Environment. Technology. Resources. Rezekne, Latvia
Proceedings of the 16th International Scientific and Practical Conference. 2025. Volume II, 321-327

A Comparative Analysis of Native iOS Frameworks for Developing Graphics Scenes in Augmented Reality Applications

Mariya Shirokopetleva
Department of Software Engineering
Kharkiv National University of RE
Kharkiv, Ukraine,
marija.shirokopetleva@nure.ua

Olena Shevchenko
Department of Software Engineering
Kharkiv National University of RE
Kharkiv, Ukraine,
daria.koshkina@nure.ua

Daria Pervieieva
Department of Software Engineering
Kharkiv National University of RE
Kharkiv, Ukraine,
olena.shevchenko@nure.ua

Loreta Savulioniene
Faculty of Electronics and Informatics
Vilniaus Kolegija
Vilnius, Lithuania
l.savulioniene@viko.lt

Paulius Sakalys
Faculty of Electronics and Informatics
Vilniaus Kolegija
Vilnius, Lithuania
p.sakalys@eif.viko.lt

***Abstract*—While defining the technology stack for creating native AR apps for iOS, developers are faced with the need to choose between two popular Apple graphics frameworks: SceneKit and RealityKit. Most available studies are dedicated to the analysis of their internal architecture and most beneficial technical capabilities. Whereas this study focuses on the fundamental comparison of the frameworks using the multi-criteria decision analysis method. For this, the significant qualitative and quantitative comparison criteria are highlighted, as well as a series of experiments are conducted to compare frameworks in terms of scene rendering performance (FPS) and mobile device resource usage (CPU, GPU). Such a formal approach makes it possible to evaluate the effectiveness of frameworks and their suitability for different project types and development teams. This allows developers to make informed decisions during the design stage according to their needs. Confirming the potential of the proposed approach in practice, an example of its application to select the most appropriate framework for two different AR projects is presented. The specificity of applying this approach in the context of the current task was highlighted.**

Keywords*— *Augmented Reality, graphics scenes, iOS, multi-criteria decision analysis.

I. INTRODUCTION

Augmented reality (AR) technologies are becoming an integral part of everyday life, with applications spanning various fields. From mobile gaming to pre-project visualisation and interactive learning, AR enhances the user experience by integrating virtual elements into the real world [1], [2], [3]. According to ARTillary Intelligence [4] in 2024, the number of AR-ready Apple devices surpassed the number of similar devices from Google.

With interactive content becoming a norm, understanding AR frameworks is a crucial skill for software engineers mastering AR. However, integrating AR into iOS applications poses a challenge due to the variety of available frameworks, each with distinct advantages.

Choosing a technology stack when starting a new project is crucial for its development cost, set of features, further updates potential, future success in general and resulting profit. Developers can decide based on their experience and intuition or utilize any suitable formal method. Employing formal analysis can provide a defensible, evidence-based justification of the choice, ensuring that the decision is robust and balanced.

For Apple devices, there are three most commonly used native frameworks that could be utilized to implement high-performance AR applications: ARKit, SceneKit (SK), and RealityKit (RK).

The primary function of ARKit is environmental scanning and real-world integration [5], [6]: detect surfaces, track device position in space, and determine depth for placing virtual objects within the real

Online ISSN 2256-070X
https://doi.org/10.17770/etr2025vol2.8610

environment. ARKit can work with LiDAR scanners available on newer iOS devices, allowing the use of three-dimensional spatial data for real-time depth mapping [7]. There is no way to get access to all this functionality without ARKit for Apple devices. ARKit relies on SK, RK, or other graphics engines to render a graphical scene in an AR application [8].

SK is the oldest Apple implementation of rendering technologies. It was introduced in 2012 and was conceived for VR [9]. It is a flexible and powerful 3D rendering engine that enables developers to easily create 3D objects, apply materials, and set up lighting, cameras, animations, and physics simulations, as highlighted in [8], [10].

RK is a modern framework introduced by Apple in 2019 [11]. It enhances 3D object rendering realism in real-world, physics simulations, and spatial audio "out of the box", supports async loading and provides a networking layer to implement state synchronization, etc., as discussed in [10], [12].

If developers want to use only native tools to ensure productivity, they need to choose between SK and RK.

II. MATERIALS AND METHODS

A. Aim and tasks of the study

Despite numerous studies on SK, and RK, most focus is made on analysing their technical capabilities, performance, and integration. However, there are no clear recommendations on selecting the appropriate framework based on project requirements, particularly considering mobile device limitations, performance demands, and rendering specifics. This lack of guidance makes it challenging for developers working on AR applications, highlighting the need for this research.

The aim of this study is to investigate the impact of SK and RK frameworks on the performance and functionality of developed scenes in AR applications for iOS. The research objectives are:

a) Highlight the criteria that can influence the choice of a framework by the software architect, define types of criteria (qualitative or quantitative), measurement scales, ranges, and their possible aggregation groups; the grouping of some criteria is used to reduce their quantity in case it doesn't confuse experts during the criteria groups' weights adjustment.

b) Conduct an experiment to measure criteria values for every alternative (SK and RK frameworks in the current case).

c) Give an example of selected criteria usage to choose SK or RK for specific project and analyse the results.

B. Selecting a formal method for framework scoring

Among the suitable formal methods of alternative selection, there are, for example, neural networks [13], the classical method of multicriteria analysis (MCDA) [14], or its modification based on fuzzy logic [15]. On the one hand, neural networks require a large amount of historical data. This is unlikely to be available to an ordinary software development company. On the other hand, the use of fuzzy logic makes sense when there is partial uncertainty about the values of the selected criteria, or when several experts disagree in their assessments. For such a task as choosing one of two frameworks according to the needs of a particular project, in case all the parameters (criteria) can be measured in terms of certain numerical values and there are no major differences in the way they are interpreted by experts, a classic MCDA is faster and more transparent. This approach also helps to better analyse the pros and cons of the alternatives and to come up with confidence-aware decisions.

For this research classical additive MCDA has been chosen.

C. Criteria identification and definition

There are several aspects that should be considered while making a choice among different graphical frameworks for AR applications. They are performance, supported features (graphics, physics), integration level of AR operations, fine-tuning abilities, simplicity of use, and limitations. Fine-tuning (or customization) ability is the ability to implement such a feature that is not supported in the framework by default.

The values of all indicators are scaled to the selected universal interval scale [0-10], where 0 means not supported or absence of feature and 10 means the best possible level of support. For such parameters as, for example, "Necessity of Performance Tuning," the values are inverted to conform to a unified logic "the bigger - the better".

The performance of AR applications is measured using several key metrics [16]. They are listed in Table I. All the metric ranges in Table I are given non-normalized because it's easier for a person to perceive them in such a format.

The primary performance metric is Frames Per Second (FPS). A high FPS ensures a seamless user experience, whereas a low FPS can cause animation stuttering and degrade usability. For AR\VR applications, the optimal FPS is a stable 60 FPS. Measuring FPS helps assess how well frameworks handle scene rendering while maintaining high performance during animations [17].

CPU usage is critical for overall application performance. It is responsible for loading assets, managing and positioning 3D objects, transferring data to GPU, processing physical interactions and application logic, integrating data from sensors, etc.

Efficient CPU usage is necessary to maintain a balance between animation smoothness and minimal resource consumption [18].

The lower boundary for CPU Frequency is set to 1 GHz since that's the value for its 'idle' mode. The upper boundary for CPU Frequency is set to 3.2 GHz since for most iPhones (the device on which the experiment was performed) with an AX chip, the maximum frequency of 'large' cores can be approximately in the range of 2.3-3.2 GHz.

TABLE I. PERFORMANCE INDICATORS

Indicator	Range	Description
Average FPS, FPS	0-60	Frames Per Second, indicate the smoothness of the application performance
Occasional FPS Drops, FPS	0-60	FPS drops cause a poor user experience
Initial FPS Drops, FPS	0-60	Min observed FPS drops on pick loads
Max CPU Frequency, GHz	1-3.2	To what max frequency the processor was 'overclocked' at peak loads; this indirectly indicates how much the CPU has been utilized
Max GPU Usage, %	0-100	Shows how much GPU was utilized at pick loads

GPU utilization plays a key role in achieving high-quality graphics and real-time interaction in AR/VR applications. The GPU is specialized for rendering computations, making it responsible for displaying 3D objects, textures, shadows, reflections, and lighting. In AR applications, the GPU enables virtual objects to blend seamlessly with the real-world camera feed. High computational loads can cause FPS drops and a poor user experience, especially in complex scenes with many objects. To maintain 60 FPS or higher, GPU performance must be optimized to prevent frame rate delays, interface lag, and excessive battery consumption. Overloading the GPU can lead to overheating and rapid battery depletion, making balanced resource management essential for efficient AR application performance [19].

Ensuring framework adaptability allows developers to optimize resource usage while considering device limitations. Effective CPU and GPU management prevents excessive performance degradation and enhances the overall user experience [20].

All the indicators listed in Table I are estimated during the experiment that is described in section II.D as well as the values themselves.

The important functional graphics features are listed in Table II. In addition to the primary criteria, the analysis also considers additional graphics functional aspects [9], [10], [11] that might be crucial to the quality of AR application being developed. They are 3D file formats support (for assets), different animation types (action based, key frame, skeletal, morph target, inverse kinematics, animation blending, etc.), photorealistic quality of the scene (PBR, basic lighting and shadowing, environmental lighting, ray tracing), physical simulations support (collisions, gravity, forces, physical properties of the objects, etc.).

Since the target type of applications considered in this paper are AR-oriented applications, the level of AR operations integration into the frameworks being evaluated impacts crucially their development process and, thus, their comparison. For example, handling collision with real-world objects, occlusion (hiding partly or completely virtual objects behind real-world objects), and external real-world light consideration.

TABLE II. GRAPHICAL FEATURES INDICATORS

Indicator	SK	RK	Description
3D File Formats (GLTF, OBJ, FBX, USDZ, USD)	8	8	SK doesn't support the USD format; RK doesn't support the FBX format
Animations	8	7	Among all animations, SK doesn't support Real-Time AR-Driven animation only while RK doesn't support Inverse Kinematics and Animation State Machines at all or has limited support for Custom Animation Curves, etc.
Photorealism	7	8	For SK is possible and achieves good results with enough tweaking, but it requires more manual work. For RK automatic environmental lighting and reflection systems help deliver a realistic blend with the real world "out of the box".
Physical Simulations	7	6	SK supports common highly customizable physics interactions (rigid bodies, fields, constraints) but is not aimed at advanced fluid/soft-body out of the box. RK is streamlined for typical AR collisions and simple rigid body dynamics but has fewer advanced features or deep customization.
Average score	**7.5**	**7.25**	

According to this indicator, SK has got 6 points while RK has got 9 points. This is due to the fact that SK supports only basic AR Anchoring and Environmental Lighting out of the box. Other functionalities like Real-Time Plane Detection, Real-Time Occlusion, Physics-Based Real-World Interaction, Motion Capture Support, etc. are supported via manual setup with ARKit or not supported at all [10].

RK points are also not maximum because object recognition works based on pre-scanned 3D object models, it also relies heavily on ARKit's spatial awareness capabilities (e.g., tracking, plane detection, and occlusion) [11]. That's why this functionality in RK is significantly weaker than the same functionality in AI-powered frameworks, such as Vuforia and Unity [21]. Manual implementation of such AI-based algorithms is possible with the use of different object boundary segmentation algorithms, for example, those described in [22].

If the desired feature is not supported by default in the frameworks, it could be possible to implement it manually. Differences between SK and RK in the abilities of feature customization and performance tuning are listed in Table III.

TABLE III. FINE-TUNING INDICATORS

Indicator	SK	RK	Description
Performance Tuning	8	5	SK supports Level of Detail (LOD) management, profiling and reducing draw calls, and simplifying node hierarchies using Xcode Instruments; RK allows to structure "entity-component-system" (ECS) data efficiently
Custo-mization	9	5	SK supports custom shaders, PBR materials, lighting, and advanced postprocessing, e.g. [23], etc. RK supports *only* simple custom materials, reflections, and lighting adjustments
Average score	**8.5**	**5**	

The indicators collected in Table IV [24] are useful because in case other aspects are equal, a product that is easier to use is likely to be preferred.

TABLE IV. USAGE SIMPLICITY INDICATORS

Indicator	SK	RK	Description
Community, Docs, Samples	9	8	SK was introduced in 2012 and RK – in 2019. Both have detailed documentation and a lot of code samples.
Graphical Scene Editor	8	7	SK has an advanced scene editor in Xcode good for deep customization and control over every aspect of a 3D scene but lacks AR-elements integration out of the box; RK uses an external Reality Composer for quick prototyping of AR apps
Scene Organization approach	8	10	SK uses an old-style node-based hierarchy, good for 3D scenes; RK provides a modern and easy-to-start ECS approach, but fewer "classic" 3D scene graph control
Necessity of Performance Tuning	5	9	SK is capable to handle moderately complex scenes with advanced graphical effects but requires manual tuning for performance optimization RK provides a generally efficient graphical engine based on Metal API
Average score	**7.5**	**8.5**	

Limitations for rated frameworks are shown in Table V.

Platform compatibility influences the potential number of application clients, and Scene scalability affects the classes of AR applications, that could be developed with the corresponding framework.

D. Experiment

To measure the performance indicators of the SK and RK in terms of parameters listed in Table I the experiment was conducted. The ARPerfTester iOS application was developed. The application allows a user to create a three-dimensional scene, add and remove objects, perform animations and analyse system behaviour under varying loads.

TABLE V. LIMITATIONS INDICATORS

Indicator	SK	RK	Description
Compatibility	10	9	SK is available for all versions of iOS/macOS/visionOS; RK is available for iOS 13+/ macOS 10.15+/visionOS all versions
Scene Complexity Scalability	7	9	SK handles moderately complex scenes, but can struggle without fine performance optimizations; RK is generally efficient for complex scenes due to Metal utilization
Average score	8.5	9	

Several test scenarios are offered to evaluate the performance of SK, and RK frameworks. The main goal of the testing is to determine how these tools handle rendering a large number of objects, processing animations, user interactions, and utilizing the device's hardware resources.

Test scenario 1: evaluates performance with an increasing number of objects (up to 100, up to 10000 faces in total). The aim is to determine how object count affects rendering performance.

Test scenario 2: test the impact of animations on the performance. The goal is to determine how the framework processes the simultaneous animation of multiple objects. Using a scene with objects (up to 100, up to 10000 faces in total), a simultaneous rotation animation was executed.

Test scenario 3: stability after object removal. The goal is to check system performance as well as memory management efficiency and resource deallocation.

For the experiment, Xcode Instruments [20] were used as the primary tool for performance analysis and debugging of iOS applications. Instruments allow real-time monitoring of device resource usage, including CPU, GPU, FPS, memory, and power consumption, providing precise data on AR application performance.

Before running the tests, the following tools were configured in Xcode Instruments:

- Time Profiler – to analyse CPU load and identify peak processing moments during scene rendering.
- GPU Profiler – to monitor GPU usage while processing 3D objects and animations.
- Core Animation – to track frame rate (FPS) in real-time and assess the stability of the frameworks.

The experiment was conducted on a physical device (iPhone 8 Plus running iOS 16.7.10). During each test scenario, Xcode Instruments recorded all performance metrics, allowing for both real-time analysis and post-test review. The collected data was exported in .trace format,

enabling a detailed further analysis of the indicators. The results of the experiment are presented in Table VI below.

TABLE VI. SUMMARY OF THE PERFORMANCE RESULTS

Metric	SceneKit	RealityKit
Average FPS, FPS	58	54
Occasional FPS Drops, FPS	4	9
Initial FPS Drops, FPS	55	9
Max CPU Frequency, GHz	1.68	2.13
Max GPU Usage, %	18	53

Throughout most of the testing, SK indicates high performance. However, short-term drops were observed while adding a large number of objects. These drops occurred mainly in the initial step of changing stages when the system processed a large number of new objects or started to execute simultaneous animations.

RK exhibited more pronounced FPS fluctuations initially (when the stage was still empty) – frame rates dropped to 9 FPS, likely due to the high computational requirements for processing initial physics effects and rendering. Subsequently, FPS stabilized at 57-60 FPS, but periodic performance drops were still recorded. These drops were not as low as for SK, but RK was stabilizing longer than SK. This effect confirms higher system resource demands.

The GPU load in the SK remained relatively low. This suggests efficient computation optimization, where the load is evenly distributed among processes.

RK showed significantly higher GPU usage. The highest peak values were observed during active object interactions when complex lighting, physics effects, and animations were being rendered.

High CPU frequency on SK was observed during the addition of new objects and animation execution, but performance remained stable after object removal. RK indicates significantly higher computational requirements compared to SK. This is due to RK's extensive use of physics simulations and lighting rendering, which increases the overall processor load.

In order to utilize the obtained values for analysis using additive convolution [14], the parameters were normalized to the selected range [0-10] according to (1):

$$x_{norm} = \frac{x_i - x_{min}}{x_{max} - x_{min}} \times 10 \qquad (1)$$

The results of the calculations are presented in Table VII.

TABLE VII. NORMALIZED SUMMARY OF THE PERFORMANCE RESULTS

Metric	SceneKit	RealityKit
Average FPS, FPS	9.7	9
Occasional FPS Drops, FPS	0.67	1.5
Initial FPS Drops, FPS	9.17	1.5
Max CPU Frequency, GHz	3.09	5.14
Max GPU Usage, %	1.8	5.3
Average score	**4.89**	**4.49**

E. Application of MCDA to determine the feasibility of frameworks' usage for different applications

Let's describe two typical applications under development that can be used to assess the feasibility of frameworks' usage for their implementation.

Application1 is being developed by a small startup team. Application1 is a product catalogue of furniture and the utility for placing 3D models in space.

Application2 is being developed by a team with experience in 3D graphics or game development. Application2 is the Educational Physics Lab that allows students to place virtual setups (pendulums, mechanical models, etc.) into the real-world environment to investigate and experiment with precise physical interactions, adjusting physical characteristics of the setup components (e.g. motor torque, friction at joints, gear ratios, angular constraints, mass distribution, damping, joint strength, etc.).

Through the analysis of application functionality, weight values for each group of indicators have been proposed for each of the applications. The results of additive convolution [14] for Application1 for both frameworks are given in Table VIII, and for Application 2 are given in Table IX.

TABLE VIII. RESULTS FOR APPLICATION1

Criterion	Weight	SceneKit		RealityKit	
		c_i	$c_i * w_i$	c_i	$c_i * w_i$
Performance	0,25	4.89	1.22	4.49	1.12
Graphical Features	0,1	7.5	0.75	7.25	0.73
AR Integration	0,25	6	1.5	9	2.25
Fine-Tuning	0,1	8.5	0.85	5	0.5
Simplicity	0,2	7.5	1.5	8.5	1.7
Limitations	0,1	8.5	0.85	9	0.9
Total			6.67		**7.2**

TABLE IX. RESULTS FOR APPLICATION2

Criterion	Weight	SceneKit		RealityKit	
		c_i	$c_i * w_i$	c_i	$c_i * w_i$
Performance	0,25	4.89	1.22	4.49	1.12
Graphical Features	0,3	7.5	2.25	7.25	2.18
AR Integration	0,15	6	0.9	9	1.35
Fine-Tuning	0,2	8.5	1.7	5	1
Simplicity	0,05	7.5	0.38	8.5	0.43
Limitations	0,05	8.5	0.43	9	0.45
Total			**6.88**		6.53

The experiment demonstrates that RK is more suitable for Application1, whereas SK is preferable for Application2.

This is because for Application1 basic photorealism and support for various AR interactions, such as gestures to move and rotate objects, are fundamental, while Application2 is impossible without accurate modeling of complex physical interactions and careful performance tuning for complex graphical objects.

III. Results and Discussion

The obtained results demonstrate that the SK is more stable in maintaining frame rates and efficiently utilizes hardware resources, particularly the GPU, thus is suitable for projects where performance and energy optimization are critical. Conversely, with the same scene project in code RK provides more photorealistic rendering and complex physics interaction, but this comes at the cost of significantly higher CPU and GPU load, which leads to FPS instability and increased power consumption. This, in general, confirms the developers' observations obtained in [10], but Apple is constantly improving RK and Metal that is used in its background (instead of OpenGL, which is used by default in SK) and, in perspective, RK should outperform the SK.

According to the features and capabilities comparison, both frameworks cover the same set of them, but there are differences in specific features and how many of them are built-in versus needed manual work. In general, SK is more powerful, mature and flexible in graphical, physical and audio features, but it needs more manual work and professional knowledge to develop them. RK turns on, by default, simple variants of these features but is not able to implement extended versions of them at all. For example, specialized shader effects (non-realistic “Cel” or “toon”-rendering with discrete colour bands, sharp lightning transitions and outlined edges) are not possible in RK at all. They could be useful in educational AR applications, artistic visualization or interactive guides, where it is necessary to help users easily distinguish virtual AR content from real-world scenes.

The same situation is in physical simulations. Powered by Bullet [25], SK allows to set up advanced physical properties of graphical objects, as for Application2. Such interactions are very hard to implement in RK which lacks the granularity needed to provide educationally accurate, highly customizable demonstrations of complex mechanical principles.

On the other side, as the study showed, RK outperforms SK in AR features support. RK was built for AR, so it has AR conveniences that SK alone doesn’t (e.g. automatic gesture handling). A crucial weakness of SK is that under visionOS it runs only in 2D mode, lacking visionOS immersive integration at all.

As a more modern development tool, RK is simpler (though less customizable) than SK for less experienced developers. SK suites developers teams who are professionals in 3D graphics.

In general, RK emerges as the recommended choice for new AR-specific projects. Developers should also consider RK to prototype and future-proof their applications, especially in the context of Apple’s spatial computing initiatives, including visionOS. Nevertheless, SK continues to serve as a reliable general-purpose 3D graphics framework, particularly useful when AR capabilities are secondary, or there are some features that are much harder or not possible to implement in RK than in SK.

IV. Conclusions

This study conducted a comprehensive analysis of the performance and capabilities of the SK and RK frameworks in the context of developing AR applications for iOS. Key evaluation criteria were identified and assessed. A method of MCDA was offered to rigorously select a framework that better meets the requirements of a specific project. It is necessary to pay attention to the peculiarity of the MCDA application in the case when there are mandatory ("must-have") features, e.g. vertex-shaders control or custom graphical post-processing effects. Such mandatory criteria should be excluded from MCDA and checked first. The following MCDA should be made among the alternatives that have passed the mandatory criteria test.

Regarding the future perspective, it is necessary to focus on the research of AI-driven methods in AR, such as detailed spatial understanding, semantic scene segmentation, and mesh classification [26],[27], which could greatly enhance AR functionality for native iOS applications.

References

[1] M. K. Shaleh Md Asari, N. M. Suaib, M. H. Abd Razak, M. A. Ahmad, and N. M. Shaleh, "Empowering Skill-Based Learning with Augmented Reality and Virtual Reality: A Case Study," in 2024 IEEE International Symp. on Consumer Technology, Kuta, Bali, Indonesia, 2024, pp. 225-229, https://doi.org/10.1109/ISCT62336.2024.10791270.

[2] B. Ton, N. Tempert, and D. Plass, "Immersive visualisation of point cloud data of railway environments," in 2024 10th International Conf. on Virtual Reality, Bournemouth, United Kingdom, 2024, pp. 233-239, https://10.1109/ICVR62393.2024.10868576.

[3] A. Dhiya' Mardhiyyah, Vincent, K. L. Mario Gracius, F. Permana, and F. I. Maulana, "LonelyScape: Increasing Attractiveness of Escape Room Game Using Augmented Reality Technology," in 2023 International Conf. on Information Management and Technology, Malang, Indonesia, 2023, pp. 795-800, https://doi.org/10.1109/ICIMTech59029.2023.10277954

[4] MobiDev,”12 Augmented Reality Technology Trends of 2025: New Milestones in Immersive Innovations,” MobiDev, 2024. [Online]. Available: https://mobidev.biz/blog/augmented-reality-trends-future-ar-technologies. [Accessed: Nov. 17, 2024].

[5] Th. A. R. Sure, “Motion Tracking in iOS Applications Using Augmented Reality,” Journal of Android and iOS Applications and Testing,vol. 8, no. 3, pp.1-5, Oct. 2023.

[6] L. Nissen et al., “Towards Preventing Gaps in Health Care Systems Through Smartphone Use: Analysis of ARKit for Accurate Measurement of Facial Distances in Different Angles,” Sensors, vol. 23, no. 9, 4486, May 2023, https://doi.org/10.3390/s23094486.

[7] Apple, “Apple Developer Documentation ARKit,” 2024. [Online]. Available: https://developer.apple.com/documentation/arkit. [Accessed: Nov. 21, 2024].

[8] J. Nhan, “Building Your First ARKit App with SceneKit,” in Mastering ARKit, Berkeley, CA: Apress, 2022, pp. 14-27.

[9] Apple, “Apple Developer Documentation. SceneKit,” 2024. [Online]. Available: https://developer.apple.com/documentation/scenekit. [Accessed: Nov. 27, 2024].

[10] “RealityKit vs SceneKit vs Metal – High-Quality Rendering,” Mar. 7, 2024. [Online]. Available: https://stackoverflow.com/questions/60505755/realitykit-vs-scenekit-vs-metal-high-quality-rendering. [Accessed: Jan. 17, 2025].

[11] Apple, “Apple Developer Documentation. RealityKit,” 2024. [Online]. Available: https://developer.apple.com/documentation/realitykit. [Accessed: Dec. 1, 2024].

[12] S. Sasmoko, F. L. Gaol, and T. Oktavia, “Augmented Reality SDK Overview for General Application Use,” International Journal of Advanced Computer Science and Applications, vol. 14, no. 11, pp. 54-60, 2023, http://doi.org/10.14569/IJACSA.2023.0141106.

[13] S. Wang, B. Mo, and J. Zhao “Deep neural networks for choice analysis: Architecture design with alternative-specific utility functions,” Transportation Research Part C: Emerging Technologies, vol. 112, pp. 234-251, Mar. 2020, https://doi.org/10.1016/j.trc.2020.01.012.

[14] A. Ishizaka and P. Nemery, Multi-Criteria Decision Analysis: Methods and Software. Hoboken, NY: Wiley, 2013, https://doi.org/10.1002/9781118644898.

[15] L. Maretto, M. Faccio, and D. Battini, “A Multi-Criteria Decision-Making Model Based on Fuzzy Logic and AHP for the Selection of Digital Technologies,” IFAC-PapersOnLine, vol. 55, no. 2, pp. 319-324, 2022, https://doi.org/10.1016/j.ifacol.2022.04.213.

[16] P. Nowacki and M. S. Woda “Capabilities of ARCore and ARKit Platforms for AR/VR Applications,” Engineering in Dependability of Computer Systems and Networks, vol. 987, pp. 358–370, 2020.

[17] A.M. Boutsi, C. Ioannidis, and S. Verykokou "Multi-Resolution 3D Rendering for High-Performance Web AR," Sensors, vol. 23, no. 15, 6885, Aug. 2023, https://doi.org/10.3390/s23156885.

[18] M. Hort, M. Kechagia, F. Sarro, and M. Harman "A Survey of Performance Optimization for Mobile Applications," IEEE Transactions on Software Engineering, vol. 48, no. 8, pp. 2879–2904, Aug. 2022, https://doi.org/10.1109/TSE.2021.3071193.

[19] F. Nusrat, F. Hassan, H. Zhong, and X. Wang. "How Developers Optimize Virtual Reality Applications: A Study of Optimization Commits in Open Source Unity Projects," in ICSE '21: Proceedings of the 43rd International Conf. on Software Engineering, pp. 473–485, Nov. 2021, https://doi.org/10.1109/ICSE43902.2021.000.

[20] “Getting Started with Instruments. WWDC Notes,” 2019. [Online]. Available: https://wwdcnotes.com/documentation/wwdcnotes/wwdc19-411-getting-started-with-instruments. [Accessed: Oct. 17, 2024].

[21] “Vuforia Instruct Is Now Vuforia Expert Capture,” May 2023. [Online]. Available: https://www.ptc.com/en/products/vuforia/vuforia-instruct. [Accessed: Jan. 17, 2025]

[22] K. Smelyakov, S. Smelyakov., and A. Chupryna, “Chapter 1. Adaptive Edge Detection Models and Algorithms,” in Advances in Spatio-Temporal Segmentation of Visual Data.Studies in Computational Intelligence, vol. 876: Springer, Cham, 2020, pp.1-51, https://doi.org/10.1007/978-3-030-35480-0_1.

[23] K. Smelyakov, M. Hvozdiev, A. Chupryna, D. Sandrkin, and V. Martovytskyi, “Comparative EfficiencyAnalysis of Gradational Correction Models of Highly Lighted Image,” in 2019 IEEE International Scientific-Practical Conf. Problems of Infocommunications, Science and Technology, Kyiv, Ukraine, 2019, pp. 703-708, https://doi.org/10.1109/PICST47496.2019.9061356.

[24] I. Gruzdo, I. Kyrychenko, G. Tereshchenko, N. Shanidze, “Metrics Applicable for Evaluating Software at the Design Stage,” in COLINS-2021: 5th International Conference on Computational Linguistics and Intelligent Systems, vol. I (2870): Main Conference, Lviv, Ukraine, 2021, pp.916-936. [Online]. Available: https://ceur-ws.org/Vol-2870. [Accessed: Nov. 21, 2024].

[25] C. Dickinson, Learning Game Physics With Bullet Physics and OpenGL, Packt Pub Ltd, 2013.

[26] R. Yenni and A. P V, "Semantic Segmentation and Spatial Relationship Modeling in Hyperspectral Imagery Using Deep Learning and Graph-Based Representations," in 2024 14th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing, Helsinki, Finland, 2024, pp. 1-4, https://doi.org/10.1109/WHISPERS65427.2024.10876420.

[27] N. Ali, A. Z. Ijaz, R. H. Ali, Z. Ul Abideen and A. Bais, "Scene Parsing Using Fully Convolutional Network for Semantic Segmentation," in 2023 IEEE Canadian Conf. on Electrical and Computer Engineering, Regina, SK, Canada, 2023, pp. 180-185, https://doi.org/10.1109/CCECE58730.2023.10288934.