Project Overview:

We are going to be making an extension library for Animal3D that wraps and interfaces a variety of Vulkan functionality with the Animal3D framework. It will include simple rendering tasks such as rendering scene composition images to the screen by using GLSL shaders through the Vulkan rendering pipeline, texturing objects, and performing basic lighting algorithms. We would also like to tackle more advanced Vulkan features such as multithreading, multi-GPU usage, and computational tasks. We aren't really sure which one is more valuable from a developmental perspective since all of them are new to us, and all of them are useful in the industry, but for whichever one we pick, we are going to wrap it into an easily-usable format.

Here we should each describe why we're interested in this sort of project from a more personal POV:

Cam:

I'm going into the AAA side of the industry, so the chances that I'll have to write a rendering engine from the ground up are slim; however, I would have to write libraries and interfaces for new APIs and hardware that work with the existing engines. This is a perfect opportunity to try that for the first time. I've also been interested in Vulkan since I first heard about it in Graphics 2, and now we finally have a chance to learn how it works and how to use it.

Ben:

Since we started learning more about graphics programming, I've been curious about the differences in both performance and functionality for different APIs. For most of our time at Champlain, we have been using GLSL for graphics projects, and I would like to expand my knowledge in graphics to other APIs that may be used in a professional environment. This project will also allow me to gain experience in adding and changing major functionality to an already existing codebase, since we will be using Animal 3D for our framework.

Questions:

1. How do we create a new project and link it to Animal?
2. Could we get a more low-level tour of Animal?
    a. E.g callbacks, how the main project uses the OpenGL project to render, updating
3. Are we allowed to make changes to other libraries in Animal, like A3Math?
4. We haven't done any technical work with compute shaders in any form, so could we possibly have a class session to check those out?
    a. How do they fit into the pipeline?
    b. How do we send data to a compute shader?
5. Are we allowed to add new libraries, for things like better in-engine UI?
    a. Could we possibly package up new libraries as dependencies of our Vulkan extension?

    b. Can we add VkBootstrap, which basically wraps up a lot of the basic info structures and one-time setup code?
6. Which of the above listed “Advanced” features should we try to wrap and implement with Animal?
    a. Cam is most interested in multithreading, but an overarching “job” system can help schedule/execute compute shaders, GPU threads, and CPU threads.