**RenderEngine Lab 07 – Shaders Galore! Revision 92**

Let’s get shaders working in the Render Engine by using callbacks to set up uniforms as needed for your shaders. The basic idea is that the RenderEngine knows nothing about the game, but is happy to call back to game methods as needed. These callbacks allow the game to set up shader uniforms for any object that needs them. By the end of this lab, you should be able to move all or at least most of the openGL calls out of the game and into the RenderEngine, except for uniform set up.

1. Set up your buffers to segregate objects by shaders/programs, if it doesn’t already.
2. Use your vertex and fragment ShaderInfo’s to implement shaders/programs. Let your RenderEngine do as much shader/program set up as you can.
3. Add a callback member / getter / setter to the appropriate structure (Renderable? ShaderInfo?) which will allow the RenderEngine to call back to the game before displaying an object that needs uniforms to be properly set for the shader that the object uses.
   1. Always initialize the callback to 0/null if you don’t already know the callback value.
4. The callback will probably have to pass a pointer to the object so the callback method can operate properly. If this doesn’t make sense at first, just implement the callback without any parameters…you can always add one or more parameters later once you understand the issues better.
5. If the callback is not null for an object, the render engine will do the callback before drawing the object.
6. ***Output***
   1. Make it obvious that your RenderEngine can handle multiple shaders.
   2. Make sure your shaders use your callback mechanism to set uniforms.
   3. Give some details how & where you implemented shaders / programs / callbacks / uniforms so your lab can be easily be evaluated / graded.
   4. Shader Examples:
      1. Use a texture shader on one or more objects.
      2. Use a color-tinted texture shader on one or more objects.
      3. Bring back your lighting / normals shader for one or more objects.
      4. Research shaders and implement some cool easy example!

Fulfillments:

1: no longer required

2: my render engine now has a function for creating shaders and programs. This function is very similar to runGame’s original function though it was modified to fit the new format.

3: I have added a callback to my shaderInfo, this allows me to store a function and call said function within renderable’s draw. There are of course getters and setters for this function. My callback function is slightly different in that I setup the function and shaders in such a way as to require a callback before drawing, as such the render engine itself does not use the callback and if I had setup render Engine to use the callback many things would need to be redesigned as my engine calls a function in my buffers that iterates over my renderables.

4: My callback does indeed pass a pointer to the object that is being drawn

5: see above explanation, as such callback is never null.

6: I have created a battleship type object which is affected by lighting as are two of my dragons. The other two dragons do not use lighting and as such are perfectly visible.

My lighting is currently static as I was having some problems with my game model not remembering data correctly, will hopefully be fixed soon but was not necessary for the lab.

Controls:

WASD – movement

RF – vertical movement in camera 0

J G – camera jumping and such

Additional:

I had a strange problem with my objconverter after I finished the lab in which it will not run saying that it cannot open the exe. I am unsure what is wrong with it and will be seeing lamb about it Tuesday.