Dean&Francis
ISSN 2959-6157

Analysis of the Current Status of Web3D Real time Rendering Implementation

Guanxin Lai

[1]Robotics and Intelligent Devices, Fuzhou University, Fujian, China
*Corresponding author: 832402211@fzu.edu.cn

Abstract:

The fundamental task of Web3D real-time rendering is to achieve high-quality, interactive 3D graphics rendering and user experience within a web browser. It significantly enriches the dimensions of human-computer interaction and has a substantial impact on the display, marketing, education, and other aspects of numerous industries. Existing research has greatly enhanced the accuracy and speed of Web3D real-time rendering for applications in complex virtual environments and on a variety of hardware devices.. Depending on the technological framework used, it can be classified into the following three categories: browser frameworks, server frameworks, and client-cloud collaborative frameworks. This paper systematically introduces representative methods for implementing rendering under these three frameworks, along with their basic processes, strengths, and weaknesses. It also compares the visual quality and associated overhead of rendering achieved under different frameworks. Finally, the paper summarizes the current challenges in Web3D real-time rendering, provides an outlook on its future development directions, and offers some new insights.

Keywords: Browser Framework; Server Framework; Client-Cloud Collaborative Framework.

1. Introduction

Web3D real-time rendering refers to the technology that instantly generates and displays interactive 3D graphics within a web browser. Its core value lies in its accessibility, enabling users to enjoy immersive experiences across platforms without installing additional plugins. With the widespread adoption of internet technology, improvements in hardware performance, and the growing demand for richer online interactive experiences, Web3D real-time rendering has gradually become a key element in meeting people‘s expectations for digital aesthetics. Web3D has driven advancements in numerous fields such as e-commerce, online education, digital twins, and the metaverse, serving as a foundational building block for next-generation immersive internet applications. Current research in this area focuses on developing lightweight Web3D solutions, collaborative dynamic light and shadow cloud rendering, and global illumination for dynamic Web3D scenes [1]. In line with these directions, previous studies have proposed

ISSN 2959-6157

several methods to advance Web3D real-time rendering. These include an integrated „computing-transmission-rendering-caching“ technical framework to overcome bottlenecks in loading, transmitting, and rendering large-scale scenes; dynamic allocation of lighting and shadow rendering tasks based on front-end device performance to enable client-cloud collaborative real-time rendering; and the introduction of a „Light Tree“ pre-storage structure combined with a heatmap-based scheduling algorithm to optimize real-time reconstruction of global illumination in dynamic scenes [2].

Throughout the development of Web3D real-time rendering, several key challenges have been encountered, primarily including the following three aspects: 1. Enabling 3D graphics to run and render within a web browser. 2. Achieving cinematic-quality, high-fidelity rendering while ensuring the content security of high-value digital assets. 3. Striking an optimal balance between superior visual quality and smooth performance.

Looking back at the development of Web3D real-time rendering, several challenges emerged. Before 2010, displaying 3D content on web pages required users to install bulky third-party plugins, which contradicted the web‘s inherent lightweight nature. The emergence of WebGL resolved this issue. WebGL is a free, cross-platform, low-level API that allows browsers to directly utilize the device‘s GPU for graphical computations via JavaScript. However, due to WebGL‘s high technical barrier, high-level JavaScript graphics libraries and engines like Three.js and Babylon.js were developed. These tools abstract the underlying APIs, significantly reducing the difficulty of use [3]. This marked the true realization of real-time rendering based on a web framework. This approach greatly reduced user interaction time, but the limited rendering capabilities of the web often led to unsatisfactory final results.To address the issue of rendering quality, server-based real-time rendering frameworks emerged. Under this framework, the primary rendering workload is handled by the server. The rendered output is then transmitted to the web front-end as an image or video stream, where the front-end only displays the final result without participating in any rendering calculations. Simultaneously, valuable 3D data remains securely stored on the server, enhancing its security.

However, while this approach significantly improves rendering quality, it places higher demands on server performance and increases the burden on network transmission. Unstable network conditions or insufficient bandwidth can lead to delays in the image stream transmission, negatively impacting the user experience. Furthermore, users cannot interact directly on the web front-end but must provide indirect feedback through the server, which reduces the real-time nature and flexibility of the interaction to some extent.Consequently, a framework combining collaborative rendering between the web front-end and the server has emerged. Within this framework, tasks with low rendering difficulty are typically assigned to the web front-end, while those with high rendering difficulty are handled by the server. The respective rendering results are then composited and presented on the web front-end. This approach ensures both smooth user interaction and high-quality rendering [4].

Focusing on the aforementioned issues and challenges, this paper analyzes and reviews the research progress and current state of Web3D real-time rendering frameworks, detailing the implementation methods and representative approaches of different Web3D real-time rendering frameworks, and provides an outlook on the potential future development directions in the field of Web3D real-time rendering frameworks.

2. The Method of Different Frameworks

2.1 The method of Web

In this framework, the primary rendering workload is handled by the web. The server transmits data such as 3D models, textures, and lighting information to the web front-end. The web front-end then utilizes engines to perform data parsing, user interaction processing, and rendering tasks, ultimately presenting the 3D rendered graphics. This implementation path significantly reduces user interaction latency. Developers use high-level APIs from frameworks like Three.js to construct a virtual 3D scene within the browser. Subsequently, the framework employs a loop mechanism to convert the scene data into a series of underlying WebGL instructions for each frame. These instructions ultimately drive the user’s local GPU to execute the rendering pipeline, calculate each frame’s image, and display it on the web canvas. All computations are completed entirely and independently within the user’s device [5]. Figure 1 shows the flowchart of 3D real-time rendering based on the web framework.

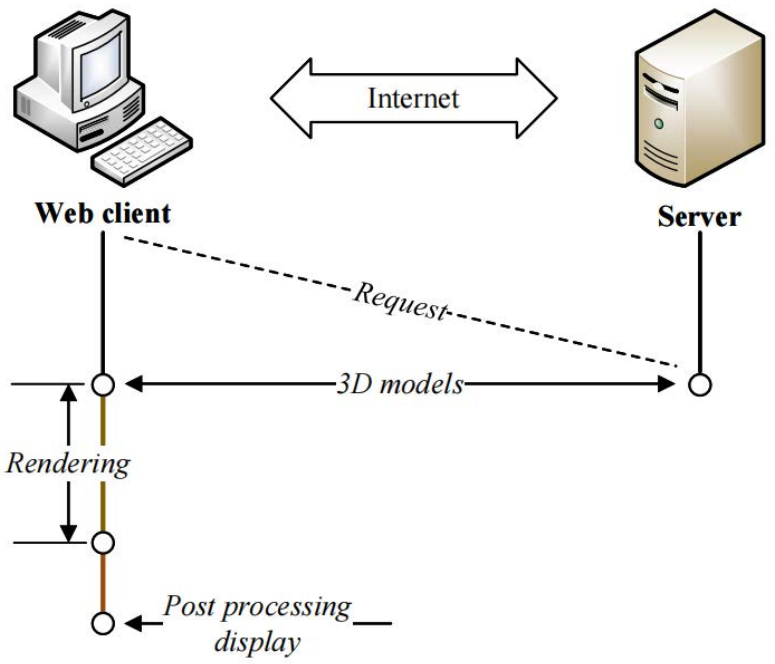

Fig. 1 The flowchart of 3D real-time rendering based on the web framework

2.2 Sever

2.2.1 The method of Sever

The user‘s interactive input is uploaded via the network to cloud servers equipped with powerful GPUs. These servers run a full 3D engine to render high-definition frames, which are then compressed in real-time into a video stream using H.265 technology. This stream is transmitted back over the network to the user‘s lightweight client. The client essentially functions as a „video player,“ with its core task being to receive and decode the video stream, then display the decoded consecutive image frames in full screen. All complex graphical computations are entirely performed in the remote cloud. Figure 2 shows the flowchart of 3D real-time rendering based on the server framework.

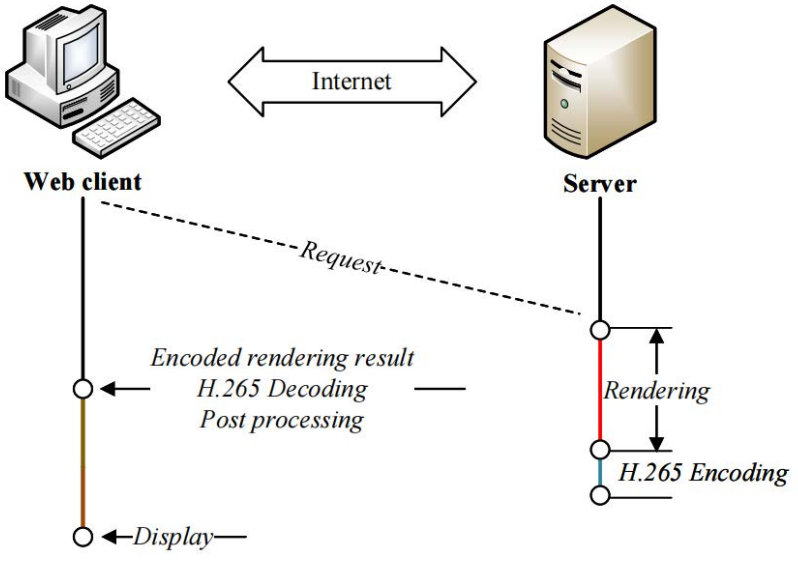

Fig. 2 The flowchart of 3D real-time rendering based on the server framework

ISSN 2959-6157

2.2.2 The theory of H.265

The workflow of H.265 begins by dividing a video frame into more flexible and larger Coding Tree Units (CTUs), followed by prediction for these units [6]: For static areas, intra-frame prediction is employed, which utilizes information from already encoded parts within the same frame to predict the current block. For moving areas, inter-frame prediction is used, which involves motion estimation and motion compensation to find the most matching block in previous and/or subsequent frames, retaining only the residual data and motion vectors. Next, the residual data after prediction undergoes transformation and quantization, concentrating the effective information and discarding visual details less perceptible to the human eye. This is a crucial step for achieving compression. Subsequently, entropy encoding is applied to all this information to form the final compressed bitstream. On the decoding end, a reverse process is executed: entropy decoding, inverse quantization, inverse transformation, and prediction compensation are performed to reconstruct the video frames. To eliminate blocking artifacts and improve reconstruction quality, in-loop filtering techniques such as deblocking filtering and Sample Adaptive Offset (SAO) are applied during the decoding process.

2.3 The method of Client-Cloud Collaborative

A central scheduler intelligently decomposes the complete rendering task based on real-time network conditions, cloud server load, and client computing power [7]. It assigns parts requiring ultra-high visual quality but less sensitive to latency to the cloud servers for rendering, while allocating tasks demanding extremely low latency to the local client for processing [8]. Subsequently, the cloud delivers its rendering results to the client. An efficient compositor on the client side then performs precise synchronization and fusion of these cloud-rendered elements with the locally rendered visuals, ultimately outputting a final image that combines both high quality and high smoothness. This approach achieves an optimal balance between visual fidelity and latency. Figure 3 shows the flowchart of 3D real-time rendering based on the client-cloud collaborative framework.

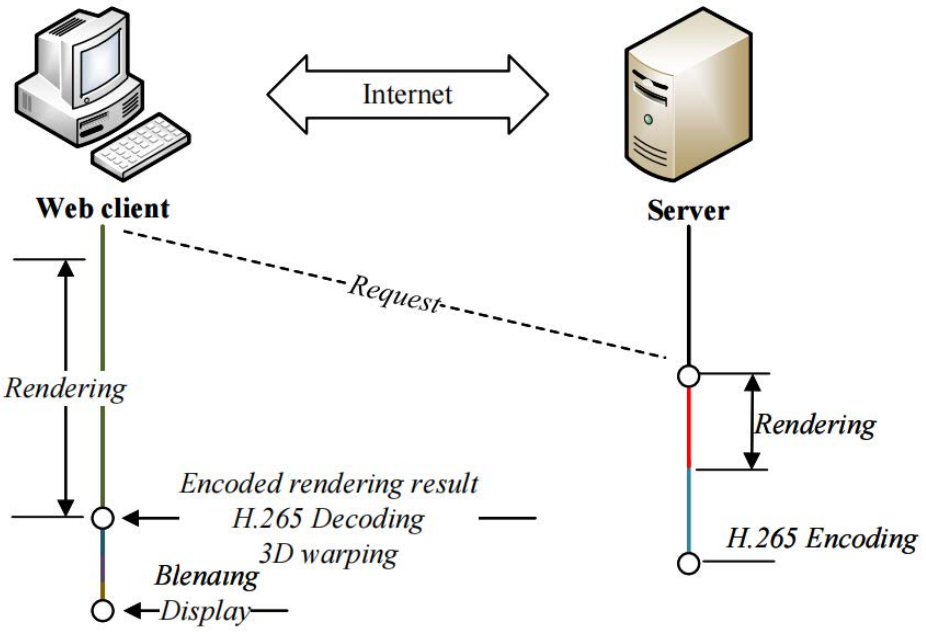

Fig. 3 The flowchart of 3D real-time rendering based on the client-cloud collaborative framework

3 Experiment

3.1 Client-Side Rendering Based on Web

3.1.1 Rendered image implemented using Three.js.

Three.js is a Web-based 3D JavaScript graphics library launched by Ricardo Cabello in 2010. It is an open-source JavaScript framework built on the WebGL API. Due to the high technical barrier of using WebGL directly, Three.js developed a highly usable graphics library by encapsulating and simplifying the WebGL interfaces. This significantly simplifies the implementation details of 3D render-

ing without losing the flexibility of WebGL, but at the cost of sacrificing some visual appeal and attractiveness of the rendered scene. Figure 4 shows a rendered image implemented using Three.js.

Fig. 4 Rendered image implemented using Three.js[9]

3.1.2 Rendered image implemented using Babylon.js.

Babylon.js is a WebGL-based Web3D engine primarily developed by Microsoft. This engine supports both JavaScript and TypeScript programming languages and can run in all browsers compatible with WebGL or WebGPU. Its latest version, „Babylon.js 4.2," introduces the Babylon Native framework and enhances support for Physically-Based Rendering (PBR), enabling soft transparent shadows for transparent objects. Compared to Three.js, it delivers better rendering results when handling specific simple scenes; however, both still have room for improvement when rendering complex scenes. Figure 5 shows a rendered image implemented using Babylon.js.

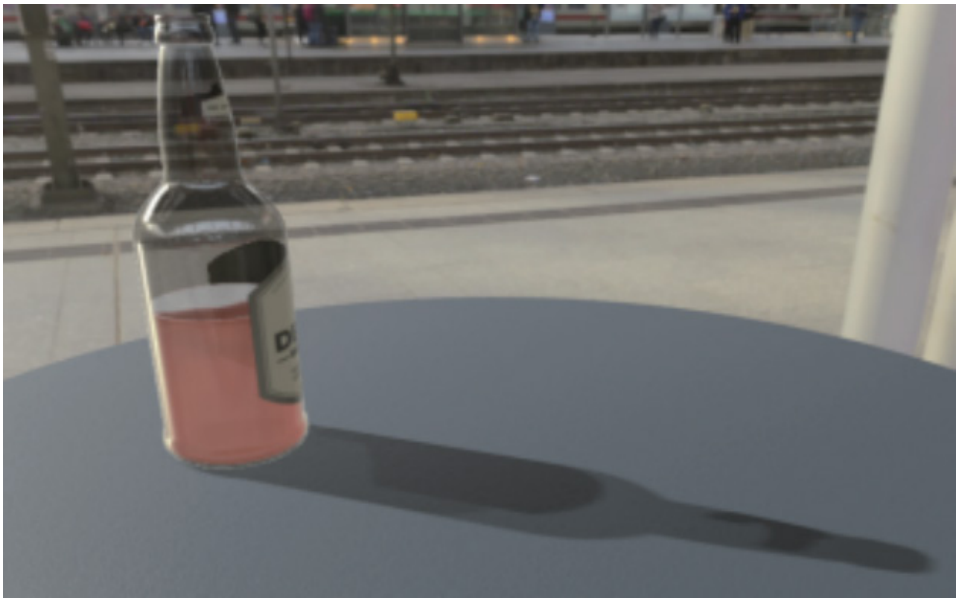

Fig. 5 Rendered image implemented using Babylon.js[9]

3.2 Server-Side Rendering

Compared to image rendering implemented through Web frameworks, the 3D rendering generated using pixel streaming technology achieves a quantum leap in visual performance: its global illumination and real-time ray tracing techniques deliver more realistic lighting and shadows, along with physically accurate reflections and refractions; an ultra-high precision PBR material system

ISSN 2959-6157

and 4K/8K resolution texture mapping endow surfaces with richer microscopic details and authentic material properties; through modeling with hundreds of millions of polygons and presentation at nanometer-level geometric precision, model structures are restored with extreme detail. Ultimately, this creates an immersive visual experience enhanced by cinematic post-processing effects such as depth of field, motion blur, and volumetric fog. In terms of overall scene realism and artistic expressiveness, it reaches a level of quality that is unattainable for traditional WebGL real-time rendering. Figure 6 shows a rendering example from Unreal Engine 4.0, where the 3D content is delivered to the web front-end via „Pixel Streaming".

Fig. 6 Rendered images implemented using H.265[9]

3.3 Server-Side Rendering

3.3.1 Rendered image implemented using Hybrid Rendering.

The following image demonstrates 3D real-time rendering achieved through Hybrid Rendering. Compared to images rendered under a server-based framework, the visuals produced by the client-cloud framework reduce the consumption of device computing power and data transmission while ensuring the quality of the rendered scene [10]. Figure 7 shows the rendering result accomplished using hybrid rendering technology via the client-cloud collaborative framework.

Fig. 7 The rendering result accomplished using hybrid rendering technology via the client-cloud collaborative framework [9]

Table 1 compares the performance characteristics of three rendering frameworks—pure Web, pure server, and cli-

ent-cloud collaborative—across five dimensions: bandwidth cost, computational cost, interactivity, rendering quality, and hardware cost. Overall, the Web framework relies on local client-side computation, offering low bandwidth and server costs along with good interactivity, but is limited in rendering quality. Server-based rendering achieves high-quality visuals but incurs high bandwidth, computational, and hardware costs, coupled with significant interaction latency. The client-cloud collaborative framework strikes a balance across these metrics. By adopting a distributed rendering strategy, it simultaneously addresses rendering quality, responsiveness, and cost-effectiveness, making it suitable for application scenarios that demand both high-quality graphics and real-time interaction.

Table 1. Multi-dimensional Comparison of Three Rendering Frameworks

Evaluation Dimension	Web	Server	Collaboration
bandwidth cost	Low	High	Medium
Rendering computation cost	Low	High	Medium
interactivity	High	Low	Medium
Render effects performance	Light	Heavy	Medium
Hardware cost	Cheap	Expensive	Medium

4 Discussion

Web3D real-time rendering, as an important issue in computer graphics, has garnered significant attention. With the improvement of hardware capabilities and rendering techniques, Web3D real-time rendering frameworks have undergone substantial changes, yet several challenges remain. Based on the client-cloud framework, which is still in its early stages of development, this section discusses the unresolved problems and prospects for future research directions.

4.1 Intelligent and Adaptive Dynamic Task Allocation Algorithms

The core of client-cloud collaboration lies in „how to split the workload," but this decision-making process is extremely complex. An ideal allocation strategy needs to perceive in real-time factors such as network bandwidth, latency fluctuations, server load, client GPU computing power, and the complexity of the current scene content. This constitutes a multi-variable dynamic optimization problem, where a change in any single factor could render the current strategy ineffective. Developing a universal algorithm capable of instantly responding to these changes and making near-optimal decisions presents the greatest challenge. Therefore, developing an intelligent scheduler based on Reinforcement Learning (RL) is proposed, with the ultimate goal of improving user experience. This involves designing a reward signal that balances rendering performance and visual quality. Key metrics could include: high frame rate stability, low end-to-end latency, low visual stutter ratio, while maintaining high visual fidelity as much as possible. Penalties would include factors like task timeouts, frame drops, and user-perceptible quality degradation. Through continuous trial-and-error and a reward mechanism, the system would autonomously learn the optimal task allocation strategy for different scenarios.

4.2 Lack of Cross-Platform, Interoperable Standardized Protocols

Currently, client-cloud collaborative rendering mostly relies on custom, proprietary solutions proposed by various companies or research teams. This leads to issues such as communication problems between the cloud and client, mismatched data transmission formats, and lack of unified synchronization mechanisms. The absence of standards results in ecosystem fragmentation. Developers may need to write different adaptation code for different cloud service providers, which significantly increases development costs and technical barriers, thereby hindering the widespread adoption and application of the technology. Consequently, promoting the establishment of open standard protocols by industry consortia—defining data formats, synchronization mechanisms, and capability negotiation methods—can help lower the development barrier and resolve this issue.

4.3 New Challenges in Security and Privacy

In client-cloud collaborative rendering, original 3D assets may be split, with some parts processed in the cloud and others on the client side. This introduces new attack surfaces: hackers could intercept communication between

ISSN 2959-6157

the client and the cloud to reconstruct or steal critical assets, or disrupt rendering results by injecting forged data packets. In the future, Trusted Execution Environments could be adopted to protect data during computation in the cloud, coupled with encryption of cloud data, to fundamentally ensure asset security.

5 Conclusion

This paper provides a review of the progress in Web3D real-time rendering frameworks, introducing representative methods and their implementation principles under different frameworks. Secondly, it compares the rendering effects of 3D scenes across different frameworks, analyzing the advantages and disadvantages of each in terms of rendering quality, speed, hardware consumption, and user interactivity. Finally, starting from the still-maturing client-cloud framework, it analyzes existing challenges such as scheduler algorithm optimization, the lack of standardized protocols, and security and privacy concerns, while proposing future directions. Based on these prospects, to enhance the effectiveness and efficiency of Web3D real-time rendering, the current technical focus should be on developing intelligent schedulers based on Reinforcement Learning (RL) or other technologies. In terms of development prospects, establishing standard protocols to break down technical barriers and implementing robust encryption and firewalls are essential to ensure data security.

References

[1] Li L, Qiao X, Lu Q, Ren P and Lin R, Rendering Optimization for Mobile Web 3D Based on Animation Data Separation and On-Demand Loading, in IEEE Access, 2020.

[2] Li K, Zhao H, Zhang Q and Jia J, CEBOW: A Cloud-Edge-Browser Online Web3D Approach for Visualizing Large BIM Scenes, in WILEY, 2022.

[3] Boutsi A M, Ioannidis C and Verykokou S, Multi-Resolution 3D Rendering for High-Performance Web AR, in MDPI, 2023.

[4] Xie C, et al., Real-Time High Realistic Web Display Method of Complex 3D Model, in IEEE Access, 2021.

[5] Liu C, Song H, Fang T, Ou Q, Yu G, You T and Ying M, Web-Cloud Collaborative Mobile Online 3D Rendering System, in WILEY, 2022.

[6] Wu J, Xu H and Vassilev T, Design of 3D Rendering Platform Based on Cloud Computing, in IEEE Access, 2016.

[7] Park M, Yoo B, Moon J Y and Seo J H, InstantXR: Instant XR Environment on the Web Using Hybrid Rendering of Cloud-based NeRF with 3D Assets, in ACM, 2022.

[8] Zhao Q P, Ten Scientific and Technical Problems in Virtual Reality, in SciEngine, 2017.

[9] Yu G, Liu C, Fang T, Jia J, Lin E, He Y and Fu S, A Survey of Real-Time Rendering on Web3D Application, in Elsevier, 2023.

[10] Hou X, Lu Y and Dey S, Wireless VR/AR with Edge/Cloud Computing, in International Conference on Computer Communication and Networks (ICCCN), 2017.