is 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

Clamping: if \( s, t > 1 \) use 1, if \( s, t < 0 \) use 0

Wrapping: use \( s, t \) modulo 1

```c
gl.texParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP )
gl.texParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT )
```

<table>
<thead>
<tr>
<th>s</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="texture" /></td>
<td><img src="image2" alt="wrapping" /></td>
</tr>
<tr>
<td><img src="image3" alt="texture" /></td>
<td><img src="image4" alt="wrapping" /></td>
</tr>
</tbody>
</table>

GL_CLAMP wrapping

GL_REPEAT wrapping
Magnification and Minification

Texel smaller than pixel – more than one texel can cover a pixel (minification)
Texel larger than pixel - more than one pixel can cover a texel (magnification)
Can use point sampling (nearest texel - fastest) or linear filtering (2 x 2 filter – less aliasing) to obtain texture values
Filter Modes

Modes determined by

- `gl.texParameteri( target, type, mode )`

```cpp
gl.texParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);

gl.texParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
```
Mipmapped Textures

- **Mipmapping** allows for prefiltered texture maps of decreasing resolutions
- Lessens interpolation errors for smaller textured objects
- Declare mipmap level during texture definition
  
  ```
  gl.texImage2D(GL_TEXTURE_*D, level, ...
  ```

Angel and Shreiner: Interactive Computer Graphics 7E © Addison-Wesley 2015
Example

Point sampling

Linear filtering

Mipmapped point sampling

Mipmapped linear filtering
Examples

- 7E/07/textureCube1
- 7E/07/textureCubev2
- 7E/07/textureCubev4
- 7E/07/textureSquare
Example-Planet Earth
Planet Earth-Picture
function triangle(a, b, c) {
    pointsArray.push(a[0], a[1], a[2]);
    pointsArray.push(b);
    pointsArray.push(c);
    // normals are vectors
    normalsArray.push(a[0], a[1], a[2], 0.0);
    normalsArray.push(b[0], b[1], b[2], 0.0);
    normalsArray.push(c[0], c[1], c[2], 0.0);
    // texture coordinates
    texCoordsArray.push(spherical(a));
    texCoordsArray.push(spherical(b));
    texCoordsArray.push(spherical(c));
    index += 3;
}
Finding Texture Coordinates
function spherical(a) {
    /*
       radius = sqrt(x^2 + y^2 + z^2)
       polar = arccos(z/radius)
       azimuthal = atan2(y, x)
       Radius = 1
    */
    longitude = Math.atan2(a[1], a[0]);
    latitude = Math.acos(a[3])
    return vec2(longitude, latitude)
}
Longitude and Latitude
Texture Mapping
Planet Earth-Mapping
Moon-Mapping