Lab 02 Shader Features

**Fulfillment**

* I have met all the requirements for the lab

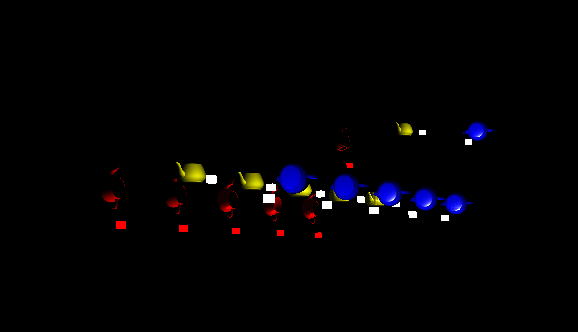
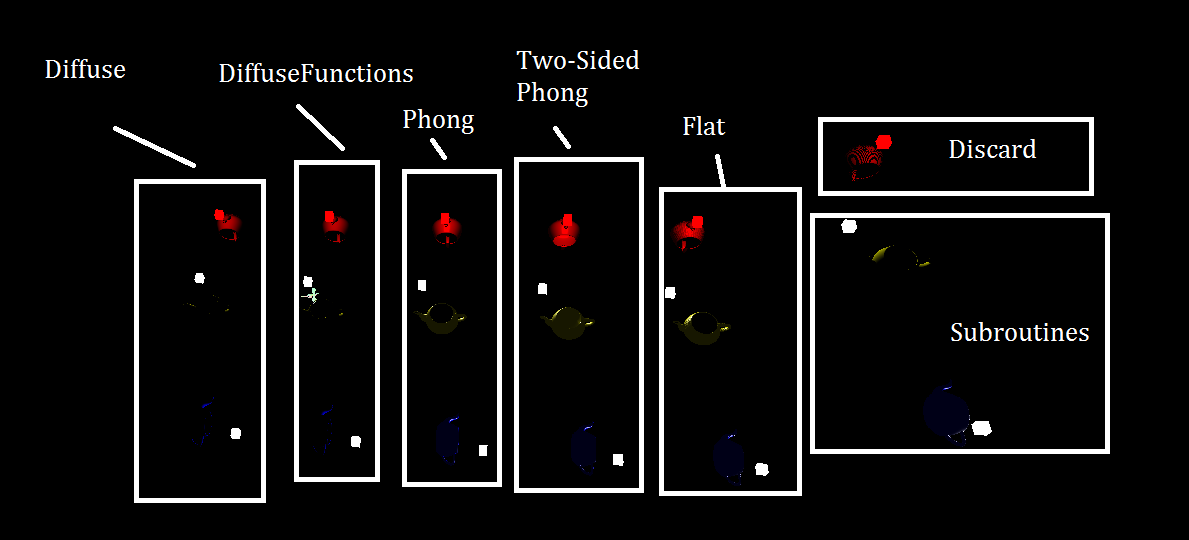
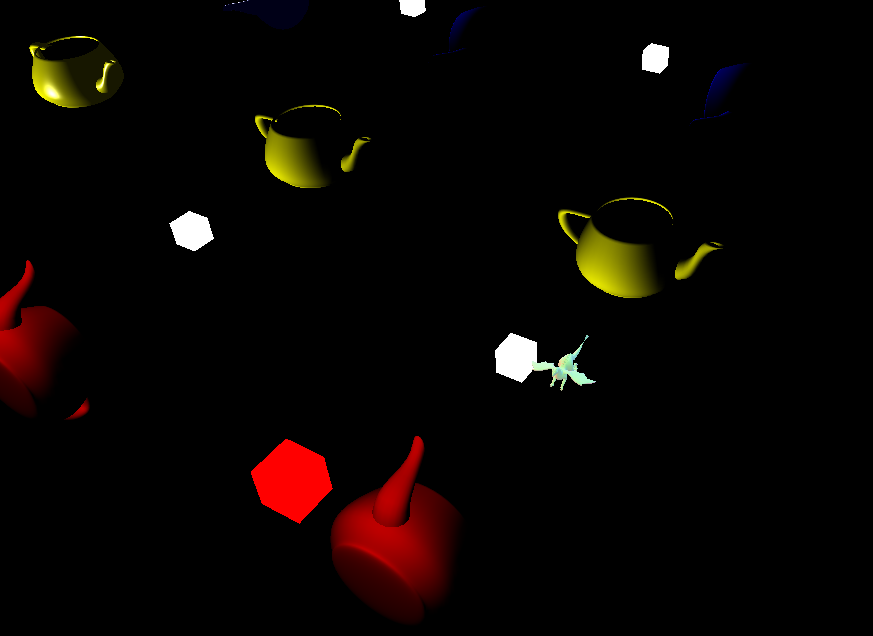
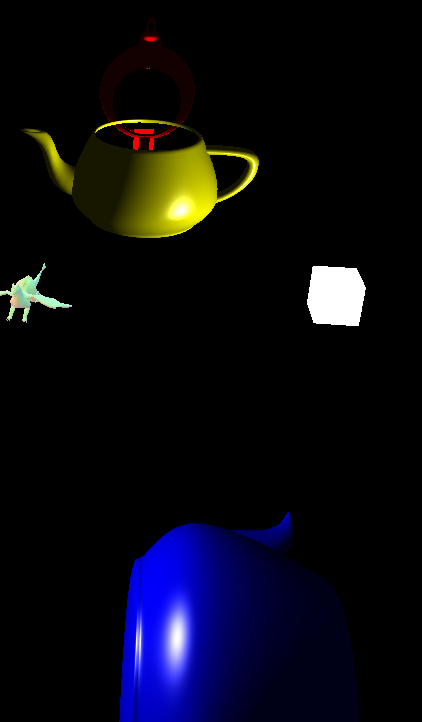
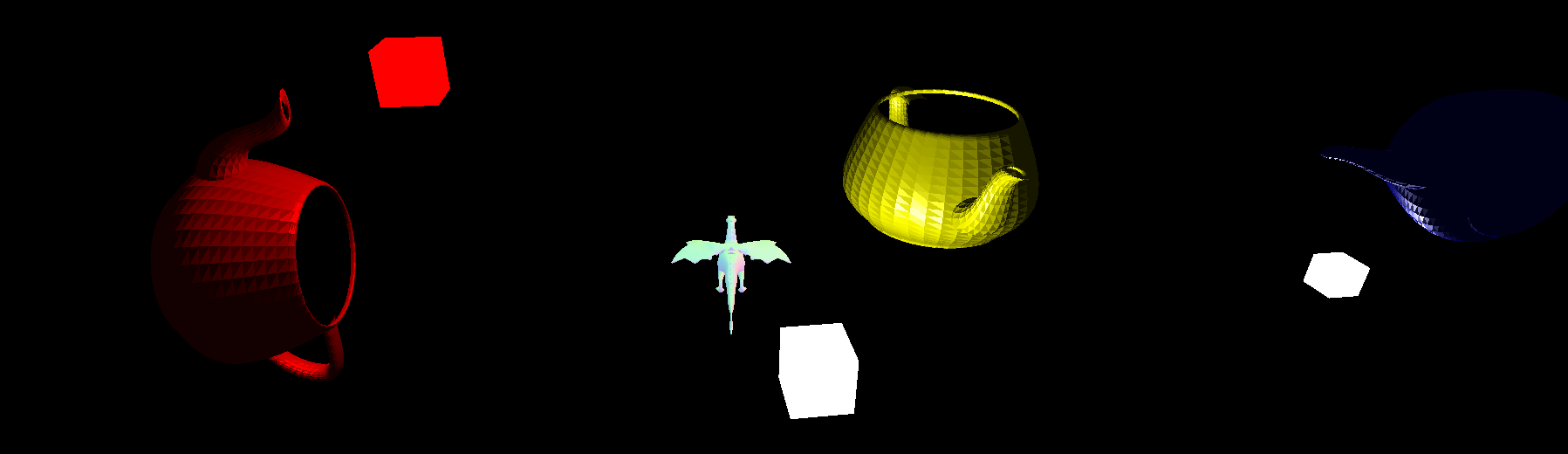
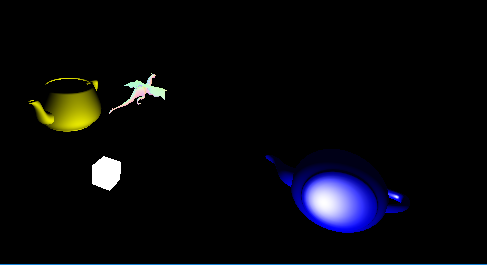
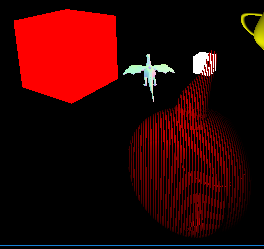
**Execution**

* I do not believe there should be any unexpected requirements for running

**Controls**

* Pressing the X key will close the application
* Pressing the P key will pause the application, pressing it again will un-pause the application
* Pressing numpad 0 will re-read the config file
* Pressing M, L, T or C will dump engine info to the console, this is pretty much exclusively used for debugging
* Pressing W will rotate BetterDargon to the left, S will rotate him to the right
* Pressing A will tilt BetterDargon forward, D backward
* Pressing Q will roll BetterDargon to the left, E to the right
* Holding space will move BetterDargon forward, in the direction he is facing, releasing will halt movement
* Right clicking and dragging the mouse will turn the camera around BetterDargon
* Scrolling in or out with the mouse wheel should zoom the camera accordingly, up to a minimum or maximum distance

**Screenshots**

* The following screenshot shows a zoomed-out view of all the demo teapots for this lab
* 
* The next screenshot is annotated manually in paint to give a nice overview of which teapots are demonstrating what shaders. Left to right, you can see the old diffuse shader for reference, the diffuse functions shader producing the same output but using functions, the phone shader, shading only one side of the teapot, the two-sided phong shader shading both sides of the teapot, the flat shader shading with no interpolation, the shader subroutines (one using the diffuse function, the other using the phong function) and, finally, the simple discard shader.
* 
* The following screenshot shows the diffuse functions shader and the regular diffuse shader side-by-side to demonstrate that they produce the same output, despite the use of functions.
* 
* The following screenshot shows the one-sided-phong shader working on teapots with culling enabled
* 
* The following screenshot shows the teapots lit using the two-sided phong shader on objects for which culling is disabled – and yes, I did add support for culling enabling and disabling for meshes in my render engine and other affected places. Note that I have set culling to be enabled for all of the objects demonstrating shaders except those using this two-sided one.
* 
* The next screenshot shows the teapots lit with the flat shading phong
* 
* The next screenshot shows two teapots being lit with the subroutine shaders. The yellow teapot has a uniform set up that passes the index for it to use the diffuse method, and the blue teapot is set up for it to use the phong method.
* 
* The final screenshot shows the discard shader discarding fragments for the red teapot, which actually looks cooler than I expected
* 

**Post-Mortem**

* It’s impossible to code something of any complexity or newness without running into any errors (the first time you do it is always the hardest), but I, again, only ran into tiny issues that were more of a time-sink than anything else, and I got them all fixed too, so none of them linger on.
* I might have re-structured some stuff for organization purposes and further subdivide my larger methods into smaller ones to avoid repetition, but I was in get-it-working-mode, not write-perfect-code-mode (which is usually the case when doing anything new).
* I actually like how some of these shaders look and work, but my favorite thing about this lab was that I managed to improve my engine along the way. I found and fixed bugs, as well as improving some things. If I had more time (like, a lot more time), I’d definitely do some much needed cleaning-up/user-friendlying to my engine so that it is more ready for shader testing and provides more, easier ways to do shader stuff.
* I actually think this lab was better-done than some of the others. I really enjoyed the mathematical explanations complete with useful diagrams as well as the images depicting the differences between the different shader outputs. I think this is a far superior way of doing things than previous labs step-by-step instructions. There is a lot more to gain from figuring something out by looking at the math/diagrams than by writing the exact code we are told to write.
* I have realized, upon putting a little more time into using my engine, that the Shader Uniform thing is actually cooler than I first thought. It is really cool because I can now log what values are being passed into my Shaders from one place. This is an amazingly useful debugging technique as it makes it clear what values are not being sent/what values are being sent incorrectly.
* I really liked how I was able to make my engine a little better along the way with this lab
  + Not only did I fix two or three bugs, but I added a material class, did the uniform thing, and refactored some code
    - I feel like, if I get a little bit of this done each lab, my engine could end up in a much better place at the end of the quarter, however, I do feel a little bad knowing that I now have a superior version of my own engine than the version I gave the others, and I worry they will run into similar issues I did with my engine.