Lab 05 Geometry shaders

**Fufillment**

* I have met all the requirements for the lab, and even did some extra this time involving mouse input

**Execution**

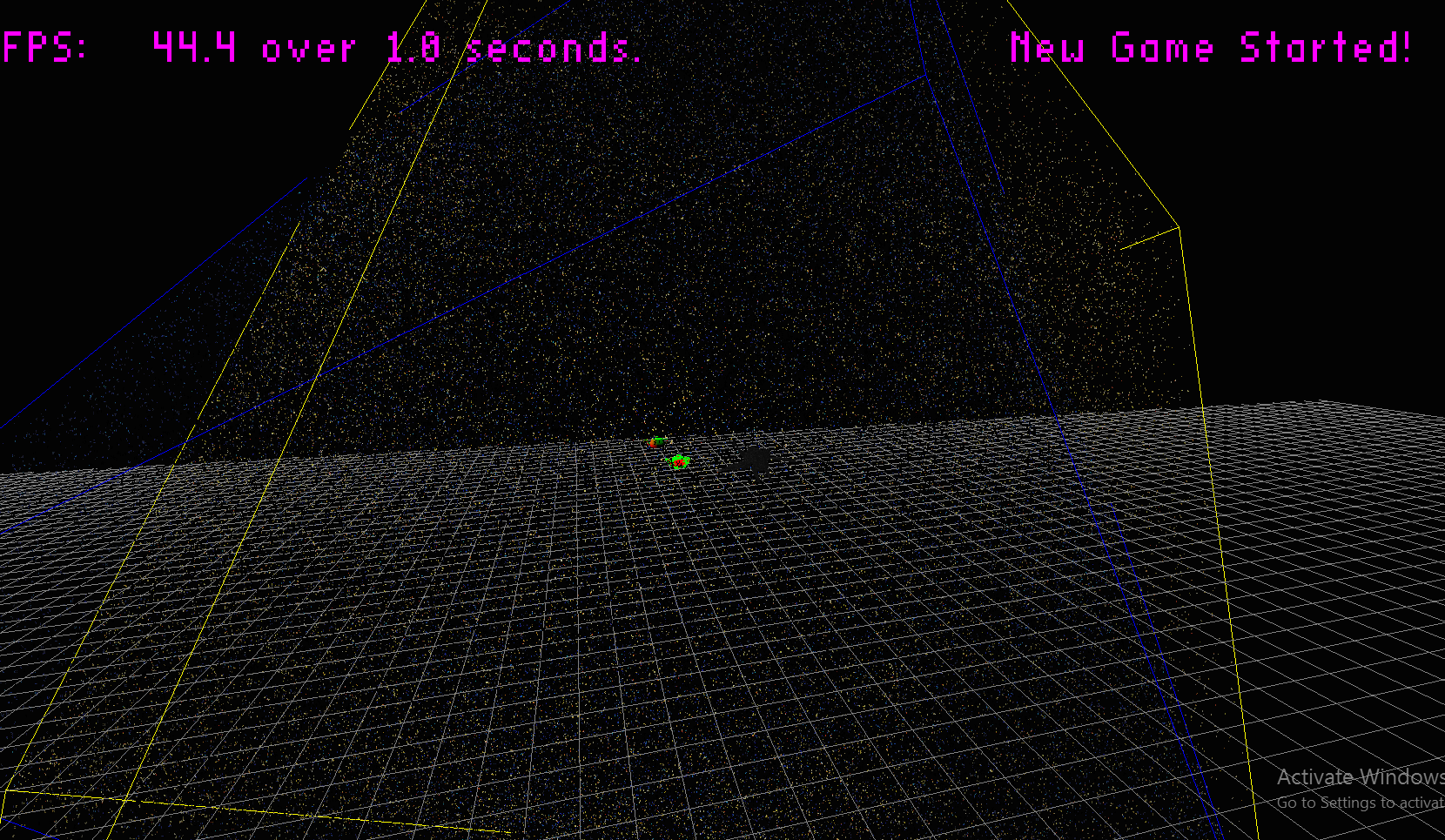
* I do not believe there should be any unexpected requirements for running

**Controls**

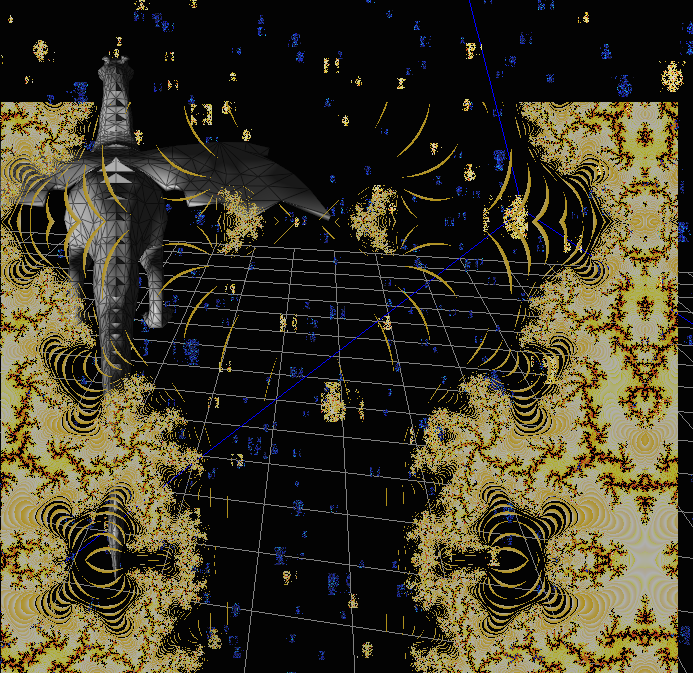
* Pressing the X key will close the application
* Pressing the P key will pause the application, pressing it again will un-pause the application
* Pressing numpad 0 will re-read the config file
* Pressing M, L, T or C will dump engine info to the console, this is pretty much exclusively used for debugging
* 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
* Pressing j will increase the wireframe distance for dargon
  + It will be altered by less if k is held, and the direction will be negated if shift is held
    - So shift+j is backwards by j, j+k is forward by a little, shift+j+k is backward slow
* Pressing 1 will make the second teapot puffier
* Pressing 2 will make it less puffy and invert it (depending on how far you go)
* Pressing N will cycle through the normal hair modes

**Screenshots (NOTE: Humorous comments included deliberately in addition to normal descriptions to make this doc more enjoyable to read. ☺)**

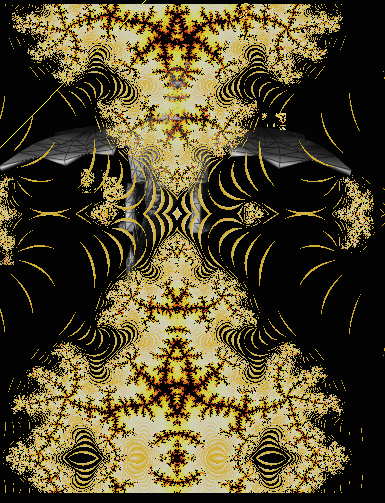
* **Billboards**
  + This screenshot shows an overview of the billboard sector. Note that the bounding volumes are cubes of size n by n by n, and there are one hundred thousand billboarded fractal shaders of each color. This may seem like a boring screenshot, because its so distant the detail cannot be seen – however, it does show that the createpoints method works – the points are all contained within their respective bounding volume – it also shows that the bounding volumes are implemented.

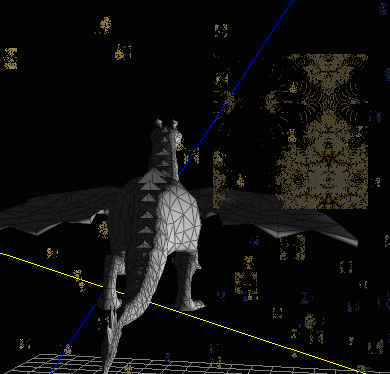


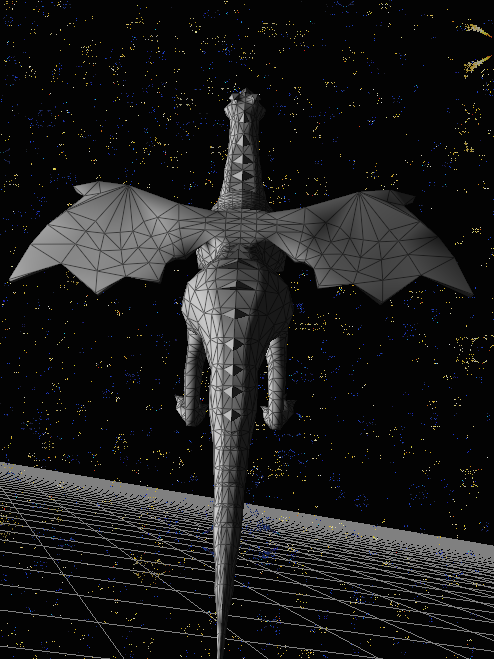
* + The next screenshot shows a close up of a billboarded fractal shader, which is very interesting and super cool looking. Note that it is discarding and does make use of a texture to choose its colors.

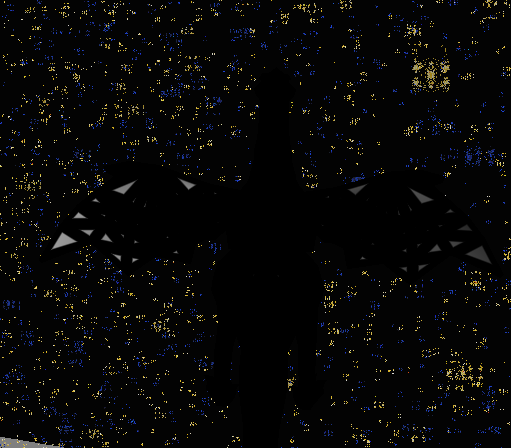


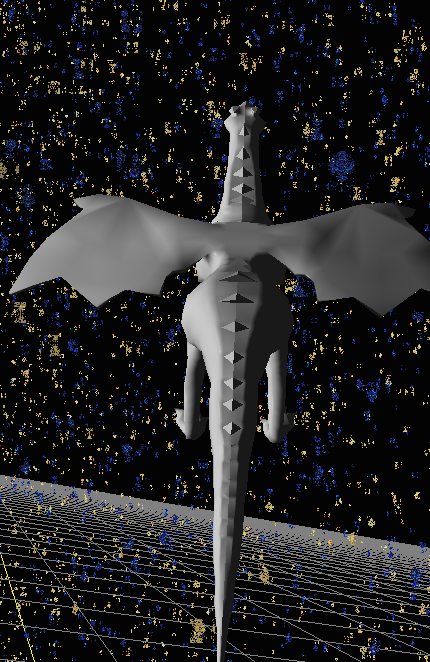
* + The next screenshot shows a fractal billboard facing the light source, as you can see it is well-lit



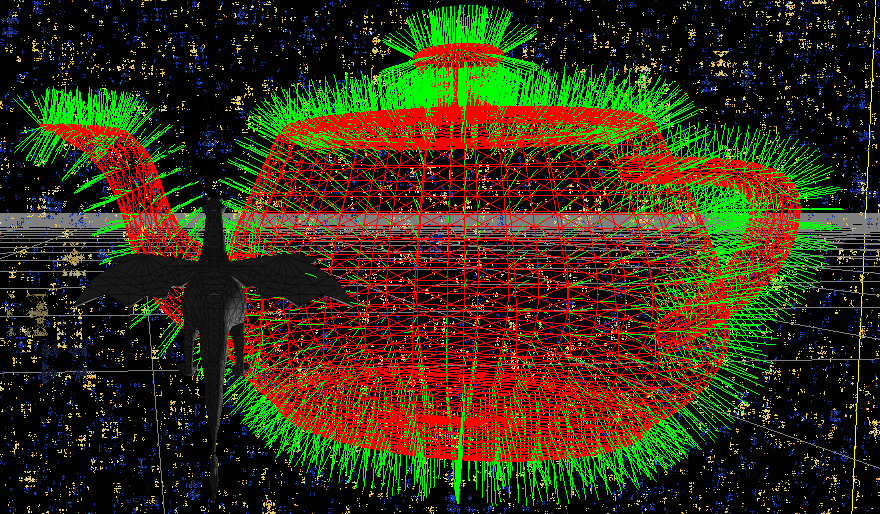
* + The next screenshot shows another fractal billboard, but this one is facing away from the light source. The contrast between the two screenshots shows that the billboards are being lit
  + 
* **Wireframe screenshots**
  + Dargon is shown here with a nice, thin wireframe

.

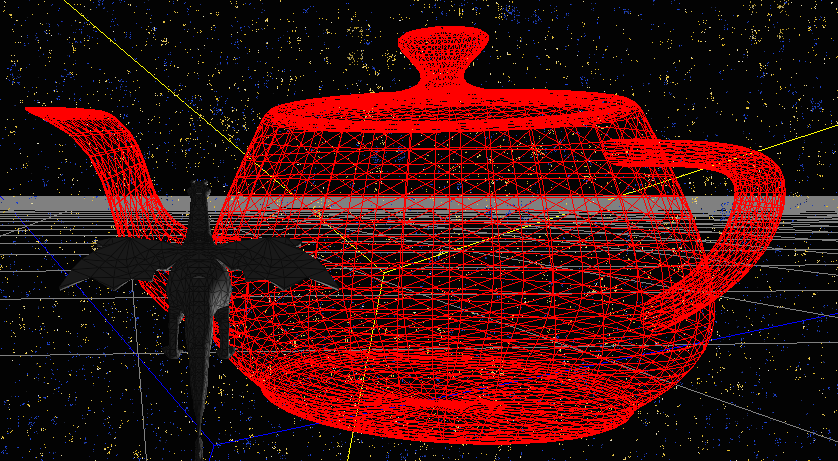
* + The next screenshot shows Dargon with a very thick wireframe, notice how because he is so low-poly you can see some of the mesh underneath despite these huge lines.
  + …
  + Here you can see dargon with wireframes disabled, not drawn with a width of zero



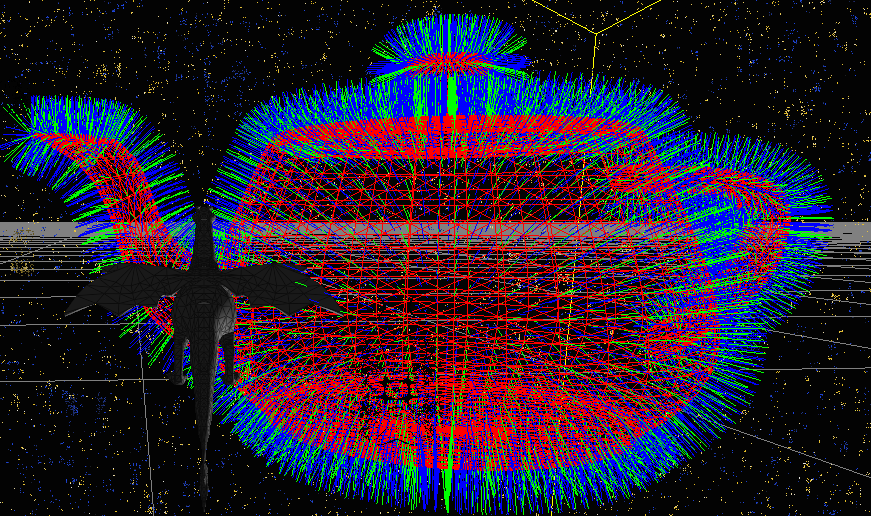
* **Normal Hair Screenshots**
  + Here you can see the teapot used to demonstrate normal hair with face normal hair disabled. Visible are the teapot lines (red) and the vertex normal hairs (green)



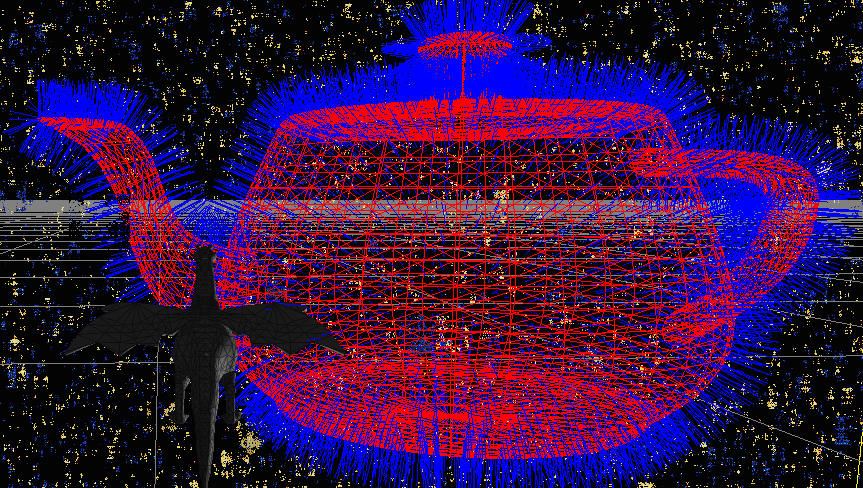
* + Here you can see the teapot with vertex and face normal disabled. Visible are the mesh lines (red)



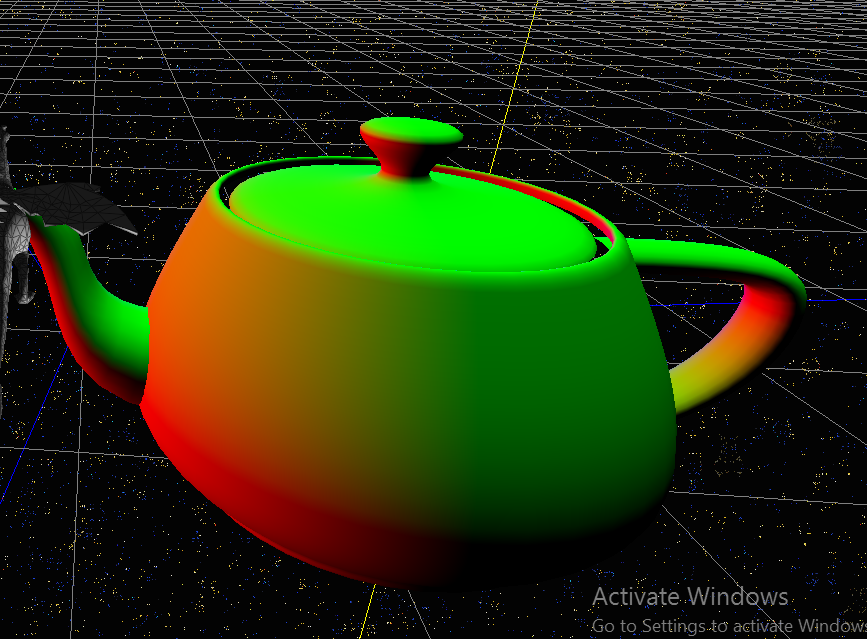
* + Here you can see the teapot with both vertex normal hairs (green) and face normal hairs (blue) as well as the mesh lines (red)



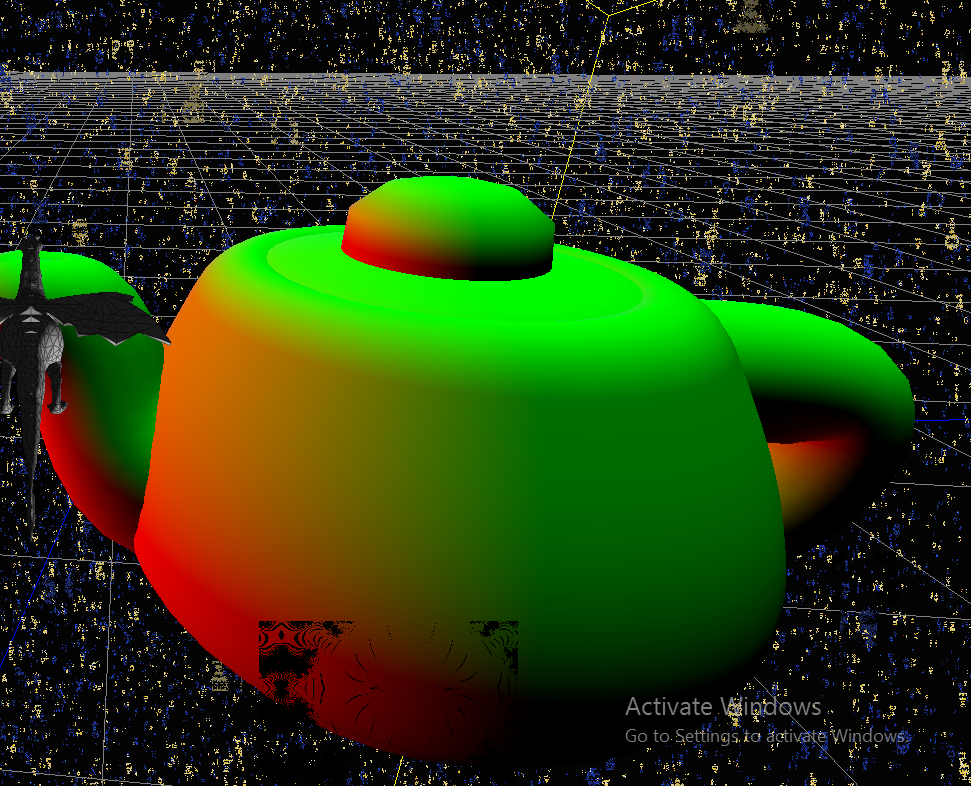
* + Finally, for completeness, included is the teapot with only face normal hairs (blue) and the teapot mesh lines (red)



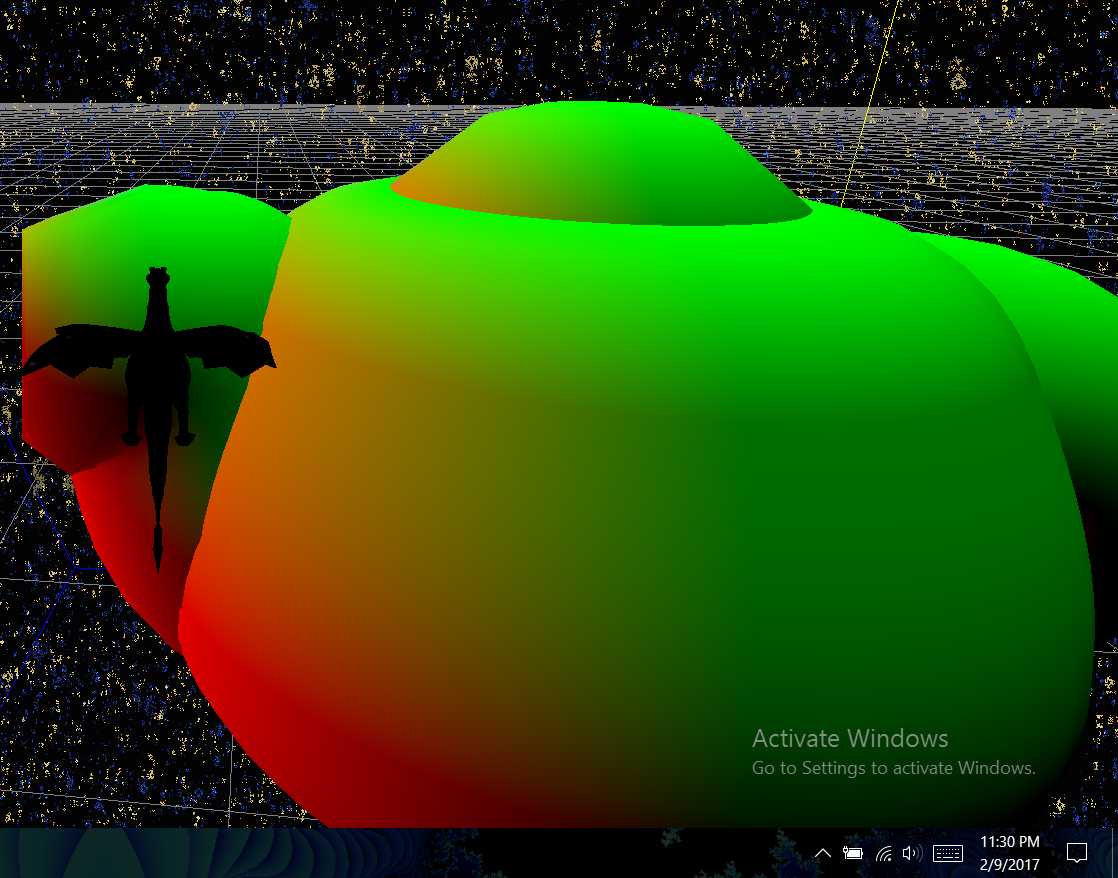
* **Something cool screenshots**
  + I made a teapot-puffer-upper for this – the first screenshot, however, is just for reference, You can see the base teapot here:



* + Now, you can see the teapot fattened up a little bit, through the application of math



* + And here you can see it even more fattened, with the same amount of math



**Post-Mortem**

* I’ve been told I can do the post-mortem later, and am taking advantage of this.
* Previous comment:
  + My sincerest apologies, but this time there will be no post-mortem, because it is 11:45 and I need to obsessively test my code before I turn it in at the last second.
    - If you’re wondering, this week was exceptionally time-consuming in other classes for a variety of reasons, also I ran into a few issues with this lab that set me back a few days, but here it is, at the last second, barely on time, not the cleanest - yet still functional.
* New Comment:
  + As is stated above, I was extremely busy this week, and am totally behind on sleep as of right now, however, I’ve found a little bit of time to re-write this (because coffee)
    - I started this lab extremely excited for geometry shaders actually, mainly because of the effective use of videos to show us cool stuff in advance. I’m not sure if you were aware, but despite our jokes about billboards, I was pretty excited to know about all this cool stuff we can do. Geometry shaders are powerful, and I’m sure I’m not the only one who has seen their potential and been excited.
    - I worked pretty thoroughly on the fractal billboard shader, because I thought it was really cool. I actually really enjoyed just fooling around with it, because I wanted to see how many I could do, and how orienting and moving them provided different effects, as well as how it affected my framerate. This was one of my favorite parts of the lab. I tested it so much, I had to delete over 700 log files from this lab.
    - I then proceeded to worry about the wireframe shader for a while while working on stuff from other classes. Honestly, I had no reason to worry because it worked almost immediately after I wrote it – I hardly made any errors. It just somehow looked more complicated than it was to me while I was reading the lab doc. I think the main thing with this was that you discussed the other two shaders in class, going over the math and what not, but did not do this at all for the wireframe shader. I’m not saying this was a bad way to go about it (maybe you wanted us to figure it out ourselves), but the contrast definitely affected the PERCEIVED level of difficulty. Again, with the math done in the doc – it was not too difficult to actually do.
    - Then, at the last minute, I managed to start the normal hair shader. This one I found less intimidating because it was thoroughly discussed in class. All I had to do was implement it. This does not mean, however, that I did not run into errors while doing it. Actually, I made a super simple to accidentally do but super hard to debug error of typing out instead of in for a variable. It produced wonky output in which the lines were the wrong color and were flashing visible and not visible. There were no GL errors and the shader compiled and linked fine, which surprised me. This ate a couple of my last minute hours. Once I got it working, however, I found I actually liked the aesthetic of the hairy teapot, and enjoyed the simple functionality of swapping between the different types of hairs in real time.
    - Finally, again due to lack of time, I rushed the something cool part, getting the simplest, quickest cool thing I could think of. I guess I perhaps figured that the coolness of my fractal shader would make up for the lack of coolness in the teapot.
    - I’d write more, but I can’t think of anything else.