A3 Lights

By Ben Cagan (bcagan)

I built upon my Vulkan system, parser, and scene graph structures from A1 and A2 in order to add support for new light objects. In particular, I created a global set of lights independent of the scene graph, taking advantage of my previously obtained local to world transforms for triangle nodes in order to implement lighting calculations in shader, with little need to rework my Vulkan code structures. From this, I was able to implement lighting in my shaders, and as a result enabling direct lighting for both Lambertian and PBR materials. The greatest challenge was implementing shadows. Given I already implemented subpasses for my HDR rendering, it was not a challenge to add additional subpasses to render a shadow map for each light, however upon doing so, I found that this method was actually not usable, as subpasses can only reference previous subpasses results from the same fragment of each corresponding attachment. Furthermore, this meant that the framebuffer had to be the same for all subpasses, meaning that shadow map resolution could not be varied.

As a result, shadow map passes were instead done in their own unique render passes, which ended up not being that much more complex to implement in my code compared to subpasses. What was a much more significant challenge was debugging the shadow map code itself. I had many bugs that I had to overcome, for instance, due to an incorrect image format choice, my shadow maps were not able to originally store depths in detail enough to distinguish between objects in my test scene. Realizing I could use the depth image viewer directly allowed me to skip the fragment shader in my shadow map passes, and resolve this bug. Similarly, debugging reading from the shadow map itself poses its own challenges. Using Nsight, I was able to render the scene in terms of the differences between the sampled shadow map depth, and the real fragment depth, enabling me to spot where shadow map sampling bugs were. In the end, I was able to fix mistakes in how I was getting the projected shadow map texture coordiantes from the rendered fragment, and was able to assess that my shadows are properly rendering. However, in the end, I was not able to make these shadows actually visible in the final render pass. In investigating in Nsight, as will be scene in my code explanation, shadows are being calculated and rendered properly, however due to issues with my image format choice, this detail is being lost by the final render pass. I have yet to debug exactly why this is happening, however as I note in the “My Lit Scene” section, I believe this is a more significant issue than just with shadows, and that the colors being rendered to the window are in general not accurate. Im also not convinced that my PBR lighting is perfectly accurate, though it does in general behave somewhat like I’d expect PBR materials to behave.  
My Lit Scene

For my scene, I used the Arerial Grass Rock texture from polyhaven by

Rob Tuytel <https://polyhaven.com/a/aerial_grass_rock>  
 EMBED video here

My intent with this scene was to light a scene entirely with point lights and spot-lights, while also showing a variety of behaviors from different color lights interacting with each other. Moreover, I wanted to use a large number of floating objects above terrain in order to highlight shadows created by the various spot lights. As a result, I modelled a mesh of a valley between two mountains in blender, using the above texture accordingly, and placed a large number of cubes above this valley, as well as numerous colored lights in order to add depth to the scene, and light it accordingly. While I think the general mood that I wanted to create was done so successfully, I am not convinced that I am rendering the spot lights properly in the scene, and in particular, the lack of visible shadows due to the aforementioned problems means that the desired depth created by shadows projected onto the mountains from the cube is lacking in the scene. There are 15 light sources in total in the scene. Running the scene in real time also highlights that there are bugs with my culling acting more often than it should, leading to the mountain, which is a quite large mesh, not being visible unless you move further away from it, something that I wish I had been able to address before A3.