Lab 06 Shadows

**Fufillment**

* I have met all the requirements for the lab, and even did some extra this time involving drawing a frustum

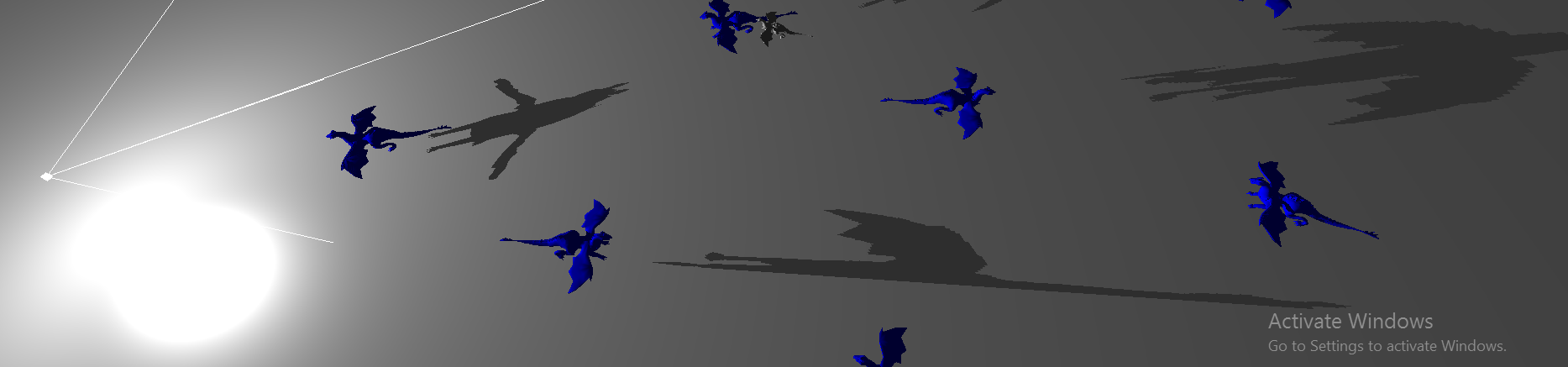
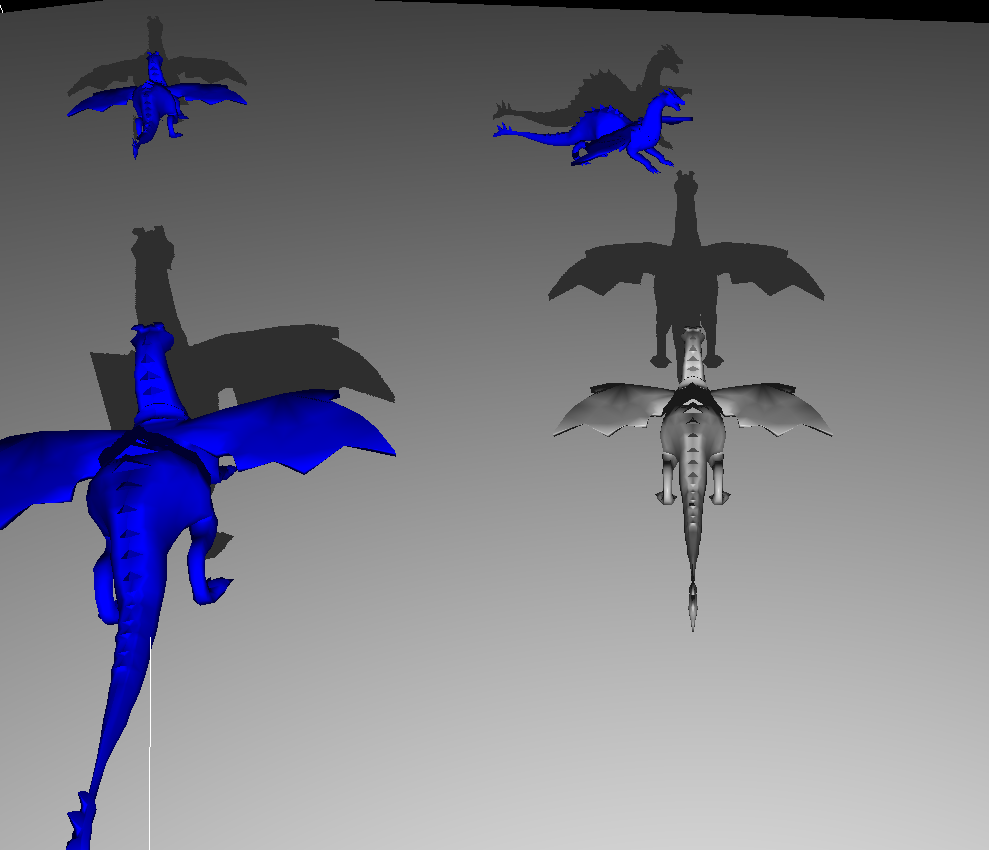
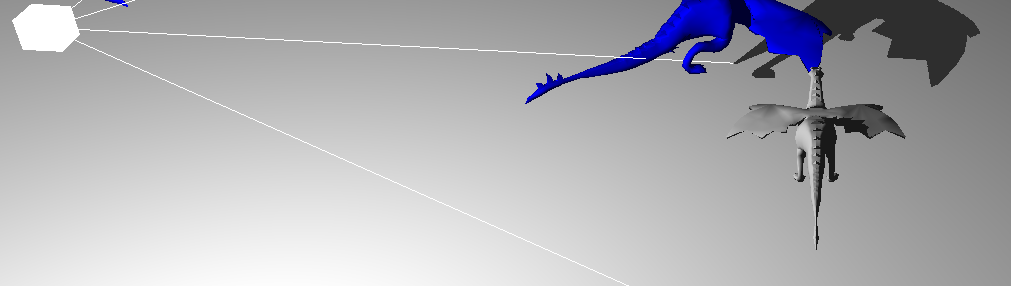
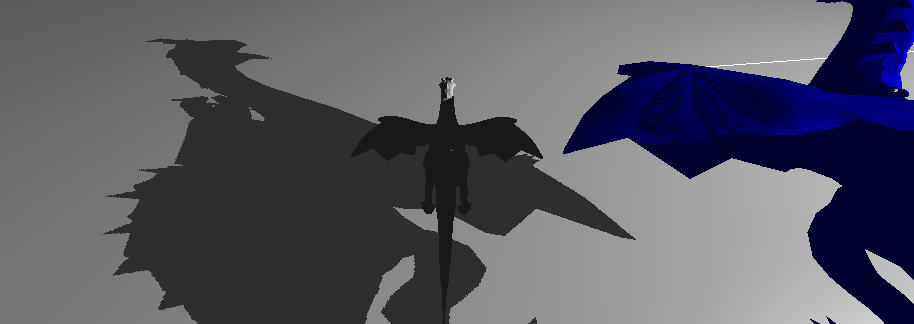
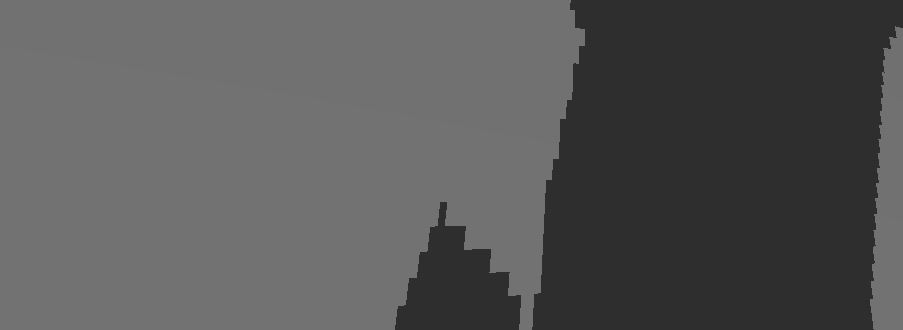
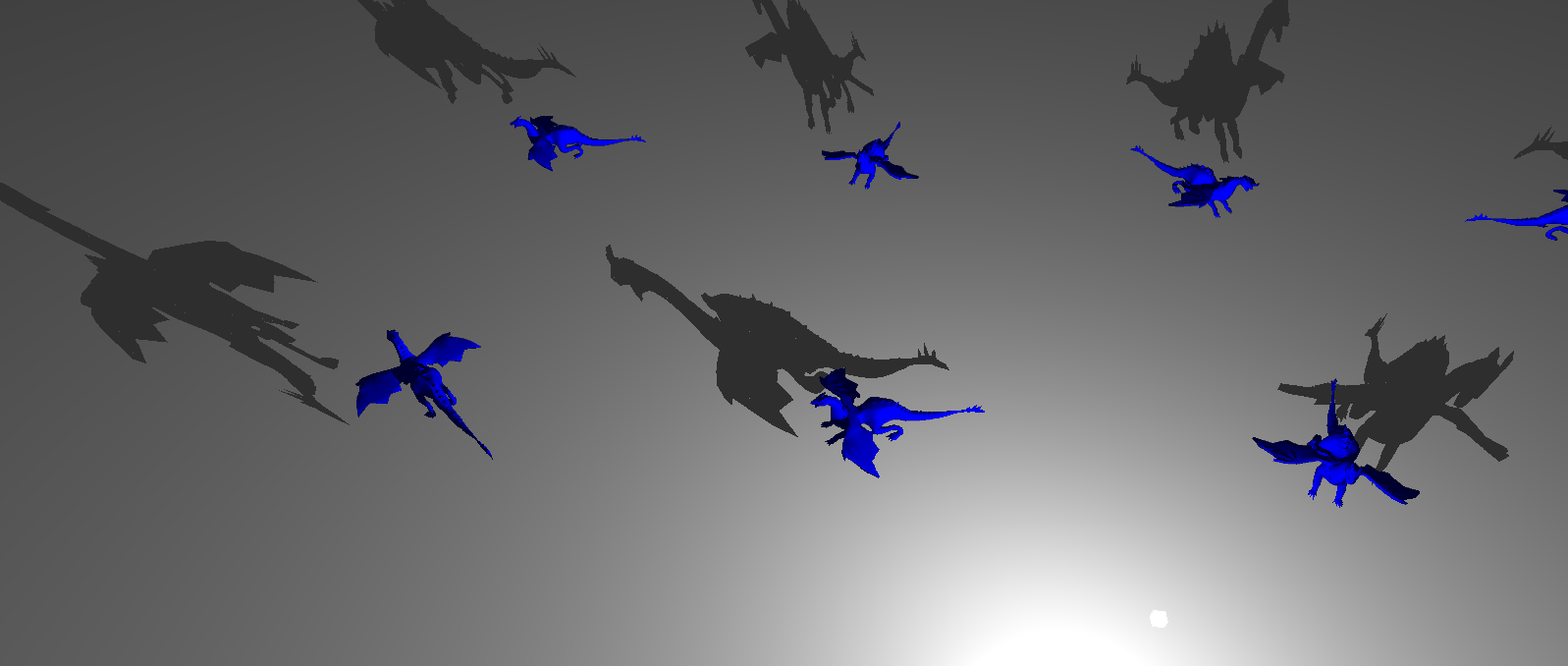
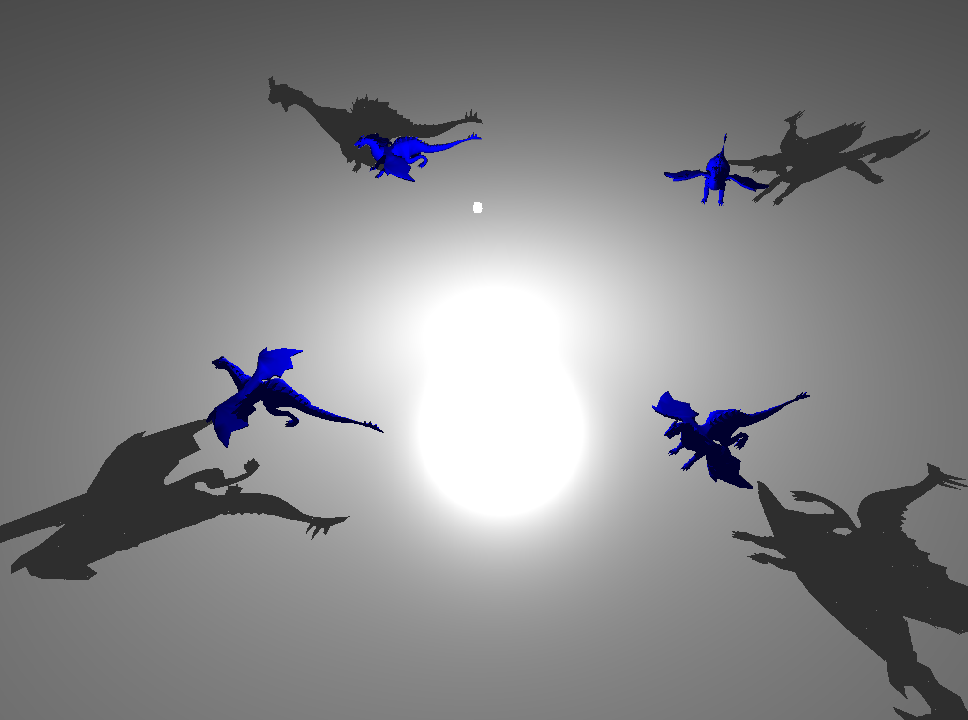
**Execution**

* 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 5 will toggle both linear and blur for basic shadows
* Pressing 3 will toggle between blur and no blur for basic shadows
* Pressing 4 will toggle between linear and nearest for basic shadows
* Pressing 6 will toggle which objects are drawn –the volumetric objects or the basic objects

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

* **Basic Shadows**
  + The first screenshot shows basic shadow mapping – the white lines are the borders of the frustum used to view the objects from the light’s point of view. You can see that the dragons within the frustum have shadows
  + 
  + This screenshot shows different dragons (all within the light frustum) being lit and having shadows. Notice how all the shadows are correct – the heads and tails and spikes are all in positions that make sense and are logical
  + 
  + The next screenshot shows how objects can cast shadows on themselves – notice how the gray dargon has his right wing shadowed by his back because the light is to the left.
  + 
  + This next screenshot shows the recommended simple blur done by averaging the adjacent shadow values. As you can see, there are multiple different levels of shadow near the edge of the shadow.
  + 
  + For reference, a screenshot showing the shadow without the blur clearly is presented here. Compare it to the above blurred shadow to very clearly see the difference.
  + 
  + This next screenshot shows the gray dargon in the shadow of the blue dargon. This shows that objects can cast shadows on multiple different objects, as well as that the compound shadow formed by both of them looks correct (see two heads to the left)
  + 
  + This next screenshot shows a very, very, very, very close close up of the basic shadow with no blur – it is used to show how the edges are not pixel perfect. Compare to the next screenshot for reference.
  + 
  + This next screenshot shows the same angle, the same object, the same light position, et cetra, lit using volumetric shadows rather than the basic shadow depth method used previously. Note how the shadow is pixel perfect! Even though we are very, very, very close to the plane, the shadow is not pixelated.
  + 
  + This next screenshot shows a wider-further-zoomed-out view of volumetric shadows. Note how they look very nice, and, though it has a terrible effect on my framerate, the shader can be used on multiple objects.
  + 
  + This screenshot shows that volumetric shadows work even when the light is moved through mouse input. Isn’t it cool?
  + 

**Post-Mortem**

* + This lab was by far the most difficult so far. I know that’s redundant – but this lab deserves that redundancy. It was very time consuming, extremely stressful, and I’m not sure it was worth all the work that went into it.
  + I ran into many issues with this, as I said before – I had tons of different “versions” of horribly incorrect things that happened. I’ll try to briefly summarize them here in hopes that they somehow help someone fix their problems in the future
    - Oh! Before I get into it – I know I’m about to list a bunch of errors whose solutions were listed in the addendum for the lab – I fixed most of these before that was posted, so these were difficult things I spent much time finding and stressed a lot over.
      * First things first, I messed up the matrices for the quad. It took me a little bit to realize that I just wanted to use the identity matrix to transform a quad already in screen space.
      * I improperly blit the framebuffer – it was difficult to find this because the output was just strange looking and it didn’t scream “hey the issue is this” – I had to slowly but steadily learn one thing about it at a time until I was finally looking in the right spot – turns out I just passed in the wrong width to my method – it’s always something simple, isn’t it?
      * The provided code for the fragment shader was incorrect – and I stated this in class. It was an easy fix, just some multiplication/division, however, the code was presented in a way that encouraged students to just accept that it was correct so this made it hard to debug because, as a student, you’ll sooner assume the mistake was in your code than the code provided to you. I STRONGLY recommend either providing fixed code to future classes OR adding a comment that indicates there is more to that. Perhaps something like “// is this correct?” or “// google this” or “// check this”
      * But really, the biggest thing of all was I didn’t feel prepared for this lab as a student, and I’m sure I’m not the only one who feels this way. I know you talked about it in class a little, and definitely covered the general concepts, but there were so many moving parts, it asked for so many engine features/workarounds that weren’t present in our engines prior to this lab, had so little time, was so large and different, that it made what could have been a fun lab a stressful, time consuming and ultimately extremely demotivating lab that showed us how little we know rather than what we can learn. I would STRONGLY recommend that if you decide to keep this lab in the future that you move it to the end of the course and cover a lot of the prerequisite concepts in much greater detail before assigning it.
  + At this point, I’m absolutely disgusted with most of my implementations for this lab. I feel like I took your code, put it into methods and did my best to call those methods in the right spots. I really feel like I’ve uglified my render engine to force it to work with this lab –and it does work but it bugs me. I know there are better solutions than manually drawing the specific objects I need to – but I was in such a rush that I did not have the time nor the mental capacity. (This sprint is by far the most difficult of all of my time at Neumont right now – I’ve never been so tired of programming. I used to do it for fun – now I just can’t wait for break so I can do anything but code for two weeks. I’m drained.) I know the code I’ve written is not only hideously ugly but also horribly inefficient – and I can see it in my game engine now. I’ve never had only sixteen FPS with so few objects on the screen. I know this is sort of intense, but I also know it should not be that intense. I would rather have had more time and done the lab correctly than to have just forced it to work in this short time and had this laggy thing on my screen
  + Overall, I think I’ve had an epiphany that I think may help you with your constant decisions about the tradeoff between giving us too little and giving us too much. I think that you need to give less code and more instruction. Teaching us about how the stencil buffer works at length, doing a demo that shows something other than the lab, talking about framebuffers and showing us their various applications – these are all things (and there are plenty more of them) that are rewarding to students and encourage learning rather than encourage forcing something to work at the last minute. Compare your course to let’s say Linear Algebra – in Linear Algebra the teacher goes over general concepts, shows plenty of examples, and then gives us problems to solve to practice. What you do is you sometimes briefly cover a concept, you talk about a problem you want us to solve, and then you give us part of the solution. I’m not saying you’re doing it wrong, but I think the preceding method would result in more students understanding concepts better and producing superior output than the method you are currently using. I know we’re kind of the guinea pigs for this course because you’re trying things a different way, and this is my serious, honest feedback about how the course is going. I think that rather than give us portions of solved problems and tell is to finish it, you should go over concepts, show them implemented in different ways and then let us loose to work our magic. Rather than saying “here’s the code for this piece figure out the rest” say “let’s talk about multi-pass rendering” then show multi-pass being used effectively on a shader that has literally nothing to do with volumetric shadows and explain as much as you can. Then do it again on a shader that has nothing to do with the first demo, or volumetric shadows. Then briefly talk about volumetric shadows and give the volumetric shadows assignment. I hope that I’m making sense, and that what I’m saying is reasonable, because I really do think that this would be an improvement.