

НОВИЙ КУРС • WWW.NEWROUTE.ORG.UA
НАНМ УКРАЇНИ • WWW.NEWROUTE.ORG.UA/NANMU
ISCU «PROTON GLOBAL» • WWW.NEWROUTE.ORG.UA/PROTON



МОНОГРАФІЯ



СГ НТМ «Новий курс»

НОВИЙ КУРС • WWW.NEWROUTE.ORG.UA
НАНМ УКРАЇНИ • WWW.NEWROUTE.ORG.UA/NANMU
ISCU «PROTON GLOBAL» • WWW.NEWROUTE.ORG.UA/PROTON

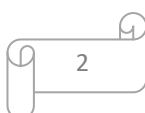


ISBN 978-617-7886-92-0
DOI: 10.61718/mon202601

ТЕОРЕТИКО-МЕТОДОЛОГІЧНІ ЗАСАДИ СУЧАСНИХ НАУКОВИХ ДОСЛІДЖЕНЬ

Монографія

Харків • СГ НТМ «Новий курс» • 2026



УДК 001:1
Т11

Теоретико-методологічні засади сучасних наукових досліджень: моногр. – Харків: СГ НТМ «Новий курс», 2026. – 299 с.

ISBN 978-617-7886-92-0
DOI: 10.61718/mon202601

Рецензенти

Штулер Ірина Юрійвна, доктор економічних наук, професор,
перший проректор ВНЗ «Національна академія управління»

Погоріла Світлана Григорівна, кандидат педагогічних наук,
доцент кафедри славістичної філології, педагогіки і методики викладання
Білоцерківського національного аграрного університету

Гетьман Ірина Анатоліївна, кандидат технічних наук, доцент,
доцент кафедри комп'ютерних інформаційних технологій
Донбаської державної машинобудівної академії

Харченко Артем Вікторович, кандидат історичних наук, доцент,
доцент кафедри мистецької освіти та гуманітарних дисциплін
Харківського національного університету мистецтв імені І. П. Котляревського

Рекомендовано до друку Вченою радою наукової установи Соціально-гуманітарна
науково-творча майстерня «Новий курс» (протокол № 1мн-2026 від 28.01.2026)

Видавець СГ НТМ «Новий курс» – діяльність у науковій, видавничій, освітній, творчій,
Інформаційній сфері з 1989 року. Свідectво про внесення суб'єкта видавничої справи до
Державного реєстру видавців, виготовлювачів і розповсюджувачів видавничої продукції
ДК № 8013 від 22.11.2023. Зареєстровано у Global Register of Publishers.
Зареєстровано у Crossref із власним префіксом 10.61718

Монографія буде корисною науковцям, викладачам, здобувачам освіти, а також широкому колу осіб, які
цікавляться питаннями розвитку сучасної науки та практики. Монографія оприлюднюється за результатами
проведення науково-практичної конференції «Трансформація світу: минуле, сьогодення, майбутнє». За
результатами проведення конференції та оприлюднення рукописів автори отримують електронні сертифікати.
Сертифікати оприлюднюються на сайті видавця (згідно Порядку підвищення кваліфікації педагогічних і науково-
педагогічних працівників, затвердженого постановою Кабінету Міністрів України від 21 серпня 2019 р. № 800).

© СГ НТМ «Новий курс», 2026
© Автори, 2026

Опубліковано на основі ліцензії Creative Commons Attribution License



НОВИЙ КУРС • WWW.NEWROUTE.ORG.UA
НАНМ УКРАЇНИ • WWW.NEWROUTE.ORG.UA/NANMU
ISCU «PROTON GLOBAL» • WWW.NEWROUTE.ORG.UA/PROTON



I науково-практична конференція (2026)
«Трансформація світу: минуле, сьогодення, майбутнє»

Україна, м. Харків – Румунія, м. Бухарест

26-28 січня 2026 року

ТЕМАТИКА КОНФЕРЕНЦІЇ

Освіта, культура, мистецтво
Спорт, фізична культура
Гуманітарні та соціальні науки, журналістика
Бізнес, послуги, адміністрування, право
Безпека та оборона
Інформаційні технології, інженерія
Виробництво, будівництво
Природничі науки, математика
Охорона здоров'я та соціальне забезпечення

ОРГАНІЗАТОРИ

Національна академія наук і мистецтв України
International Scientific-creative Unit «Proton Global»
Соціально-гуманітарна науково-творча майстерня «Новий курс»

www.newroute.org.ua

Зміст

	Стор.
Передмова	7
Розділ перший	
Освітні, культурно-мистецькі, спортивно-оздоровчі та соціальні аспекти	8
1.1. <i>Бедь Наталія Михайлівна</i> Ефективність занять атлетичною гімнастикою в системі фізичної реабілітації молодих чоловіків віком (20-25 років)	8
1.2. <i>Koryahin Viktor Maksymovych</i> Comprehensive analysis of the athlete training system	23
1.3. <i>Faichuk Olena, Surgova Svitlana</i> Training of social workers in the context of modern civilizational challenges: educational aspect	26
1.4. <i>Бовсунівська Наталія Миколаївна</i> Музичний всесвіт польського та єврейського етносів Волині у XIX – першій половині XX ст.	34
1.5. <i>Kukharska Liliia</i> Methodological Principles of Ensuring Educational Service Quality in Higher Education Institutions	45
1.6. <i>Рубський В'ячеслав Миколайович</i> Аналіз та інтерпретація отриманих результатів	51
Розділ другий	
Історико-філософська, гуманітарна, культурологічна, філологічна та літературознавча специфіка	65
2.1. <i>Прищепя Тетяна Валеріївна</i> Гармонія як семіотична модель міжкультурного діалогу в романі Германна Гессе «Сіддхартха»	65
2.2. <i>Ничкалюк Віталій Григорович</i> Діяльність академіка К. Г. Воблого в умовах інституційного становлення вітчизняної академічної науки в Україні (перша половина XX ст.): історичний вимір	71
2.3. <i>Хоменко Тетяна Анатоліївна</i> Давньогерманська сакральна лексика	87
2.4. <i>Крочак Аліна Вячеславівна, Заповівський Микола Володимирович</i> Лексичні засоби німецько-українського перекладу термінології	99
Розділ третій	
Особливості бізнесу, адміністрування, юриспруденції, міжнародні відносини та безпекові виміри	104
3.1. <i>Яковенко Роман Валерійович</i> Комунікації як складова людського потенціалу в системі управління якістю служби технічного сервісу	104
3.2. <i>Halyna Lukianova</i> Contemporary trends in the development of legal mechanisms for ensuring information security within the european legal order	119
3.3. <i>Гаркуша Сергій Анатолійович</i> Інтеграція автоматизованих систем бухгалтерського обліку у механізми прозорості та сталості економіки	128
3.4. <i>Ціватий Вячеслав Григорович</i> Геополітична шахівниця в Центральній Азії XXI століття: дипломатія та азійський вектор зовнішньої політики провідних держав світу (теоретико-методологічний, геостратегічний та міжнародно-регіональний дискурси)	138

3.5.	<i>Решетник Сергій Миколайович</i>		
	Безпека життєдіяльності у надзвичайних ситуаціях техногенного характеру	...	153
3.6.	<i>Ірина Босак</i>		
	Формування та використання системи інформаційного забезпечення в управлінні підприємством	...	164
3.7.	<i>Гринчишин Ярослав Миколайович, Західний Роман Вікторович</i>		
	Ринок нерухомості як сегмент фінансового ринку	...	169
3.8.	<i>Вишневецька Оксана Василівна</i>		
	Управління персоналом в умовах кризи: сучасні техніки та технології	...	179
3.9.	<i>Лебега Павло Валерійович</i>		
	Еколого-правовий статус громадян України	...	195
3.10.	<i>Квасницька Раїса Степанівна, Авраменко Олександр Володимирович, Слатенко Максим Олександрович</i>		
	Фінансові послуги поштових операторів на фінансовому ринку України	...	197
3.11.	<i>Стадник Мирослава Євгенівна</i>		
	Готельно-ресторанний бізнес: виклики та засоби їх подолання	...	213
Розділ четвертий			
	Виробничий потенціал, інженерія, сфера послуг та екологічна складова їх сталого розвитку	...	228
4.1.	<i>Зацерковна Роксоляна Станіславівна, Зацерковний Ростислав Григорович</i>		
	Аналітика великих даних у плануванні друкованої рекламної продукції	...	228
4.2.	<i>Габрель Микола Михайлович, Град Юрій Ігорович</i>		
	Феномен образності в архітектурі сакральних-меморіальних просторів	...	233
4.3.	<i>Захарій Грачов, Захарій Грачов</i>		
	Дослідження європейського досвіду реновації панельної житлової забудови	...	242
4.4.	<i>Габрель Микола Михайлович, Стовпак Олександр Володимирович</i>		
	Проблематика ревіталізації промислових об'єктів в Україні та методика їх вирішення, на прикладі заводу Сатурн в Тернополі	...	248
4.5.	<i>Romanyuk Oleksandr Nykyforovych, Zavalniuk Yevhen Kostiantynovych, Bobko Oleksiy Leonidovych, Novoseltsev Oleksandr Oleksandrovych, Stakhov Oleksiy Yaroslavovych</i>		
	Basic features of computer graphics system design	...	255
4.6.	<i>Myroslava Kadyliak</i>		
	Development of the concept for the design and structure of premium CHOCOZOO chocolate packaging	...	270
4.7.	<i>Савченко Ольга Михайлівна</i>		
	Інтеграція вимог охорони праці в сучасні поліграфічні виробництва	...	279
4.8.	<i>Романюк Олександр Никифорович, Лесько Олександр Йосипович, Ціхановська Олена Михайлівна</i>		
	Моделі дистанційного керування підприємствами	...	284
4.9.	<i>Савченко Ольга Михайлівна, Огірко Ігор Васильович</i>		
	Конкурентна диференціація цифрових методів термотрансферу	...	290
	Післямова	...	297

ОБИРАЙТЕ ПЕРШИХ!
ДОВІРЯЙТЕ СПРАВЖНІМ!
ЦІНУЙТЕ УНІКАЛЬНІСТЬ!

соціологів та інших фахівців. Не менш важливим є раннє залучення місцевої громади та тристороннє партнерство влади, бізнесу і громадського сектору, що підвищує легітимність проєктів і сприяє залученню фінансування. Водночас варто зберігати автентичність промислової архітектури, інтегруючи історичні елементи у нові функції, а також впроваджувати гнучке функціональне зонування для адаптації просторів до потреб міста. Активна інформаційна підтримка та промоція успіхів проєктів (через ЗМІ, культурні події, відкриті покази) сприяють позитивному сприйняттю оновлень та залученню інвесторів. Такі узагальнені рекомендації формують інтегрований підхід до ревіталізації колишніх промислових зон, який відповідає виявленим архітектурно-соціальним викликам.

1. Сич О. А. Ревіталізація як механізм забезпечення сталого розвитку міст // *Європейські перспективи*. 2016. Вип. 1. С. 27-33.
2. Левченко О. В., Бірментале В. В. Ревіталізація: історія, досвід і перспективи просторового розвитку // *Вісник студентського наукового товариства ДонНУ імені Василя Стуса*. 2020. Том 2, №12. (Опубліковано 25.11.2020).
3. Гнатюк Л. Р., Мельник М. В. Ревіталізація промислових об'єктів на прикладі м. Київ // *Теорія та практика дизайну*. 2019. Вип. 16. С. 52-67. DOI: 10.18372/2415-8151.16.14330.
4. Сеньковська Я. Т. Функціонально-планувальна реструктуризація територій промислових об'єктів міста (на прикладі м. Львова): дис. ... канд. архіт. наук. Львівська політехніка. 2017.
5. Стаття Пляменицька О. А., Гнатюк Л. Р., Гуменюк І. І. (МЕТОДИЧНІ ПІДХОДИ ДО РЕВІТАЛІЗАЦІЇ ТА РЕНОВАЦІЇ ІСТОРИЧНИХ БУДІВЕЛЬ (АНАЛІЗ ДОСВІДУ)).
6. Стаття Боднар Т., Ясінський М. (МОЖЛИВОСТІ РЕВІТАЛІЗАЦІЇ ПРОМИСЛОВИХ ТЕРИТОРІЙ МІСТ НА ПРИКЛАДІ ВУЛИЦІ ЗАМКОВОЇ У МІСТІ ЛЬВОВІ).
7. ДБН Б.2.2-12:2019 "Планування та забудова територій" (як базовий ДБН для планувальних рішень; він також відсилає до ДСП 173-96 у питаннях СЗЗ)
8. Миколаєнко Н. 2019 (Напів)ревіталізовані: 4 приклади переосмислення індустріальних просторів в Україні. <https://mistosite.org.ua/articles/napivrevitalizovani-4-priklady-pereosmyslennia-industrialnykh-prostoriv-v-ukraini>.
9. Перехрест О. Креативний Львів: як заводи і парки розвивають місто <https://www.village.com.ua/village/city/lviv/261163-creative-lviv>
10. Mark Williams., Zollverein: converting a coal mine 2024 <https://ww3.rics.org/uk/en/modus/built-environment/resilient-infrastructure/zollverein-converting-coal-mine.html>

UDC 004.92

Romanyuk Oleksandr Nykyforovych
Doctor of Technical Sciences, Professor
ORCID: 0000-0002-2245-3364
Vinnitsia National Technical University
Zavalniuk Yevhen Kostiantynovych
ORCID: 0009-0005-1202-4653
Vinnitsia National Technical University
Bobko Oleksiy Leonidovych
ORCID: 0009-0000-1753-279X,
Vinnitsia National Technical University
Novoseltsev Oleksandr Oleksandrovych
ORCID: 0009-0005-3152-1794
Vinnitsia National Technical University
Stakhov Oleksiy Yaroslavovych
Doctor of Philosophy, Senior Lecturer
ORCID: 0000-0002-4901-3211
Vinnitsia National Technical University

4.5. Basic features of computer graphics system design

The paper examines the basic requirements for building computer graphics software systems. The general requirements for 3D rendering systems are analyzed, as well as the requirements for parallelizing computational rendering processes, and the requirements for implementing texturing and shading subsystems. Keywords: computer graphics software systems, rendering, parallelization, texturing, shading.

Introduction. Computer graphics systems are a basic element of modern information technologies and are used in many application areas, including scientific activities, engineering design, computer modeling, medicine, education, digital art, film production, as well as in virtual and augmented reality environments. The increase in the complexity of graphic scenes, the volume of processed data, and the requirements for the reliability of visualization necessitate the formation of clearly defined and scientifically substantiated requirements for the creation of computer graphics systems. Such requirements relate to hardware architecture, software, algorithms for processing graphic information, as well as aspects of performance, scalability, and interaction with other computing platforms. The formulation of these requirements is the basis for the correct design and further development of the relevant systems.

Literature review. An analysis of current scientific publications shows that in recent years, computer graphics systems have been intensively researched in conditions of increased requirements for image realism, computational speed and processing flexibility. A significant part of the work is focused on optimizing real-time rendering [1], hardware ray tracing and the development of graphics pipelines for VR/AR [2-3], which leads to a revision of traditional architectural approaches.

Research on rendering methods demonstrates a shift away from exclusively rasterization approaches to hybrid schemes [4] using ray tracing, which in turn increases the requirements for computational accuracy, memory bandwidth and energy efficiency. Lighting and shading models, in particular the Gouraud and Phong methods, remain relevant as basic or simplified approaches and define the requirements for interpolation procedures, vector normalization and scene complexity control.

Special attention is paid to the issues of normalization [5-7] and numerical stability, since computational errors can accumulate in complex multi-pass scenes and negatively affect the final image quality. Adaptive triangulation [8-9] is important in geometric processing processes, which allows reducing computational costs if coordinated with lighting and texturing algorithms.

Texturing methods, including bump mapping, normal mapping, and parallax approaches [10], are considered an effective way to increase visual fidelity without complicating geometry, while at the same time creating increased demands on the fragment processing stage and shader programs. The introduction of machine learning methods [11-14] is also noted, which allow reducing the computational load, but require more resources and increased architectural adaptability of systems.

A summary of the analyzed sources shows that modern research is focused on forming holistic system requirements for computer graphics as integrated software and hardware solutions aimed at achieving a high level of rendering realism, stable performance, numerical reliability, and flexibility of the graphics pipeline.

The purpose of the work is to analyze and systematize the requirements for building computer graphics software systems.

1. Basic requirements for parallel computing in computer graphics systems. One of the basic requirements for creating modern computer graphics systems at the hardware level is to achieve high computing performance, which is directly related to the effective organization of parallel computing. Graphics tasks have a pronounced parallel nature, since most operations are performed independently for a large number of elements, such as vertices, geometric primitives, fragments or pixels. This determines the orientation of modern graphics systems to the use of specialized graphics processors, which contain hundreds or thousands of computing cores and are capable of performing the same type of operations on large data sets in parallel.

The requirement for a high level of parallelism extends to all stages of the rendering pipeline, starting with geometric processing and ending with final image post-processing. At the vertex calculation stage, coordinate transformations, vector normalization, calculation of auxiliary attributes, and data preparation for subsequent stages are performed in parallel. Each vertex of the scene can be processed separately, which allows scaling calculations with increasing geometric complexity of the models. The system must provide such a level of parallelism that an increase in the number of vertices does not cause a proportional increase in processing time.

The highest demands for parallel processing arise at the rasterization and fragment computation stages. This is where the largest amount of operations is performed, as each fragment or pixel [15] requires separate color calculations, lighting, texture sampling, and visibility checks. The graphics system must support massively parallel execution of fragment shaders, reducing dependencies between threads and avoiding conditional jumps that reduce the efficiency of SIMD architectures [16] of graphics processors. In this context, the requirements for parallelism directly affect the structure of shader programs and the way calculations are organized.

An essential aspect is scalability, i.e. the ability of the system to effectively use additional computing resources if they are available. This applies both to increasing the number of computing units within a single GPU and to using multiple graphics processors within a single system. In multiprocessor configurations, the requirements for parallelism are complemented by the need for balanced load distribution between GPUs, correct synchronization of calculations, and reduction of data exchange overhead. The system should be organized so that the performance gain when adding a new GPU is as close to linear as possible.

Parallel computing in computer graphics systems is closely related to the organization of data streams. Massive parallelism is possible only if memory is accessed efficiently and conflicts are minimized when accessing it. Therefore, the requirements for parallelism are complemented by the need for high memory bandwidth and a rational scheme for its use. Modern graphics processors are equipped with high-speed video memory such as GDDR6 or HBM, which allows servicing thousands of streams simultaneously. At the same time, software solutions must make maximum use of cache memory, reduce the number of irregular accesses, and avoid excessive copying of data between the central and graphics processors.

A separate group of requirements is formed by the need for effective parallel implementation of lighting and shading algorithms. Modern lighting models, even within the framework of classical approaches, contain a significant number of vector operations, normalizations and calculations of scalar products. The system must provide the ability to perform such operations in parallel without losing numerical stability and physical correctness of the results. This is especially relevant for methods in which lighting is calculated at the fragment level, where the number of parallel streams per frame can reach tens of millions.

The parallelism requirement also extends to image post-processing steps, such as filtering, tone mapping, color correction, and anti-aliasing. These operations are typically applied to each pixel independently, making them well-suited for parallel execution. The graphics system must provide the ability to implement these algorithms in the form of parallel computing cores optimized for processing large amounts of data in real time.

Another important requirement is determinism of computation. In complex graphics systems, especially in interactive and embedded environments, it is necessary to ensure predictable frame generation times. Excessive asynchrony or uncontrolled dependencies between threads can cause performance fluctuations and frame rate instability. Therefore, systems must support synchronization mechanisms and parallel task scheduling that combine high performance with time stability.

Modern graphics programming interfaces have a significant impact on the implementation of parallel computing requirements. Low-level APIs, such as Vulkan and DirectX 12, give developers direct control over resource allocation, command queue management, and thread synchronization. This allows for more complete use of the parallel architecture of the GPU, reduces driver overhead, and provides finer control over the execution of graphics and computing tasks. Support for such interfaces is a prerequisite for modern computer graphics systems focused on high performance and scalability.

The structure shown in the diagram (Fig. 1) illustrates the principles of implementing parallel computing at different stages of the rendering pipeline and their hardware support by modern graphics processors. On the left is shown the sequence of the main stages of rendering – from vertex processing and geometric assembly of primitives to rasterization, fragment processing and final image post-processing. Each of these stages is organized in such a way that the processing of individual elements (vertices, primitives, fragments or pixels) can be performed independently, which creates the prerequisites for massive parallelism. This is consistent with the requirement described in the text to distribute computations between a large number of threads of the same type without strict inter-thread dependencies.

Parallelization at Rendering Pipeline Stages

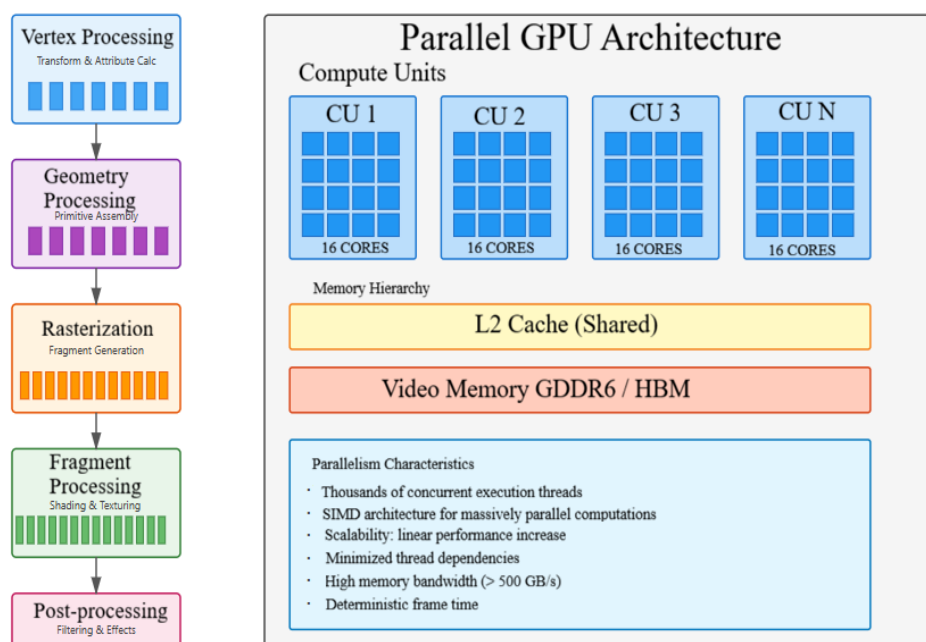


Fig. 1. Scheme of parallelization of calculations at different stages of the graphics pipeline

The right part of the diagram shows a generalized model of the parallel GPU architecture, which provides the implementation of such an approach. Compute units contain a set of simple cores that simultaneously execute instructions on different data within the SIMD model. It is these units that are mapped to vertex, geometric and fragment calculations, and the number of active threads can reach thousands or more. This organization of hardware resources meets the requirements of scalability, since an increase in the number of compute units allows us to increase performance without fundamentally changing the rendering algorithms.

The diagram also highlights the role of the memory hierarchy, including a shared second-level cache and high-speed video memory such as GDDR6 or HBM. They ensure efficient data exchange between parallel threads and minimize memory access latency during massive fragment and post-processing operations. This directly complements the requirements for high memory bandwidth and the need for rational organization of access to it in conditions of massive parallelism.

The diagram demonstrates that parallelism in modern computer graphics systems is not a local property of individual stages, but is inherent at all levels – from algorithms and shader programs to the organization of computational units and memory.

In general, the requirements for parallelism in computer graphics systems cover hardware architecture, memory organization, rendering algorithms, and the use of software interfaces. Efficient massively parallel execution of calculations is a necessary condition for achieving high image quality, real-time processing, and stable operation of systems with increasing scene complexity. It is the parallel organization of calculations that largely determines the potential of modern and promising graphics systems and their ability to meet the growing demands of computer graphics.

2. General requirements for visualization software systems . Requirements for visualization systems are not formed separately from specific methods, but as a generalized result of the need to maintain high rendering quality, optimal performance and flexibility of the graphics pipeline.

In modern graphics engines, the requirements for scalability of calculations, support for hardware ray tracing and the use of hybrid lighting models that combine rasterization and ray tracing are of particular importance. In this context, a computer graphics software system must ensure the coordinated operation of all stages of the graphics pipeline, taking into account the growing requirements for realism, interactivity and adaptability of visualization.

One of the basic requirements for visualization systems is the ability to process three-dimensional scenes with a large number of geometric primitives in a mode close to real time. This necessitates the need to support massively parallel computing and forms requirements for the architecture of graphics processors, which must effectively perform the same type of operations on large data sets. The system must be designed taking into account the balance between the geometric complexity of the scene and the speed of its processing, since excessive detailing without appropriate optimization mechanisms leads to a decrease in performance. Accordingly, the requirements for visualization systems include means of controlling the level of detail, adapting the quality of visualization and rational use of computing resources.

Rendering methods define the basic requirements for the organization of the graphics pipeline [17]. The system must support a multi-stage process of converting geometric data into a raster image, which includes coordinate transformations, clipping, projection, rasterization, and fragment processing. Each of the stages imposes its own limitations on computational accuracy, memory bandwidth, and execution latency.

Accordingly, modern rendering methods must ensure stable results regardless of the scale of the scene and the number of light sources, which is of particular importance for scientific visualization, architectural design and digital prototyping. On the other hand, this directly affects the requirements for numerical stability, memory bandwidth and the efficiency of implementing random samples in a parallel GPU environment.

Rendering methods with an increased level of realism form the requirements for the system from the standpoint of the accuracy of physical light models and the efficiency of their calculation. Physically correct rendering [18], which has become a de facto standard in modern graphics systems, requires the software platform to support complex material models, multiple light reflections and stochastic sampling methods. It is based on solving the light transport equation, which describes the propagation and interaction of light energy in a scene.

Along with physical correctness, there is a requirement for computational efficiency of rendering methods. Classical ray tracing algorithms provide a high level of visual realism, but their computational complexity limits their application in real-time tasks. Therefore, modern computer graphics systems must support optimized or approximation approaches that allow reducing the number of calculations without significantly degrading image quality. The efficiency requirement also includes the possibility of parallel execution of algorithms on the GPU and the use of specialized hardware blocks to accelerate ray tracing.

An important characteristic is the adaptability of rendering methods to different execution conditions. Graphics systems must dynamically change rendering parameters depending on the complexity of the scene, available computational resources and time constraints. Adaptive approaches allow us to reduce the number of rays, simplify lighting models or reduce the accuracy of calculations in less visible areas of the scene. This approach is especially relevant for interactive applications, where frame rate stability has a higher priority than maximum photorealism.

A key requirement for rendering methods is their compatibility with programmable shader pipelines. Modern graphics APIs allow for the implementation of custom lighting, shadow, and material algorithms using shader languages. This requires rendering methods to be formalized as algorithms that can be efficiently implemented in a parallel GPU environment. They should follow the concept of data streaming and minimize dependencies between computational stages that can limit parallelism.

One of the important requirements for modern rendering systems is to support hybrid methods [4], which combine rasterization and ray tracing. This approach allows rasterization to be used for basic geometric processing of the scene, and ray tracing to calculate complex lighting effects, reflections, and refractions. The use of hybrid rendering provides a compromise between performance and image quality, allowing the system to dynamically switch between methods or combine them depending on the importance of parts of the scene, the camera position, and the available computing resources.

Within such hybrid approaches, fast basic lighting can be implemented using the Phong method [19], which, due to its simplicity and computational efficiency, allows for rapid estimation of ambient, diffuse, and specular illumination of surfaces. The Phong method is well suited for processing a large number of geometric primitives in near-real time. For complex scenes where realistic optical effects need to be rendered, ray tracing is used as a complement to the Phong method.

Adaptive rendering techniques are used to improve efficiency: reducing detail for distant or peripheral objects, using different Levels of Detail, calculating lighting per-vertex or per-pixel depending on the importance of areas of the scene. In the case of ray tracing, this may include limiting the depth of tracing for non-priority areas, increasing the number of rays in key areas, selective noise filtering, and reusing the results of previous frames to reduce calculations.

Combining the adaptive Phong method with ray tracing within hybrid rendering provides:

- The optimal balance between quality and performance, allowing us to maintain a stable frame rate in real time.
- Scalability across platforms – from powerful workstations to mobile devices or interactive simulators.
- Rendering flexibility – dynamic switching between methods and algorithm combinations depending on the context.
- Efficient use of resources – adaptive regulation of calculations depending on the importance of objects and the characteristics of user perception.

The requirement for stability and predictability of rendering methods is important. For real-time systems, in particular in virtual and augmented reality, it is necessary to avoid sharp fluctuations in frame generation time. This requires deterministic behavior and controlled computational complexity from algorithms. Methods with high peak loads can negatively affect the user experience even with high average system performance.

Modern research is creating additional requirements for rendering methods due to the integration of neural networks.

Neural rendering [11-14] uses deep learning to approximate complex lighting models, restore details, or generate intermediate frames. This requires rendering methods to be compatible with neural network models and to support data exchange between traditional graphics algorithms and machine learning modules.

Consistency of results requires special attention, since neural approaches can be unstable or sensitive to changes in input data.

Additionally, this creates a need for hardware accelerators and close interaction between classical graphics pipelines and deep learning modules.

The requirement for scalability of rendering methods is the ability of algorithms to work effectively on both powerful desktop systems and limited mobile or embedded platforms.

This involves the development of multi-tier implementations, where the base layer provides minimal functionality, and advanced capabilities are activated when additional resources are available.

This approach allows us to use a single software framework for different classes of devices.

Rendering systems are also typically characterized by a set of structural requirements for their components.

Figure 2 shows a typical structural diagram of a rasterization-based rendering system.

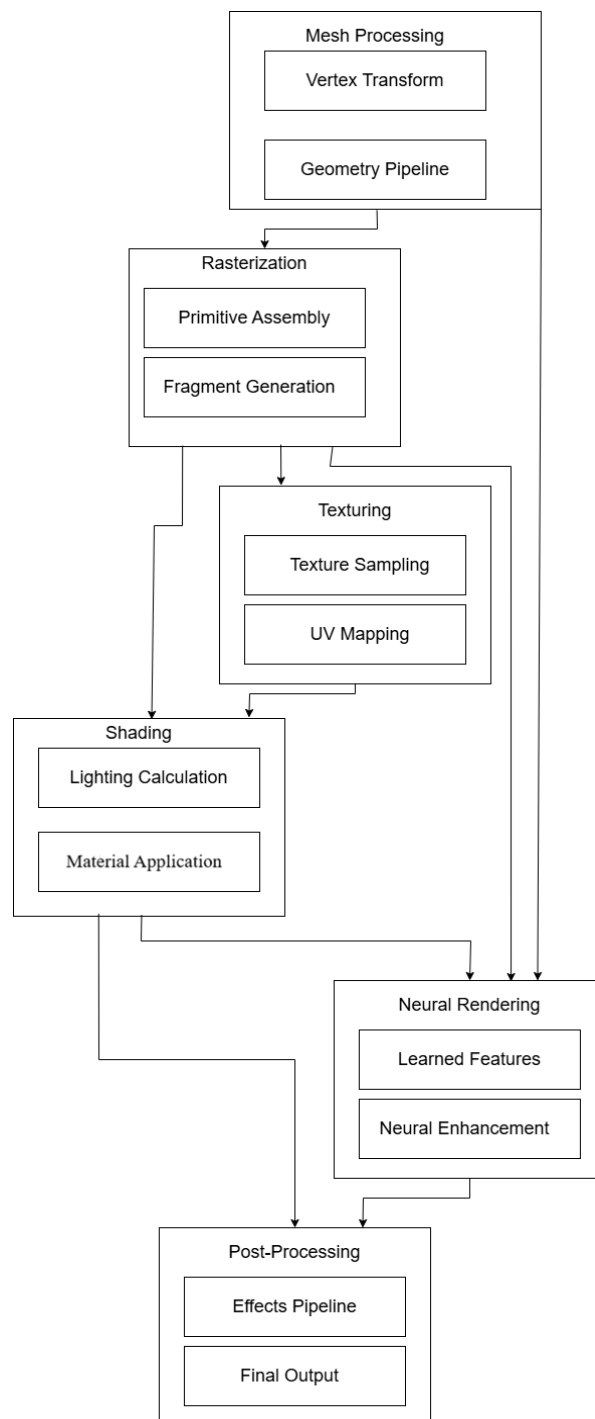


Fig. 2. General structural diagram of the visualization system

Such requirements are formed taking into account the specifics of graphics calculations, the stages of image formation, and the need for clear coordination between individual functional modules. The architecture of the rendering system must reflect the sequential movement of data along the graphics pipeline – from the initial geometric description of the scene to the formation of a completed image, ready for display or storage.

The initial structural link of the rendering system (see Fig. 2) is the polygonal model processing subsystem. [19]. The requirements for its organization are determined by the need for effective processing of geometric data that describe the shape and spatial arrangement of scene objects. The structure of this subsystem should ensure the correct storage and processing of vertex coordinates, normals, and topological relationships between mesh elements. Of particular importance is the vertex transformation stage, which brings the geometry to the coordinate systems required for further processing. From a structural point of view, this module should be clearly delimited, but at the same time integrated into a common geometric pipeline.

The geometric pipeline as part of the mesh processing subsystem should be organized in such a way as to ensure consistent preparation of primitives for the rasterization stage. Its structure should include the execution of coordinate normalization operations, clipping, and the formation of a set of attributes necessary for further discretization. An important structural requirement is the absence of procedures related to shading or texturing in this module, which allows maintaining a clear division of responsibility between the rendering stages.

The next mandatory component of the architecture is the rasterization subsystem. [19]. Its structural role is to transform geometric primitives into a set of discrete elements suitable for further pixel calculations. This subsystem includes the stage of primitive assembly, which forms triangles or other basic shapes from a set of vertices. The structural requirement is to ensure the consistency of this stage with the results of geometric processing and the correct order of subsequent operations.

The key element of rasterization is fragment generation, during which vertex attributes are interpolated within the primitive and a set of fragments is created, which are passed to the shading subsystem. The system architecture should provide for the possibility of performing fragment clipping and depth accounting operations at this stage, which allows reducing the load on subsequent modules. Such a structural organization helps to increase the overall system performance.

The texturing subsystem is a separate functional block, the requirements for the structure of which are determined by the need for the correct combination of geometric data with texture resources. It must be closely related to the results of rasterization, but maintain logical autonomy. The texturing structure includes modules for working with UV coordinates and sampling texture values. An important requirement is to ensure the correct mapping of textures to the surfaces of objects regardless of the complexity of their shape.

The texture sampling module should be structurally separated from the lighting calculations, since its purpose is to obtain the necessary data from texture maps. This approach allows us to change filtering methods, levels of detail, and texture storage formats without making changes to other parts of the system. In addition, the structure of the subsystem should take into account the intensity of memory access, which is characteristic of texturing operations.

The central place in the structure of the rendering system is occupied by the shading subsystem [19]. The requirements for its organization are determined by the need to calculate the color characteristics of fragments taking into account the lighting conditions and material properties. The structure of this module should provide for the reception of data generated at the previous stages and the formation of color values, which are subsequently used to create an image. An important requirement is a clear distinction between the calculation of lighting and auxiliary operations associated with the use of texture coefficients.

The lighting computation module should be designed to support different approaches to modeling the interaction of light with surfaces. This provides the system with flexibility and the ability to adapt to different requirements for rendering quality. At the same time, the structural consistency of the shading subsystem with other modules is a necessary condition for the stable operation of the entire graphics pipeline.

Modern rendering architectures are increasingly supplemented with neural components [11-14], which places additional requirements on the system structure. The neural module must be integrated in such a way as to have access to intermediate or final results of classical rendering. Its placement in the system structure may vary depending on the tasks, but the general requirement is consistency of data formats and the ability to share the results of calculations.

The component responsible for using the trained features needs to support additional information channels and intermediate representations. This means that the system structure must be flexible enough so that the integration of neural methods does not require a complete overhaul of the traditional pipeline. The neural image enhancement module must be logically separated from the classical post-processing algorithms, but at the same time compatible with them in terms of input and output data.

The final structural stage is the post-processing subsystem. The requirements for its organization are determined by the need to perform on-screen effects and form the final visual result. The structure of this module should allow flexible adjustment of the sequence of effects and expansion of their set without interfering with the operation of the

previous subsystems. The final output module should be separated from the main calculations and be responsible for preparing the image for display or storage.

Structural requirements for rendering systems are reduced to logical decomposition into interconnected but autonomous subsystems, clear organization of data flows, and the possibility of further development of the architecture. The structural scheme, covering all the main stages – from geometry processing to post-processing and neural modules – corresponds to modern approaches to building visualization systems and creates a basis for their further improvement.

In general, the requirements for visualization systems are formed as an integrated set of characteristics that arise from the support of modern methods of geometric processing, rendering, shading and texturing. The system must provide a balanced combination of image quality, performance, numerical stability and scalability, which is a necessary condition for its effective use in scientific, engineering and applied tasks. Analysis of classical and modern methods of computer graphics through the prism of requirements allows us to form a holistic idea of the principles of designing graphics systems, focused on further development and adaptation to new technological challenges.

3. Basic requirements for shading subsystems . Shading methods significantly influence the formation of requirements for visualization systems, since they determine the approach to calculating surface illumination and, despite the active development of physically correct and hybrid rendering, remain in demand in many application areas. The classical algorithms of Gouraud and Phong [19], despite their relatively simple algorithmic structure, continue to be widely used in real-time systems, embedded graphics environments, educational platforms, and industrial solutions. This necessitates the need to clearly define the requirements for their implementation.

Gouraud shading [19] is based on calculating the illumination at the vertices of a polygonal mesh followed by linear interpolation of color values over the polygon surface. This approach requires the visualization system to provide sufficient geometric detailing of the models, since the quality of the reproduction of chiaroscuro transitions directly depends on the number, spatial arrangement, and regularity of the vertices. The low density of the polygonal mesh leads to smoothing of local illumination maxima, in particular specular highlights, the appearance of striped transitions, and a decrease in the expressiveness of surfaces. In this regard, the system must support effective tessellation mechanisms, correct attribute interpolation, and matching the geometric complexity of the scene with the capabilities of the hardware.

Within the Gouraud method, specific requirements are also formed for the bidirectional reflectance distribution functions (BRDF) used. Since the illumination is determined only at the vertices, the BRDF models must be sufficiently smooth and have moderate nonlinearity in angular parameters. The use of sharply directed or energy-concentrated specular components leads to significant interpolation errors and loss of glare inside polygons. Therefore, in systems focused on Gouraud shading, it is advisable to use simplified empirical BRDFs, in particular the Lambertian diffuse model or smoothed versions of the Phong and Blinn–Phong models with limited reflection sharpness.

The Phong method [19], unlike the Gouraud approach, involves interpolation of normals within the polygon and calculation of illumination for each image fragment. This significantly increases the load on the computational subsystem of the graphics system, in particular on the performance of the GPU and the efficiency of the implementation of shader programs. One of the main requirements is to ensure high accuracy of normal interpolation and their correct normalization before use in lighting models. Even slight deviations of the normal length from a single value can cause noticeable artifacts, in particular distortions of the intensity of the diffuse and specular components.

The application of the Phong method expands the permissible range of bidirectional reflectance distribution functions, since the illumination is calculated at the fragment level. This allows the use of more complex and more directional BRDF models, in which the reflectance intensity significantly depends on the directions of light, view and surface normal. At the same time, such models impose increased requirements for numerical stability of calculations, accuracy of vector normalization and compliance with energy consistency. The system must ensure that the applied BRDFs do not violate the energy balance and do not cause uncontrolled increase in brightness during normal interpolation or changing material parameters.

In general, the Gouraud and Phong methods pose different but complementary requirements for shading systems. The Gouraud implementation is focused on geometric detail and simple reflection models, while the Phong method requires increased normalization accuracy, significant computational resources, and correct use of BRDFs. Taking these requirements into account is a necessary condition for building stable, efficient, and visually consistent rendering systems, especially in real-time environments and hybrid rendering pipelines.

In modern computer graphics systems, in particular within the framework of the Phong method, considerable attention is paid to the mathematical description of the interaction of light with the surfaces of objects, since it is on this

that the level of realism and visual reliability of images depends. One of the main tools for such a description is the bidirectional reflectance distribution function (BRDF) [20-24], which determines the dependence of the intensity of reflected radiation on the direction of light incidence and the direction of observation. The requirements for BRDF are formed taking into account both the physical laws of light propagation and the practical limitations associated with the implementation of rendering algorithms in software and hardware environments.

First of all, the BRDF must comply with the physical laws that describe the processes of light reflection in the real world. In particular, the function must satisfy the energy conservation condition, according to which the total amount of reflected energy does not exceed the amount of incident radiation. Violation of this condition leads to distortion of the visualization results and the appearance of unrealistic lighting effects. It is also essential to adhere to the principle of reciprocity, according to which the value of the BRDF remains unchanged when the directions of incidence and observation are mutually interchanged, which ensures the consistency of the lighting results regardless of the chosen calculation method.

To ensure visual quality of images, the BRDF must be continuous and smooth throughout the entire permissible range of angles. Abrupt changes in the function values can cause artifacts, noise, or unwanted brightness transitions on the surfaces of objects. This is especially evident when modeling materials with a complex microstructure, where the nature of light reflection significantly depends on the orientation of the surface and the direction of illumination.

An important requirement is the universality of the BRDF, which consists in the ability to describe a wide range of materials within the framework of a single mathematical model. This is achieved by using parametric functions, the values of which can vary depending on the physical properties of the material, such as surface roughness, reflection coefficient or refractive index. This approach allows us to use a single model for different types of surfaces and simplifies the process of setting up materials in graphics systems.

A separate place among the requirements for BRDF is occupied by computational efficiency. In rendering tasks, especially in real-time systems, lighting calculations are performed repeatedly for each pixel or fragment of the image. Therefore, the mathematical apparatus of BRDF should be relatively simple and well adapted to optimization and parallel calculations on graphics processors. Even in offline visualization systems, where accuracy is a priority, excessive complexity of the function can lead to a significant increase in rendering time, which reduces the practical feasibility of its application.

In the formulas of bidirectional light reflection functions, the greatest computational complexity is usually represented by the components associated with modeling the microgeometry of the surface and the angular dependence of reflection. These include, first of all, microfacet normal distributions, which are often described by exponential or fractional-rational functions and require the calculation of powers, exponent expressions, or complex fractions. Significant costs of computational resources are also associated with geometric shading and masking terms, which take into account the mutual overlap of micro-inequalities and usually contain conditional transitions, roots, or divisions. The combination of these components significantly complicates the calculation of BRDF, especially in real-time problems, which necessitates the use of simplified models or analytical approximations.

The BRDF should also be suitable for use in global illumination algorithms, in particular in ray tracing methods and stochastic approaches. To do this, the function should be correctly defined for all possible reflection directions and convenient for constructing probability distributions that are used when sampling directions in Monte Carlo methods. This helps to increase the convergence of the algorithms and reduce the noise level in the final image.

Equally important is the predictability of the behavior of the BRDF when changing its parameters. Small adjustments to the material characteristics should cause smooth and logical changes in the visual result, which facilitates the process of artistic and technical tuning of the scene. This property is also useful in the context of automated optimization methods and the application of machine learning algorithms in computer graphics tasks.

From a practical point of view, the BRDF should be compatible with common standards and approaches used in modern graphics engines and software libraries. The focus on physically based rendering requires the use of models that are easily integrated into standard visualization pipelines and provide consistent results on different hardware platforms. Taken together, compliance with these requirements allows for the creation of efficient and reliable light reflection models, which is a necessary basis for building modern computer graphics systems with a high level of realism and performance.

Fig. 3 shows an example of a block diagram of a module for calculating the pixel color intensity based on the well-known Blinn-Phong BRDF [20-24]. This module is the central element of the architecture of the shading subsystem based on the Phong method.

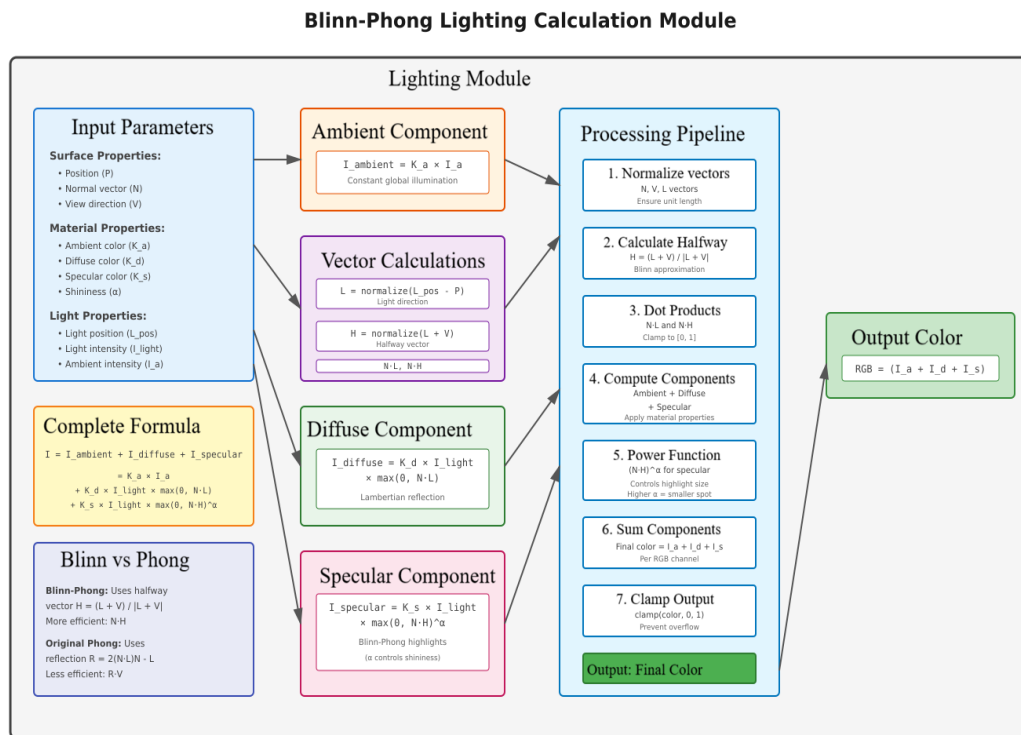


Fig. 3. Block diagram of the module for determining the pixel color intensity within the shading subsystem based on the Blinn-Phong BRDF

The lighting calculation module, built on the basis of the Blinn-Phong approach, has a clearly structured internal organization, focused on the phased formation of the color characteristics of the fragment. Its architecture reflects the sequence of logical operations necessary to take into account the geometry of the scene, the properties of materials and the parameters of light sources in order to obtain a visually correct result. Each element of the structure performs a separate function, while remaining integrated into the overall lighting calculation process.

The initial component of the module is the input parameters block, which accumulates all the information necessary for further processing. It contains data characterizing the position and orientation of the surface in space, the direction of observation, and also describes the optical properties of the material. The parameters of the light source are taken into account separately, in particular its spatial location and radiation intensity. The structural role of this block is to form a single consistent data set, which serves as the basis for all subsequent stages of the calculation.

The next element of the structure is the component responsible for taking into account the general background lighting. It provides a basic level of brightness that does not depend on the specific direction of the light flux or the orientation of the surface. Within the module, this component plays a supporting but important role, since it prevents the complete loss of visibility of objects in poorly lit areas of the scene. Its result is further combined with other lighting components.

An important structural unit of the module is the vector data preparation and analysis block. Its purpose is to form directions that describe the mutual location of the surface, light source and camera. In this block, vectors are brought to a consistent form, which allows for correct comparison of their orientations. Such a structural organization simplifies further calculations and ensures the stability of the results regardless of the spatial parameters of the scene.

After completing the vector preparation, the module proceeds to the formation of lighting components related to light scattering. The corresponding structure block estimates the degree of influence of the light source on the surface, taking into account its orientation and material characteristics. It is this component that determines the main visual appearance of the object, its volume and perception of shape. Structurally, this block is closely related to material and light data, but remains independent of other types of lighting effects.

A separate position in the module architecture is occupied by the component responsible for the formation of specular reflections. Its functioning is based on the analysis of the directions of observation and illumination and the

surface parameters that determine the nature of the shine. The structure of this block allows us to model different types of surfaces - from matte to highly reflective - by adjusting the corresponding parameters. Due to this, greater expressiveness of the image is achieved without significantly complicating the computational process.

Coordination of the work of all computational components is carried out using a processing pipeline, which determines the order of operations within the module. Its structure involves step-by-step data processing, starting with the normalization of input values and ending with the combination of the results of individual components. The pipeline also implements mechanisms for controlling permissible values, which allows us to avoid incorrect or unstable results. This approach increases the reliability of the module and ensures predictable behavior under different lighting conditions.

The final element of the structure is the final color formation block. At this stage, the results of calculating the background, diffuse and specular contributions are combined into a single color representation. The structure of this block involves the final correction of values and preparation of data for transmission to subsequent rendering stages or directly to the output system. The resulting result is complete from the point of view of the lighting module and can be used without additional processing.

The structure of the lighting module according to the Blinn-Phong model is characterized by a sequential, hierarchically ordered organization, in which each block performs a specific function and interacts with others through clearly defined interfaces. This approach ensures the clarity of the module's logic, simplifies its inclusion in the overall architecture of the rendering system, and creates the prerequisites for further adaptation or expansion without violating the integrity of the entire system.

Vector normalization [5-7] forms a separate group of requirements for shading subsystems, as it directly affects the correctness of lighting models and the stability of the visual result. The rendering system must ensure the unit length of surface normals, light directions, view vectors and reflected rays before using them in lighting calculations. Violation of this condition leads to distortion of scalar products, errors in calculating the intensity of lighting and the appearance of visual artifacts.

Fig. 4 shows an example of a block diagram of the vector normalization module of the shading subsystem.

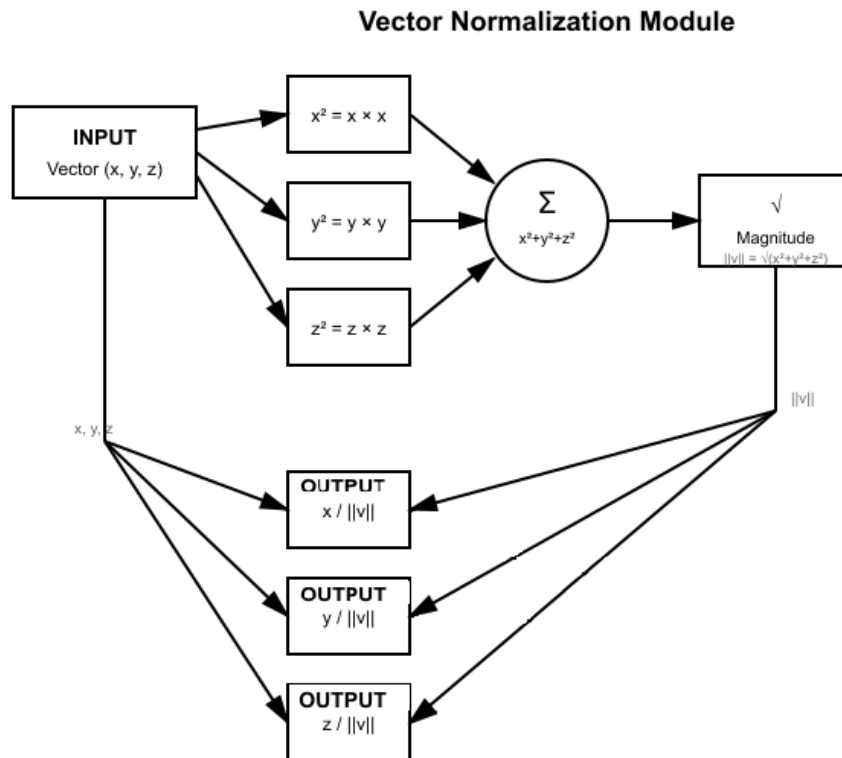


Fig. 4. Block diagram of the vector normalization module within the shading subsystem

The given module is intended for normalization of a three-dimensional vector and has a simple sequential data processing structure. At the input, the module receives the vector coordinates, which are used in parallel to calculate their squares. Then these values are aggregated to determine the length of the vector, after which the operation of reducing each coordinate to a unit length by scaling is performed. At the output, a normalized vector is formed, the direction of which coincides with the initial one, and the length is equal to one. Such a structure ensures the correct preparation of vector data for further use in lighting calculations and geometric transformations.

The primary requirement for normalization algorithms is numerical robustness. With millions of floating-point operations performed during rendering, normalization methods must minimize the accumulation of rounding errors, avoid division by zero or under-valued values, and ensure reproducibility of results across hardware platforms. This is especially true for normalizing interpolated normals in shaders, where even small errors can cause noticeable lighting artifacts.

Another important requirement is the geometric and physical correctness of the normalization. The normalization process must preserve the vector direction and its physical content. For normals, this means maintaining orthogonality to the surface, and for illumination and view vectors, compliance with the adopted photometric or radiometric model. Normalization must not result in the loss of information necessary for subsequent stages of the graphics pipeline.

It is essential that normalization methods are adaptable to rendering conditions. Modern scenes feature a wide range of scales, levels of detail, and dynamic lighting characteristics. Therefore, normalization algorithms must work correctly in both low-contrast scenes and high dynamic range conditions, ensuring visual stability and preventing loss of detail or oversaturation.

Equally important is computational efficiency. Vector normalization is performed massively for vertices and fragments, so the corresponding algorithms must be adapted to parallel execution on the GPU. The system must support hardware-accelerated normalization or efficient approximation methods with controlled error, which allow achieving a balance between accuracy and performance and reducing the number of operations and memory accesses.

A separate group of requirements is the consistency of normalization between different stages of the rendering pipeline. The methods applied at the vertex level must be compatible with the fragment processing and post-processing algorithms. Violation of this consistency can cause lighting instability, visual discontinuities, or incorrect interaction with shadows and reflections.

In the context of modern and promising rendering systems, adaptive vector normalization methods are of particular importance, which involve dynamic changes in the accuracy of calculations and the normalization method depending on the geometric, screen, or time context of the scene. A promising direction is context-dependent geometric normalization, in which normals are processed taking into account the local curvature of the surface and the density of the polygonal mesh. In areas with high curvature, more accurate normalization is used with a full calculation of the vector length, while approximation methods with reduced computational complexity are allowed in almost flat areas. This approach reduces lighting noise, prevents the accumulation of numerical errors, and increases the stability of the visual result, which is especially relevant when using normal mapping and parallax texturing methods.

The use of adaptive and context-aware normalization also helps improve temporal stability of the image in dynamic scenes, where camera movement, geometry deformation, or lighting animation can cause noticeable intensity fluctuations. Matching the accuracy of the normalization to the local geometric properties of the surface provides smoother interpolation of normals in space and time, reduces flicker, and improves the perception of fine details without significantly increasing the computational load.

In conclusion, the main requirements for vector normalization in rendering systems are to ensure numerical stability, physical correctness, adaptability, high performance, and consistency with the entire graphics pipeline. Meeting these conditions is a necessary prerequisite for creating reliable, scalable, and visually stable computer graphics systems.

An important aspect of building visualization systems is also the requirements for adaptive triangulation of geometric models [8-9]. Since graphics processors are optimized for processing triangles, the system must provide an efficient representation of arbitrary surfaces in the form of a triangular mesh. Adaptive triangulation involves changing the density of the polygonal mesh depending on the complexity of the geometry, the position of the camera and the requirements for the quality of visualization. This allows us to reduce computational costs without significantly degrading the image quality, but requires close integration of the geometric stage with other components of the graphics pipeline.

From the perspective of shading methods, adaptive triangulation directly affects the quality of lighting. For the Gouraud method, an insufficient number of triangles can lead to a loss of visual information, while the Phong method is less sensitive to a decrease in mesh density, but at the same time requires significantly more computational resources. This creates a requirement for the system to ensure a coordinated interaction between triangulation algorithms and the choice of shading method in order to achieve an optimal ratio between quality and performance.

Separately, the requirement of temporal stability of adaptive triangulation is put forward. In dynamic scenes, frequent restructuring of the triangular grid can cause flickering or sudden changes in lighting, which negatively affects

perception. Therefore, adaptive triangulation algorithms must be deterministic and provide a smooth change in the level of detail, which is especially relevant for interactive systems and virtual reality environments.

In summary, the requirements for Gouraud and Phong shading methods, BRDFs, vector normalization, and adaptive triangulation form an important layer of general requirements for building computer graphics systems. They reflect the close relationship between the geometric complexity of scenes, the mathematical accuracy of calculations, and the computational efficiency of the graphics pipeline, ensuring the creation of flexible, stable, and productive graphics systems for a wide range of applications.

4. Basic requirements for texturing subsystems. The texturing subsystem occupies an important place in modern visualization systems, since it largely determines the degree of detail, realism and informativeness of images formed during rendering. Its functional purpose is to apply two-dimensional or multidimensional textures to the surface of three-dimensional objects in order to reproduce their physical and optical characteristics, in particular color, roughness, transparency, microrelief and other material parameters. In this regard, a set of interrelated requirements is imposed on the texturing subsystem, covering image quality, performance, scalability, flexibility and rational use of computing and memory resources.

One of the basic requirements is the correctness of texture mapping, which implies the consistency of the object geometry with texture coordinates. The subsystem must provide reliable mechanisms for parameterizing surfaces, including support for classic UV texturing, multiple sets of UV coordinates, as well as alternative approaches such as projective, triplanar, and volumetric texturing. This allows us to avoid distortions, gaps, and unwanted repetitions of textures, especially on models with complex topology. Support for automated and semi-automated texture coordinate generation algorithms is also important, which simplifies work with large scenes and procedurally generated geometry.

An essential requirement is to ensure high quality texture rendering, which directly affects the perception of the final image. The subsystem must guarantee correct interpolation of texture coordinates during rasterization, reducing numerical errors and visual artifacts. This requires support for modern filtering methods, in particular, nearest, bilinear, trilinear, and anisotropic [25], which allows maintaining texture clarity at oblique viewing angles. The system must be able to adapt filtering parameters depending on the distance to the object, scale, and hardware capabilities.

A separate block of requirements is related to the use of mip levels and hierarchical texture storage structures. Mipmapping helps reduce aliasing and flickering when changing the scale and reduces the load on the memory subsystem. The texturing subsystem should support automatic generation of mip levels, correct selection of the level of detail and smooth transitions between them without noticeable visual jumps. It is also advisable to implement adaptive or procedural approaches to detail control taking into account the characteristics of the scene and camera movement.

One of the most important requirements is efficient memory management, since texture data usually occupies a significant portion of video memory. The system must provide fast access to textures, support caching, asynchronous loading and unloading of resources, as well as optimal placement of data in the GPU memory. It is important to support various texture formats, including compressed ones, which allows us to reduce the amount of memory without significant loss of image quality.

The main requirements include support for multi-layer textures and combining different texture maps. Modern systems use not only color textures, but also normal, height, roughness, metallicity, transparency and other auxiliary data maps. The texturing subsystem must ensure their consistent use in shader programs and support for physically based material models, which allows achieving a high level of realism without significantly complicating the geometry.

Increased requirements for the texturing subsystem arise when using relief texturing methods [10], which allow simulating complex surface microgeometry without a real increase in the number of polygons. Such approaches significantly improve the visual result, but at the same time complicate the processes of sampling and processing texture data at the fragment stage of rendering, which imposes additional requirements for the accuracy of calculations, performance, and consistency of work with other components of the graphics pipeline.

The application of bump mapping requires correct processing of height maps and accurate calculation of surface derivatives, on the basis of which the perturbed normals are formed. Since such maps usually have limited bit depth, the system must be resistant to numerical errors and provide smoothing of sharp gradient changes that can cause noise or lighting artifacts. Stable normalization of perturbed normals after interpolation and consistency of relief scales with the parameters of the lighting model, as well as optimized access to height maps, are important.

Normal mapping, unlike bump mapping, directly uses normal maps that encode the orientation of the surface in tangent space. This creates requirements for correct work with tangent bases, in particular for storing and interpolating tangent, binormal, and normal direction vectors. The subsystem must ensure consistency between geometry, texture coordinates, and normal maps, as well as correct transformation of normals from texture space to scene or lighting space. Support for different normal map formats and correct handling of edge cases associated with specular UV scans or abrupt changes in surface orientation are important.

Parallax methods of relief texturing, in particular parallax mapping, steep parallax mapping and parallax occlusion mapping, form an even more complex set of requirements. They are based on the iterative displacement of

texture coordinates depending on the viewing direction and heightmap values, which leads to multiple sampling of texture data for each fragment. In this context, the texturing subsystem must provide high throughput of access to textures, minimize sampling delays and effectively use caching. It is important to be able to adaptively control the complexity of calculations, for example, by dynamically changing the number of iterations depending on the distance to the camera, viewing angle and available resources. The system must also be resistant to the appearance of artifacts on the silhouettes of objects and correctly combine parallax methods with mipmapping and anisotropic filtering.

In general, support for bump mapping, normal mapping, and parallax relief texturing methods places increased demands on the texturing subsystem in terms of computational accuracy, data access efficiency, memory management, and integration with lighting and shader programming subsystems. The implementation of these approaches must be accompanied by adaptive rendering mechanisms that allow maintaining a balance between visual quality and performance in real-time conditions.

It is important to tightly integrate the texturing subsystem with the lighting and shader programming subsystems. It should provide flexible means of accessing texture data in vertex, geometric, and fragment shaders, ensuring high performance and scalability. It is necessary to support modern graphics APIs and standards, which allows us to effectively use the hardware capabilities of various graphics processors, in particular in physically based lighting models such as PBR, where the role of texture data is crucial.

Special attention is required for procedural and dynamic texturing requirements. The subsystem should support real-time texture generation based on mathematical models, noise, fractals, or physical simulations. This allows for reduced static resource usage and increased variability in visual effects, which is important for modeling dynamic surfaces such as water, fire, smoke, or material degradation processes.

Performance, scalability, and reliability are among the most important requirements for a texturing subsystem, especially in interactive applications and virtual and augmented reality systems. The subsystem must effectively use parallel computing and hardware acceleration, adapt to different hardware configurations, and correctly handle errors associated with the absence or corruption of texture resources. In addition, it must be convenient for extension and modification, which is important for educational, scientific, and experimental systems.

Fig. 5 shows an example of a block diagram of a possible texturing subsystem.

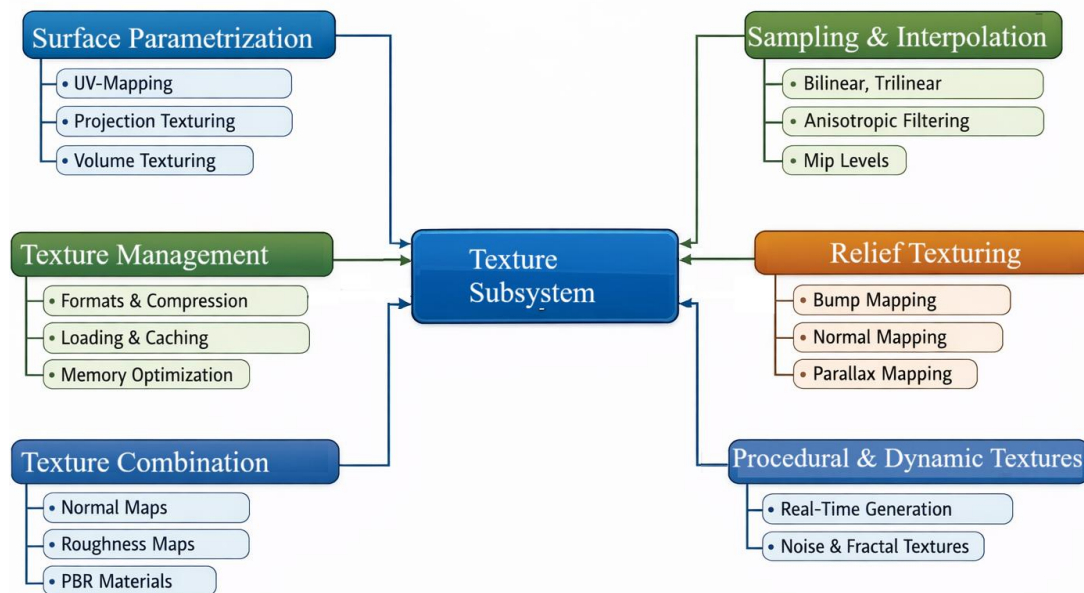


Fig. 5. Structural diagram of the texturing subsystem

The surface parameterization block (see Fig. 5) is responsible for establishing a correspondence between the geometric elements of objects and the texture space. This block uses UV mapping, projection, and volume texturing methods, which allows us to correctly apply textures to surfaces of arbitrary shape.

The texture management module handles texture resources at the storage and usage stages. It includes support for various compression formats and methods, loading and caching mechanisms, and memory optimization, which helps improve the efficiency of the rendering system.

The sampling and interpolation functions are implemented in the corresponding module, which defines the method of obtaining texture values during image formation. It uses bilinear and trilinear interpolation, mip-levels, and anisotropic filtering, which allows to reduce distortion and ensure stable image quality under different display conditions.

The relief texturing module is designed to simulate small geometric surface details without increasing the number of polygons. For this, bump mapping, normal mapping, and parallax mapping methods are used, which affect the lighting and perception of the object's shape.

Texture blending allows us to use multiple texture maps simultaneously, including normal maps, roughness maps, and physically correct material parameters. This allows us to create complex materials with realistic visual properties.

The procedural and dynamic textures module focuses on creating textures directly during rendering. It includes algorithms for generating noise and fractal structures in real time, which increases the flexibility of the system and reduces the need to use pre-prepared texture images.

In general, the requirements for the texturing subsystem are reduced to ensuring high quality visualization, rational use of resources, flexible work with different types of textures and materials, and close interaction with other components of the graphics system, which is a necessary condition for creating modern and productive computer graphics systems.

Conclusions. Summarizing the above, it should be noted that modern computer graphics systems are formed on the basis of a comprehensive approach that takes into account the hardware, algorithmic and software aspects of the graphics pipeline. Effective parallelization of calculations, accurate implementation of shading methods, stable vector normalization, adaptive triangulation and a developed texturing subsystem create an interdependent structure that determines the performance, scalability and quality of images. Coordination of these components ensures realistic reproduction of complex scenes in real time, support for numerical and temporal stability of rendering and efficient use of computational and memory resources. Fulfillment of these requirements creates the prerequisites for the development of flexible and reliable visualization systems that can adapt to the growing complexity of graphics tasks and various areas of practical application.

1. Akenine-Möller T., Haines E., Hoffman N. Real-Time Rendering. 4th ed. Boca Raton: Taylor & Francis, 2018. 1198 p.
2. He C., Sun C. Research on real-time graphics rendering and interaction optimization strategies in virtual reality. *Applied Mathematics and Nonlinear Sciences*. 2024. Vol. 9, no. 1. P. 1-22.
3. Hu X., Wu X., Ma M., Xu X., Gu Y., Wang G., Xu Y., Meng X., Wang L. Efficient VR rendering: Survey on foveated, stereo, cloud, and low-power rendering techniques. *Virtual Reality & Intelligent Hardware*. 2025. Vol. 7, no. 5. P. 421-452.
4. Li K., Wang G., Zhao H. Design and implementation of heuristic hybrid rendering. *Proc. SPIE 13105, International Conference on Computer Graphics and Data Processing (ICCAID 2023)*, 03 Mar. 2024, SPIE – Society of Photo-Optical Instrumentation Engineers. 2024. Article 1310505.
5. Cigolle Z.H., Donow S., Evangelakos D., Mara M., McGuire M., Meyer Q. A survey of efficient representations for independent unit vectors. *Journal of Computer Graphics Techniques*. 2014. Vol. 3, no. 2. P. 1-30.
6. Romanyuk O., Zavalniuk Y., Mykhaylov P., Chekhmestruk R., Titova N., Romanyuk S. Accelerated Vector Normalization for Rendering Tasks. *14th International Conference on Advanced Computer Information Technologies (ACIT)*, České Budějovice, Czech Republic, 2024. P. 640-644.
7. Romanyuk O. N., Zavalniuk Y. K., Bobko O. L., Arsenyuk I. R., Romanyuk O. V., Sachaniuk-Kavets'ka N. V., Nykyforova L. E., Kotyra A., Kozbakova A., Kalizhanova A. Method of combined vector normalization of 3-D objects. *Proc. of SPIE "Photonics Applications in Astronomy, Communications, Industry, and High Energy Physics Experiments 2024"*, Lublin, Poland, 27–29 June 2024. Vol. 13400. 1340000 – 1-8.
8. Romanyuk O., Zavalniuk Y. Deep Learning-Based Determination of Optimal Triangles Number of Graphic Objects Polygonal Model. *5th International Workshop on Intelligent Information Technologies and Systems of Information Security*, Khmelnytskyi, 28 Mar. 2024. Vol. 3675. P. 39-51.
9. Wang M., Jing J., Gao S., Bian P., Ma Y., Zhou N. Improved adaptive tessellation rendering algorithm. *Technol. Health Care*. 2023. Vol. 31, Suppl. 1. P. 81-95.
10. Романюк О. Н., Майданюк В. П., Новосельцев О. О. Класифікація методів рельєфного текстурвання для високореалістичного рендерингу. *Наукові праці Вінницького національного технічного університету*. 2025. № 3. С. 1-11.
11. Weinreich C., De Oliveira L., Houdard A., Nader G. Real-Time Neural Materials Using Block-Compressed Features. *Computer Graphics Forum*. 2024. Vol. 43, no. 2. P. e15013.
12. Tewari A., Thies J., Mildenhall B., Srinivasan P., Treitsch E., Yifan W., Lassner C., Sitzmann V., Martin-Brualla R., Lombardi S., Simon T., Theobalt C., Nießner M., Barron J. T., Wetzstein G., Zollhöfer M., Golyanik V. Advances in Neural Rendering. *Computer Graphics Forum*. 2022. Vol. 41, no. 2. P. 703-735.
13. Mildenhall B., Srinivasan P., Tancik M., Barron J.T., Ramamoorthi R., Ng R. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. *Communications of the ACM*. 2021. Vol. 65, no. 1. P. 99-106.
14. Sztrajman A., Rainer G., Ritschel T., Weyrich T. Neural BRDF Representation and Importance Sampling. *Computer Graphics Forum*. 2021. Vol. 40, no. 6. P. 332-346.
15. Романюк О. Н., Майданюк В. П., Романюк О. В., Бобко О. Л., Новосельцев О. О. Розподілення обчислень і балансування навантаження у високодуктивних системах паралельного рендерингу. *Наукові праці Вінницького національного технічного університету*. 2025. № 3. С. 1-11.
16. Романюк О. Н., Бобко О. Л., Снігур А. В. Метричне оцінювання розпаралелення рендерингу. *Наукові праці Вінницького національного технічного університету*. 2025. № 2. С. 1-12.
17. Romanyuk O.N., Bobko O.L., Zavalniuk Y.K., Titova N.V., Romanyuk S.O., Stakhov O.Y. Analysis of graphics pipelines. *Innovation in der modernen Wissenschaft: Innovative Technology, Informatik, Sicherheitssysteme, Verkehrsentwicklung, Architektur und Bauwesen, Physik und Mathematik*. Buch 30. Teil 1. Chap. 4. ScientificWorld-NetAkhatAV, 2024. P. 71-80.
18. Pharr M., Jakob W., Humphreys G. Physically Based Rendering : From Theory to Implementation. 4th ed. Cambridge, MA : The MIT Press, 2023. 1312 p.
19. Романюк О. Н., Романюк О. В., Чехмestрук Р. Ю. Комп'ютерна графіка. Вінниця : ВНТУ, 2023. 147 с.
20. Romanyuk O., Zavalniuk Y., Romanyuk O., Kotlyk S., Bobko O., Melnyk O. New Illumination Model for Rendering. *15th International Conference on Advanced Computer Information Technologies (ACIT)*, Sibenik, Croatia, 2025. P. 723-727.
21. Romanyuk O., Lytvynenko V., Zavalniuk Y. Modified cosine-quadratic reflectance model. *Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska*. 2025. Vol. 15, no. 1. P. 45-48.
22. Romanyuk O., Zavalniuk Y., Titova N. V., Kaduk O., Wójcik W., Kalimoldayev M., Shermantayeva Z. Optical system visualization of combined reflectance model based on cubic and quadratic functions. *Proc. SPIE 12985, Optical Fibers and Their Applications 2023*, Lublin, Poland, 129850C (20 Dec. 2023). P. 129850 C. 1–129850. C-8.
23. Romanyuk O., Zavalniuk Y., Romanyuk O., Snigur A., Titova N., Maidaniuk V. The Development of Physically Correct Reflectance Model Based on Logarithm Function. *13th International Conference on Advanced Computer Information Technologies (ACIT)*, Wrocław, Poland, 2023. P. 483-487.
24. Romanyuk O., Zavalniuk Y., Pavlov S., Chekhmestruk R., Bondarenko Z., Koval T., Kalizhanova A., Iskakova A. New surface reflectance model with the combination of two cubic functions usage. *Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska*. 2023. Vol. 13, no. 3. P. 101-106.
25. Романюк О. Н., Дудник О. О. Анізотропна фільтрація з використанням текстурних карт вагових коефіцієнтів. *Математичні методи обробки даних*. 2017. Т. 19, № 3. С. 34-41.

НОВИЙ КУРС • WWW.NEWROUTE.ORG.UA
НАНМ УКРАЇНИ • WWW.NEWROUTE.ORG.UA/NANMU
ISCU «PROTON GLOBAL» • WWW.NEWROUTE.ORG.UA/PROTON

Наукове видання

**ТЕОРЕТИКО-МЕТОДОЛОГІЧНІ ЗАСАДИ
СУЧАСНИХ НАУКОВИХ ДОСЛІДЖЕНЬ**

Монографія

ISBN 978-617-7886-92-0

DOI: 10.61718/mon202601

В межах І науково-практичної конференції (2026)
«Трансформація світу: минуле, сьогодення, майбутнє»
Україна, м. Харків – Румунія, м. Бухарест
26-28 січня 2026 року

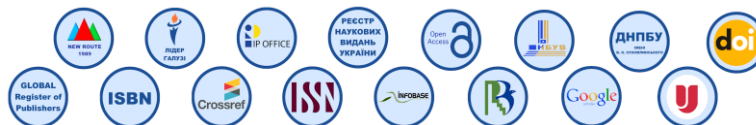
Видання змішаними мовами
Відповідальний секретар – Кучина Т. І.
Формат 60х90 1/8, А4, гарнітура «Times New Roman»
Авторські аркуші – 32,5

Creative Commons Attribution License CC BY

**ОБИРАЙТЕ ПЕРШИХ!
ДОВІРЯЙТЕ СПРАВЖНІМ!
ЦІНУЙТЕ УНІКАЛЬНІСТЬ!**

Видавець СГ НТМ «Новий курс»
Наукова установа
Пр. Перемоги, 77, оф. 179, м. Харків, 61174, Україна
Тел.: +380500301905, +380962250903
Telegram, Viber: +380970440309
www.newroute.org.ua, info@newroute.org.ua

Свідectво про внесення суб'єкта видавничої справи до
Державного реєстру видавців, виготовлювачів і розповсюджувачів
видавничої продукції ДК № 8013 від 22.11.2023
Зареєстровано у Global Register of Publishers
Зареєстровано у Crossref із власним префіксом 10.61718



Прислудуйтесь

Підписка <http://surl.li/vvfqkp>
Фейсбук-сторінка www.facebook.com/newroute1989
Телеграм <https://t.me/newroute1989>
Інстаграм www.instagram.com/newroute1989
Вайбер <http://surl.li/nbtqz>
Фейсбук-група www.facebook.com/groups/secnr

Будемо раді подальшій співпраці!