# Investigation about the graphical engine Vulkan API among other available technologies

Jagoba Pérez

February, 2017

# Contents

**Abstract**

Latest years videogames have obtained a strong influence in the entertainment industry. This new gold mine has produced a new sector where different vendors have created videogames related products - such as gaming specific equipment, merchandising, etc.

Technologies like Virtual Reality (VR), smartphones, consoles have broaden the frontiers of videogames. Today the market remains increasing demanding a better experience with a lower price. Therefore, Information Technology (IT) companies invest money on developing new technologies for taking the best advantages of current hardware.

The aim of the article is to analyze the advantages of the *Vulkan* engine Application Programming Interface (API) and compare it among other available technologies in the market. This article also shows introduces which are the required tools for developing with *Vulkan*. As the engine is very young a benchmark test is proposed for verifying that the Software Development Kit (SDK) is, at least, as good as its ancestor.

# Chapter 1

# History

*Khronos* group is an industrial consortium dedicated to the creation of APIs with open standard and free. This standards should allow the creation and spread of multimedia content on multiple devices. Some of their most relevant projects are: *OpenGL*, a multiplatform API oriented to graphics; *OpenCL*, an API for multiplatform computation; *OpenSL*, an optimized API for 3D and *MIDI* reproduction on integrated devices.

In 2014, this group get together for a kickoff meeting with the *Valve Corporation*, where the creation of an API for the next generation of graphic was proposed. In the *SIGGRAPH* conference that toke place that year - where people and companies related with the computational graphics gathers - professionals of the multimedia sector wishing to contribute to the project were summoned.

For creating this API they were based *Mantle*, a specification created by *AMD*. The source code was relinquised to *Khronos* with the intention of generating a new low level standard similar to *OpenGL*.

The following year, *Khronos* presented *Vulkan* in the *Game Developers Conference 2015*. Until that moment the API was known as *glNext*. One company bounded to *Valve*, called *LunarG*, demonstrated how they were able to develop a driver for *GNU Linux* using the graphical card *Intel Graphics HD 4000* - despite that until that date *Mesa* drivers were not compatible with *OpenGL 4.0*.

It won't take long until have acceptance. In August 2015, *Google* announced that *Vulkan* would be supported for the incoming *Android* Operative System (OS) versions. From the version 24 of *Android* this API can be used. Beginning 2016 the first release version of *Vulkan* was published joined together to the specification.

*Unity Technologies* announced that from the version 5.6 of *Unity*, *Vulkan* would be supported.

In theory, *Vulkan* could be used for be employed with hardware of parallel computing, for controlling billions of Graphical Processing Unit (GPU) cores, in little wearable devices, 3D printers, vehicles, VR and basically anything kit out with a GPU.

Nowadays, the development kit is available for *Windows* and *GNU Linux* systems. There is no plan of supporting *MacOS* but, as is explained in section 2, the community is developing a compatible API.

# Chapter 2

# Features

One of the greatest challenges of *Vulkan* is to offer a high hardware performance, with the lowest consumption possible and reducing the overhead. Nowadays, mobile devices have the chance of running games with a great graphical quality. One method that is employed is to reduce the use of the Central Processing Unit (CPU). This reduction is performed by a reduction in the number of delegated tasks, instead the responsibility of the GPU is increased: use of batch tasks in the GPU, exclusive use of the CPU for rendering and computation.

Reaching to all devices is other of the main objectives. *Vulkan LunarG* implementation is available from *Windows 7* to *Windows 10*, *Tizen*, *GNU Linux* and *Android*. The SDK is can be installed in both *Windows* and *GNU Linux* desktop environments. In *MacOS* is being developed by third parties for achieving the execution of Vulkan applications. *The Brenwill Workshop* is a development company specialized on graphics software. They have proposed to create a middleware between the *OpenGL* and *Metal* API implementations for getting the support of *Vulkan* on *MacOS* and *iOS*. The solution is called *Molten*, and it is part of an available graphic development framework.

*Vulkan* implements several improvements at technical level, which are a good reason for being the best successor of *OpenGL*:

- Native muticore CPUs scaling support.
- Use of intermediate binary format called *SPIR-V*: *Vulkan* generates its own code for generating shaders, so that the drivers do not require the development of a compiler for being able to render them. As the shaders are compiled, a higher range of shaders can be used per scene. The driver just is in charge of optimize and generate the final compiled code.
- Unified management of computing kernels and graphical shaders: both API are mixed that, until now, were separated.
- Oriented to objects, there is no global state.
- States are not tied into a context, they are cached in a buffer.
- Multithreading support.
- More control over memory and synchronization.

There are features that have been proposed, but will be implemented in the future. For instance, providing support to multiple GPUs of different model. The need of using Scalable Link Interface (SLI) for being able to use several graphical cards at the same time.

On the Figure 2.1 the existing differences between the two API can be appreciated more graphically. As you may notice among the features of *Vulkan*, a closest access to the hardware from the applications has been attempted. Thanks to this, the need of a huge error and memory management is removed. Only a intermediate language is employed for using shaders (*SPIR-V*), and the API is the same for developing on both mobile applications and desktop applications.

Knowing all these information could be concluded that *Vulkan* increases the work load that the developers already have. However, things are not as they appear. *Vulkan* can be applied at three different levels:

1. Use directly all these features for having a full control over the engine.
2. Use and share libraries and layers for accelerating the development process.
3. Use already existing and optimized engines over the *Vulkan* API.

The first option is the less frequent since developers have to start from scratch. But it could be a good option for making good benchmarks. *Khronos* expects the second option to be a rich area of innovation among the community and companies. As the libraries can be published as little bundles, they could be shared as Open Source that require improvements and updates. The last option could be the most tentative considering that the hard work has already done by industry giants, developers have only to pay some royalties to the propietaries of the engine; for instance *Unity Engine*.

On the Figure 2.2 can be appreciated how the *Vulkan* architecture is distributed. As explained, the *Vulkan* implementation is placed in a lower level that the game engine. It is located over the hardware so that many of its features can be exploded directly. An interface is set up for implementing other tools and frameworks joined together with *Vulkan* in the future.

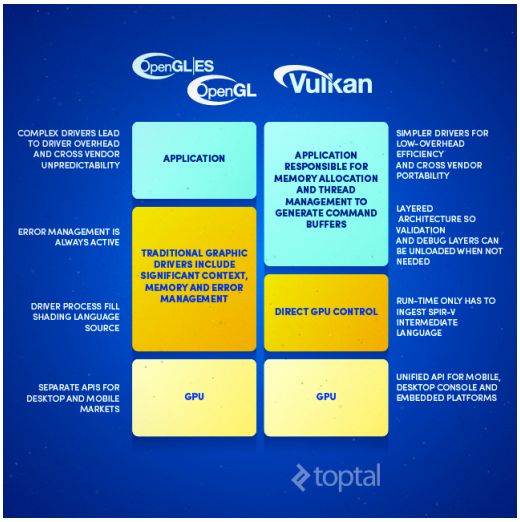


Figure 2.1: Features of *OpenGL* and *Vulkan* Comparison


Figure 2.2: *Vulkan* Ecosystem

# Chapter 3

# Benchmark

As long as *Vulkan* approaches to be an *OpenGL* replacement, it is required to compare if *Vulkan* is able to offer a throughput greater than *OpenGL* - or at least similar. For this reason I have performed a benchmark with two different games. With each game I have played twice a round, one time with *OpenGL* and another time with *Vulkan*. I have played the same scenes with the two implementations of the technologies for avoid having less computing load with one of the technologies. The game screen resolution has been set up *1920x1080*, with the highest graphic quality.

The specifications of the computer I have done the tests with are the following: *Intel Core i7-4790* processor, 16 GB of DDR3 RAM running at 1,600 MHz. and an *AMD Radeon R9 380 Series* graphic card. I have updated the *AMD* drivers to the latest version for having the latest improvements.

I have used the software *PresentMon64* for getting the measurements, which is free for use in the *GitHub* repository GameTechDev/PresentMon at *GitHub*. This application can measure the times that takes to draw the frame, time that takes to render the frame, synchronization time...

During the test I have not notice any problem when playing - that is the game experience has been excellent. Based on the collected data the Table 3.1 has been constructed. As you can see, the average of OpenGL has been greater that *Vulkan* average. However, *Vulkan* has a greater frame stability as the standard deviation is lower than the *OpenGL* frame rate. Both technologies have a greater frame average than the frame rate the players currently demand - about 60 FPS playing with a resolution of *1920x1080*.

It has to be taken into consideration that *Vulkan* drivers have not the same maturity level as *OpenGL*. In the future it is for sure that the *Vulkan* results will be improved. As conclusion of this test, I think that today *Vulkan* could be a good alternative to *OpenGL*, at least from the multimedia point of view.

| Juego | API | Average | Maximum | Minimum | Standard Deviation |
|---|---|---|---|---|---|
| Doom 2016 | OpenGL | 149 | 184 | 7 | 11.169 |
| Doom 2016 | Vulkan | 85 | 194 | 36 | 9.786 |
| Dota 2 | OpenGL | 137 | 200 | 6 | 33.17 |
| Dota 2 | Vulkan | 122 | 200 | 9 | 30.719 |

Table 3.1: Benchmark Results. Measured as Frames Per Second (FPS).

# Chapter 4

# A quick look

If this article has attracted your attention, I am going to propose you a way to take a closer look to the development kit of *Vulkan*. LunarG/VulkanSamples GitHub repository contains several executable examples using the LunarG implementation of *Vulkan*.

In the first place it is required to prepare the tools for developing. Ensure that the graphic card supports *Vulkan*. In the Table 4.1 can be checked if your model of graphic card has drivers that provide support to *Vulkan*. Very few integrated cards of *Intel* cards support it, so make sure you are using a good graphic card. So if you find problems when compiling the source, the problem can be cause for a lack of support.

Preparation of the development environment is different in *Windows* and *GNU Linux*, but generally the ingredients are:

1. CMake: C/C++ compiler.
2. Python 3.
3. Git
4. Glslang: required for compiling compilar OpenGL Shading Language (GLSL) to SPIRV.

Under *Windows* you have to install an Integrated Development Environment (IDE), which in this environment could be *isual Studio* par excellence. Then, download and install the LunarG SDK *Vulkan* implementation. After creating a new C++ project it is required to point out where the SDK is, and add the library dependencies *vulkan-1.lib* and *glfw3.lib*.

On the *GNU Linux* hand, download and install the LunarG SDK *Vulkan* implementation. There is an executable file in the package that eases the installation tasks. It is required to install some libraries in addition to the previous dependencies:

1. libxcb1-dev: XCB library, employed for communicating with the *X-Window* system.
2. xorg-dev: Xorg development files.
3. libglm-dev: GLM library, employed for lineal algebra operations.

| Tarjeta gráfica | *Windows* | *GNU Linux* |
|---|---|---|
| NVIDIA | 368.69+ | 367.27+ |
| AMD | Crimson Edition 16.3+ | Crimson Edition 16.3+ |
| Intel | 15.40.20.4404+ | Mesa 12.0+ |

Table 4.1: *Vulkan* Drivers

4. mesa-dev: MESA development libraries.

After all requirements are install, proceed to clone the repository where the example is located. After the project is downloaded, the *update_external_sources* file has to be executed. This script downloads and compiles all dependencies that require this project (*glslang* and *spriv-tools*). Finally, the project has to be compiled and executed, such as it is indicated in the *README.md* file which can be found in the project.

# Chapter 5

# Games

Still has not passed enough time to implement emphVulkan in a huge variety of games. But there are some companies that have implement the API for develop their engines.

The first company to launch a game supporting this API has been *Oxide Games* with the game *Ashes of Singularity*, a strategy on real time game. It has been the first game to support *DirectX 12* too. It is strange that they have released the game under Windows, having the chance of supporting other platforms.

*Doom*, is one of the first emphAAA games that implemented *Vulkan* a few months after it was put on sale. Normally, *ID Software*, the company that has developed this saga, has typically developed their games with the *OpenGL* API. In this case, they have develop the game using their engine *ID Tech 6* - based on *OpenGL* - and then they have added support to *Vulkan* with excellent results.

*Valve* has implemented *OpenGL* in their engine *Source 2*. So that games as *Dota 2* have been the first titles to offer multiplatform support. *Ark: Survival Evolver* is other game offering multiplatform support with an excellent quality.

Other videogame examples that implement *Vulkan*, in desktop as much in mobile devices, are *Vainglory* (Figure 5.1), *Rust*, *Need for Speed: No Limits*...


Figure 5.1: *Vainglory* running on a *Samsung Galaxy S7 Edge*

# Chapter 6

# Alternatives

Even though I mentioned that there are several alternatives to *Vulkan*, I propose an overlook of some development kits that are widely used by the industry on different systems.

## 6.1 Metal

It is an API that provides a high performance access to the GPU. Allows a low level access to the hardware, which maximizes the graphical quality and the computation power for applications runnng under *iOS*, *macOS* and *tvOS*. *Metal* owns a simplified API, precompiled shaders and multithread support. It is focused for developers that want push to the limit the integrated graphic cards in *Apple* mobile devices.

Released on 2015 as an alternative to *OpenGL*. Seeks a more direct access to the hardware of the devices - the same objective as *Vulkan*. Being closer to the *metal* of the machine the resource consumption and access is reduced. Some of the new features includes a better synchronization between the GPU and the CPU for reducing the amount of data the CPU has to provide to the GPU: the logic behind is that the each time the graphic card needs to draw a scene, the CPU has to send the commands explicitly. By means of the resource sharing, the graphic card can access to the data with no need of intermediary.

## 6.2 OpenGL

Standard specification that defines a multilanguage and multiplatfrom API. The initial release was performed in 1992. Introduced as a solution to the graphical problems existing at early 80s: companies had to outsource programmers for writing specific drivers for each kind of hardware. Many times these kind of solutions were not the most optimal, without considering the financial implications.

During mid-90s, the multimedia industry tried to unify two most relevant technologies: *Microsoft*'s *Direct3D* and *OpenGL*. The objective was to create an industry graphical libraries standard. Because of a lack of funding and supporting the project failed two years after the proposal.

*OpenGL* is available to anyone, but it is not Open Source since the source code is not available. This API is an open specification that describes the interface the programmer uses and expected behavior. The implementation that is Open Source is called *Mesa3D*. It is a good start point for the programmers that want to introduce themselves to the graphical libraries world. Nowadays *OpenGL* does not have a huge relevance in desktop environments, a few companies uses this standard as a first option for developing. This specification is very used for the creation of graphical engines, simulators and mobile devices applications.

## 6.3 DirectX

Made up by a collection of APIs developed for ease the task related with multimedia, videos and - above all - videogames. Consists of different libraries for drawing in 2D, 3D, input device processing, network communications, audio playing and recording...

It is only available in Microsoft's platforms, such as *Windows* or *Xbox*. However, it is being developed an Open Source implementation for *Unix* systems - this project is called *WineHQ*.

The initial release was at ending 1995s. In the past, games were executed directly under *DOS* system, where no specific API was employed for multimedia development. During those years, the market demand of multimedia applications was going greater and greater, and there is no native solutions. The first version of *DirectX* - also called *Direct3D* - was integrated into *Windows 95*, as a substitute of Device Control Interface (DCI) and *WinG Windows 3.1* API.

*Direct3D* is focused in game development. During many years the API implementation has been increased, and many optimizations have been included. When the *Xbox* consoles were put on sale a specific API was created for the consoles - same happened with the mobile system *Windows Phone*. *Microsoft* still supports to *OpenGL* inside their desktop systems. This is because applications like Computer Assisted Design (CAD) and simulators consumes that API. For taking advantage of *OpenGL* features, *Microsoft* has created a half-support between *DirectX* and *OpenGL*. They offer to developers different choices upon implementing their solutions.

# Chapter 7

# Conclusions

*Vulkan* is quite promising API which has two strong points: it is supported by some of the most relevant gaming companies and has an emphasis on the community to take part in the develop of libraries and improvements. This makes the technology available to anyone wanting to put its bit. Furthermore speeds up the development of applications due to the wheel is not reinvented in each new project.

The implementation of *Vulkan* is called *LunarG* and has all documentation and SDK on its website. Remember that the development tools are only available in *Windows* and *GNU Linux* and there is no support for *iOS* and *macOS* systems.

The throughput of *Vulkan* is a bit lower than *OpenGL*, but the stability of the frame rate is better. Companies have to improve their drivers due to this technology is new and requires time to optimize the graphical libraries. Despite these facts, many companies have make their games compatible with this API, which is a remarkable point.

I expect promising future feature to be released soon. Meanwhile people's hardware has to be updated to be compatible with *Vulkan*.

# Glossary

**API** Application Programming Interface. 1–3, 6, 9, 11–13

**CAD** Computer Assisted Design. 12

**CPU** Central Processing Unit. 2, 11

**DCI** Device Control Interface. 12

**FPS** Frames Per Second. 5, 6

**GLSL** OpenGL Shading Language. 7

**GPU** Graphical Processing Unit. 1–3, 11

**IDE** Integrated Development Environment. 7

**IT** Information Technology. 1

**OS** Operative System. 1

**SDK** Software Development Kit. 1, 2, 7, 13

**SLI** Scalable Link Interface. 3

**VR** Virtual Reality. 1

# Bibliography


[1] *A Brief Overview Of Vulkan API*. URL: https://www.toptal.com/api-developers/a-brief-overview-of-vulkan-api (visited on 02/20/2017).

[2] *GitHub - LunarG/VulkanSamples: Vulkan Samples*. URL: https://github.com/LunarG/VulkanSamples (visited on 02/21/2017).

[3] *Home « The Brenwill Workshop Ltd.* URL: http://brenwill.com/ (visited on 02/20/2017).

[4] *List of games with Vulkan support*. en. Page Version ID: 766281997. Feb. 2017. URL: https://en.wikipedia.org/w/index.php?title=List_of_games_with_Vulkan_support&oldid=766281997 (visited on 02/21/2017).

[5] *LunarG*. en. Page Version ID: 740923674. Sept. 2016. URL: https://en.wikipedia.org/w/index.php?title=LunarG&oldid=740923674 (visited on 02/20/2017).

[6] *SIGGRAPH*. en. Page Version ID: 763095929. Feb. 2017. URL: https://en.wikipedia.org/w/index.php?title=SIGGRAPH&oldid=763095929 (visited on 02/20/2017).

[7] *Vainglory Vox Lane - Samsung Galaxy S7 Edge - API Vulkan - YouTube*. URL: https://www.youtube.com/ (visited on 02/21/2017).

[8] *Vulkan*. es. Page Version ID: 96973445. Feb. 2017. URL: https://es.wikipedia.org/w/index.php?title=Vulkan&oldid=96973445 (visited on 02/20/2017).

[9] *Vulkan - Graphics and compute belong together*. URL: https://www.khronos.org/api/vulkan/faq/ (visited on 02/20/2017).

[10] *Vulkan - Industry Forged*. URL: https://www.khronos.org/vulkan/ (visited on 02/20/2017).