Texture Mapping

A way of adding surface details

Two ways we can achieve this:

Model the surface with more polygons
  • Slow downs rendering
  • Hard to model fine features

Map a texture to the surface
  • Image complexity does not affect complexity of processing
Map textures to surfaces

The polygon can have an arbitrary size and shape. Set as a vertex attribute.
Textures in OpenGL ...

• **glEnable(GL_TEXTURE_2D)**
  turn on the 2D texture store

• **glTexImage2D**
  declares a texture’s size, color components (RGBA, etc), data type (byte, float...), pixel data

• **glBindTexture**
  “bind” the given texture to the active store. Only one texture can be bound at a time. All future configuration and co-ordinates correspond to this texture. For the fixed pipeline, you can only use one texture for rendering. For shaders, you bind textures to uniforms.
Textures in OpenGL Continued...

**glTexParameteri**

Used to set texture configuration:

How are the texture values interpolated?
GL_NEAREST vs GL_LINEAR
- GL_NEAREST rounds to nearest texel
- GL_LINEAR linearly interpolates the texels

Does the texture repeat itself?
GL_REPEATE vs GL_CLAMP
  - Say we have a texture coordinate of (-0.1,1.1)
    - GL_CLAMP changes it to (0.0,1.0)
    - GL_REPEATE changes it to (0.9,0.1)

More options for Texture Parameters can be found here:  
http://www.opengl.org/sdk/docs/man/xhtml/glTexParameteri.xml
Examples of use can be found in the Texture class in the framework
Textures in CS 4620 Framework ... 

Takes the burden of:

- Loading texture files as texture maps (~ glTexImage2D)
- Setting up the texture parameters (~ glTexParameteri)
- Managing the texture units (~ glBindTexture)

Wrapper classes for working with 1D, 2D and 2D Mip-Mapped textures.

Simple interface for using textures with GLSL.
Textures in CS 4620 Framework ...

private Texture2D texture;

public void init(GLAutoDrawable drawable) {
    super.init(drawable);

    final GL2 gl = drawable.getGL().getGL2();

    try {
        texture = new Texture2D(gl,
                "data/textures/sample.jpg");
    } catch (IOException e) {
        System.out.print("Can't load texture: ");
        System.out.println(e.getMessage());
        Terminate();
    }
}
protected void drawTexturedQuad(GL2 gl) {
    texture.use();

    // Draw stuff

    texture.unuse();
}
Texturing in GLSL/Pipeline
Texturing in GLSL

New elements:

sampler2D (type)

vec4 texture2D(sampler2D,vec2) (function)

vec2 in_TexCoord (attribute)
In the OpenGL app, we have to bind the desired texture to the sampler uniform

**Inside Init()**

```java
// Load the 2D texture
texture = new Texture2D(gl, 
    "data/textures/sample.jpg");

// Get the sampler uniform
samplerUniform = 
    textureShaderProgram.GetUniforms().
    get("sampler");

// Load, compile and link the shaders
textureShaderProgram = new 
    Program(gl, vertexFileName, 
        fragmentFileName);
```

**Inside Render()**

```java
texture.use(); // Make it the active texture unit
textureShaderProgram.use(); // Activate the shader

// Bind the active texture unit to the sampler uniform
TextureUnit.getActiveTextureUnit(gl).bin
dToUniform(samplerUniform);

draw(gl); // Render your scene

// Revert the changes
textureShaderProgram.unuse();
texture.unuse();
```
Figure out the coordinate that we want to sample from using `in_TexCoord`

```glsl
attribute vec2 in_TexCoord
varying vec2 coord;

void main() {
  gl_Position = ...

  coord = vec2(in_TexCoord);
}
```
Texturing in GLSL – Fragment Shader

Take the coordinate data from the vertex shader and sample the appropriate pixel from the desired texture

```glsl
varying vec2 coord;
uniform sampler2D sampler;

void main() {
    gl_FragColor = texture2D(sampler, coord);
}
```