TEXTURE MAPPING
Texture Mapping

- A way of adding surface details

- Two ways can achieve that goal:
  - 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

- Use `glTexCoord2f(s, t)` to specify texture coordinates for each vertex in object space

- State machine: texture coordinates remain valid until you change them or exit texture mode via `glDisable(GL2.GL_TEXTURE_2D)`

Example:

```
gl.glBegin(GL2.GL_QUADS);
  gl.glTexCoord2f(1, 1);
  gl.glVertex3f(1.0f, 1.0f, 0.0f);
  gl.glTexCoord2f(0, 1);
  gl.glVertex3f(-1.0f, 1.0f, 0.0f);
  gl.glTexCoord2f(0, 0);
  gl.glVertex3f(-1.0f,-1.0f, 0.0f);
  gl.glTexCoord2f(1, 0);
  gl.glVertex3f( 1.0f,-1.0f, 0.0f);
gl.glEnd();
```
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

- **glTexParameteri**
  - set texture configuration: how does the texture wrap? How are nearest-pixel values interpolated?

- **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.
Textures in CS 4621 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 4621 Framework ...

```java
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();
    gl.glBegin(GL2.GL_QUADS);
    {
        gl.glTexCoord2f(1, 1);
        gl.glVertex3f( 1.0f, 1.0f, 0.0f);
        gl.glTexCoord2f(0, 1);
        gl.glVertex3f(-1.0f, 1.0f, 0.0f);
        gl.glTexCoord2f(0, 0);
        gl.glVertex3f(-1.0f,-1.0f, 0.0f);
        gl.glTexCoord2f(1, 0);
        gl.glVertex3f( 1.0f,-1.0f, 0.0f);
    }
    gl.glEnd();
    texture.unuse();
}
Texturing in GLSL
Texturing in GLSL

- New elements:
  - sampler2D (type)
  - texture2D (function)
  - gl_MultiTexCoord0 (uniform variable)
Texturing in GLSL – Vertex Shader

- Figure out the coordinate that we want to sample from `gl_MultiTexCoord0`

```glsl
varying vec2 coord;

void main() {
    gl_Position =
        gl_ModelViewProjectionMatrix * gl_Vertex;

    coord = vec2(gl_MultiTexCoord0);
}
```
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);
}
```
Texturing in GLSL – OpenGL App

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).bindToUniform(samplerUniform);

draw(gl); // Render your scene

// Revert the changes
textureShaderProgram.unuse();
texture.unuse();
```
Examples ...

- Simple texturing example using the fixed pipeline
- The same example, but using GLSL
- Toon shader with 1D texture map