Questions on the Final Project

Mon, 16 Dec 2013 16:18:34

```
> One other concern of mine is, I've gotten multiple objects to render
> properly, but did so by having multiple shaders; when went to submit, my
> submission was rejected because I had "an odd number of .glsl files."
> Should I just go ahead and the way I do the multiple objects?
```

That's my (extremely simplistic) attempt to ensure that all the shader source files were submitted. On the theory that vertex and fragment shaders must be paired (i.e., one of each in every shader program), the 'try' scripts require that you submit at least two .glsl files, and if you submit more than two there must be an even number of .glsl files.

Unfortunately, if you're using the same fragment (or vertex) shader in more than one shader program, this means that you have to duplicate it, which is kind of a waste of space but works with the simplistic test I'm using.

Mon, 16 Dec 2013 08:55:44

```
> I didn't know if you were aware of this or not, but the final doesn't put
> JOGL on the classpath as all of our other assignments have had. I was able
> to submit by submitting the two jars with it, but I didn't know if that was
> how you wanted it done or not.
```

No, that wasn't the intent. Thanks for letting me know.

I've modified the 'try' scripts to add the two JOGL jar files if they aren't submitted. (The scripts also add Jama 1.0.3 if a Jama jar file isn't submitted.)

Sat, 7 Dec 2013 13:03:04

```
> When drawing multiple objects where each object uses a different shader,
> where do the multiple shaders and objects get specified? I know shaders are
> read and compiled with a shaderProgId and (I think?) you need to use
> multiple vertex buffers, but I'm not sure how you associate individual
> objects/vertex buffers with its correct shaders.
```

For each shader program you want to use, the first step is to load, compile, and link the vertex and fragment shaders for that program. This can be done by calling shaderSetup() if you're working in C or C++, or by calling readAndCompile() if you're working in Java. See the init() function in the textureMain.* source file for the texture mapping assignment for an example.

Those functions return a shader program ID to you. In order to use this shader program, just call glUseProgram() with the appropriate shader program ID; any objects you draw from that point onward

will be drawn using that shader program. To change shaders, just call glUseProgram() again with the next shader program ID.

To draw multiple objects, you have to set up both a vertex buffer and an element buffer for each object. The basic sequence for a single object is as follows:

- clear the existing shape (if any)
- create the object shape
- allocate a vertex buffer ID
- bind the vertex buffer ID as the current vertex buffer
- copy the vertex data into the buffer
- allocate an element buffer ID
- bind the element buffer ID as the current element buffer
- copy the element data into the buffer
- record the buffer IDs and the vertex count somewhere

See the createShapes() function in textureMain.* for an example. For variables that are being texture mapped, remember that "vertex data" includes locations and texture coordinates; if you're associating color with the vertices, "vertex data" includes locations and colors; etc. To set up multiple objects, you just need to repeat this process once for each object. To keep track of the IDs and vertex counts for the objects, the easiest way is to create global arrays which you index in parallel - e.g., for object N, you would use vbufid[N], ebufid[N], and nverts[N].

The basic sequence for drawing one object is this:

- select the shader program to be used for this object
- bind the vertex and element buffers
- set up all attribute variables used in the shader program
- set up all uniform variables used in the shader program
- draw the object

See the display() function in textureMain.* for an example. You'll need the vertex and element buffer IDs and the vertex count that you saved when you set up the vertex and element buffers for this object.
