DOI 10.52209/1609-1825_2025_1_483 UDC 004.924

Comparison of Graphics Application Public Interfaces DirectX 12 and Vulkan

[1]**DANENOVA Gulmira,** *Cand. of Tech. Sci., Associate Professor, guldan72@mail.ru,*
[1]**KOKKOZ Makhabbat,** *Cand. of Ped. Sci., Associate Professor, makhabbat_k@bk.ru,*
[1]**AKHMETZHANOV Talgat,** *Cand. of Tech. Sci., Associate Professor, akhmetzhanov_t@mail.ru,*
[1]**SAILAU KYZY Zhuldyz,** *PhD, Acting Associate Professor, s_k_zhuldiz@mail.ru,*
[1]***ABILDAEVA Gulnur,** *Master's Degree, Senior Lecturer, g.abildaeva@kstu.kz,*
[1]NPJSC «Abylkas Saginov Karaganda Technical University», N. Nazarbayev Avenue, 56, Karaganda, Kazakhstan,
**corresponding author.*

Abstract. *This study is aimed at studying modern graphical APIs, studying their performance, capabilities, as well as comparing them with previous generation graphical interfaces. The purpose of the study is to study the performance of these interfaces when used under the same conditions. It is necessary to find out which of the interfaces represents not only the best performance, but also the most stable implementation. In the course of the article, the main innovations that are characteristic of both new generation graphical interfaces were first considered. First of all, these are changes in the field of asynchronous calculations, namely changes in the pipeline. The most obvious use of these technologies is in game development, but graphical interfaces are also widely used in science to visualize results. The performance and stability results of both interfaces were obtained by comparing them with the previous generation interface during the study.*

Keywords: *graphics API, DirectX 11, DirectX 12, Vulkan, Nvidia, architecture.*

Introduction

In the modern world of computer graphics, Graphics APIs (Application Programming Interfaces) play a crucial role in software development, ranging from gaming applications to complex computer-aided design (CAD) systems. Each year, the demand for higher graphics quality increases, requiring developers not only to implement new technologies but also to optimize performance to deliver high-quality graphics without the need for excessively powerful systems.

The growing complexity of graphical tasks and the diversity of hardware solutions pose additional challenges for developers, making the use of a standard API especially important. A graphics API abstracts the specific features of different graphics devices, allowing developers to work with a unified interface where the responsibility for driver development lies with hardware manufacturers. This simplifies software development and enhances its performance.

The aim of this study is to conduct a comparative analysis of the modern graphics APIs, DirectX 12 and Vulkan. While the previous generation API, DirectX 11, is still in use, the newer, low-level interfaces of DirectX 12 and Vulkan offer developers greater opportunities for performance enhancement and graphics optimization. This article examines the key features of these APIs, their architectural differences, and their impact on the performance of graphical applications.

The objectives of the study include:

1. Conducting a comparative analysis of the performance and stability of DirectX 12 and Vulkan.

2. Comparing the new APIs with the previous generation interface, DirectX 11, to identify their advantages and disadvantages.

3. Evaluating the modern graphics APIs DirectX 12 and Vulkan efficiency based on parameters such as frame rate and GPU load.

Thus, this study aims to determine which of the modern graphics APIs offers the best combination of performance and efficiency, as well as to provide a rationale for selecting an API for graphics application development.

Literature review

With the above said, the review the performance and stability of the two graphics API between themselves and the comparison of them with graphics API of the previous generation

have been conducted. Firstly, we have conducted the comparative analysis of the performance and stability of DirectX 12 and Vulkan.

A graphics API is essential because it allows developers to avoid dealing with the vast number of variations in graphics hardware. Instead, they work with an interface that shifts the responsibility for creating hardware drivers to the graphics hardware manufacturers, enabling software developers to use a unified API [1].

The first such interface was SGI's IRIS GL, which allowed for both 2D and 3D graphics on devices running the IRIX operating system, developed in 1992. In 1994, this evolved into OpenGL, with the primary distinction being that OpenGL allowed for software implementations of features unavailable in hardware. In 1995, Microsoft developed DirectX, which became OpenGL's main competitor [2]. In 2006, the rights to OpenGL were transferred to the Khronos Group, which in 2014 began developing a modern, low-level graphics API intended to compete with DirectX [3].

Today, DirectX 11 remains the most widely used graphics API from the previous generation, but more developers are now implementing support for the newer DirectX 12 or Vulkan, or even completely abandoning the legacy interface. These new low-level APIs are designed with performance optimization in mind [4]. For many developers today, the choice lies between DirectX 12 and Vulkan, and the following outlines the key features of these two modern graphics APIs.

One key feature of DirectX 12 and Vulkan is fast «draw call» preparation. In 3D rendering, a draw call commands the creation of a polygonal mesh, with more objects requiring more draw calls. Shorter preparation times in DirectX 12 reduce CPU load, minimize GPU idle time, and allow more objects to be displayed on screen, also improving load balancing in multi-core systems.

DirectX 12 introduced the Pipeline State Object (PSO) [2], which stores pipeline states (input assembler, pixel shader, etc.) in a unified, immutable object. PSOs can be quickly changed, enabling hardware and drivers to efficiently translate PSOs into hardware instructions, reducing overhead and improving draw call performance.

In DirectX 11, there is a single instruction queue for rendering, which can lead to inefficiencies. DirectX 12 and Vulkan, however, allow separate queues for graphics and compute tasks, with the CPU and driver distributing GPU resources between them, similar to CPU Hyper-Threading.

The asynchronous queue system in DirectX 12 and Vulkan is termed «Multi-Engine» [2,5]. While tasks in separate queues may have dependencies, Multi-Engine supports concurrent execution of computational instructions, making it a more accurate description than «asynchronous computing», which applies to a narrower range of tasks.

AMD GPUs benefit from Multi-Engine, while NVIDIA chips are less efficient with it due to architectural limitations, with only the Pascal architecture supporting Multi-Engine [5]. The easiest architecture to analyze is AMD's Graphics Core Next (GCN), which underpins all recent AMD GPUs. GCN's strengths and weaknesses make it particularly suited for Multi-Engine. Designed to handle both GP-GPU computing and graphics rendering, GCN is built to offload much of the task of saturating the GPU with parallelism to the hardware, rather than relying on the driver or application. Even early GCN chips support simultaneous execution of multiple compute queues alongside a graphics rendering queue, thanks to two types of command processors: the Graphics Command Processor and the Advanced Compute Engine.

Since the third generation of GCN (Tonga and Fiji chips), the architecture also includes separate schedulers for shader and compute instructions, allowing the processor to dynamically allocate computing resources between different instruction queues. GCN facilitates relatively smooth context switching between compute units, where a unit waiting for data from a long-running operation can take on new tasks from the command processor, saving its register contents in external storage. This external storage is a high-speed integrated cache in GCN, enabling efficient context switching. GCN's control logic is also capable of optimizing GPU utilization by using instructions from separate queues, filling small pipeline gaps efficiently.

The situation with Multi-Engine support in NVIDIA GPUs is far from being as transparent as in the case of AMD. NVIDIA materials, which are in the public domain, do not give a clear answer to all questions. Kepler, Maxwell and Pascal GPU architectures are generally allowed to deal with a mixed load under the control of DirectX 12 and Vulkan. Reason to this is based largely on third-party sources and does not claim to be the ultimate truth.

Unlike AMD, NVIDIA has chosen to split its GPUs into primarily consumer or professional models, starting with the Kepler architecture. The first ones are initially deprived of a lot of computational functions that are useless in game tasks, such as fast execution of double precision calculations. In addition, on the way from the Fermi architecture to Kepler, and then Maxwell, the developers consistently reduced the GPU control logic, shifting some of

the functions to the driver.

Mixed load support even in mainstream NVIDIA chips has expanded significantly since Kepler. Small chips of the Kepler architecture are able to work with a single command queue, whether it's graphics or a purely computational task. In the big Kepler and first-generation Maxwell chips, a separate block was introduced to receive computing Hyper-Q queues, but a separate computing load simultaneously with graphics is only possible under the proprietary CUDA API. In addition, the computing queue can use one and only one of the 32 slots of the CUDA Work Distributor block, which distributes chains of operations between individual streaming multiprocessors [6].

Dynamic power distribution between the graphics and computing queues appeared only in Maxwell of the second generation, but there is a critical limitation: redistribution occurs only at the draw call boundary, which means that the driver needs to allocate the streaming multiprocessor group necessary for a particular task in advance. This gives rise to scheduling errors that cannot be eliminated on the fly. In addition, Maxwell suffers heavy losses from a context change, as intermediate results of calculations are stored in RAM, while the L1 cache and shared memory of the GPU are completely cleared. Under such conditions, the rather short idle time of individual SMs is not as much detrimental to performance as a context change.

Secondly, we have compared the features of DirectX 12 and Vulkan with graphics API of the previous generation.

DirectX 12 got many new rendering features with updates levels 12_0 and 12_1. But unlike previous versions of DirectX, version 12 is not meant to bring the world something never seen before, as was the case with shaders in DirectX 8 and polygon tessellation in DirectX 11. Some features of feature levels 12_0 and 12_1 improve the quality of certain effects, while others are used in advanced rendering algorithms. And yet, most of the points of feature levels 12_0 and 12_1 serve to make the GPU perform faster a number of already known tasks, which otherwise create a large load on the bandwidth of texture mapping units, the memory bus [2].

The additional processing power unlocked by the new API versions allows for richer game graphics with more detailed textures and objects. In some games like Ashes of the Singularity, the choice of API is crucial due to the large number of draw calls required for many units on screen. However, the adoption of new APIs is still limited, and the diversity of user hardware prevents developers from making DirectX 12 and Vulkan-exclusive content widely available.

Modern GPUs are no longer just «graphics processors». Their architecture, featuring many execution units like ALUs or CUDA cores, is capable of handling various tasks suited for GP-GPU, including industrial work, cryptocurrency mining, and machine learning [7].

GP-GPU techniques have been applied in games, with NVIDIA adapting the PhysX API for GPUs after acquiring Ageia. However, no commercial game has fully showcased the potential of non-graphical computations like NVIDIA’s PhysX demos. This is because even the best GPUs lack sufficient resources for large-scale physics calculations without affecting frame rates, especially with new priorities like ultra-high-definition resolution and VR.

General-purpose computing in games isn't limited to physics; techniques like screen space ambient occlusion, reflections, shadow mapping, and global illumination can also be implemented using GP-GPU methods. The boundary between graphics and computation exists only in the application and API architecture, where tasks are processed in separate instruction queues, a concept known as asynchronous computing.

The API layer controlling the GPU has become leaner compared to DirectX 11, where tasks like memory management and queue synchronization were handled automatically. While this allows for performance optimization, it requires programmers to account for various GPU architectures to avoid performance issues [2].

Since Microsoft's 2018 introduction of ray tracing, only DirectX 12 initially supported the technology through DXR. In December 2020, Vulkan added ray tracing support as well [8]. Khronos, with NVIDIA's help, facilitated the transition of ray tracing to Vulkan. Vulkan supports GLSL and HLSL, while DXR supports only HLSL [9]. Additionally, Vulkan’s Deferred Host Operations allow acceleration structures to be created across multiple processor cores, key for Bounding Volume Hierarchy (BVH), a method used in both ray tracing and collision calculations to simplify complex object computations.

In addition to being used in video games, graphics APIs also play an important role in research to visualize results [10]. The reason for using a low-level interface is to have full control over the rendering process and the possibility of increasing performance. To unleash the full potential of Vulkan, the Datoviz library was created, which allows you to achieve the highest available performance [11,12]. One of the reasons why scientific developers prefer Vulkan is its multi-platform. While DirectX 12 is a step up from DirectX 11, with support not

only on Windows 10 and Windows 11, but also on Xbox Series X and Linux, Vulkan is available on Windows XP, Windows 7, Windows 10, Windows 11, SteamOS, Android, Red Hat Linux Enterprise, Tizen, Ubuntu [5]. For example, with the help of Vulkan, a PolyBench port was created for mobile platforms [13]. PolyBench is a collection of benchmarks containing static control parts. The purpose is to uniformize the execution and monitoring of kernels, typically used in past and current publications.

Methodology

Thirdly, we have evaluated the modern graphics APIs DirectX 12 and Vulkan efficiency based on parameters such as frame rate and GPU load.

We have used two parameters to do this. First parameter is frame rate. The more frames per second the graphics card can render, the better. The second parameter is the loading of the GPU. The more graphics card resources the graphics API can use, the better. Stability will be best compared by looking at how far the maximum and minimum frame rates will differ from the average.

A rather small number of games support both DirectX 12 and Vulkan, since these graphics APIs are more similar to each other than between their predecessors (OpenGL for Vulkan and DirectX 11 for DirectX 12), since both interfaces are low-level, and also support ray tracing technology.

To compare graphics API performance, we will be using an Nvidia RTX 2060 Super based gaming PC. Table 1 shows the characteristics of the gaming PC:

- Processor: Intel Core i7 9700k;
- Video Card: ASUS RTX 2060 Super Strix Gaming;
- Motherboard: ASUS TUF Z390-Plus Gaming;
- RAM: DDR-4 16 Gb Kingston HyperX, 2x8 Gb, 2666 MHz;
- SSD: KINGSTON 240Gb;
- Operating System: Windows 10 Ultimate.

Since DirectX 12 and Vulkan are very similar, there are very few games that have support and good optimization of both graphical interfaces. Ashes of Singularity, Red Dead Redemption 2, Serious Sam 4 will be used as examples of comparing DirectX 12 and Vulkan. For comparison DirectX 11 and Vulkan will be used: Rainbow Six Siege 6, Dota 2.

For comparison DirectX 11 and DirectX 12 will be used: Battlefield V, Civilization VI. Games are the best way to compare graphics API, because the graphics in games use the maximum range of technologies that interfaces offer us. For example shaders or ray tracing.

The study will measure the frame rate, and the workload of the GPU. In most cases, these games already have built-in benchmarks that can be used, but for games such as Dota 2 and Battlefield V, scripts have been prepared that would help reproduce the same sequence of actions to compare the performance of interfaces. Measurements were taken 10 times for each game and interface, and the best indicators were selected in the results.

Research results

In our case, the research results display the minimum, average and maximum frame rates that were achieved during the test run, as well as the workload of the GPU. GPU load is an important indicator, since with V-sync turned off, we will be able to observe how much the graphics API can use the resources of the video card. V-sync limits the maximum frame rate to the monitor's frequency, thus preventing the video card from running empty and keeping its power.

Table 1 shows the results of Ashes of Singularity runs at maximum graphics settings and 1440p resolution. The built-in benchmark was used for runs.

Table 2 shows the results of Red dead redemption 2 runs with maximum graphics settings and a resolution of 1440p. The built-in benchmark was used for the runs. The game does not support DirectX 11.

Table 3 shows the results of Serious Sam 4 runs at maximum graphics settings and 1440p resolution. The built-in benchmark was used for runs.

Table 4 shows the results of Rainbow Six Siege runs at maximum graphics settings and 1440p resolution. The built-in benchmark was used for runs. The game does not support DirectX 12.

Table 5 shows the results of Dota 2 runs with maximum graphics settings and 1440p resolution. For runs a prepared replay of the game was used. The game does not support DirectX 12.

Table 6 shows the results of Battlefield V runs at maximum graphics settings and 1440p resolution. For runs, a script was used that reproduced the same sequence of actions in a single player campaign. This was necessary to create the most approximate conditions for the correct comparison of graphics APIs. The game does not support Vulkan.

Table 7 shows the results of measurements of Civilization 6 with maximum graphics settings and a resolution of 1440p. The built-in benchmark was used for runs. The game does not support Vulkan.

Results and discussion

On Figures 1, 2 we can see visualized results of research. Figure 1 represents the comparison of average frame rate for all games and all used graphics API. There we can see that in

Table 1 – The results of Ashes of Singularity runs

Graphics API	Minimum frame rate, fps	Maximum frame rate, fps	Average framerate, fps	GPU load, %
DirectX 11	19	62	25.7	42.3
DirectX 12	41	69	54.1	97.2
Vulkan	37	72	53.6	92.5

Table 2 – The results of Red dead redemption 2 runs

Graphics API	Minimum frame rate, fps	Maximum frame rate, fps	Average framerate, fps	GPU load, %
DirectX 12	10	97	56.5	96.8
Vulkan	26	81	58.2	97.1

Table 3 – The results of Serious Sam 4 runs

Graphics API	Minimum frame rate, fps	Maximum frame rate, fps	Average framerate, fps	GPU load, %
DirectX 11	41	81	52.1	97.8
DirectX 12	37	77	56.8	98.1
Vulkan	69	101	73.4	83.9

Table 4 – The results of Rainbow Six Siege runs

Graphics API	Minimum frame rate, fps	Maximum frame rate, fps	Average framerate, fps	GPU load, %
DirectX 11	44	207	55.8	96.8
Vulkan	25	168	57.3	97.1

Table 5 – The results of Dota 2 runs

Graphics API	Minimum frame rate, fps	Maximum frame rate, fps	Average framerate, fps	GPU load, %
DirectX 11	158	213	190.8	84.7
Vulkan	165	223	193.4	96.3

Table 6 – The results of Battlefield V runs

Graphics API	Minimum frame rate, fps	Maximum frame rate, fps	Average framerate, fps	GPU load, %
DirectX 11	101	142	113.1	64.8
DirectX 12	25	201	145.5	97.9

most cases API of new generation perform better, than DirectX 11. Figure 2 represents difference between minimum and maximum frame rates. There we can see that in most cases Vulkan offers more consistent performance, while DirectX 12 works with predictions, heavy frame

Table 7 – The results of measurements of Civilization 6

Graphics API	Minimum frame rate, fps	Maximum frame rate, fps	Average framerate, fps	GPU load, %
DirectX 11	24	61	47.3	95.3
DirectX 12	42	82	59.8	98.8

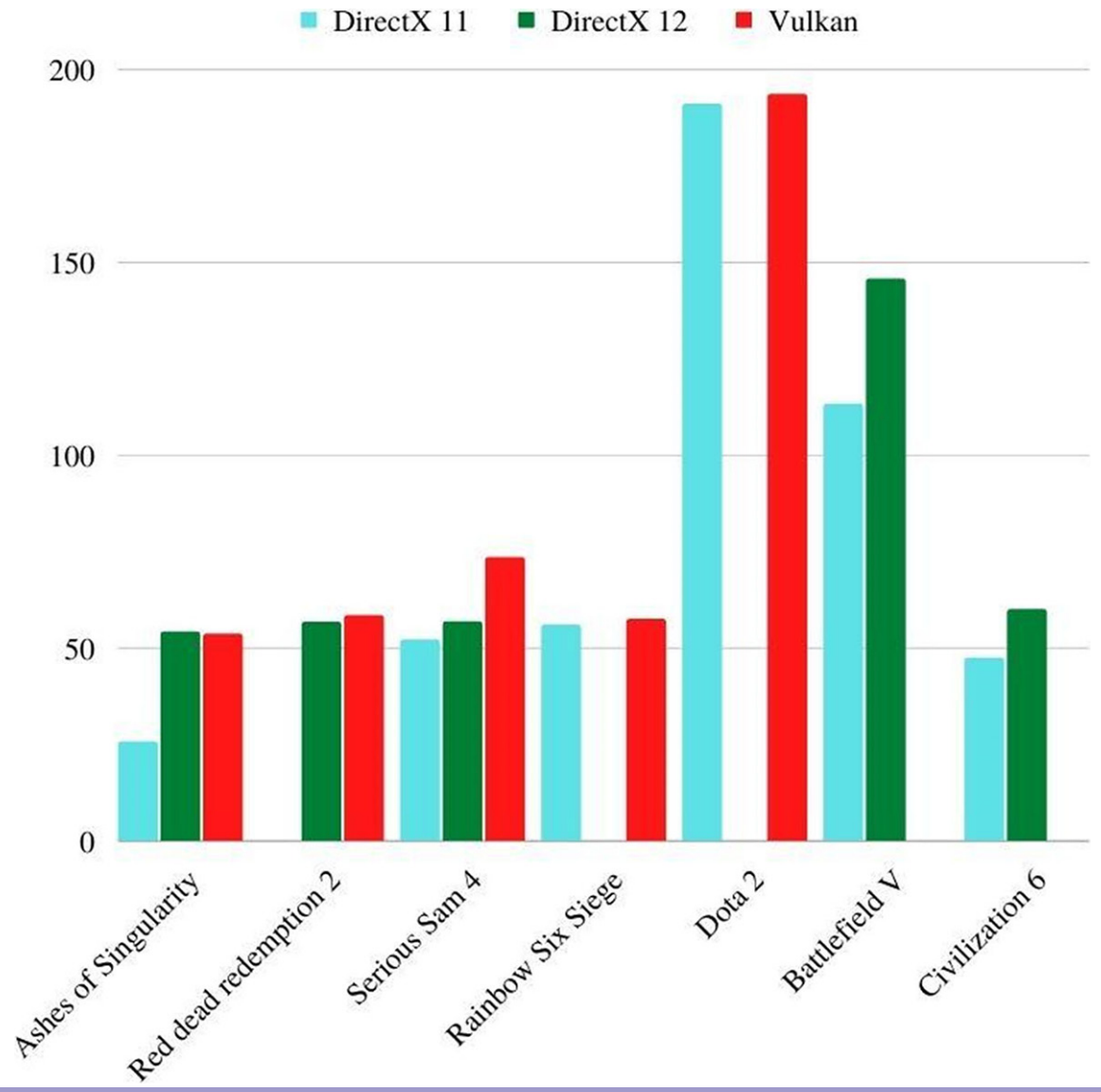

Figure 1 – Average frame rates of graphics API

drops often occur, sometimes even worse than DirectX 11. For Figure 1 the higher the value is the better. For Figure 2 the lower the value is the better.

Looking at the research results, we can see that in the vast majority of examples, next-generation graphics APIs produce higher frame rates (10 to 50 percent gains) and also make better use of the graphics card's power. For example, in Ashes of the Singularity, the next-generation APIs use almost all of the processing power, when DirectX 11 could only use about 60 percent on average [14]. This is due to the ability of modern APIs to distribute the load evenly across processor cores, and now software developers have more access directly to the hardware. The latter, of course, can create many problems for small studios, since it will be difficult for them to take into account all the variety of possible architectures, but for this there are ready-made game engines that take care of this for developers.

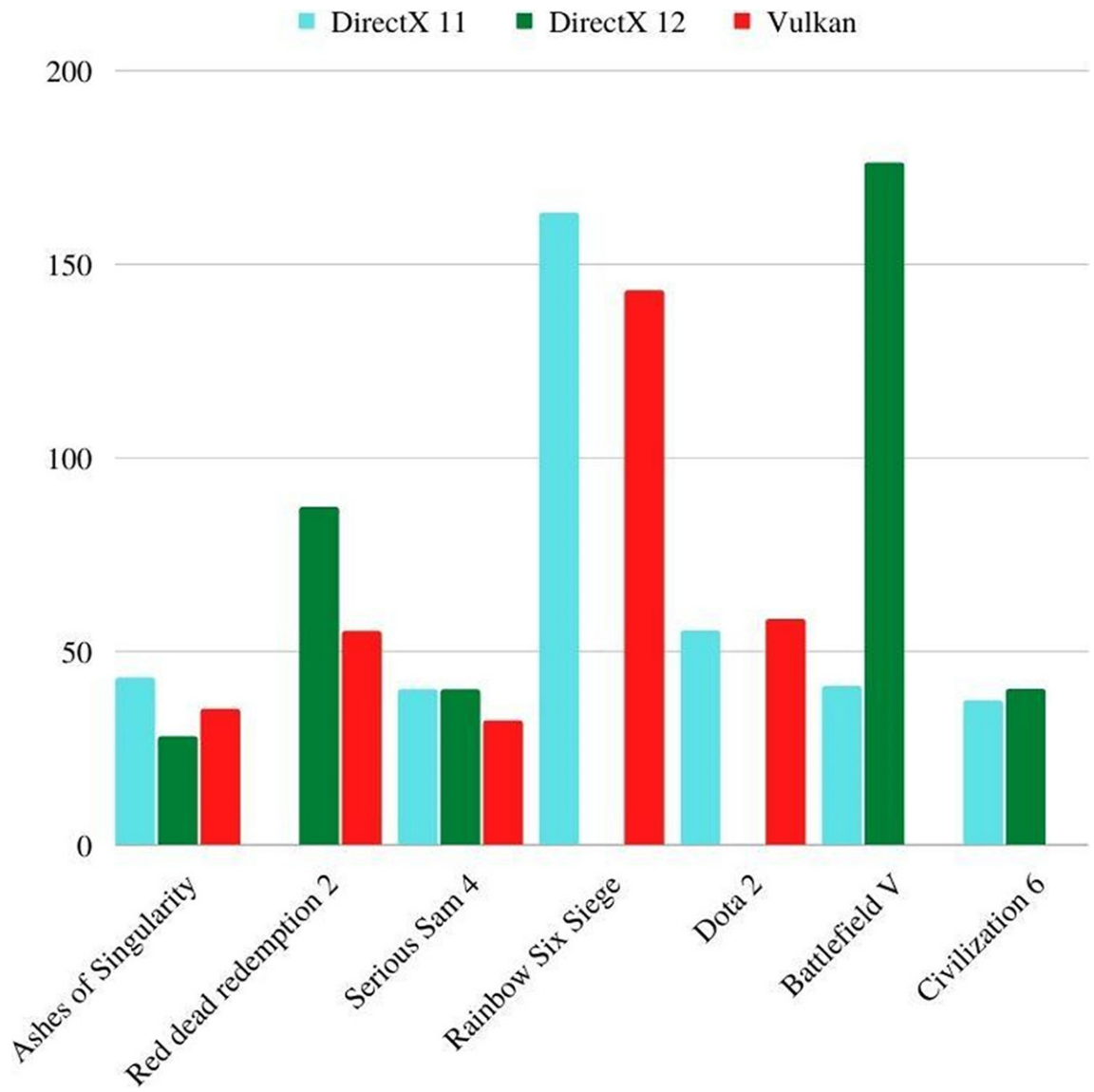

Figure 2 – Difference between minimum and maximum frame rate

In the case of Dota 2 and Rainbow Six Siege, we can only see a small framerate increase on Vulkan compared to DirectX 11, but we can see a significant framerate stabilization on Vulkan for Rainbow Six Siege, and for Dota 2 we can see a uniform improvement in all frame rate measurements. This behavior can be explained by insufficient optimization of the new graphics API.

If we look at the comparison of the two new graphics APIs, in the case of Red Dead Redemption 2 and Serious Sam 4, Vulkan performs significantly better. In Red Dead Redemption 2, we can see not only higher average frame rates, but also more stable frame rates. While Vulkan's frame rate sags relative to the average is only up to 50%, DirectX 12's frame rate sags up to 18% relative to the average. For Serious Sam 4, a similar trend can be observed. But this picture looks more like poor optimization of DirectX 12, since the performance is not much better than the previous version. For Ashes of the Singularity, we can see that DirectX 12 performs slightly better than Vulkan, although the difference is not too big.

Looking at the results of Dota 2 and Rainbow Six Siege, we can see that in these games Vulkan performs only 1-2% better, while fully loading the GPU. The result is far from in favor of Vulkan, but this can be justified by the use of its earlier versions, as well as insufficient optimization by the developers.

DirectX 12 compared to DirectX 11 in Battlefield V and Civilization 6 performs significantly better, by 20%, however, in the case of Battlefield V, you can observe a significant drop in frame rate at times when a huge number of particles are rendered, and this drop was noticed stably during time of all test situations.

Conclusion

All the objectives of the study were successfully achieved. The comparative analysis demonstrated that modern graphics APIs, DirectX 12 and Vulkan, provide significant advantages for developers in terms of performance and flexibility compared to DirectX 11. Key architectural features were examined, such as the Pipeline State Object (PSO) in DirectX 12 and the Multi-Engine capabilities in both APIs, allowing for more efficient load balancing between the CPU and GPU.

Based on the comparative analysis, the following conclusions can be drawn.

1. In terms of performance, Vulkan performs on average 2-3 percent worse than DirectX 12, but instead produces more stable frame rates, which is often a more important factor.

DirectX 12 often suffers severe fps drops, when Vulkan, offering slightly lower performance, provides much better stability. If we compare those options when only one modern graphical interface was available and DirectX 11, then DirectX 12 was significantly more successful than Vulkan, with way better performance and stability.

2. Some results showed that DirectX 11 was just as good or slightly better than the next generation of graphics API. One of the reasons why DirectX 11 wasn't much worse in some runs is the lack of experience with the new graphics API, as well as the increased responsibility of developers for many things that the interface previously did for them automatically. Ashes of Singularity is a good example of the progress of new graphics API. Here we see a significant increase in performance and stability. This is a good sign, because the more experience developers have with new generation of graphics API, the more significant performance gains we can see.

3. When it comes to ray tracing features, Vulkan is in the lead due to its great features, but DRX (DirectX ray tracing) is gradually adding more and more new features. For scientific research, preference also remains for Vulkan, primarily for its cross-platform. And this is a huge advantage for new gaming platforms, which do not belong to Microsoft. For example we can use Google Stadia. The presence of Vulkan support allows developers to release their games on a wider range of platforms, which is an undeniable advantage. Summing up, we can say that in general, Vulkan is a more preferred graphics API than its competitor.

4. In this study, the authors conducted an in-depth analysis of the performance and stability of DirectX 12 and Vulkan, evaluated GPU resource utilization efficiency, and provided recommendations for selecting a graphics API for various types of applications. The authors also analyzed the architectural features of AMD and NVIDIA GPUs, proposing optimization methods for graphics applications based on the type of GPU. The article presents the authors' practical results and research, supported by experimental data and comparative performance analysis.

REFERENCES

1. M. Nwadiugwu. The roles of graphics API, graphic hardware, and the graphics pipeline, ResearchGate [Online], 2015. Available: https://www.researchgate.net/publication/283052466_
2. Microsoft. (2021, Dec. 30). DirectX 12 programming guide [Online]. Available: https://docs.microsoft.com/en-us/windows/win32/direct3d12/directx-12-programming-guide
3. M. Bailey. Introduction to the vulkan graphics API, in SA '18: SIGGRAPH Asia 2018, Tokio, Japan, 2018, pp. 1-225.
4. Microsoft. (2014, aug. 12). Developer Blog. DirectX 12 – High Performance and High Power Savings [Online]. Available: https://devblogs.microsoft.com/directx/directx-12-high-performance-and-high-power-savings/
5. J. Leech, T. Hector (2023) Vulkan® Documentation and Extensions: Procedures and Conventions [Online]. Available: https://www.khronos.org/registry/vulkan/specs/1.3/styleguide.html
6. T. Amert, N. Otterness, M. Yang, J.H. Anderson, F.D. Smith. GPU Scheduling on the NVIDIA TX2: Hidden Details Revealed, in 2017 IEEE Real-Time Systems Symposium (RTSS), Paris, France, 2017, pp. 104-115.
7. J. Ghorpade. GPGPU Processing in CUDA Architecture, Advanced Computing: An International Journal, vol. 3 (1), pp. 105-120, Feb 2012.

8. M. Rusch, N. Bickford, N. Subtil. Introduction to Vulkan Ray Tracing, in Ray Tracing Gems II, Berkeley: Apress, 2021, pp. 213-255.
9. C. Wyman, A. Marrs. Introduction to DirectX raytracing, in Ray Tracing Gems, Berkeley:Apress, 2019, pp. 21-47.
10. C. Ioannidis, A. Boutsi. Multithread rendering for cross-platform 3D visualization based on Vulkan API, in 3rd BIM/GIS Integration Workshop and 15th 3D GeoInfo Conference 2020, vol. XLIV-4/W1-2020, London, UK, 2020, pp. 57-62.
11. P. Ramachandran, G. Varoquaux. Mayavi: 3D visualization of scientific data, Computing in Science & Engineering, vol. 13 (2), pp. 40-51, Oct 2010.

DirectX 12 және Vulkan графикалық қосымшаларының жалпы интерфейстерін салыстыру

[1]ДАНЕНОВА Гульмира Тулендиевна, *т.ғ.к., доцент, guldan72@mail.ru,*
[1]КӨККӨЗ Махаббат Мейрамқызы, *п.ғ.к., доцент, makhabbat_k@bk.ru,*
[1]АХМЕТЖАНОВ Талгат Бураевич, *т.ғ.к., доцент, akhmetzhanov_t@mail.ru,*
[1]САЙЛАУ ҚЫЗЫ Жұлдыз, *PhD, доцент м.а., s_k_zhuldiz@mail.ru,*
[1]*АБИЛДАЕВА Гулнур Балтабаевна, *магистр, аға оқытушы, g.abildaeva@kstu.kz,*
[1]«Әбілқас Сағынов атындағы Қарағанды техникалық университеті» КеАҚ, Н. Назарбаев даңғылы, 56, Қарағанды, Қазақстан,
**автор-корреспондент.*

Аңдатпа. *Бұл зерттеу қазіргі заманғы API графикалық интерфейстерін, олардың өнімділігін, мүмкіндіктерін зерттеуге, сондай-ақ оларды алдыңғы буын графикалық интерфейстерімен салыстыруға бағытталған. Зерттеудің мақсаты – бірдей жағдайларда пайдаланылған кезде осы интерфейстердің өнімділігін зерттеу. Интерфейстердің қайсысы ең жақсы өнімділікті ғана емес, сонымен қатар ең тұрақты іске асыруды қамтамасыз ететінін анықтау қажет. Жұмыс барысында алғаш рет жаңа буынның екі графикалық интерфейсіне де тән негізгі инновациялар қарастырылды. Ең алдымен, бұл асинхронды есептеу саласындағы өзгерістер, атап айтқанда құбырдағы өзгерістер. Бұл технологиялардың ең айқын қолданылуы ойын дамытуда, бірақ графикалық интерфейстер нәтижелерді визуализациялау үшін ғылымда кеңінен қолданылады. Екі интерфейстің өнімділігі мен тұрақтылығының нәтижелері оларды зерттеу барысында алдыңғы буын интерфейсімен салыстыру арқылы алынды.*

Кілт сөздер: *графикалық API, DirectX 11, DirectX 12, Vulkan, Nvidia.*

Сравнение общедоступных интерфейсов графических приложений DirectX 12 и Vulkan

[1]ДАНЕНОВА Гульмира Тулендиевна, *к.т.н., доцент, guldan72@mail.ru,*
[1]КОККОЗ Махаббат Мейрамовна, *к.п.н., доцент, makhabbat_k@bk.ru,*
[1]АХМЕТЖАНОВ Талгат Бураевич, *к.т.н., доцент, akhmetzhanov_t@mail.ru,*
[1]САЙЛАУ ҚЫЗЫ Жұлдыз, *PhD, и.о. доцента, s_k_zhuldiz@mail.ru,*
[1]*АБИЛДАЕВА Гулнур Балтабаевна, *магистр, старший преподаватель, g.abildaeva@kstu.kz,*
[1]НАО «Карагандинский технический университет имени Абылкаса Сагинова», пр. Н. Назарбаева, 56, Караганда, Казахстан,
**автор-корреспондент.*

Аннотация. *Данное исследование направлено на изучение современных графических интерфейсов API, изучение их производительности, возможностей, а также сравнение их с графическими интерфейсами предыдущего поколения. Цель исследования – изучить*

производительность этих интерфейсов при использовании в одинаковых условиях. Необходимо выяснить, какой из интерфейсов обеспечивает не только наилучшую производительность, но и наиболее стабильную реализацию. В ходе работы впервые были рассмотрены основные нововведения, характерные для обоих графических интерфейсов нового поколения. Прежде всего, это изменения в области асинхронных вычислений, а именно изменения в конвейере. Наиболее очевидное применение этих технологий – в разработке игр, но графические интерфейсы также широко используются в науке для визуализации результатов. Результаты производительности и стабильности обоих интерфейсов были получены путем сравнения их с интерфейсом предыдущего поколения в ходе исследования.

Ключевые слова: *графический интерфейсы API, DirectX 11, DirectX 12, Vulkan, Nvidia.*

REFERENCES

1. M. Nwadiugwu. The roles of graphics API, graphic hardware, and the graphics pipeline, ResearchGate [Online], 2015. Available: https://www.researchgate.net/publication/283052466_
2. Microsoft. (2021, Dec. 30). DirectX 12 programming guide [Online]. Available: https://docs.microsoft.com/en-us/windows/win32/direct3d12/directx-12-programming-guide
3. M. Bailey. Introduction to the vulkan graphics API, in SA '18: SIGGRAPH Asia 2018, Tokio, Japan, 2018, pp. 1-225.
4. Microsoft. (2014, aug. 12). Developer Blog. DirectX 12 – High Performance and High Power Savings [Online]. Available: https://devblogs.microsoft.com/directx/directx-12-high-performance-and-high-power-savings/
5. J. Leech, T. Hector (2023) Vulkan® Documentation and Extensions: Procedures and Conventions [Online]. Available: https://www.khronos.org/registry/vulkan/specs/1.3/styleguide.html
6. T. Amert, N. Otterness, M. Yang, J.H. Anderson, F.D. Smith. GPU Scheduling on the NVIDIA TX2: Hidden Details Revealed, in 2017 IEEE Real-Time Systems Symposium (RTSS), Paris, France, 2017, pp. 104-115.
7. J. Ghorpade. GPGPU Processing in CUDA Architecture, Advanced Computing: An International Journal, vol. 3 (1), pp. 105-120, Feb 2012.
8. M. Rusch, N. Bickford, N. Subtil. Introduction to Vulkan Ray Tracing, in Ray Tracing Gems II, Berkeley: Apress, 2021, pp. 213-255.
9. C. Wyman, A. Marrs. Introduction to DirectX raytracing, in Ray Tracing Gems, Berkeley:Apress, 2019, pp. 21-47.
10. C. Ioannidis, A. Boutsi. Multithread rendering for cross-platform 3D visualization based on Vulkan API, in 3rd BIM/GIS Integration Workshop and 15th 3D GeoInfo Conference 2020, vol. XLIV-4/W1-2020, London, UK, 2020, pp. 57-62.
11. P. Ramachandran, G. Varoquaux. Mayavi: 3D visualization of scientific data, Computing in Science & Engineering, vol. 13 (2), pp. 40-51, Oct 2010.