Final Year Project Report

Zakariya Oulhadj

OUL20475392
oulhadjz@roehampton.ac.uk

January 17, 2023

Department of Arts and Digital Industries
University of Roehampton
London, United Kingdom

Submitted in partial fulfilment of the requirements for the degree of
Bachelors of Science in Computer Science

Supervisor Dr. Charles Clarke

I dedicate this report to my family and friends

Decleration

I hereby certify that this report constitutes my own work, that where the language of others is used, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions, or writings of others. I declare that this report describes the original work that has not been previously presented for the award of any other degree or any other institution.

Contents

1 Introduction

This is some example text.

1.1 Purpose of the project

1.2 Goals/Requirements

- Lightweight model viewer - No need to install heavy applications such as Blender/Unity/Unreal - Easy to use - No technical knowledge required

2 Literature or technology review

2.1 Version Control System

- Git (https://git-scm.com/) - Repository hosted at https://github.com/ZOulhadj/vmve/ - Branches - Main (Major releases) - Develop (Active development)

2.2 Task Management

- GitHub Issues - GitHub Kanban

2.3 Build System

- CMake (https://cmake.org/) - Using CMake version 3.23

2.4 Programming Language

2.5 Debugging

Throughout this project, debugging will be used extensively to fix crashes, bugs and generally ensuring that the application runs as expected. The primary tool that will be used for debugging is Visual Studio 2022 [1]. This is an IDE (Integrated Development Enviroment) that includes various tools such as debugging, performance analysis etc.

- RenderDoc (https://renderdoc.org/) - Render pipeline and frame analysis - AMD Radeon GPU Profile (https://gpuopen.com/rgp/) - Low-level GPU hardware data

2.6 Compiler

Developing a program that runs directly on the hardware requires a compiler. This is a program that parses source code and generates CPU instructions that the computer will be able to understand and therefore, execute. MSVC (Microsoft Visual Compiler) also known as CL will be the compiler of choice. This is a compiler built into the Microsoft Visual Studio IDE.

2.7 Additional Tools

Some additional tools will be used that will further improve the ease of development. Visual Studio Assist X [2] is one such tool. This is a Visual Studio extension that provides many useful features that the based IDE does not provide such as reliable symbol renaming, file symbol outling, quick file searching and much more.

3 Design

3.1 Architecture

VMVE will be a combination of two projects. The "Engine" project also known as the core of VMVE will contain the fundemental implementation details. This includes the window, renderer, ui and other technical details. This project will be distributed as a library file (.lib) that other projects can import for specific use cases.

The "VMVE" project will include the "Engine" project by importing the .lib file.

There is another key reason as to why, I will isloate the core of VMVE into its own project instead of combinding it into one program. The reason being is have the core be a library that can be used in not just VMVE but also other projects in the future.

- Why did I choose C++ over other languages? - Provides various helpful features such as function overloading, templates and basic STL containers - Low-level access to memory - Fast performance - Cross platform - - Reason for choosing 64bit program. - Deferred rendering pipeline - Functional over object oriented - POD (Plain old data types) - Less boilerplate code (getters and setters) - Easier access to member variables - Why am I going to be using Vulkan instead of OpenGL? - Finer control of rendering pipeline - Aiming to learn low-level GPU architecure - Using Vulkan means that we avoid OpenGLs state machine architecture. - Reduced driver overhead - Allows

for finer control of multithreading - - Editor UI design and the reason for it for this particular application? - UI allows for interaction with underlying system during runtime. - Viewport style editor meaning that all controls can be positioned around the viewport and the main rendering occurs at the center of the screen.

3.2 Programming Style

- Functions names like so "ExampleFunctionName()" - Brackets are like so

4 Implementation

- precompiled header for reduced compilation time -

4.1 Window System

- Cross platform library

4.2 Rendering System

- Renderer context - Combine Vulkan resource allocating objects into single object for increased clarity - Frames in flight - Double and Triple Buffering - Buffers - Single large buffer rather than multiple buffers for each frames - Deferred rendering framebuffers/images - 64 bit world positions - Shader resources - SPV binary format - Pipelines - Pipeline cache - Pipeline derivitive - Delta Time - Matrix projection transformation (model to projection space)

$$MVP = Projection * View * Model * Pos \tag{1}$$

The lighting model that is implemented in Blinn-Phong which derives from the original Phong model developed by By Bui Tuong Phong in a paper published in 1975. Purpose of lighting model is to calculate approximations of lighting based on the real world. [5]

Ambient is a constant value that is used instead of global illumination as it requires more computational resources and more complex algorithms.

Diffuse is the intensity that light has on an objects surface based on the angle between the lights position and surface normal.

Specular is the part of an object that has the highest levels of light intensity.

- Phong lighting - Ambient + Diffuse + Specular + Shadow

$$L = reflect(-L, N) \tag{2}$$

4.3 User Interface

- Using a editor style user interface

4.4 Encryption System

4.5 Logging system

- Buffer based logging system

4.6 Website

- Creation of website for presenting and showcasing features.

4.7 Binaries

- Prebuilt binaries for different operating systems

5 Evaluation

5.1 Programming language

- C++ 20 - Checking for error codes first programming style

5.2 Choice of rendering API

- Vulkan vs DirectX Vulkan is cross platform whereas DirectX is Windows-only. Xbox requires DirectX.

5.3 Time Management

- Have all outstanding GitHub issues been resolved? - How effective was the use of project management tools during development. - Key stages of the project - Initial core systems/graphics development stage - System redesign - Encryption development stage

5.4 Performance

- Compilation times - Benchmarking - Frame times - Startup time - no pipeline cache vs pipeline cache - pipeline derivatives - Shutdown time -

6 Related Work

7 Conclusion

- Have the original requirements been met? -

8 Reflection

- Learn a lot in terms of solving problems - Patience - Could have planned better by designing systems before implementing them. - Project devlogs hosted on YouTube https://www.youtube.com/@ZOulhadj

9 Future Work

- To add more features - Easy to use API for other projects to use renderer. - Implementation of DirectX for more platform support May provide increased performance on Windows since Microsoft have insider knowledge on performance requiremens and overall system design. - Cross platform support

10 Glossary

Glossary	Description
Vulkan	API used for GPU communication
C++	Low Level programming language
CL	Microsoft C Compiler and Linker
MSBuild	Microsoft Build Enviroment

11 Abbreviations

Acronym	Abbreviation	Description
CPU	Central Processing Unit	The brains of the computer
GPU	Graphics Processing Unit	The graphics processing of computer
RAM	Random Access Memory	The computers main memory
WSI	Window System Integration	Vulkans Window System Integration
AES	Advanced Encryption Standard	Method used for encryption
GLSL	OpenGL Shader Language	GPU Programming Language
LIB	Static Library	Library imported by client applications
DLL	Dynamic Link Library	Library that is linked at runtime
PCH	Precompiled Header	A header that is compiled at compilation
CLI	Command Line Interface	The command line
VMVE	Vulkan Model Viewport and Exporter	Custom model format
API	Application Programmable Interface	A way of interacting with libs
EXE	A Windows executable format	A windows runnable application
VMA	Vulkan Memory Allocator	AMDs custom memory allocator implementation
IDE	Integrated Development Enviroment	Development Enviroment

12 Appendices

Code snippets, images and other resources

13 References

[1] Microsoft Visual Studio 2022 (https://visualstudio.microsoft.com/)

[2] Visual Studio Assist X (https://www.wholetomato.com/)

[3] C++ Core Guidelines

[4] https://registry.khronos.org/vulkan/specs/1.3/html/vkspec.html

[5] https://en.wikipedia.org/wiki/Blinn