3-бөлім Раздел 3 Section 3

Информатика Информатика Computer Science

IRSTI 519.687.1

Application of Vulkan technology for 3D Visualization of large computing data which change over the time

Mustafin M.B., Al-Farabi Kazakh National University
Almaty, Kazakhstan, E-mail: mustafin.mb@gmail.com
Akhmed-Zaki D.Zh., University of International Business
Almaty, Kazakhstan, E-mail: darhan_a@mail.ru
Turar O.N., Al-Farabi Kazakh National University
Almaty, Kazakhstan, E-mail: Olzhas.Turar@kaznu.kz

In this work, a high-performance application for visualization of large-size grid models (about a million cells) with using of Vulkan technologies was developed. Vulkan is a new software interface (API) which controls the graphic processor (GPU). Vulkan became a low-level API, thanks to which the entire GPU capability was used, such as memory and synchronization control, error checking, creation of commands performed by graphic processor, etc. Thus it shows high performance with less load on the CPU. For the operation of the application without interruption, double buffering of vertex buffer and multi-threading of the processor was used. The results of each 100th iteration of the Jacobi method for solving the Poisson's equation, namely data of each iteration, were taken for the 2D and 3D model visualization. Using of the above given methods, grid model examples are given. As a result of this work, a prototype of a visualizer was developed and presented, and it can be used for any results of numerical mathematical modeling on structured and unstructured 3D grids.

Key words: Vulkan, 2D, 3D, computer graphics, visualization, double buffering, grid model, multithreading.

Использование технологии Vulkan для 3D-визуализаций больших вычислительных данных, изменяющихся со временем

Мустафин М.Б., Казахский национальный университет имени аль-Фараби,
г. Алматы, Казахстан, E-mail: mustafin.mb@gmail.com
Ахмед-Заки Д.Ж., Университет международного бизнеса,
г. Алматы, Казахстан, E-mail: darhan_a@mail.ru
Турар О.Н., Казахский национальный университет имени аль-Фараби,
г. Алматы, Казахстан, E-mail: Olzhas.Turar@kaznu.kz

В данной работе было разработано высокопроизводительное приложение для визуализации сеточных моделей больших размеров (около млн. ячеек), с использованием технологий Vulkan. Vulkan – это новый программный интерфейс (API) управляющий графическим процессором (GPU). Vulkan является низкоуровневым API, благодоря чему была использована вся возможность графического процессора такие как: управление памятью и синхронизацией, проверки на ошибки, создание команд выполняемых графическим процессором и т.д. Таким образом, показывает высокую производительность при меньшей нагрузке на центральный процессор. Для работы приложения без прерывания была использована двойная буферизация буфера вершин и многопоточность процессора. Для визуализаций 2D и 3D модели были взяты результаты каждой сотой итерации метода Якоби для решения уравнения Пуассона. Используя вышеуказанные методы, приведены примеры сеточной модели. В результате данной работы был разработан и представлен прототип визуализатора, которую можно использовать для любых результатов численного математического моделирования на структуированных и неструктурированных 3D сетках.

Ключевые слова: Vulkan, 2D, 3D, компьютерная графика, визуализация, двойная буферизация, сеточная модель, многопоточность.

Уақыт өте келе, үлкен есептелетін деректерді 3D визуализациялау үшін Vulkan технологиясын пайдалану

Мустафин М.Б., Әл-Фараби атындағы қазақ ұлттық университеті,
Алматы қ., Қазақстан, E-mail: mustafin.mb@gmail.com
Ахмед-Заки Д.Ж., Халықаралық бизнес университеті,
Алматы қ., Қазақстан, E-mail: darhan_a@mail.ru
Тұрар О.Н., Әл-Фараби атындағы қазақ ұлттық университеті,
Алматы қ., Қазақстан, E-mail: Olzhas.Turar@kaznu.kz

Бұл жұмыста Vulkan технологияларын қолдану арқылы үлкен өлшемді тор модельдерін (миллионға жуық ұяшық) визуализациялау үшін жоғары өнімді бағдарлама жасалды. Вулкан – графикалық процессорды (GPU) басқаратын жаңа бағдарламалық интерфейс (API). Вулкан – төменгі деңгейлі API, соның арқасында графикалық процессордың бүкіл мүмкіндіктері пайдаланылды, мысалы, жады мен синхрондау, қателерді тексеру, графикалық процессорлармен орындалатын командаларды құру және т.б. Осылайша, орталық процессордың аз жүктеу кезінде жоғары өнімділікті көрсетеді. Бағдарламаның үздіксіз жұмыс істеуі үшін төбелер буферінің қос буферлеуін және процессордың көп ағындылығы қолданылды. 2D және 3D модельдерін визуализациялау үшін Пуассон теңдеуін шешу үшін Якоби әдісінің нәтижелері, яғни әрбір 100-ші итерацияның деректері алынған. Жоғарыда келтірілген әдістерді қолдану арқылы торлық моделінің мысалдары келтірілген. Осы жұмыстың нәтижесінде визуализатордың прототипі әзірленіп ұсынылды, оны құрылымдық және құрылымдық емес 3D торларында сандық математикалық модельдеудің кез келген нәтижесіне қолдануға болады.
Түйін сөздер: Vulkan, 2D, 3D, компьютерлік графика, визуализация, қос буферлеу, торлы модель, көп ағындылық.

1 Introduction

Vulkan is an API for graphics and computing devices. Like an OpenGL, Vulkan allows to display real-time 3D graphics with high performance, as well as a higher pproductivity and a lower load on CPU. Vulkan technology was based on AMD Mantle technology. Compared with OpenGL technology, Vulkan is a low-level API and generally has similar capabilities. This allows to use all the features of GPU for computing, to get low-level access to GPU and control its functioning.

Unlike from OpenGL, Vulkan shaders are represented by a binary intermediate representation of SPIR-V programs. SPIR-V is the only supported shader language for the Vulkan. It is adopted at the level of API and used for creation of conveyors designed to control the device and for the performance of work of application.

This paper proposes a real-time visualization of large grid models with using of Vulkan technologies on the typical personal computers equipped with a discrete graphic card. To store and update data, double buffering of vertex buffers and multithreading C ++ 11 was used. The proposed approach to the visualization of the grid model optimizes the speed of drawing. The paper also presents the visualization of a numerical algorithm for solving the Poisson algorithm 2D with a size of 1000×1000 (10^6 cells) and 3D with a size of $33 \times 33 \times 11$.

Let's note that the presented application can be used for any results of numerical mathematical modeling on structured and unstructured 3D grids.

2 Literature review

Currently, the OpenGL Shader Language (GLSL) became the main part of programming with the using of OpenGL library [1–3]. With help of GLSL, you can use the power of graphics processor for display and computing. Based on GLSL, the RGL library was created, which offers three-dimensional visualization in real time [4,5]. At work [6,7], the capabilities of the graphics processor and the shader language of the OpenGL library were used, and the results of numerical mathematical modeling were visualized.

Recently, Vulkan technology is gaining great popularity in the field of visualization [8]. Vulkan is a cross-platform graphical and computing API developed by the Kronos Group consortium [9]. Vulkan is an OpenGL recipient, but it differs very much. The main difference is that to make the OpenGL driver, is the responsibility of the programmer [10,11]. Another difference between Vulkan technology and OpenGL is the shader language. Vulkan supports the only shader language SPIR-V [12]. In the literatures [13–15], you can get acquainted with the specifications of the Vulkan technology.

At work [16], languages and libraries for multi-threaded programming are described, also, how to develop programming skills, and various testing and debugging methods developed for multi-threaded programs over the past 20 years are described and demonstrated.

3 Materials and methods

The purpose of this work was to develop a high-performance application for visualizing of results of numerical mathematical modeling,by the using of Vulkan technology.

In older APIs such as OpenGL, the driver controls synchronization and memory, checks for errors during the running of the application. This is convenient for programmers, but it takes CPU time. In the Vulkan, all responsibilities are transferred to the programmer, i.e. almost all status tracing, memory and synchronization management [13].

3.1 Coordinate systems

Vulkan works with segments and triangles, representing their vertexes as points in three-dimensional space. They are called vertices and are shown in Figure 1 [3].

For beginning 50×50 sizes grid was created and filled with random colors. Each square of the grid consists of two triangles, which set by the coordinates [7]. In the functions of initVertices(), the vector was filled in, which contains the coordinates and color of each vertex of the triangle.

3.2 Memory and resources

Almost for all computing systems, including Vulkan, memory is extremely important. There are two basic types of memory in Vulkan: CPU memory (host memory) and GPU memory (device memory). When creating a new object in Vulkan, you will need memory to store data. The regular CPU memory is used for it.

Vulkan works with data, and data is stored in resources. There are two basic types of resources in Vulkan: buffers and images [10]. A buffer is a simple linear piece of memory that

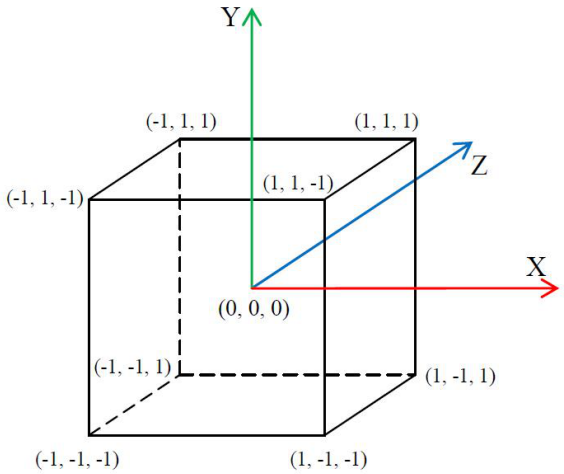

Figure 1: Coordinate system of vertices

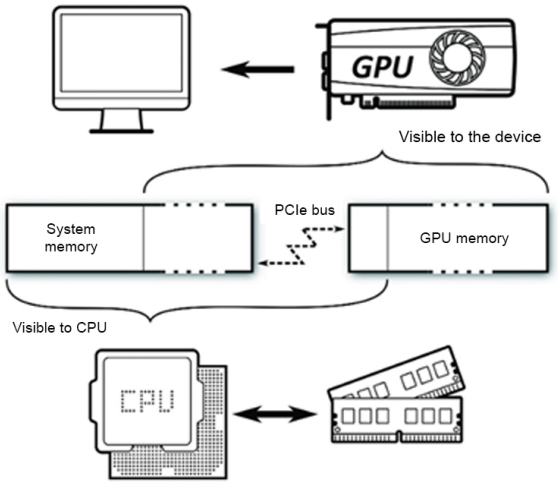

Figure 2: CPU and GPU memory

can be used for various purposes. They are used to store linear structured and unstructured data, which can be in format or simply be bytes in memory.

3.3 Device memory management

When Vulkan works with data, this data should be stored in device memory (device memory). In Figure 2, the GPU and CPU memory circuit, each with its own memory [10].

3.4 Device memory allocation

Device memory allocation is represented as a VkDeviceMemory object created with the help of vkAllocateMemory() functions, the prototype of which is shown below [10, 11, 14]:

```
VkResult vkAllocateMemory (
VkDevice device,
const VkMemoryAllocateInfo* pAllocateInfo,
```

```
const VkAllocationCallbacks* pAllocator,
VkDeviceMemory* pMemory
);
```

After allocating the memory of the device, it can be used to store data. Thus, vertexBuffer called buffer was created and used to store the vertex data. After all works we get the following Figure 3:

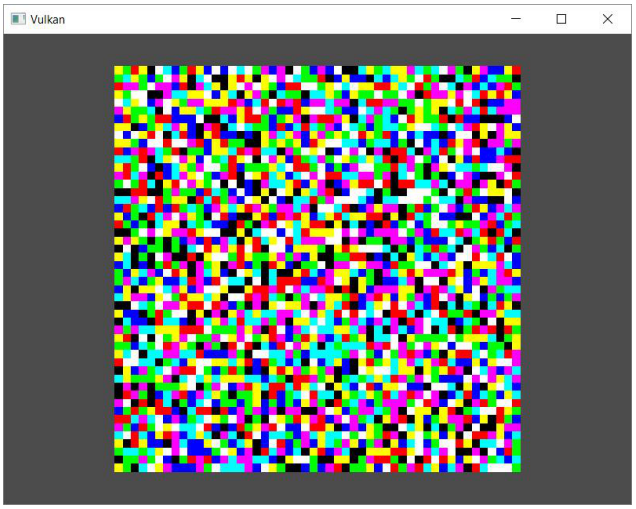

Figure 3: Grid with size of 50×50

When visualizing in real time, new data will come in, and this is the color data of each cell. Since the coordinates and color of the cells are recorded in one buffer, the coordinates will be overwritten, and this is the waste of a precious time of CPU. To solve this problem, a separate buffer was created for these colors. As a result, we get two buffers: posBuffer - for storing coordinate data and colBuffer - for storing color data. The program will work in real time, which means data will be flowing continuously. Let's create a new colBuffer1 buffer to store the new data.

3.5 Buffer update

To update the data inside the buffer, the vkCmdUpdateBuffer function was used [15]. vkCmdUpdateBuffer() copies data directly from CPU memory to buffer memory. Data is collected from the CPU memory when calling, and upon returning from vkCmdUpdateBuffer (), one can free up this memory or write new data to it.

3.6 Double buffering of vertex buffer

Double buffering is a data preparation method that provides the ability of out-turn of the finished result without interruption [6]. This method is commonly used when working with frame buffer. This paper describes how to use a similar approach for buffering of color data on a model of sheet.This method is as follows. A second buffer is created, and the data is filled only on it. As soon as the reading process is completed, the buffers change places, and data output will start from the second buffer, and new data will be filled into the first buffer.

This is similar to double frame buffering and is used for vertex buffers. Figures 4 and 5 show examples of drawing each buffer.

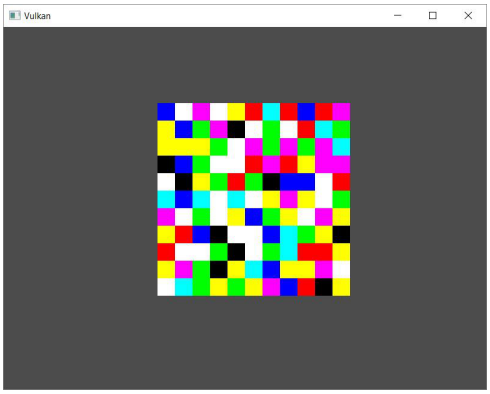

Figure 4: Drawing of the colBuffer

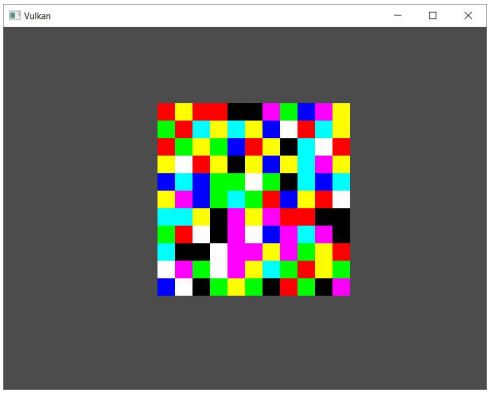

Figure 5: Drawing of the colBuffer1

When reading and writing new data, the drawFrame() drawing function stops and the program will not react until the data is initialized. Despite the fact that Vulkan processes on the CPU and GPU occur asynchronously, interactive control of a model as the rotation and transfer the model using the mouse and keyboard should be done on the CPU. If all actions on the central processor will occur in the same thread, we will not be able to interact with the three-dimensional model while writing to the buffer. For example, if we call the read and write function every ten seconds, and this function will be executed in six seconds, as a result we will get a program that does not respond at the time of initialization and only works for the remaining time. A visual example is shown below in Figure 6.

Figure 6: Green - Drawing, gray - Copy (A-host to host, B - host to device)

In order the input data recorded without stopping the work of drawing, it must be performed in parallel. For this we will use C ++ 11 multithreading [16]. Modern computers have multi-core processors, in which multithreading is performed by the fact that several processes are executed on different cores. Using the multithreading feature, we pass the command to read and fill the buffer into another thread, and it will be executed in parallel without affecting into the drawing. That is, when the input data from the first buffer is displayed on the screen, the second buffer will be updated in parallel. As soon as the second buffer is ready, the buffers change places. A visual example is shown in Figure 7.

As you can see in Figure 6 and 7, initialization of data takes place in two stages: reading and writing data into the system memory (Gray A), and copying from system memory to GPU memory (Gray B). It takes longer time to initialize the reading and writing of new data into the system memory, i.e. host to host.

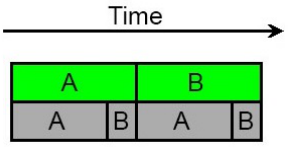

Figure 7: Green - Drawing, gray - Copy (A-host to host, B - host to device)

4 Results and discussion

To get the result, a personal computer (Core i7 3770 3.40 GHz, 8Gb DDR3) equipped with a discrete graphics card (nVidia GeForce GTX650Ti, 1Gb GDDR5) was used. As an input data, the results of each 100th iteration of the Jacobi method for solving the Poisson equation [17, 18] with a size of 1000×1000 for 2D and $33 \times 33 \times 11$ for 3D, the quantity of iterations of 20,000 were taken. The Dirichlet boundary conditions for the problem under consideration are: u(0, y) = 0, u(1, y) = 0, u(x, 0) = 0 u(x, 1) = 0. To visualize the three-dimensional model was used fotmat GRDECL [19]. On Figure 8 the prototype of the application that visualizes the result of the Poisson equation grid model is shown.

Measurements were made and the results presented in Table 1 were obtained.

Table 1. Results of measurement.

Dimension	2D,1000×1000	3D, $33 \times 33 \times 11$
Quantity of cells	10^6	11979
FPS (frames per second)	1750	4415
copy host to host, ms	8438	97
copy host to device, ms	14	0

When running the application for a grid model with the above mentioned dimensions, the more time takes copying host to host, i.e. reading and writing data into the system memory, which depends on computer resources: reading and writing speed of the hard disk, data transfer speed via the Internet. Copying data from system memory to device memory depends on the following GPU characteristics: memory bus, memory interface, memory transfer speed, memory bandwidth, type of dedicated video memory. Characteristics of the graphics processor on which the work was performed are shown in Table 2.

Table 2. Characteristics of the graphic card nVidia Geforce GTX650Ti.

Memory bus	PCI Express x16 Gen3
Memory interface	128 bit
Memory data transfer rate	5400 MGhz
Memory bandwidth	86.40 GB/s
Type of dedicated video memory	1024 MB GDDR5

5 Conclusion

By using the Vulkan technology, a high-performance application was developed which visualizes real-time grid models of numerical mathematical modeling. In order the application

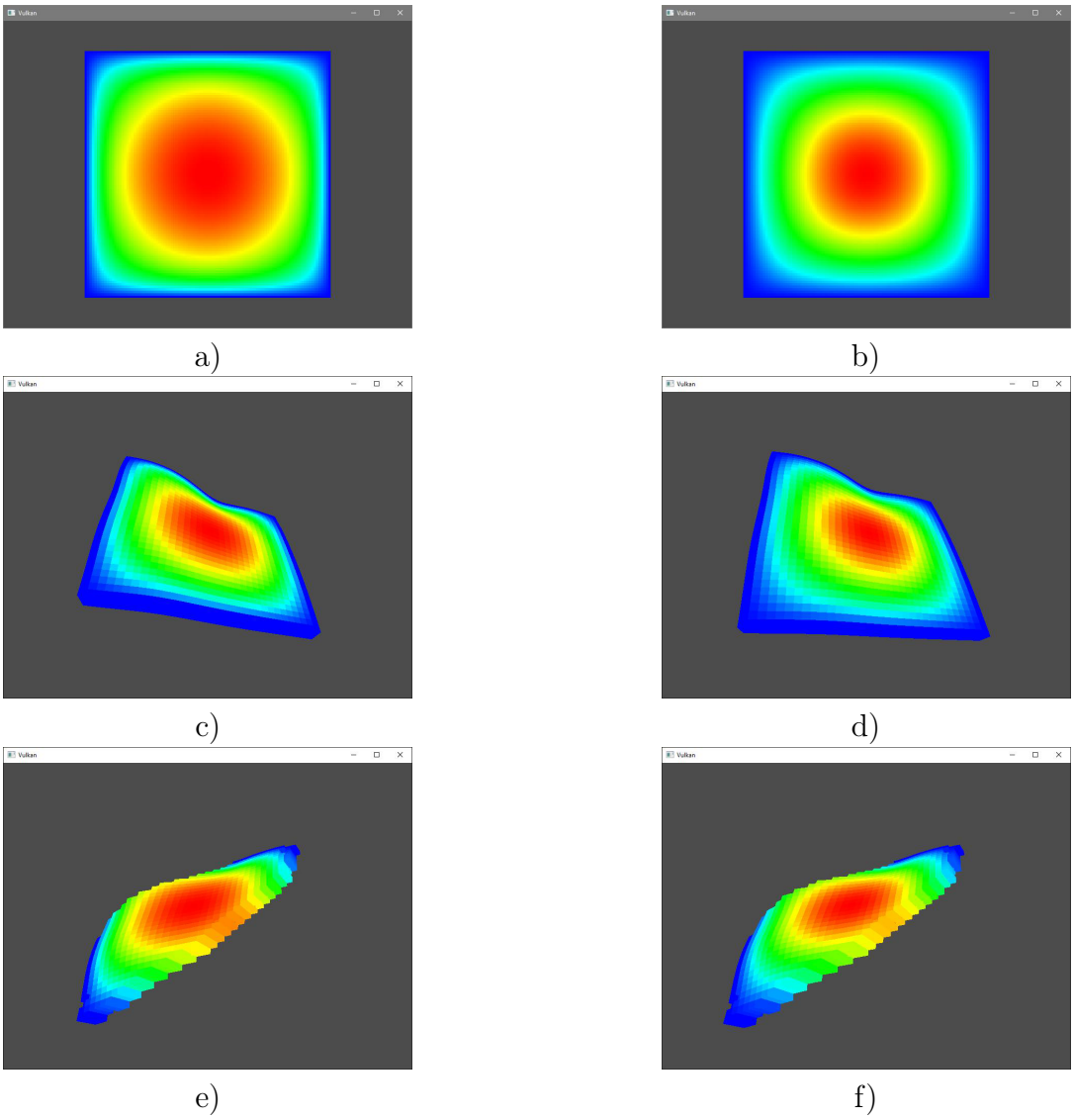

Figure 8: Examples of visualizations of the grid model of the results of the Jacobi iterative method for solving the Poisson equation: a) 2D model of the first iteration, b) 2D model of the latest iteration, c) 3D model of the first iteration, d) 3D model of the latest iteration, e) 3D model with inactive areas of the first iteration, f) 3D model with inactive areas of the latest iteration

worked without interruption, double buffering of vertex buffers and multithreading of C ++ 11 was used. The double buffering method of vertex buffers used for color data buffering on the model of oil reservoir. Such an approach to the visualization of the grid model optimized the speed of the application. An application that visualizes 3D graphics in real time, with high performance and lower CPU load, was introduced. For example, the results of the Poisson equation 2D and 3D were taken. The ready application will be used for numerical mathematical modeling results' visualization.

6 Acknowledgements

This work was performed as part of the grant funding Science Committee of the Ministry of Education and Science of the Republic of Kazakhstan on the topic "Development of intellectual high-performance information system for analysis of oil production technologies «iFields-II»".

References

[1] Wolff D., *OpenGL 4.0 Shading Language Cookbook* (Birmingham: Packt Publishing Ltd., 2011), 63-71.

[2] Sellers G., Wright R.S. Jr., Haemel N., *OpenGL SuperBible: Comprehensive Tutorial and Reference (7th Edition).* (Addison-Wesley, 2015) 848.

[3] Dave Shreiner et al., *OpenGL programming guide : the official guide to learning OpenGL, version 4.3.* (Addison-Wesley, 2013), 101-112.

[4] Adler, Daniel G, Oleg Nenadic and Walter Zucchini., "RGL : A R-library for 3D visualization with OpenGL." (2003).

[5] Mullen T. et al., "Real-time modeling and 3D visualization of source dynamics and connectivity using wearable EEG", *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Osaka, 2184-2187, 2013.

[6] Badretdinov M.R., Badretdinov T.R., Borshyuk M.S., "Primeneniye biblioteki opengl dlya vizualizatsii rezul'tatov chislennogo matematicheskogo modelirovaniya na setkakh bol'shoy razmernosti [Application the opengl library to visualize the results of numerical mathematical modeling on high-dimensional grids]", *Herald UGATU*, no 4 (2015): 84-94.

[7] Abraham F., Celes W., "Distributed Visualization of Complex Black Oil Reservoir Models" *Eurographics Symposium on Parallel Graphics and Visualization (2009)*, EGPGV (2009), Munich, Germany, 87–94.

[8] "Vulkan." Khronos Group, accessed November 13, 2017, https://www.khronos.org/vulkan/.

[9] "Vulkan." Nvidia developer, accessed December 9, 2017, https://developer.nvidia.com/Vulkan/.

[10] Sellers G., *Vulkan. Developer's Guide. Official guide*, trans. A. B. Boreskova (DMK Press, 2017), 394.

[11] Pawel Lapinski, *Vulkan Coockbook. Work through recipes to unlock the full potential of the next generation graphics API—Vulkan* (Birmingham: Packt Publishing Ltd., 2017), 151-169.

[12] "SPIR Overview." Khronos Group, accessed March 3, 2018, https://www.khronos.org/spir/.

[13] "C++ Abstraction library for Vulkan API." L. O. Tolo., accessed March 3, 2018, https://github.com/ larso0/bp.

[14] "Vulkan Memory Allocator." Advanced Micro Devices Inc., accessed February 26, 2018, https://github.com/GPUOpenLibrariesAndSDKs/VulkanMemoryAllocator.

[15] Khronos Vulkan Working Group, *Vulkan 1.0.98 - A Specification (with KHR extensions). 1.0.98. Jan. 2019.* (Khronos Group, 2019), 57-74.

[16] Richard H. Carver, Kuo-Chung Tai., *Modern multithreading: implementing, testing, and debugging multithreaded Java and C++/Pthreads/Win32 programs.* (USA: WILEY-INTERCIENCE, 2006), 112-124.

[17] Verzhbizckiy V.M., *"Chislennyye metody (matematicheskiy analiz i obyknovennyye differentsial'nyye uravneniya) [Numerical methods (mathematical analysis and ordinary differential equations)]"*, (Moscow: High school, 2001), 98-105.

[18] Samarskiy A.A., Nickolayev Ye.S., *"Metody resheniya setochnykh uravneniy [Methods of grid equations solving]"* (Moscow: The science, 1987), 130.

[19] Schlumberger, *Eclipse reference manual: technical description* (2002) 2683.

[20] Loix T. et al., "Layout and performance of the power electronic converter platform for the VSYNC project", *2009 IEEE Bucharest PowerTech* (2009): 1-8.

[21] Parminder Singh, *Learning Vulkan* (Birmingham: Packt Publishing Ltd., 2016), 10-12.

[22] Karen L. et al. , "Visualization in Science Education" , *Alberta Science Education Journal*, vol. 41, no 1 (2011): 22-30.

[23] John K. Gilbert, Miriam Reiner, Mary Nakhleh , *Visualization: Theory and Practice in Science Education* (Springer, 2008), 205-244.

[24] Linda M. Phillips, Stephen P. Norris, John S. Macnab , *Visualization in Mathematics, Reading and Science Education* (Springer, 2010), 65.

[25] Korakakis G. et al. , "3D visualization types in multimedia applications for science learning: A case study for 8th grade students in Greece" , *Computers and Education*, vol. 52, no 2 (2009): 390-401.