TECHNOLOGIES AND ENGINEERING

Journal homepage: https://technologies-engineering.com.ua/en
Vol. 26, No. 4, 2025

Received: 11.04.2025
Revised: 22.08.2025
Accepted: 17.09.2025

UDC 621.3

DOI: 10.30857/2786-5371.2025.4.3

Research on methods for optimising the performance of VR applications in low-bandwidth browsers

Yan-Viktor Latyshev*

Postgraduate Student
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"
03056, 37 Beresteiskyi Ave., Kyiv, Ukraine
https://orcid.org/0009-0000-6133-2736

Hanna Vlasiuk

Doctor of Technical Sciences, Professor
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"
03056, 37 Beresteiskyi Ave., Kyiv, Ukraine
https://orcid.org/0000-0002-7382-0435

Abstract. The relevance of the research lay in the need to improve the performance of WebVR applications in browser environments with limited network bandwidth, which is important for users with slow or unstable Internet connections. The aim of the work was to develop and evaluate effective methods that reduce scene loading times, optimise the amount of data transferred, and maintain a stable frame rate without compromising the quality of interaction. The study used experimental modelling of a virtual reality environment with limited network parameters. The work used open libraries for three-dimensional graphics with the ability to progressively load 3D resources. Several optimisation methods were implemented and compared, including multi-level model detailing, deferred loading, server-side geometry simplification, data compression, and texture decompression. The effectiveness was evaluated based on loading time, traffic volume, smoothness of playback, and system response. The results of the study showed that combining multi-level detailing with texture compression can reduce the amount of data transferred by up to 70% without noticeable loss of image quality. It was found that the use of progressive loading significantly reduces the initial scene rendering time, while client-side caching reduces repeat traffic by 90%. The most effective strategy was found to be the initial loading of simplified models with the subsequent asynchronous addition of detailed objects, which supports stable operation even at low Internet speeds. The practical value lies in the application of the developed methods in the creation of WebVR applications for education, medicine, commerce, and entertainment, especially in conditions of limited internet connection. This contributes to expanding the audience and reducing infrastructure costs

Keywords: WebXR; progressive loading; LOD; caching; texture compression; Three.js; A-Frame

Introduction

Modern virtual reality (VR) technologies are increasingly being integrated into the web environment thanks to the development of application programming interfaces (APIs) such as WebVR (Web Virtual Reality) and WebXR (Web Extended Reality). However, the performance of such solutions depends on the quality of the internet, which complicates access in regions with slow or unstable connections. Traditional VR applications that require the transfer of large amounts of 3D data are ineffective in such conditions. This necessitates the optimisation of VR content to take into account network limitations. The main task is to maintain a balance between visualisation quality and the amount of data transferred. Deterioration in quality reduces the user experience, while maintaining full detail complicates scene loading. This requires a comprehensive approach to optimisation, including the use of level-of-detail (LOD) models, resource compression, progressive loading, and efficient caching.

Globally, the current state of VR content optimisation for low-bandwidth web environments has been characterised by the active development of adaptive algorithms that take into account network fluctuations and limited resources. In particular, research focused on integrating WebXR

Suggested Citation:
Latyshev, Ya.-V., & Vlasiuk, H. (2025). Research on methods for optimising the performance of VR applications in low-bandwidth browsers. *Technologies and Engineering,* 26(4), 37-43. doi: 10.30857/2786-5371.2025.4.3.

*Corresponding author

with networks to reduce latency and optimise traffic. For example, H. Lee & Y. Hwang (2022) evaluated the use of VR technologies in web environments for educational purposes, emphasising the importance of an adaptive approach to resources and optimising interaction in conditions of limited bandwidth. F. Maura *et al*. (2024) worked on algorithms that automatically adjust the quality of VR video depending on network speed using Wi-Fi. Their research showed that such algorithms reduce video "freezing" by 20%, which is useful for creating smooth VR applications in browsers.

The research focused on streaming VR content in a web environment focused on user experience quality, where adaptive encoding, progressive loading, and caching methods were key. B. Fanini & G. Gosti (2024) developed a pipeline for immersive analytics based on WebXR, addressing network bottlenecks and performance issues in collaborative 3D environments, reducing data volume and latency in low-bandwidth conditions. J. Woo *et al.* (2024) proposed a method that predicts internet speed using artificial intelligence (Gated Recurrent Unit models) and adjusts video quality, reducing latency by 11% in mobile networks. This approach can be useful for adapting WebXR applications to low-speed conditions. B. Fanini *et al*. (2021) presented the ATON framework for creating immersive Web3D/WebXR applications, which adapts interfaces and rendering to devices, including resource optimisation for low-speed networks through compression and progressive loading.

B. Sobota *et al*. (2023) investigated the use of web-based 3D environments in primary and secondary education for children with multiple disabilities, demonstrating the practical applicability of such environments in conditions of limited technical resources and confirming the possibility of adapting content for poor network conditions. L. Franke & D. Haehn (2020) in their review of modern web visualisations emphasised the role of WebXR alongside the Web Graphics Library in optimising performance, allowing the creation of high-quality 3D experiences with less network load through efficient compression and caching. K.A. Mills *et al*. (2024) developed an approach to using VR games and multimodal 3D environments in education, which contributes to the effective optimisation of web VR content and improves the quality of interaction in educational systems. Overall, these works pointed to a trend towards using frameworks such as A-Frame and Three.js, machine learning, and edge computing to adapt VR content, but not enough attention has been paid to comprehensive optimisation for low-bandwidth browsers, especially in combination with LOD, progressive loading, and caching. For example, A.-M. Boutsi *et al.* (2023) emphasised the role of multi-resolution rendering schemes using the Three.js library and glTF formats with Draco compression to reduce the loading time of 3D models in networks with limited bandwidth. Similarly, a study by S.-M. Wang *et al*. (2022) described support for progressive asset loading and adaptive resource optimisation, which allows web VR applications to reduce the first render time on low-speed networks.

Scientific research in the field of WebVR is actively developing, but much of it is focused on high-speed connections or native VR environments. Insufficient attention is paid to the specifics of how VR applications function in browsers under unstable network conditions. The aim of the study was to develop effective optimisation strategies that minimise loading delays, reduce the amount of data transferred, and ensure a stable frame rate, while maintaining the quality of user interaction with the system even under conditions of limited computing resources.

Materials and Methods

The study focused on optimising WebVR applications for browsers under low network bandwidth conditions (less than 2 Mbit/s). The methodology was developed to ensure a stable frame rate of 60 frames per second (FPS), reduce loading delays, and maintain the quality of the user experience. The study used data from S. Pacheco-Gutierrez *et al*. (2021), Three.js (n.d.), and A-Frame Documentation (n.d.). A WebVR/WebXR test scene was created using the Three.js and A-Frame libraries. The scene contained 3D models (up to 120,000 polygons), physically correct textures, dynamic lighting, and animation that simulated typical VR applications. Testing was conducted at a network speed of 1-2 Mbit/s, reflecting real conditions in regions with slow internet.

Three methods were tested: dynamic level-of-detail (LOD) replacement, texture compression in .basis format, and progressive loading with caching via Service Workers. The experiments were conducted on laptops (Intel UHD Graphics) and smartphones (Android/iOS) in Chrome, Firefox, and Edge browsers with WebXR support. The glTF models had simplified (3-5 thousand polygons) and detailed (up to 100 thousand) versions. .basis textures were selected based on network speed via navigator.connection.downlink. The scene was created with LOD, textures were compressed with the Basis Universal tool, and Service Workers cached the data. Performance was evaluated based on loading time (performance.now()), FPS, data volume, and graphics processing unit (GPU) load, measured using Chrome DevTools. The sample included several scenarios, each with 5-15 objects, to evaluate the methods under different conditions. Table 1 summarises the data for each scenario. The methodology allowed to test the effectiveness of the combination of LOD, texture compression, progressive loading, and caching under low bandwidth conditions. The choice of methods was justified by their proven effectiveness in other studies (Pacheco-Gutierrez *et al*., 2021; Žádník *et al*., 2022; Liu *et al*., 2023).

Table 1. *Description of test scenarios*

Scenario	Number of objects	Content type	Main parameters
Scenario 1	5	Static objects	Simplified models, 512 x 512 textures
Scenario 2	8	Mixed	Detailed models, 1024 x 1024 textures

Table 1. Continued

Scenario	Number of objects	Content type	Main parameters
Scenario 3	12	Dynamic objects	Animation, 2048 x 2048 textures
Scenario 4	15	Complex scene	LOD, .basis compression, delta updates

Source: *developed by the authors based on experimental data*

These approaches were implemented in detail to assess their effectiveness in practical conditions. In particular, dynamic LOD replacement involved loading two versions of models – simplified (3-5 thousand polygons) and detailed (up to 100 thousand polygons). Simplified models were loaded first, and detailed models were loaded asynchronously after scene initialisation. Formally, this approach is described as two-phase scene initialisation:

$$T_{total} = T_{init}^{low} + T_{defer}^{high}, \quad (1)$$

where T_{init}^{low} – initialisation time for simplified models, and, T_{defer}^{high} – asynchronous loading of detailed models.

Additionally, a method of adaptive texture replacement using the .basis format has been implemented. Several texture options (512 x 512, 1024 x 1024, 2048 x 2048) have been prepared for each object. The principle was calculated using the formula:

$$T = \frac{S * 8}{B}, \quad (2)$$

where S – size of transmitted data, B – network bandwidth.

For example, for S = 3 MB and B = 1 Mbit/s, T was 24 seconds. When using Basis Universal, the size was reduced to 0.6 MB, which corresponded to only 4.8 seconds. To reduce the load on the GPU, simplified 256 x 256 normal maps were used to simulate a complex surface (according to Chrome DevTools). Shaders (graphics processing programs) were also optimised: instead of complex mathematical effects, simple textures were used, the number of light sources was limited to two, and materials were changed to less resource-intensive ones (MeshStandardMaterial instead of MeshPhysicalMaterial). The optimised loading code is shown in Figure 1.

```
const modelLow = await loadGLTF('models/scene_low.glb')
scene.add(modelLow)

initScene().then(() => {
  loadGLTFAsync('models/scene_high.glb').then(modelHigh => {
    scene.remove(modelLow)
    scene.add(modelHigh)
  })
})
```

Figure 1*. Optimised loading code*
Source: *developed by the authors based on data from Three.js (n.d.), A-Frame Documentation (n.d.)*

The "delta change transmission" method was considered separately – instead of a complete scene update, only changes in the positions, rotations and states of objects are transmitted. For example, instead of JavaScript Object Notation (JSON) with a full description of 1,000 objects (~2 MB), only ID arrays and change vectors were transmitted, which reduced the average payload size to 40-60 KB per frame (Fig. 2).

```
[
  {"id": "chair_2", "position": [1.2, 0, 3.3], "rotation": [0, 90, 0]},
  {"id": "door_1", "open": true}
]
```

Figure 2*. Structure of the delta package for updating scene objects in JSON format*
Source: *developed by the authors based on materials from S. Pacheco-Gutierrez et al. (2021)*

Thus, the developed methodology provided a comprehensive verification of the impact of key WebVR application optimisation techniques in low-bandwidth browsers. It combined LOD levels, adaptive texture compression, and caching mechanisms, allowing to evaluate not only individual performance parameters but also system stability during dynamic scene updates. The methodology was reproducible because it uses open-source tools Three.js and A-Frame.

Results and Discussion

Initial tests showed that loading full-size models and textures with a resolution of 2048x2048 at a speed of 1 Mbit/s resulted in a first render time of 12.4 seconds and a frame rate of no more than 24 FPS. Testing was conducted across four scenarios covering static, mixed, dynamic, and complex scenes. A comparison of methods showed that LOD with deferred loading of detailed models provided the fastest initial rendering. Texture compression provided the greatest reduction in latency, although low-quality textures slightly reduced visual quality. Delta updates were most effective for dynamic scenes, transmitting only 2-3% of the data from the full scene state with an update time of less than 100 ms. The combination of all methods allowed to achieve a stable 60 FPS on devices with a weak graphics core (Intel UHD) at speeds of less than 1 Mbit/s (scenario 4). Key metrics: reduction in GPU frame time from 48 ms to 15 ms, first render time from 9.6 s to 2.8 s, data volume from 24 MB to 3.1 MB (Table 2).

Table 2. *Comparison of optimisation method performance*

Method	First render time (s)	FPS	Data volume (MB)	GPU frame time (ms)
Without optimisation	12.4 ± 0.5	24	24.0 ± 1.0	48 ± 2
LOD	4.2 ± 0.2	57-60	6.5 ± 0.3	20 ± 1
Compression .basis	5.1 ± 0.3	55-60	3.1 ± 0.2	18 ± 1
Delta updates	6.8 ± 0.4	58-60	4.0 ± 0.2	17 ± 1
Combination of methods	2.8 ± 0.2	60	3.1 ± 0.2	15 ± 1

Source: *developed by the authors based on experimental data*

The results showed that each optimisation method has its advantages depending on the type of content. In particular, the use of LOD proved to be most effective for static scenes with a large amount of geometry (scenario 1), as it allows simplified models to be displayed quickly and detailed ones to be loaded gradually. LOD reduced the time to first render to 4.2 ± 0.2 seconds, providing a stable 57-60 FPS even on devices with basic graphics. Texture compression in .basis format is best suited for projects with a large amount of graphic material (scenario 2), providing a significant reduction in loading time without noticeable loss of quality. The amount of data transferred was reduced to 3.1 ± 0.2 MB, which gave the best results on slow networks. Delta updates showed the best results for dynamic scenes with a large number of moving objects (scenario 3), as they only transmit changes in states, reducing traffic several times over. Delta updates minimised scene update latency, delivering a stable 58-60 FPS even in dynamic scenarios. The combined use of all methods ensures versatility and stable performance regardless of scene complexity or network speed.

A detailed review of the results showed that the LOD method was most beneficial in cases where fast first rendering was key. Scene appearance time was reduced by more than three times compared to the baseline, and the reduction in GPU load allowed for consistently high FPS even on devices with limited resources. This was particularly noticeable on laptops with Intel UHD, where performance improved the most. The only exception was Firefox, where short drops sometimes occurred during the gradual loading of details, but they did not have a critical impact on the user experience. The results for compressing textures in .basis format were different. Its main advantage was the maximum reduction in the amount of data transferred, which resulted in a significant gain in bandwidth. This effect is especially important in slow networks, where .basis made it possible to avoid long delays and maintain smooth scene rendering. At the same time, performance varied slightly depending on the platform: on iOS, hardware support accelerated decompression, while on Android, with less powerful hardware, additional delays were observed. This indicated that the effectiveness of the method is highly dependent on the specific execution environment.

Delta updates had the greatest impact on reducing network traffic. The amount of data transferred was reduced tenfold, which had a positive effect on the interactivity of dynamic scenes. As a result, the system's response time approached real time, and the frame rate remained virtually unchanged. On mobile devices, the mechanism worked flawlessly in Chrome and Edge, while in Firefox there were cases of request accumulation, which caused short pauses in the update. Despite this, the average performance remained within a comfortable range of 58-60 FPS. Comparing the methods according to key metrics (Table 2) reveals a clear distribution of strengths: LOD provided the best optimisation of start-up time, .basis offers the most effective reduction in network traffic, and delta updates reduce GPU load while ensuring high system responsiveness. The combined use of these techniques allowed to combine their advantages: rendering started in less than three seconds, data volumes were reduced by almost eight times, and the load on the GPU fell threefold compared to the initial conditions.

Analysis of results on different devices confirmed the versatility of the combined approach. On laptops with integrated graphics, performance reached a stable 60 FPS even at network speeds below 1 Mbps. Android smartphones showed slightly higher latency due to slower decompression, but the average frame rate did not drop below 55 FPS. iOS devices proved to be the most stable thanks to hardware optimisation, making them the most convenient platform for applying such techniques. Thus, all three methods have different effects on key metrics, but their combination provides the most balanced result for different types of scenes and different technical conditions. This ensures an acceptable quality of experience even on low-bandwidth networks and devices with weak graphics cores.

A comparison with other studies revealed similarities in approaches, but also revealed important differences. For example, A.-M. Boutsi *et al*. (2023) achieved a 46% reduction in render time using multi-resolution 3D rendering, which is similar to the author's LOD, but their methodology was not focused on low-bandwidth scenarios (tests were conducted mainly at speeds >5 Mbit/s). In contrast, the approach in this article showed stable 57-60 FPS even at <2 Mbit/s, making it suitable for a wider range of use cases. B. Fanini *et al*. (2021) reduced the data volume by 50% in WebXR using the ATON framework, which is partially consistent with the authors' results (73% for LOD), but the lack of integration with texture compression limited the overall effect. In the current work, the combination of methods yielded a more balanced result: a reduction in render time to 2.8 s while reducing the data volume by more than seven times. A similar optimisation effect was demonstrated by J. Wang *et al*. (2022), who implemented direct training of a voxel feature grid for high-precision surface reconstruction from RGB-D sequences in the GO-Surf project. Their method combines local geometric and visual features in a multi-level structure and provides

fast reconstruction with minimal deviation between synthesised and real data without the need for post-processing.

M. De Fré *et al*. (2024) showed that optimisation through caching can reduce the number of server requests by 70%, which is consistent with the authors' results, but their research focused primarily on video conferencing. The current results for VR content showed even greater relevance of this approach in interactive environments, where repeated requests significantly affect the user experience. M. Lim *et al*. (2025) achieved a 30% reduction in latency at speeds of 2-4 Mbit/s in the WIDE-VR project, but their system was optimised for a wider range of network conditions. In contrast, this study showed that under narrow constraints (<2 Mbit/s), the efficiency can be even higher: FPS increased from 24 to 60, and response time was reduced by more than half, making the approach more relevant for weak network environments. Similar results in terms of latency reduction were confirmed by S. Uddin *et al*. (2025), who investigated the effectiveness of low-latency adaptive bitrate algorithms within the framework of Hypertext Transfer Protocol adaptive streaming and showed that such algorithms provide a consistently high quality user experience even with fluctuations in network bandwidth.

Similar parallels can be found in the work of I. Yeregui *et al*. (2025), who investigated the optimisation of VR environments in the context of 5G networks. Although their results demonstrated a high level of adaptation and reduced latency thanks to network infrastructure, the authors of the current study focused on a more universal scenario – browser limitations, where 5G is not always available. This revealed a difference in areas of application: their approach is geared towards technologically advanced environments, while the authors' approach has practical value even in countries with lower quality network services. According to the results of W.J. Kim & B.B. Khomtchouk (2024), processing was accelerated by 63% thanks to WebAssembly. Although their work did not cover LOD or texture compression, it demonstrated the promise of combined technologies. The current results showed that combining traditional optimisations with modern tools such as WebAssembly or WebGPU can provide even better results, which is consistent with the system optimisation approach proposed by D. Honcharenko *et al*. (2023), who used multi-criteria analysis to select optimal technologies in IoT systems for monitoring physical indicators. In this sense, the authors' approach laid the foundation for future research, where classical methods will be integrated with new web technology standards.

It is also important to note the results of the study by C. Slocum *et al*. (2021), who presented the VIA system with selective loading of only those 3D objects that are in the user's field of view. This made it possible to reduce the scene rendering time by 50% at speeds of about 10 Mbit/s. However, this approach proved to be less effective in low-bandwidth scenarios, where the load remains significant even with partial model loading. The authors' research showed that a combination of LOD, delta updates, and texture compression allows for greater flexibility. This indicates that the proposed approach can be effective in a wider range of network conditions and be more resistant to extreme constraints than traditional visibility methods.

Thus, the authors' combined approach not only surpasses individual solutions in terms of versatility, but also demonstrates better adaptability to low-bandwidth conditions. This has been confirmed by direct comparisons with previous works: while other researchers have mostly tested systems in environments with medium or high network speeds, the current study has shown that even under minimal conditions (<2 Mbit/s), a significant performance improvement can be achieved without compromising the quality of the user experience.

Conclusions

The study evaluated methods for optimising WebVR applications for browsers under low bandwidth conditions (less than 2 Mbit/s). Experiments using the Three.js and A-Frame libraries confirmed that a combination of dynamic LOD replacement, texture compression in .basis format, and delta updates with caching via Service Workers significantly improves performance. The first render time was reduced from 12.4 to 2.8 seconds, the amount of data transferred was reduced from 24 MB to 3.1 MB, and the frame rate stabilised at 60 FPS on devices with a weak graphics core (Intel UHD) at network speeds of less than 1 Mbit/s. The use of LOD with initial loading of simplified models (3-5 thousand polygons) and asynchronous addition of detailed ones (up to 100 thousand polygons) ensured fast initial rendering. Compression of .basis textures reduced their size by 6-8 times, which reduced loading delays by 80%, as shown by measurements using performance.now.() Delta updates reduced the amount of data to 40-60 KB per frame, which proved to be effective for dynamic scenes. The combination of these methods outperformed individual approaches, ensuring stable performance. Additionally, analysis of the results showed that the combined approach allowed maintaining a stable frame rate even when the number of objects in the scene was increased to 15, confirming its scalability. It was also found that the use of simplified 256 x 256 pixel normal maps reduced peak GPU loads by 30%, and optimising shaders by limiting light sources to two reduced graphics processing time by 20% compared to the initial settings. Prospects for further research include integrating machine learning to predict network conditions and using WebGPU to accelerate graphics processing in browsers, which could further reduce latency and improve rendering quality.

Acknowledgements

None.

Funding

None.

Conflict of Interest

None.

References

[1] A-Frame Documentation. (n.d.). *Asset management system*. Retrieved from https://aframe.io/docs/1.7.0/core/asset-management-system.html.

[2] Boutsi, A.-M., Ioannidis, C., & Verykokou, S. (2023). Multi-resolution 3D rendering for high-performance web AR. *Sensors*, 23(15), article number 6885. doi: 10.3390/s23156885.

[3] De Fré, M., van der Hooft, J., Wauters, T., & De Turck, F. (2024). Demonstrating adaptive many-to-many immersive teleconferencing for volumetric video. In *Proceedings of the 15th ACM multimedia systems conference* (pp. 453-458). New York: Association for Computing Machinery. doi: 10.1145/3625468.3652192.

[4] Fanini, B., & Gosti, G. (2024). A new generation of collaborative immersive analytics on the web: Open-source services to capture, process and inspect users' sessions in 3D environments. *Future Internet*, 16(5), article number 147. doi: 10.3390/fi16050147.

[5] Fanini, B., Ferdani, D., Demetrescu, E., Berto, S., & d'Annibale, E. (2021). ATON: An open-source framework for creating immersive, collaborative and liquid web-appsfor cultural heritage. *Applied Sciences*, 11(22), article number 11062. doi: 10.3390/app112211062.

[6] Franke, L., & Haehn, D. (2020). Modern scientific visualizations on the web. *Informatics*, 7(4), article number 37. doi: 10.3390/informatics7040037.

[7] Honcharenko, D., Mokin, V., & Protsenko, D. (2023). Building an information system for monitoring physical indicators based on the Internet of Things technology. *Information Technologies and Computer Engineering*, 20(2), 99-108. doi: 10.31649/1999-9941-2023-57-2-99-108.

[8] Mills, K.A., Brown, A., & Funnell, P. (2024). Virtual reality games for 3D multimodal designing and knowledge across the curriculum. *The Australian Educational Researcher*, 51, 2323-2353. doi: 10.1007/s13384-024-00695-3.

[9] Kim, W.J., & Khomtchouk, B.B. (2024). WebAssembly enables low latency interoperable augmented and virtual reality software. *arXiv*. doi: 10.48550/arXiv.2110.07128.

[10] Sobota, B., Korečko, Š., & Mattová, M. (2023). Web-based 3D virtual environments utilization in primary and secondary education of children with multiple impairments. *Electronics*, 12(13), article number 2792. doi: 10.3390/electronics12132792.

[11] Lee, H., & Hwang, Y. (2022). Technology-enhanced education through VR-making and metaverse-linking to foster teacher readiness and sustainable learning. *Sustainability*, 14(8), article number 4786. doi: 10.3390/su14084786.

[12] Lim, M., Bentaleb, A., & Zimmermann, R. (2025). WIDE-VR: An open-source prototype for web-based VR through adaptive streaming of 6DoF content and viewport prediction. In *Proceedings of the 16th ACM multimedia systems conference* (pp. 221-227). New York: Association for Computing Machinery. doi: 10.1145/3712676.3718334.

[13] Liu, K., Wu, N., & Han, B. (2023). Demystifying web-based mobile extended reality accelerated by WebGPU. In *Proceedings of the 2023 ACM on internet measurement conference* (pp. 145-153). New York: Association for Computing Machinery. doi: 10.1145/3618257.3624833.

[14] Maura, F., Casasnovas, M., & Bellalta, B. (2024). Experimenting with adaptive bitrate algorithms for virtual reality streaming over Wi-Fi. *arXiv*. doi: 10.48550/arXiv.2407.15614.

[15] Pacheco-Gutierrez, S., Niu, H., Caliskanelli, I., & Skilton, R. (2021). A multiple level-of-detail 3D data transmission approach for low-latency remote visualisation in teleoperation tasks. *Robotics*, 10(3), article number 89. doi: 10.3390/robotics10030089.

[16] Slocum, C., Huang, J., & Chen, J. (2021). VIA: Visibility-aware web-based virtual reality. In *Proceedings of the 26th international conference on 3D web technology* (article number 7). New York: Association for Computing Machinery. doi: 10.1145/3485444.3487641.

[17] Three.js. (n.d.). *DRACOLoader: A loader for geometry compressed with the Draco library*. Retrieved from https://threejs.org/docs/#examples/en/loaders/DRACOLoader.

[18] Uddin, S., Grega, M., Leszczuk, M., & ur Rahman, W. (2025). Evaluating HAS and low-latency streaming algorithms for enhanced QoE. *Electronics*, 14(13), article number 2587. doi: 10.3390/electronics14132587.

[19] Wang, J., Bleja, T., & Agapito, L. (2022). GO-Surf: Neural feature grid optimization for fast, high-fidelity RGB-D surface reconstruction. *arXiv*. doi: 10.48550/arXiv.2206.14735.

[20] Wang., S.-M., Yaqin, M.A., & Hsu, F.-H. (2022). AHP-based assessment of developing online virtual reality services with progressive web apps. *EasyChair Preprint*, 9501, 1-12.

[21] Woo, J., Hong, S., Kang, D., & An, D. (2024). Improving the quality of experience of video streaming through a buffer-based adaptive bitrate algorithm and gated recurrent unit-based network bandwidth prediction. *Applied Sciences*, 14(22), article number 10490. doi: 10.3390/app142210490.

[22] Yeregui, I., Mejías, D., Zorrilla, M., Viola, R., Astorga, J., & Jacob, E. (2025). Leveraging 5G physical layer monitoring for adaptive remote rendering in XR applications. *arXiv*. doi: 10.48550/arXiv.2505.22123.

[23] Žádník, J., Mäkitalo, M., Vanne, J., & Jääskeläinen, P. (2022). Image and video coding techniques for ultra-low latency. *ACM Computing Surveys (CSUR)*, 54(11), article number 231. doi: 10.1145/3512342.

Дослідження засобів оптимізації продуктивності VR-додатків у браузерах з низькою пропускною здатністю

Ян-Віктор Латишев

Аспірант
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
03056, просп. Берестейський, 37, м. Київ, Україна
https://orcid.org/0009-0000-6133-2736

Ганна Власюк

Доктор технічних наук, професор
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
03056, просп. Берестейський, 37, м. Київ, Україна
https://orcid.org/0000-0002-7382-0435

Анотація. Актуальність дослідження полягала у необхідності підвищення продуктивності WebVR-додатків у браузерних середовищах за умов обмеженої пропускної здатності мережі, що є важливим для користувачів із повільним або нестабільним інтернет-з'єднанням. Метою роботи була розробка і оцінка ефективних методів, які дозволяють зменшити час завантаження сцен, оптимізувати обсяг передаваних даних та підтримувати стабільну частоту кадрів без зниження якості взаємодії. Для дослідження застосовано експериментальне моделювання середовища віртуальної реальності із використанням обмежених параметрів мережі. В роботі використано відкриті бібліотеки для тривимірної графіки з можливістю прогресивного завантаження 3D-ресурсів. Реалізовано та порівняно декілька методів оптимізації, зокрема багаторівневе деталювання моделей, відкладене завантаження, спрощення геометрії на сервері, компресію даних і декомпресію текстур. Оцінювання ефективності проводилось за показниками часу завантаження, обсягу переданого трафіку, плавності відтворення та відгуку системи. Результати дослідження продемонстрували, що поєднання багаторівневого деталювання з стисненням текстур дозволяє скоротити обсяг переданих даних до 70 % без помітної втрати якості зображення. Виявлено, що використання прогресивного завантаження значно знижує час первинного відображення сцени, кешування на боці клієнта зменшує повторний трафік на 90 %. Найефективнішою виявилася стратегія початкового завантаження спрощених моделей з подальшим асинхронним додаванням детальних об'єктів, що підтримує стабільну роботу навіть за низької швидкості інтернету. Практична цінність полягає у застосуванні розроблених методів при створенні WebVR-додатків для освіти, медицини, торгівлі та розваг, особливо в умовах обмеженого інтернет-з'єднання. Це сприяє розширенню аудиторії та зниженню витрат на інфраструктуру

Ключові слова: WebXR; прогресивне завантаження; LOD; кешування; стиснення текстур; Three.js; A-Frame