VMVE Project Report

Zakariya Oulhadj

OUL20475392
oulhadjz@roehampton.ac.uk

January 31, 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
Second marker Alex Collins

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.

January 31, 2023

Date

z.oulhadj

Signature

Contents

1 Introduction

This report will provide a detailed insight into all aspects of the project including requirements, design, implementation and future work.

The project being developed is called VMVE which stands for Vulkan Model Viewport and Exporter.

3D Computer Graphics

- Context of application

1.1 Purpose of the project

- State the problem being addressed and why it is important to address it - large applications consist of many requirements such as powerful hardware, applications

- Key stakeholders

1.2 Goals and Requirements

In order to address these issues in this domain, a set of requirements must be defined that aims to achieve the desired goal of this project.

1.3 Aims and Objectives

- Refer to milestone 2 document.

1.4 Justification

- Why this project is suitable for BSc.

Lightweight meaning application should be highly efficient in regards to rendering and memory usage.

Ease of use so that users with no prior experience

Useful Another vital requirement is that it should be useful

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

2 Technology review

This project made use of various technologies at different stages throughout the development process and was a key aspect in helping achieve the final goal.

Technologies are categorised into two areas based on the impact they have on the project such as direct and indirect influence. A technology that has direct impact means that it assisted in some way the implementation of the project. Whereas, indirect impact are technologies that are used in some areas not directly responsible in the projects outcome.

2.1 Version Control System

A version control system (VCS) is a tool used for backing up and/or collaborating with developers on a project. The use of a VCS was an obvious choice as this would provide a platform on which the project source code could be hosted. This gives me the peace of mind knowing that if for some reason a local copy of the project is lost or corrupted then another copy is safely hosted on the servers managed by the VCS.

Another key feature of a VCS is project management. These types of systems provide various tools that greatly benefit developers. One such feature is known as a "commit" which records any changes made to a particular repository at that moment in time. Developers use commits to view changes that occur at each stage but also, provides means of reverting to previous states of particular sections, files or even an entire repository. Due to the length and complexity of this project, tools such as this provided by VCS are invaluable throughout the development process.

The specific version control system that was chosen was Git [1]. This is the most popular free and open-source VCS and is highly recommended.

Git can be used in several different ways such as installing Git onto a server manually and interacting with Git through that server. Another way is by using existing platforms that are built on top of Git. A few examples include GitHub, GitLab and Bitbucket. By far the most popular option is GitHub and is the platform that I am most familiar with.

The VMVE project is hosted on GitHub as a private repository in the following location https://github.com/ZOulhadj/vmve/.

2.2 Task Management

- GitHub Issues - GitHub Kanban

2.3 Hardware

The majority of the development was done on a Thinkpad T14s Gen 3 laptop with the following specifications. - AMD Ryzen 7 Pro 6850U - Integrated GPU - 32GB DDR6 RAM -

2.4 Build System

Developing a program that runs directly on the underlying operating system requires a compiler. This is a program that parses source code and generates assembly instructions that the CPU 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 that comes bundled with the Microsoft Visual Studio IDE.

2.5 Programming Language

The programming language of choice was C++. Due to the nature of application, there were several requirements that had to be met such as a high performing programming language as well as low-level memory access in order to specifically manage how memory is handled within the application.

2.6 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 [2]. This is an IDE (Integrated Development Enviroment) that includes various tools such as debugging, performance analysis etc.

[3]

As mentioned above, the hardware used for development made use of a AMD CPU. Therefore, in order to access insightful performance metrics on a per-frame basis a specific piece of software was required that could access the underyling drivers. The AMD Radeon GPU Profiler [4] is one such application.

2.7 Additional Tools

Some additional tools will be used that will further improve the ease of development. Visual Studio Assist X [5] 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 VCS Architecture

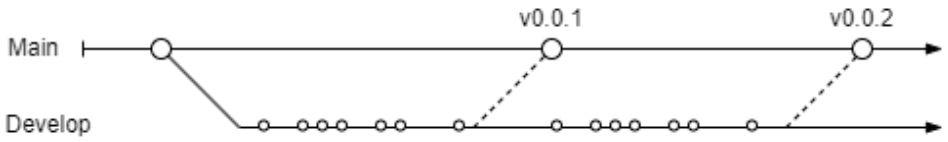

Figure 1: Git branch design

3.2 Project 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 subsystems. 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.

A high-level overview of the project architecture can be seen in figure 2.

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. goes beyond 4GB address space limit in 32bit programs. supports modern hardware

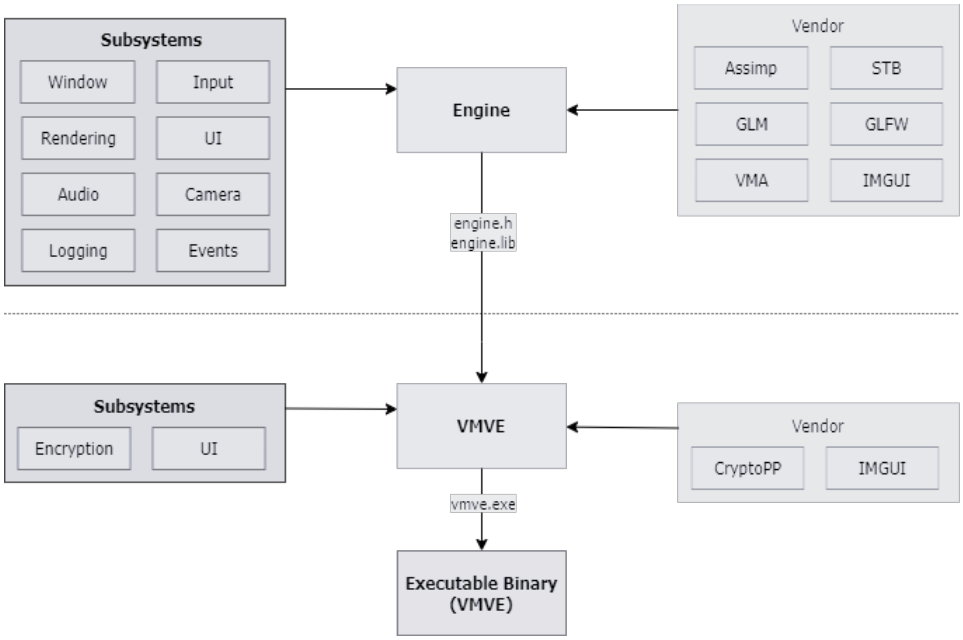

Figure 2: Project Architecture

- Deferred rendering pipeline better lighting performance when there are lots of lights - 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 and have a deeper understanding of low-level GPU architecure - Using Vulkan means that we avoid OpenGLs state machine architecture. can introduce bugs - Reduced driver overhead - Allows for finer control of multithreading - Better performance potential - Error checking can be disabled when shipping application. - 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. - VMVE file format - Header - Data - Encryption - AES (128, 256 bits for key length)

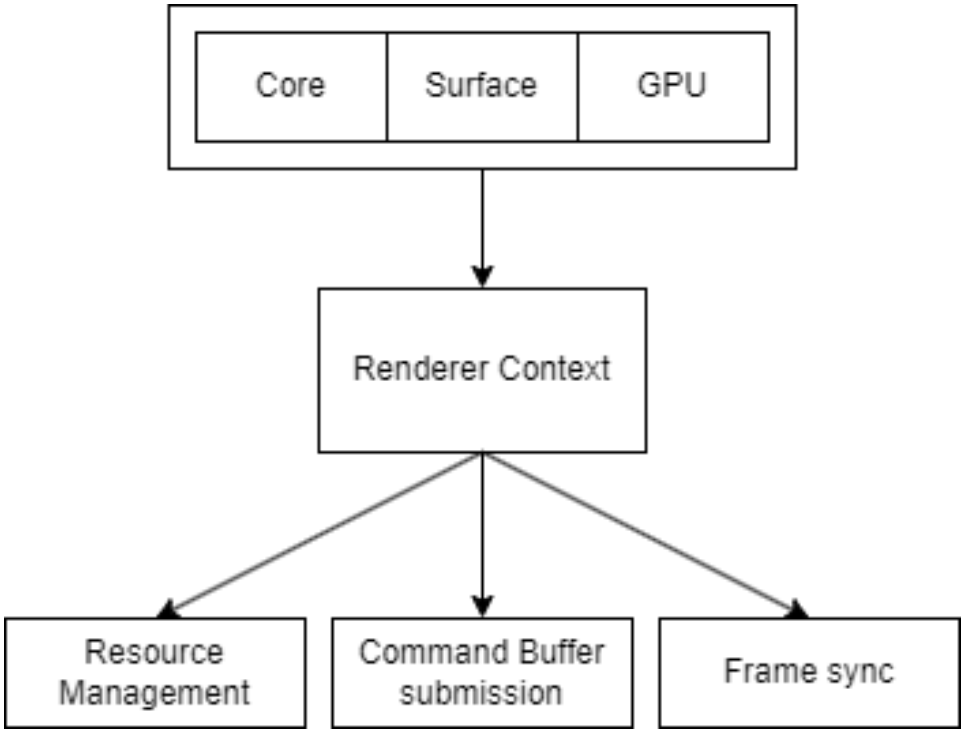

Figure 3: Renderer Architecture

3.3 Renderer Architecture

3.4 Programming Style

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

4 Implementation

- precompiled header for reduced compilation time (could this be - profiled?) This works by combining compiling all header files - once and simply being reused across all compilation units - instead of recompiling the same header file wherever it is used.

4.1 Window System

- Cross platform - window/input callbacks

4.2 Rendering System

This is one of the key systems of the engine and subsequently VMVE.

Linking against actual driver function calls instead of linking to the common Vulkan loader. [6]

- 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

- Quaternion camera - prevents gimbal locking - Matrix projection transformation (model to projection space)

$$MVP = P \times V \times M \times Pos \tag{1}$$

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. [7]

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

$$A = G + P \tag{2}$$

The next step is implementing diffuse lighting. This takes into account the light direction L and the normal N for a surface at a particular pixel. The formula below calculates the intensity at which light reflects off the of an object based on the angle the surface is against a light source.

The dot product of the light direction and surface normal returns a value between the ranges of -1 to 1 depending on how parallel the directions are. If this value is less than 0 then it's clamped as a result of the max function which returns the largest value for the given two parameters.

$$\mathrm{I} = \max(\vec{\mathrm{L}} \cdot \vec{\mathrm{N}}, 0) \tag{3}$$

$$\tag{4}$$

Having calculated the intensity value we can now simply multiply it with the objects surface colour.

$$D = I \times C \tag{5}$$

Figure 4 demonstrates the use of diffuse lighting which shows how each surface of the cube is lit differently based on its angle to the directional light source.

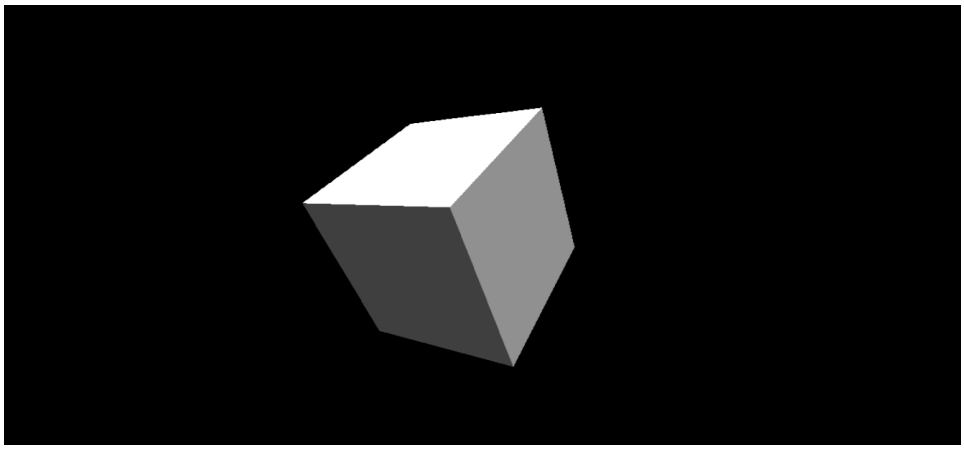

Figure 4: Diffuse Lighting

The final step in the Blinn-Phong lighting model is specular highlighting. This effect adds...

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

When evaluating the performance of the project it's important to understand what aspects are being measured and how exactly they affect the application. There are two main areas where performance can be measured

- Compilation
- Runtime

"Compilation" refers to the process of building the application in an offline setting. The speed at which the project is compiled directly affects the developer. On the other hand, "Runtime" means the performance of the application while it's running and will affect the users.

CPU timing C++ comes with the ¡chrono¿ library which provides std::chrono::high_resolution_clock. This is a CPU timer that can record and measure differences in time using different time units such as nanoseconds, milliseconds, seconds etc.

Visual Studio IDE Performance Profiler

- Benchmarking CPU timing GPU timing

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

6 Related Work

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 here

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 - Networking support allowing for multiple users within the same environment to create virtual enviroments.

In regards to the VCS (version control system) architecture, a potential change that can be done is to add a third branch called “Beta”. The purpose of this branch would be release relatively stable but yet unfinished features to users. This would allow user to use new features and test them before an official version is released. Overall this would reduce the number of bugs in final versions and be beneficial to users who are keen in using recently developed features. Figure 5 shows how the VCS architecture could look like with the addition of a “Beta” branch.

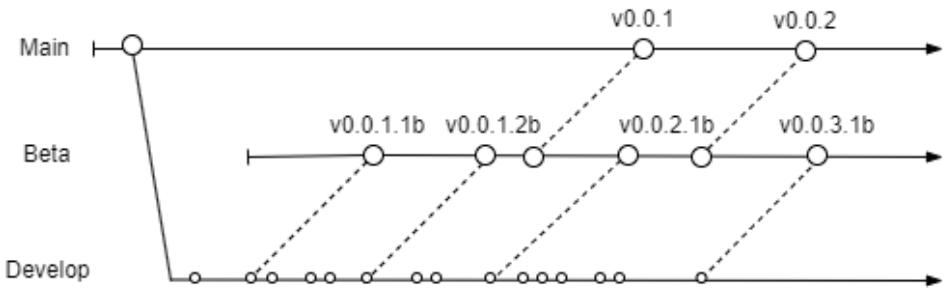

Figure 5: Potential future git branch design

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] "Git version control system," https://git-scm.com/.

[2] "Microsoft visual studio 2022," https://visualstudio.microsoft.com/.

[3] "Renderdoc," https://renderdoc.org/.

[4] "Amd radeon gpu profiler," https://gpuopen.com/rgp/.

[5] "Visual studio assist x," https://www.wholetomato.com/.

[6] "Vulkan loader," https://gpuopen.com/learn/reducing-vulkan-api-call-overhead/.

[7] "Blinn-phong reflection model," https://en.wikipedia.org/wiki/Blinn%E2%80%93Phong_reflection_model.