CPD Plan Draft #4

Harris C. McRae, 47083573

Topic

The chosen topic is the **Vulkan Graphics Library & GLSL (OpenGL Shader Language)**.

Over the years I've been passionate in lower-level development of game engines. To this extent I've made multiple iterations of 2D engines with growing complexity, and recently branched out into experimenting with 3D. While I can handle most inner workings of the engines so far including collisions & rudimentary physics systems, event management, and so on, to date I haven't managed graphics at an API level.

To achieve this I want to seek to learn the Vulkan graphics library as well as GLSL for render effects. While Vulkan is a complex system and cannot be mastered in such a timeframe, it will open the gate for me to experiment further. At the same time, I will be getting more used to using C++; right now, I can understand the systems behind C++, but cannot easily apply it at a higher level. Overall, this project will develop my skills in fields that will prove useful in both my career and personal hobbies.

I intend to create a simple environment renderer using C++ and the Vulkan API. Ideally, a user would be able to manipulate basic 3D shapes and view them; if this cannot be achieved, preset shapes should be able to be viewed. Multiple premade shader effects will be included and can be switched between when viewing to create different effects.

Areas of Improvement

- **Vulkan & GLSL**

Low-level computer graphics API / languages used in a variety of systems.
Bloom's Taxonomy -
Current: Remember
Target: Apply

- **Low-level API & conventions**

By learning a complex API such as Vulkan, I will further familiarize myself with navigating and understanding documentation for complex systems.
Bloom's Taxonomy -
Current: Understand
Target: Analyze

- **C++ programming**

Generally considered one of the more complicated, widely used languages in lower level systems. Confidence with concepts used in C++ can be transferred easily to similar languages i.e. Rust.
Bloom's Taxonomy -
Current: Understand
Target: Analyze

Project Deliverables

Main Program

A 3D rendering program capable of displaying complex pre-determined models which can be minimally interacted with (rotate and pan the model). The program will have multiple models and multiple graphical shader effects that can be swapped between during runtime by the user.

Logbook / notes

A running notebook of findings on Vulkan as I go, as well as notes on the structure of the main program which acts as a point of reference for future modifications.

Milestones

The milestones apply to the main program in particular, and should be considered to be offramps IF further progress is not possible.

Milestone 1: Basic Program

A version of the main program that does not display advanced graphical capabilities, such as shaders and complex environments. This version will still demonstrate user interface and input capabilities (rotating & panning models, swapping models) with basic, textured models.

Logbook & documentation is consistent with the features demonstrated in the program so that further development may be possible.

Milestone 2: Advanced

Basic shaders are now present and interchangeable similarly to the models. There are more complex features in the models themselves, such as translucency and other techniques. Documentation is up-to-date.

Milestone 3: Ideal

Shaders are now capable of more complex visual effects. Models may include more complex parts, such as movement of limited aspects and animated textures. Documentation is up-to-date.

Milestone 4: Further development

Models can now be modified and saved to memory by the user. Some pre-made models do exist, but there is now a method to create new ones. Tools to edit models should not be extremely complex but can create and manipulate primitives, apply textures, and load in new shaders. Documentation is up-to-date, and a guide on using the editor exists.

Learning Outcomes

L01 - Program design & documentation

Further developed ability to create complex programs, including using appropriate, efficient data structures and algorithms, structuring a program so to allow ease of modification, and user interface design.
Should be able to analyze concepts to determine fitness and modify to meet own needs.

L02 - 3D Computer Graphics and related mathematics

Able to utilize Vulkan API features, including creating and managing graphical window contexts, creating and rendering shapes, and an understanding of the graphical rendering pipeline,

including an understanding of matrix operations and other mathematical concepts for computer graphics.

Should be able to understand and apply these concepts in new situations.

L03 - Low-level programming & C++

A greater level of understanding of features present in lower level programming languages, such as pointer & memory management, garbage collection, and manipulation of data structures. Additionally, a level of competence with C++, understanding the syntax and functions of the language.

Measures of Success

All of these measures satisfy the learning outcomes to varying degrees. Improvement should be noticed in all 3 areas.

MS01 - Minimal Success

Demonstrates a basic understanding of the Vulkan API & can create window contexts and display shapes. No application of GLSL shaders. Competent in C++ programming and program design. Achieves Milestone 1.

MS02 - Moderate Success

Demonstrates an understanding of the Vulkan API & can create and manage window contexts and display textured shapes and models. Basic understanding of GLSL shaders, with primitive shader effects present. Competent in C++ programming and program design. Achieves Milestone 2.

MS03 - Good Success

Demonstrates an understanding of the Vulkan API & can create and manage window contexts, display textured shapes and models, and apply mathematical operations more creatively to produce animated aspects of models. Good understanding of GLSL shaders with more complex shader effects present. Competent in C++ programming and program design. Achieves Milestone 3.

MS04 - Great Success

Demonstrates a good understanding of the Vulkan API and, as well as earlier mentioned feats, can dynamically create models during runtime. User interface is complex enough that editing models is possible. Good understanding of GLSL shaders with more complex shader effects

present. Can create new techniques in C++ to facilitate user modification behavior. Achieves Milestone 4.

Required Resources

The project will not require anything physical outside of a personal computer.

Development will be done with C++, using the Vulkan API and GLSL (OpenGL Shader Language) as libraries. Additional libraries may be included for mathematical operations and other functionality but a focus of learning will be on the two features mentioned above.

The main resources for learning are:

- Vulkan
  - Vulkan Official Learning resources
  - Vulkan Documentation
- GLSL
  - The Book of Shaders
  - Shadertoy BETA
- C++
  - C++ Reference

Timeline

Note that deliverables from week 3 onwards can count as an off-ramp if progress is delayed or otherwise prevented. These will demonstrate reaching the learning outcomes, albeit at a reduced level.

Week 1

Tasks:
- Begin familiarization with the Vulkan API through online tutorials and documentation.

Goals:
- Tester program demonstrates simple shape rendering.

Deliverables:
- None

Week 2

Tasks:
- Further learning of Vulkan API, such as window management, more complex rendering techniques, et cetera.

- Start developing the main program, introducing basics of Vulkan to create a render window.

Goals:

- Main program should open to a blank window

Deliverables:

- None

Week 3

Tasks:

- Learn GLSL scripting and integrate it into the test program.
- Integrate basic environment rendering & user input into the main program.

Goals:

- Main program can render panoramas
- Tester programs can demonstrate shader effects.

Deliverables:

- MS01 should be met.

Week 4

Tasks:

- Complex shader creation
- Implement shader swapping to main program
- Design more basic environments, and store them to be loaded into the program.

Goals:

- Main program can hot-swap shaders and environments.
- Complex shaders should be tested.

Deliverables:

- MS02 should be met.

Week 5

Tasks:

- Implement complex shaders into main program for hot-swapping
- Design more environments.
- OPTIONAL: Environment editing mode for main program should start development.

Goals:

- Main program can hot-swap between multiple environments and shaders.

Deliverables:

- MS03 should be met.

Week 6

Tasks:
- Polish off program, prepare for presentation.
- OPTIONAL: Environment editing mode should be further developed and completed.

Goals:
- All outstanding work completed.
- OPTIONAL: Environments can be edited and saved through the program itself so that they can be loaded again.

Deliverables:
- OPTIONAL: MS04