Simulation and Optimization of Large Passenger Ship Based on Unity 3D

Ya Zhang, Tong Sun, Zhicong Tao, Juncen Wu, Xianda Meng

Abstract—**With the rapid advancement of virtual reality (VR) technology and maritime engineering, the demand for high-performance simulation systems has significantly increased. However, when developing large-scale 3D ship virtual simulation often encounters challenges, such as excessive polygon counts in models and unsustainable computing resource consumption resulting in memory overload, system lag, and delayed loading. This paper describes the development process of a virtual reality simulation for the cruise ship "Tianhai New Century", alongside the system comprising four optimization strategies and the corresponding experimental results: (1) A six-degree-of-freedom (6-DOF) mathematical model of ship motion is established, and a hybrid modeling approach combining NURBS curves, polygon meshes, and primitive modeling to construct a high-precision 3D digital ship model. (2) A simulation system integrating virtual scene roaming, Unity Graphical User Interface (UGUI), and dynamic sailing scenarios has been developed using Unity3D; (3) Four optimization strategies—including a dual-layer clipping algorithm, code-loading optimization, texture compression, and hull surface mesh simplification. Experimental results demonstrate a 4× improvement in rendering efficiency (from 30 FPS to 120 FPS) and reduce the memory consumption by 50.17% (from 2.81 GB to 1.40 GB), reducing graphics rendering complexity, vertex counting and shadow calculation overhead, etc., and providing a scalable solution for subsequent simulation.**

Index Terms—**Virtual reality, 3D modeling, Unity 3D, Optimization strategy, ship design**

I. Introduction

VIRTUAL reality (VR) is an advanced computer technology that generates immersive three-dimensional environments through computational systems [1], combining virtual imagery with realistic perceptions to create and present lifelike simulated experiences for users [2]. With the continuous development of software and hardware capabilities, coupled with sustained enhancements in user experience design, VR has emerged as a pivotal tool in the digital era, exerting broad influence across diverse fields [3], [4], [5].

Against this backdrop, the International Maritime Organization (IMO) conducted a revision of the International Convention on Standards of Training, Certification and Watchkeeping for Seafarers (STCW Convention) in 2010, mandating the enhancement of simulation training standards for navigation simulators [6]. In view of the fact that large passenger ships, as an important part of maritime transport, face significant challenges in navigation and passenger service, it is of great significance to develop a navigation simulation system for large passenger ships in order to improve the crew's overall knowledge of large passenger ships and their ability to cope with a variety of sea conditions and service needs, as well as to provide passengers with a more comfortable and convenient travel experience.

In recent years, domestic virtual reality technology has rapidly advanced in the maritime sector, enabling the development of diverse ship types such as LNG carriers, engineering vessels, and dredgers [7], [8], [9]. However, due to the large number of object models of the whole ship, the real-time performance in the simulation process may be affected, resulting in excessive memory occupation of the scene computer, and even problems such as stuck and unsmooth switching of scenes may occur. To address these challenges, this study focuses on a large passenger ship simulation system and proposes a hybrid modeling methodology combined with comprehensive scene optimization techniques. Key strategies include: (1) A dual-layer clipping algorithm to dynamically manage rendering workloads; (2) Code-loading optimization and texture compression to reduce computational overhead; (3) Surface mesh simplification of the main hull to minimize vertex counts while preserving model fidelity. These optimizations collectively enhance simulation efficiency by reducing memory usage, graphical rendering complexity, and triangular facet redundancy. The system architecture is illustrated in Fig. 1.

Manuscript received January 2, 2025; Revised May 10, 2025.

Ya Zhang is an associate professor of School of Navigation and Ship Engineering, Dalian Ocean University, Dalian, 116023, China (*Corresponding author provided to e-mail:zhangya_0426@163.com)

Tong Sun is a postgraduate student at School of Navigation and Ship Engineering, Dalian Ocean University, Dalian, 116023, China (e-mail:suntong621520@163.com).

Zhicong Tao is a postgraduate student at School of Navigation and Ship Engineering, Dalian Ocean University, Dalian, 116023, China (e-mail: 2821036288@qq.com).

Juncen Wu is a postgraduate student at School of Navigation and Ship Engineering, Dalian Ocean University, Dalian, 116023, China (e-mail: 867647802@qq.com).

Xianda Meng is a postgraduate student at School of Navigation and Ship Engineering, Dalian Ocean University, Dalian, 116023, China (e-mail: 2571815174@qq.com).

II. Constructing 3D Visualisation Model of Virtual Scene

A. Motion mathematical model construction

1) Coordinate system

Before building a mathematical model of motion, it is first necessary to confirm the coordinate system. Using the fixed coordinate system E_1-$\xi_1\eta_1\varsigma_1$ and the kinematic coordinate system E-$\xi\eta\varsigma$. To simplify the equation, the origin E is placed on the centre of gravity of the hull so that this coordinate system moves with the motion of the hull, as

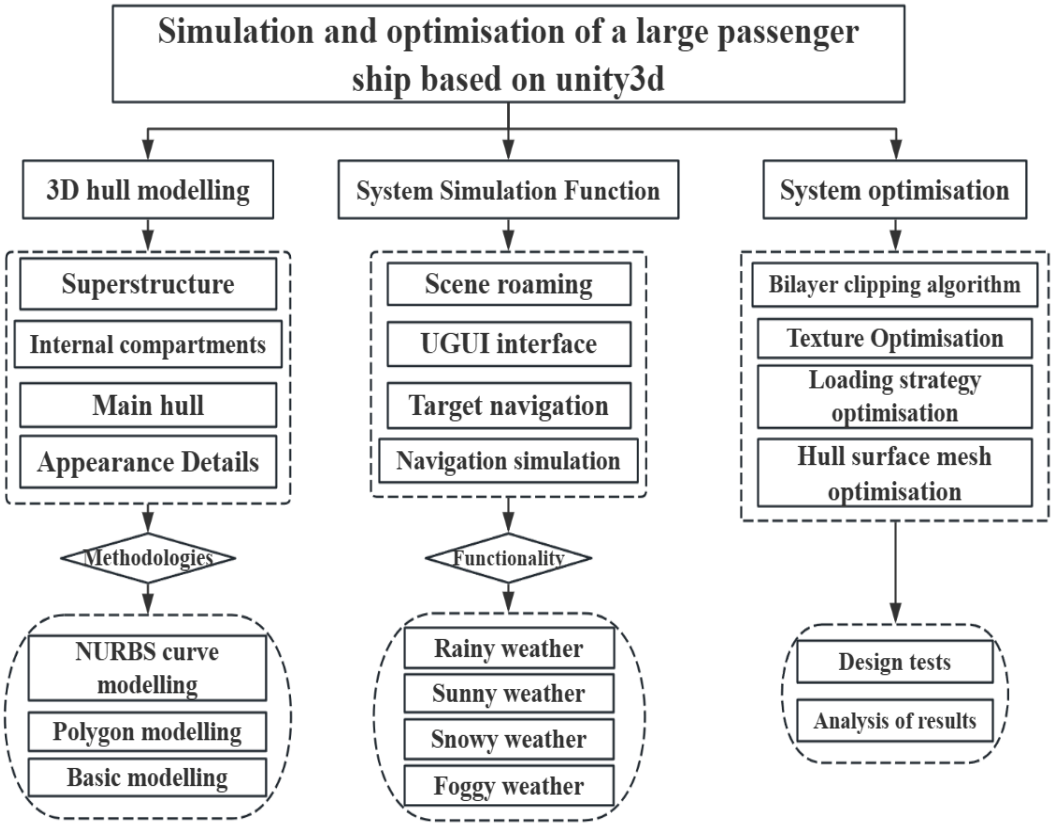


Fig. 1. General framework of the simulation system

shown in Fig. 2. Provide that the $E_1\xi_1$ points due north, the $E_1\eta_1$ points due east, and the $E_1\varsigma_1$ points to the centre of the earth. The $E\text{-}\xi\eta\varsigma$ system is a ship-attached coordinate system with the origin at the focus of the ship, with the $E\xi$ pointing to the bow, the $E\eta$ pointing to the starboard side, and the $E\varsigma$ pointing to the keel.

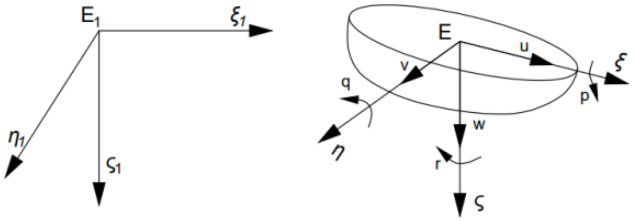


Fig. 2. Fixed and kinematic coordinate systems

2) Ship motion equation

According to the principle of rigid body mechanics, the six-degree-of-freedom equation of motion for ship motion can be described as [10]:

$$\begin{aligned} &m(\dot{u}-vr+wq)=X \\ &m(\dot{v}-wp+ur)=Y \\ &m(\dot{w}-uq+vp)=Z \\ &I_\xi\dot{p}+(I_\varsigma-I_\eta)qr=K \\ &I_\eta\dot{q}+(I_\xi-I_\varsigma)rp=M \\ &I_\varsigma\dot{r}+(I_\eta-I_\xi)pq=N \end{aligned} \tag{1}$$

Due to the complexity of maritime conditions, the following assumptions are adopted: (1) The ship operates in infinitely wide waters; (2) The influence of waves on propeller and rudder forces is neglected; (3) Forces acting on the ship, propeller, rudder, and waves are analyzed independently; (4) Interactions among the ship, propeller, and rudder are treated under static water maneuverability conditions. Forces and moments are calculated based on Newton's laws of motion and the Mathematical Model Group (MMG) framework. The (6-DOF) mathematical equations of ship motion in the kinematic coordinate system are formulated as follows:

$$\begin{aligned} &(m+m_\xi)(u'-vr+wq)=X_H+X_P+X_R+X_{wind}+X_{wave} \\ &(m+m_\eta)(v'-wp+ur)=Y_H+Y_P+Y_R+Y_{wind}+Y_{wave} \\ &(m+m_\varsigma)(w'-uq+vp)=Z_H+Z_P+Z_R+Z_{wind}+Z_{wave} \\ &(I_{\xi\xi}+J_{\xi\xi})p'+(I_{\varsigma\varsigma}-I_{\eta\eta})rq=K_H+K_R+K_{wind}+k_{wave} \\ &(I_{\eta\eta}+J_{\eta\eta})q'+(I_{\xi\xi}-I_{\varsigma\varsigma})rp=M_H+M_P+M_R+M_{wind} \\ &(I_{\varsigma\varsigma}+J_{\varsigma\varsigma})r'+(I_{\eta\eta}-I_{\xi\xi})pq=N_H+N_P+N_R+N_{wind}+N_{wave} \end{aligned} \tag{2}$$

Where: m is the mass of the ship; $X_H, Y_H, Z_H, K_H, M_H, N_H$ are the forces and moments of the ship's hull in the direction of the corresponding six degrees of freedom; variables labelled P、R are propeller and rudder forces and moments, respectively; variables labelled $wind, wave$ and $current$ are wind, wave and current forces and moments, respectively; $m_\xi, m_\eta, m_\varsigma$ are the added mass of the ship in the longitudinal, transverse and vertical directions, respectively; u, v, w, p, q, r are the translational velocities of the transverse, longitudinal and vertical swings and the angular velocities of the transverse, longitudinal and bow swings of the $E\xi$, $E\eta$ and $E\varsigma$ axes, respectively; $I_{\xi\xi}$, $I_{\eta\eta}$, $I_{\varsigma\varsigma}$ are the moment of inertia around the ξ , η and ς axes, respectively.

B. Motion mathematical model construction

The construction of 3Dscenes serves as the cornerstone of virtual simulation and a critical determinant of simulation quality. Large passenger ships, characterized by numerous functional rooms and complex power systems, require extensive modeling tasks encompassing a wide range of components. These tasks often lead to increased computational resource demands—particularly memory consumption—resulting in prolonged system response times and reduced efficiency in overall ship digital modeling. In this study, the "Tianhai New Century" large passenger ship is utilized as a case study. Professional 3D modeling software (Rhino) and rendering software (KeyShot) are employed for virtual digital modeling [11]. During the modeling process, a hybrid strategy is developed by integrating NURBS curve modeling, polygon modeling, and primitive modeling techniques [12].This strategy dynamically selects optimal modeling methods based on object-specific characteristics, significantly improving modeling efficiency while enabling systematic classification and organization of components. These advancements establish a robust foundation for real-time interactive responsiveness and scene loading. The detailed workflow of the hybrid modeling methodology is illustrated in Fig. 5.

1) NURBS curve modeling

NURBS (Non-Uniform Rational B-Spline) curve modeling is a technique that constructs precise curves using multiple control points. These control points are highly flexible and can be freely adjusted to fine-tune the shape of the curve. Additionally, multiple NURBS curves can be combined to form complex surfaces [13]. Given the unique and complex surface modeling requirements of ship hulls, the application of NURBS curve modeling techniques enhances the smoothness and accuracy of hull surfaces while optimizing modeling efficiency. This approach ensures that the digital model of the ship hull more accurately reflects the physical characteristics of real-world structures [14].

2) Polygon modeling

Polygonal modeling is a widely adopted technique in the 3D modeling field, enabling the rapid and accurate construction of complex shapes. This method involves key sub-object elements, including vertices, edges, borders, polygons, and elements. Unlike traditional approaches limited to triangular or quadrilateral faces, polygonal modeling supports the creation of arbitrary polygonal surfaces, significantly enhancing modeling flexibility. While preserving geometric fidelity, this technique allows optimization of face counts by reducing redundant polygons and simplifying structural complexity. For irregular objects in the superstructure of passenger ships—such as chimneys, chairs, lifeboats, beds, foremasts, mid-masts, staterooms, and recreational facilities (e.g., basketball courts)—polygonal modeling is employed to achieve high-detail representations.

3) Basic modeling

Basic modeling focuses on objects with regular geometries and simplified appearances, such as windows, doors, railings, stairs, and communication domes. These components are efficiently modeled using Rhino, a professional 3D modeling software, ensuring rapid completion while maintaining structural accuracy.

C. Model rendering

The realism of model rendering depends on several key steps, including texture mapping, material configuration, and lighting arrangement, which collectively determine the visual fidelity and appeal of the rendered model. First, textures are meticulously adjusted using real photographs and high-definition images sourced from the internet, processed through Photoshop software. These refined textures are then applied to detailed components of the ship, such as logos, signage, interior furnishings, and other intricate elements, significantly enhancing the model's realism and detail richness. Lighting effects are optimized through a hybrid approach that combines pre-baked global illumination with real-time local lighting, achieving a balance between rendering performance and visual quality [15]. Finally, the model is rendered using KeyShot, a professional 3D rendering software, is used to produce a complete and highly realistic digital asset. The results are illustrated in Fig. 3.

Fig. 3. 3D rendering of the Tianhai New Century passenger ship model

D. Construction of the marine environment

The accuracy of wave modeling plays a critical role in determining the overall fidelity of ocean scene simulations [16]. To balance high realism with computational efficiency in marine environment simulation, this study leverages the Ceto Ocean System plugin within the Unity3D platform to develop a virtual ocean environment that achieves both visual authenticity and operational efficiency By meticulously adjusting the core parameter data, which encompasses but is not limited to mesh subdivision accuracy, wave dynamic characteristics (such as speed, period, and attenuation rate), environmental variables (including wind speed control), and visual effect intricacies (like the generation of water foam), we have successfully attained precise control and a high degree of customization over the marine environment. The realized marine environment is depicted in Fig. 4.

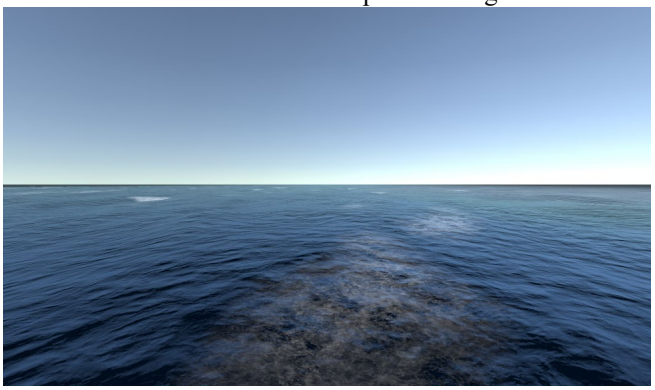


Fig. 4. Creating a marine environment in unity3D

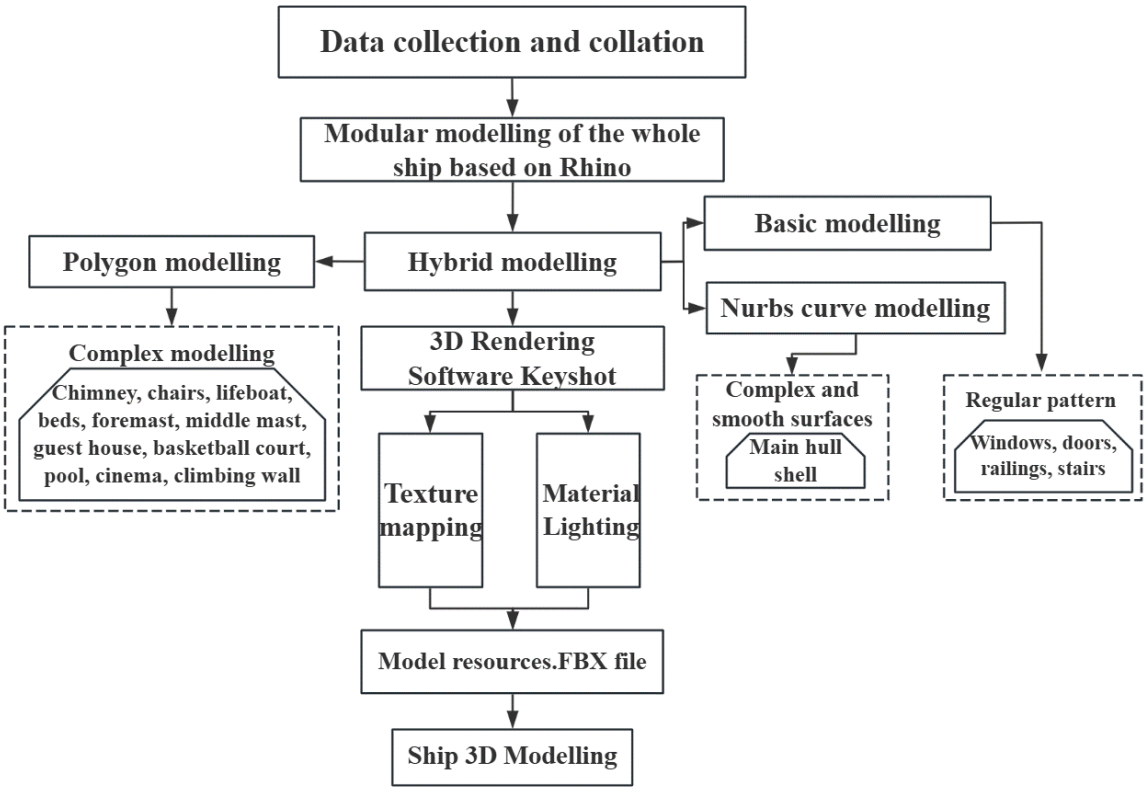


Fig. 5. Modelling flowchart

III. Passenger Ship Virtual Simulation System Implementation and Optimisation

A. Virtual roaming of ships

The system employs Unity3D as the 3D interactive simulation development platform, implementing functionalities such as virtual roaming, user interface (UI), navigation environment simulation, and scene optimization. Virtual roaming technology enables the digital reconstruction of real-world environments, allowing users to freely navigate and interact within the simulated virtual space while observing from multiple viewpoints and positions [17]. Two distinct roaming modes are supported: (1) Navigation Roaming: Characters automatically traverse predefined paths or scripted routes to collect environmental data; (2) Manual Roaming: A first-person perspective mode where users control character movement (forward/backward/left/right) via the WASD keys and mouse, enabling seamless viewpoint transitions. By the roaming design, passengers can virtually tour various areas of the passenger ship before departure or during the voyage, including guest rooms, restaurants, entertainment facilities, etc. to know and familiarize themselves with the ship in advance, and help crew and passengers better understand the escape procedures and safety measures. The virtual roaming of the scene is shown in Fig. 7.

During 3D virtual ship roaming, collision events analogous to real-world physical interactions frequently occur between objects during ship motion or character interactions. The system must accurately detect collisions and generate corresponding feedback to simulate realistic post-collision effects; failure to do so may result in model penetration, compromising the authenticity of the virtual environment. To ensure precise and efficient collision detection, this system employs a hierarchical bounding volume (BVH) algorithm, a graphics-based real-time collision detection method. The core strategy involves: (1) Coarse Approximation: Using slightly oversized bounding volumes with simplified geometries (e.g., boxes, spheres) to approximate complex object contours; (2) Hierarchical Refinement: Constructing a tree-like hierarchy to iteratively refine geometric approximations until they closely match the object's detailed characteristics. The collision detection process follows two stages: Broad Phase: Rapidly exclude non-colliding object pairs by checking overlaps between bounding volumes, which is computationally cheaper than direct geometric intersection tests; Narrow Phase: Perform precise intersection tests only on overlapping bounding volumes, significantly improving algorithmic efficiency. This method can not only meet the accurate design requirements, but also help to reduce the workload and complexity in the development process.

B. Unity Graphical User Interface (UGUI)

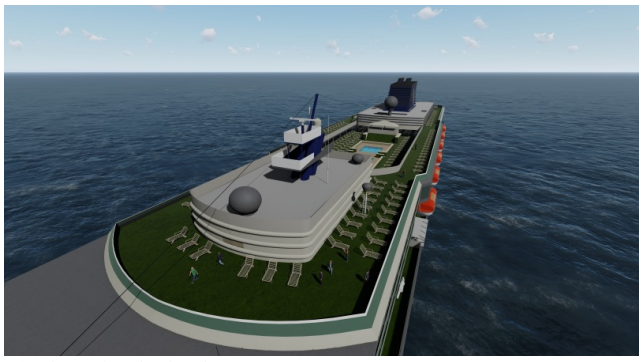
The virtual ship system integrates a wide range of devices, scenes, and functional areas. To assist users unfamiliar with the ship's layout, a dual-mode navigation interface is designed: (1) 2D Map Navigation: Provides guided tours to direct users to specific destinations; (2) 3D Eagle-Eye Navigation: Hawkeye Navigation greatly simplifies the processing of complex three-dimensional ship models. Displays real-time ship position, destination, estimated arrival time, and other critical data. Interface Layout: Main Interface: Features a high-quality ship rendering as the central visual element; Right panel: Includes the main secondary interface for navigating to each deck layer. The secondary UI interface is embedded with more detailed tertiary interfaces, such as deck-level zoning and weather selection, so that users can quickly locate and navigate to their target area;3D Model Interaction: The eagle-eye navigation simplifies complex 3D ship models, allowing users to zoom, rotate, and pan via

mouse operations for comprehensive exploration. This dual-mode navigation system ensures an intuitive and efficient user experience, particularly for users unfamiliar with the ship's layout. The 2D map provides clear guidance, while the 3D eagle-eye view offers a detailed, interactive representation of the ship's structure and surroundings. The UI design balances functionality and visual appeal, the UI interface is shown in Fig. 6.

C. Navigational environment simulation

Simulating realistic maritime weather conditions encountered during ship navigation enhances the immersive experience of virtual reality (VR) training systems, enabling students to acquire and master operational skills across diverse sea states, thereby improving pedagogical efficiency [18].This system employs Unity3D's particle system to simulate four typical maritime weather scenarios: rainfall, snowfall, fog, and daylight variations. Key parameters of the particle system—including particle size, velocity, color gradients, and emission rates—are precisely calibrated to control: (1) Cloud & Fog Dynamics: Density distribution and flow velocity; (2) Precipitation Effects: Particle size, directional trajectories, and terminal velocity of raindrops/snowflakes; (3) Lighting Conditions: Real-time adjustments to daylight intensity and spectral characteristics are implemented. To optimize computational performance, the system implements resource preloading—precomputing and caching frequently used weather effects—to mitigate rendering latency caused by dynamic resource loading, is shown in Fig. 8.

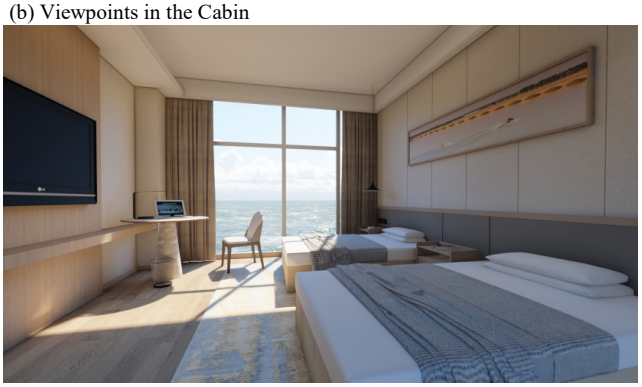
Fig. 6. UI Interactive Interface



(a) Viewpoints in the air

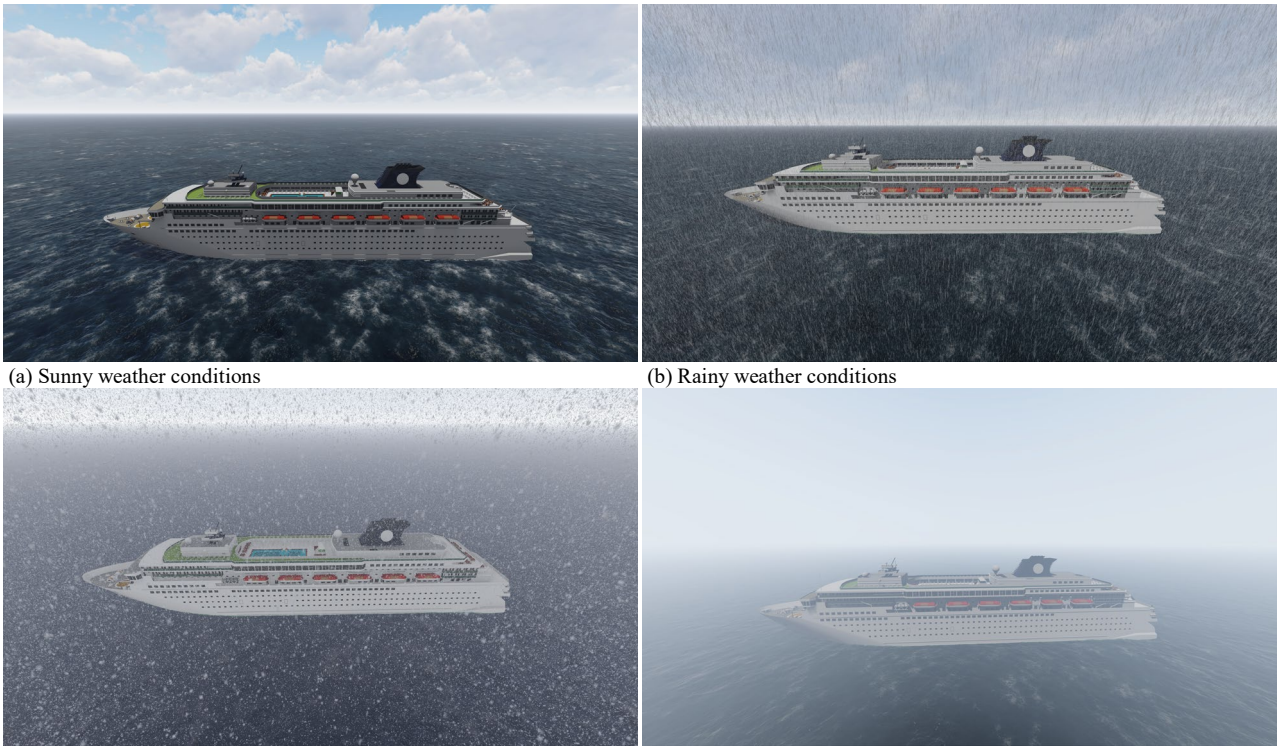
(b) Viewpoints in the Cabin

(c) Viewpoints in the guest room



(d) Viewpoints in the lobby

Fig. 7. Virtual roaming of different scenes



(a) Sunny weather conditions (b) Rainy weather conditions

(c) Foggy conditions (d) Snowy conditions

Fig. 8. Navigational environment simulation charts

D. System optimization

While maintaining fixed hardware specifications and model rendering precision, the accumulation of 3D objects and visual effects in virtual scenes leads to progressive degradation in real-time rendering performance. Such performance deterioration can manifest as frame drops or operational latency. To overcome these technical barriers and achieve enhanced scene fidelity with improved interactivity, this research develops a systematic optimization framework for complex 3D virtual environments.

1) Dual-layer clipping algorithm

The dual-layer clipping algorithm comprises two core phases [19]:Fade-In/Fade-Out Level of Detail (LOD) Algorithm: Efficiently removes model elements outside the camera's field of view; Dynamically selects detailed or coarse clipping based on the model's visible area ratio; Addresses the "popping" artifact in traditional Level of Detail (LOD) transitions through height compensation [20], ensuring smooth inter-level transitions.

When level A is switched to A_1, the height difference is Δh The total transition time t, and the height is compensated first, followed by the switching of levels. There are shown in Equation (3), where Δt is the current time that has been transitioned.

$$h_{A_1} = h_A + \frac{\Delta h}{t} \times \Delta \mathrm{t} \tag{3}$$

The second layer of the clipping algorithm, known as the slow culling algorithm, refines the culling process by optimizing traditional 3D vector calculations into 2D vector computations. This is achieved through projective transformations of the view vectors, normalizing the coordinates to $\vec{V} = (0,0,1)$, Suppose a certain triangle face piece is ΔABC, and that $A(x_1 y_1 z_1), B(x_2 y_2 z_2)$, and $C(x_3 y_3 z_3)$ then the normal ΔABC vector d of $\vec{n}$ is computed as:

$$\vec{n} = \overrightarrow{AB} \times \overrightarrow{BC} = \begin{vmatrix} \vec{x} & \vec{y} & \vec{z} \\ x_2 - x_1 & \mathrm{y}_2 - y_1 & \mathrm{z}_2 - z_1 \\ x_3 - x_1 & \mathrm{y}_3 - y_1 & \mathrm{z}_3 - z_1 \end{vmatrix} \tag{4}$$

Multiplying the field of view vector $\vec{v} = (0,0,1)$ with the normal vector $\vec{n}$ yields the formul:

$$\vec{n} \cdot \vec{v} = |\vec{n}| \times |\vec{v}| \times \cos\theta \tag{5}$$

where θ is the angle between the field of view vector of the slow culling algorithm and the plane normal vector of the triangular surface piece. The two rejection distances are set at the same time, the start rejection distance Q and the end rejection distance R are set, and the camera distance from the object is d. The opacity is adjusted by the length of the distance to achieve smooth transition switching, and the slow rejection algorithm is shown in Fig. 9.

2) Loading strategy optimization

In the Unity3D engine, the game logic thread and rendering thread operate in serial execution by default. Since rendering tasks typically depend on game logic computations, the engine prioritizes game logic processing before initiating rendering tasks. To enhance performance, this study implements parallel execution of these threads. Resource Loading Strategies: Synchronous Loading: Resources are loaded sequentially in the main thread, blocking other operations until completion; Asynchronous Loading: Employs multithreading to load resources in the background while the main thread continues executing tasks. For scenarios involving heavy game logic or rendering workloads, the system adopts asynchronous loading to: (1) Enable non-blocking background resource loading; (2) Process multiple loading tasks concurrently; (3) Optimize

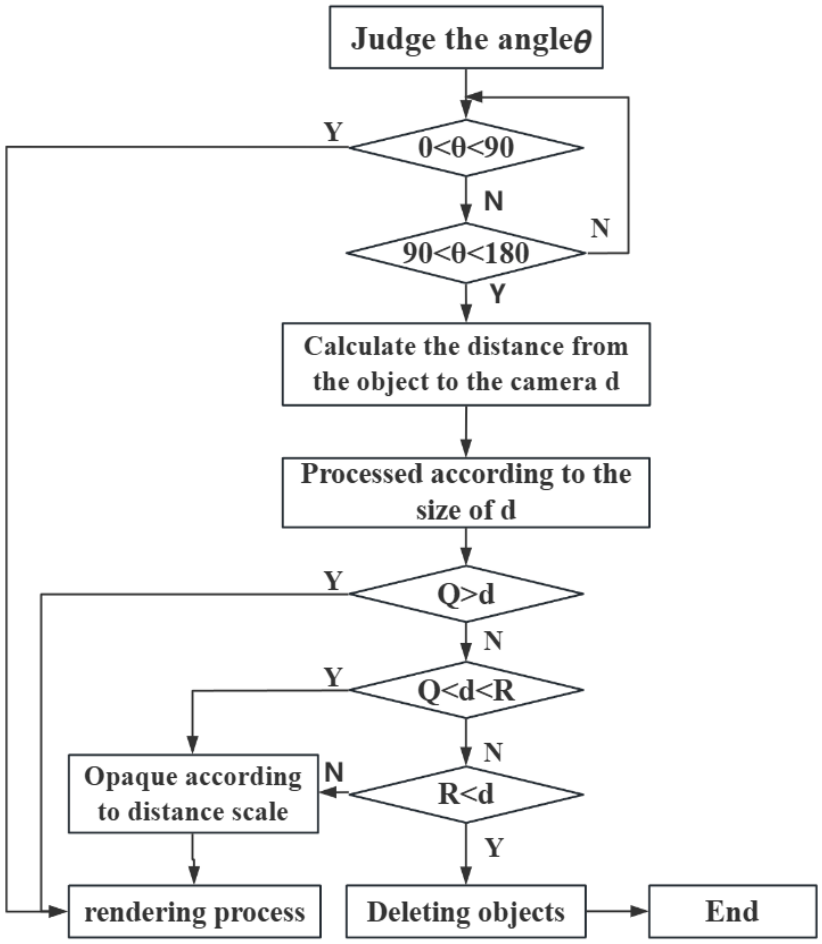


Fig. 9. slow culling algorithm

application performance and responsiveness; (4) Maintain scene continuity during resource loading. This approach significantly improves resource utilization and system efficiency. A comparative analysis of loading methods is illustrated in Fig. 10.

3) Texture optimization

Texture mapping is critical for achieving photorealistic scene rendering. Inside the vessel suit part thick in this large number of similar textures. To optimize texture resources—particularly for repetitive patterns in interior furnishings—this study implements a dual strategy:(1) Power-of-tow (PoT) Scaling: All textures are resized to PoT dimensions (e.g., 512×512, 1024×1024) using Photoshop, ensuring compatibility with GPU memory alignment requirements. Non-PoT textures are automatically upscaled by Unity to the nearest PoT size during compilation [21]. (2) Compression: Texture resolutions are selectively reduced via lossy/loss less compression while preserving visual clarity. This approach minimizes memory waste caused by texture

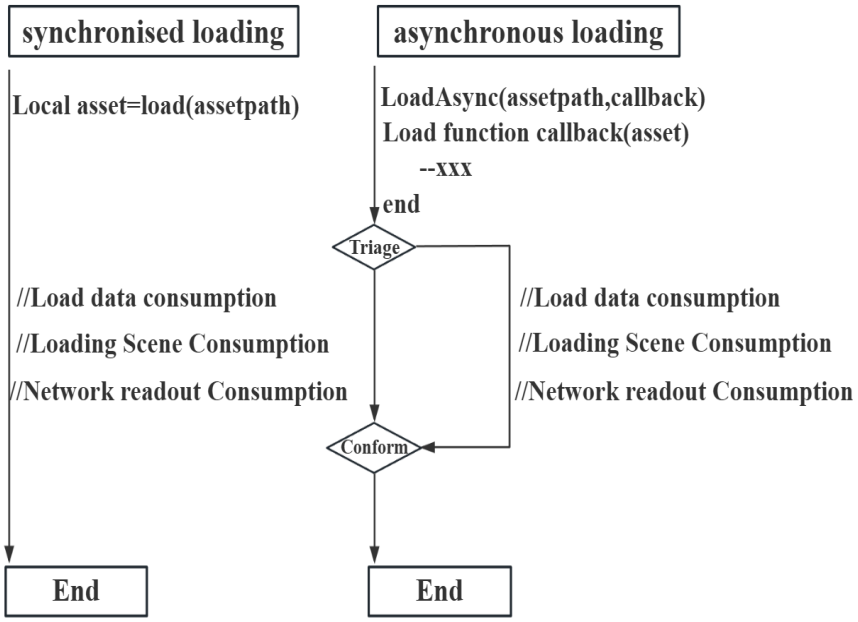


Fig. 10. Loading mode operation diagram

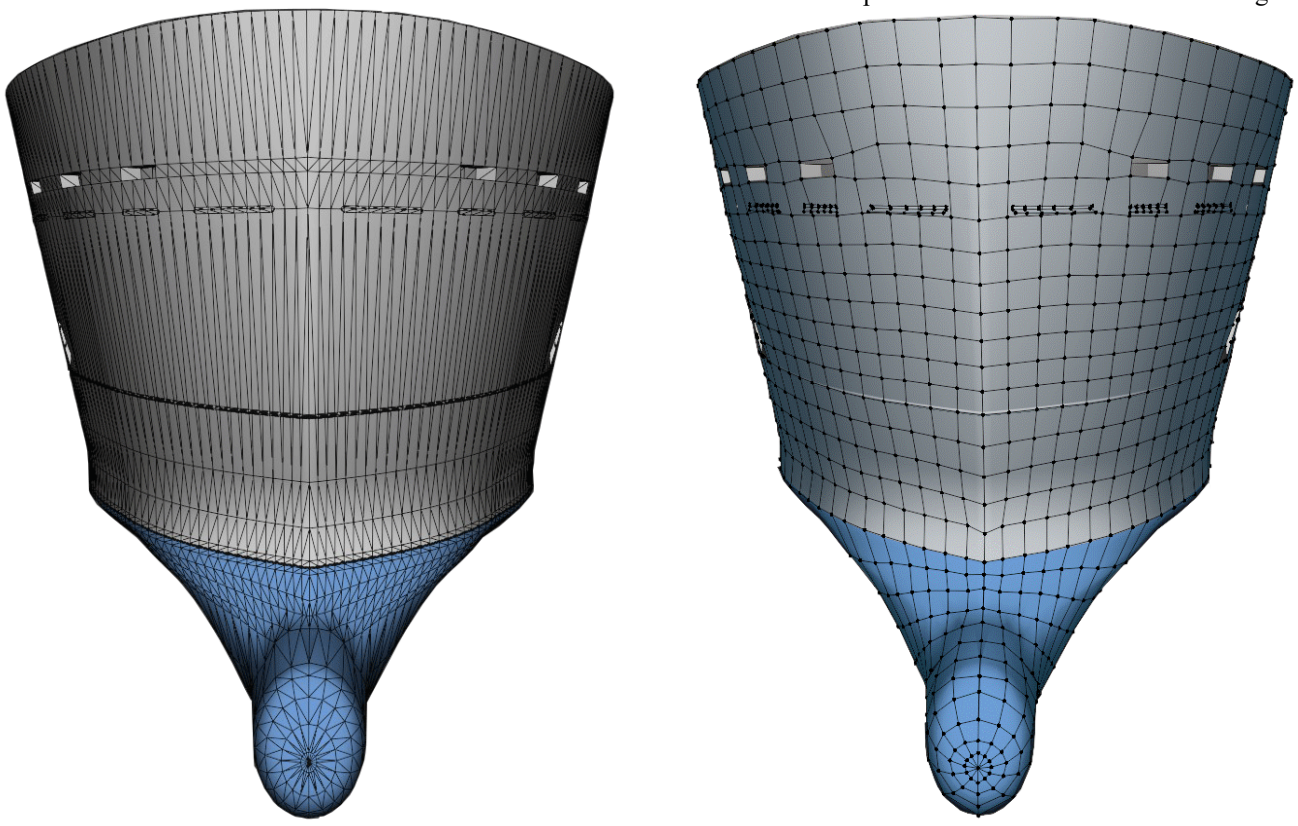
misalignment and enhances rendering efficiency across diverse hardware platforms.

4) Hull surface mesh optimization

Optimizing the surface mesh refers to a key process for the main hull shell model, The hull surface optimization process comprises three stages: (1) Re-meshing: Converts NURBS-based hull surfaces into quadrilateral-dominant polygonal meshes using Rhino, preserving curvature continuity [22]; Reduces polygon counts by 40–70% while maintaining geometric fidelity; (2) Retopology :Exports meshes as .FBX files for Blender retopology; Utilizes the retopoFlow add-on to reconstruct optimized topology with edge loop alignment; Before and after re-topology is shown in Fig. 11. (3) Mesh Refinement: Performs local density adjustments and normal smoothing; Finally the optimal form of the model is obtained.

E. Experimental design and analysis

The hardware configuration is as follows: CPU: 12th Gen Intel® Core™ i5-12400F @2.50 GHz, Memory: 32 GB, GPU: NVIDIA GeForce RTX 4060; experimental testing was conducted using Unity3D.



(a) Non-quadrilateral models before retopology

(b) Modelling after retopology

Fig. 11. Hull shell retopology before and after comparison

1) Dual-Layer Clipping Algorithm, Single-Layer Clipping Algorithm, and Comparative Experimental Study on Integrated Optimization Strategies

This experiment benchmarks the integrated optimization strategies (incorporating the dual-layer clipping algorithm) against conventional single-layer approaches, with dedicated analysis on the standalone performance of the dual-layer framework. The workflow includes: Fade-In/Fade-Out LOD Processing: Load and render only objects within the camera's field of view (FOV); Cull distant objects outside predefined FOV thresholds. Secondary Visibility Culling: Perform fine-grained visibility checks on in-view objects; Remove occluded elements in blind zones. The remaining three optimization techniques are added to the double-layer clipping algorithm to achieve a comprehensive optimization strategy. Now set the three-layer LOD level Screen Size as shown in Table I. The starting distance Q of the slow culling algorithm is set to 400cm, and the culling termination distance R is set to 600cm. Comparing the rendering frame rate (Frame Per Second, FPS) of each algorithm, for the convenience of experiments, the algorithms are summarized as shown in Table II. The experimental results are shown in Fig. 12.

TABLE I
ALGORITHM

Algorithm name	Abbreviation	Peculiarity
Fade-in/out hierarchical detail algorithm	FLOD	Single layer cut
Slow culling algorithm	FCULL	Single layer cut
Double cut	DCut	Fade-in/out hierarchical detail algorithm Slow culling algorithm
Integrated Optimization Framework	IOF	Four optimization strategies

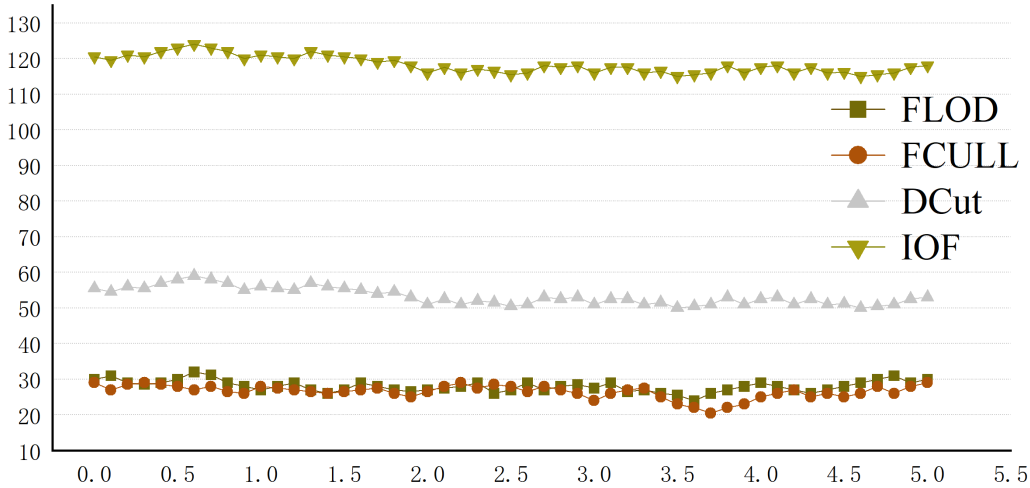


Fig. 12. Dual-Layer Clipping Algorithm, Single-Layer Clipping Algorithm, and Comparative Experimental on Integrated Optimization Strategies

TABLE II
LOD RATING AND SCREEN SIZE SETTINGS

LOD Rating	0	1	2
Screen Size	0.6	0.3	0.1

Fig. 12 demonstrates that the single-layer FLOD (Fade-In/Fade-Out LOD) and FCULL (Field of View Culling) achieve 30 FPS, while the dual-layer clipping improves performance to 57 FPS—a 90% enhancement. The Integrated Optimization Framework (IOF) further boosts the frame rate to 120 FPS, achieving a fourfold improvement over baseline methods.

The experiment compares synchronous (single-threaded) and asynchronous (multi-threaded) loading strategies in the large passenger ship simulation system. The test scenario utilizes the navigational environment simulation module from Section 3.3, loading diverse weather conditions in Unity3D.At the same time, Frame time data is used to show the comparison of rendering efficiency. Frame time is an index to measure the time required for a frame rendering, and it is also a very important concept in Unity3D, which directly affects the performance and fluency of the game. As shown in Fig. 13.

Fig. 13 reveals that single-threaded loading yields unstable frame rates (15 ±2 FPS), while multi-threaded loading stabilizes performance at 9 ±1 FPS. Demonstrating that parallelized resource allocation effectively mitigates CPU-GPU synchronization delays and improves computational efficiency.

2) Comprehensive optimization experiment

This experiment evaluates the effectiveness of the proposed optimization techniques using Unity3D as the testing platform. The optimized "Tianhai New Century" ship model serves as the experimental group, while the non-optimized model acts as the control group. These optimizations are interdependent, collectively enhancing multiple performance metrics and forming a mutually reinforcing system.

Relevant Performance Metrics: Tris: Number of triangular facets; Verts: Number of vertices; Batches: Total number of draw calls after batching; SetPass Calls: Number of state changes during rendering; Shadow Casters: Number of shadow casting calculations. The data optimisation comparison is shown in Fig. 14.

To comprehensively evaluate the impact of optimization strategies on system resources and validate the performance advantages of the dual-layer clipping algorithm and comprehensive optimization framework, this study conducts multiple experimental comparisons. Key metrics include CPU rendering frame rates, GPU loading times, model triangular facet counts, and vertex counts under four conditions: (1) No optimization; (2) Fade-in/fade-out level of detail (LOD) algorithm; (3) Slow culling algorithm with dual-layer clipping; (4) Comprehensive optimization strategy. As shown in Fig. 15 and Fig. 16.

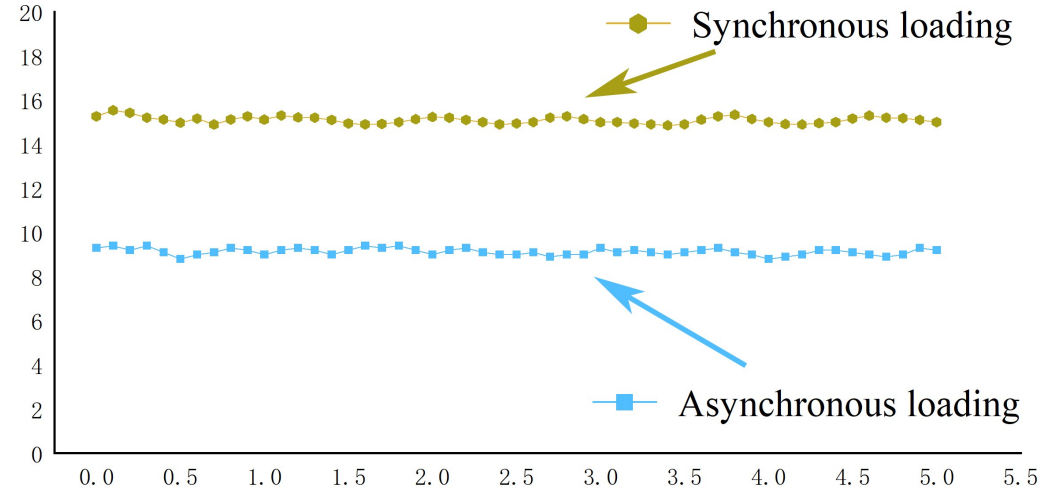


Fig. 13. Comparison of Frame time experiments

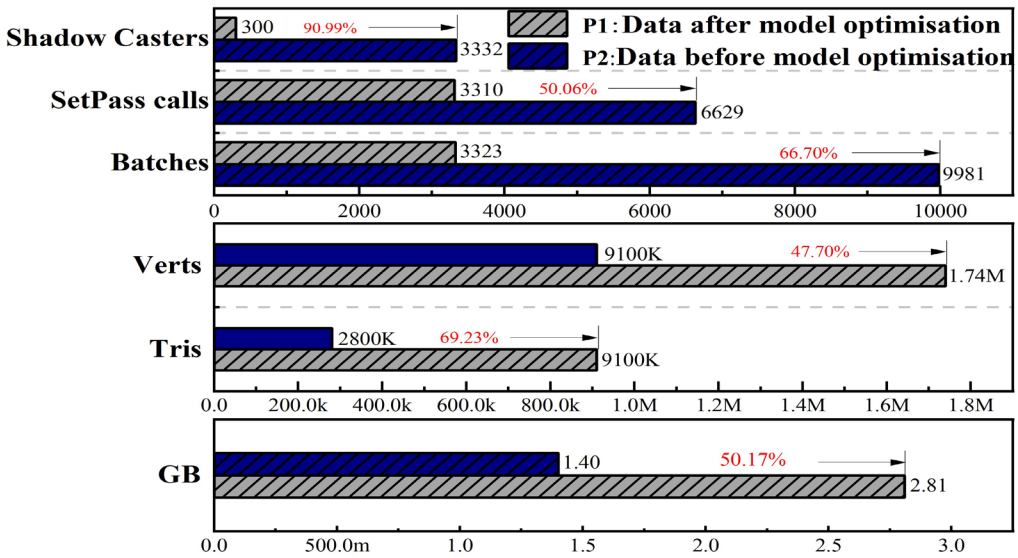


Fig. 14. Optimization Comparison Chart

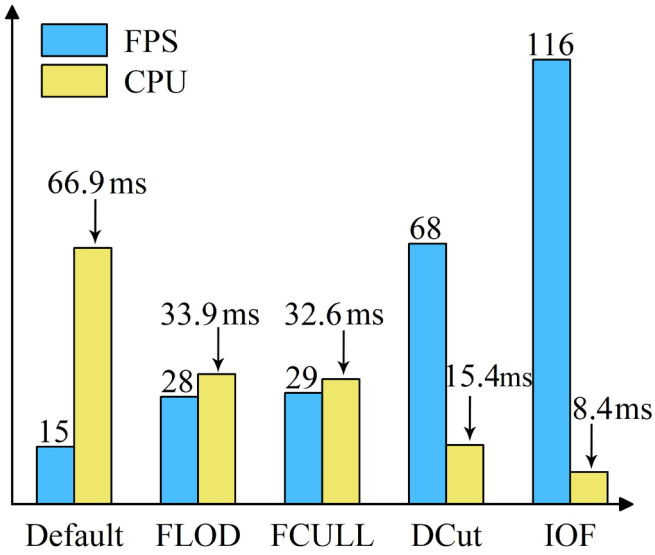


Fig. 15. Comparison of scenario experiment data 1

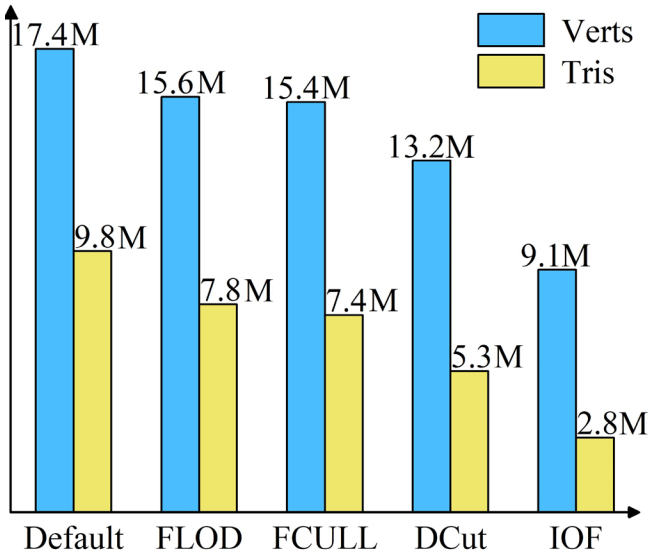


Fig. 16. Comparison of scenario experiment data 2

As summarized in Fig. 14, the proposed optimizations achieve substantial performance gains: triangular facets (Tris) decrease by 47.70% (9.1M→2.8M), vertices (Verts) by 69.23% (17.4M→9.1M), and memory consumption by 50.18% (2.81GB→1.40GB). GPU workload metrics improve significantly—batches, SetPass calls, and shadow casters are reduced by 66.70%, 50.06%, and 90.99%, respectively—demonstrating streamlined rendering pipelines and reduced computational redundancy.

Analysis of Fig. 15 and Fig. 16, the Integrated Optimization Framework (IOF) achieves a greatly improves improvement in frame rate (15 FPS → 120 FPS) compared to baseline methods (Dcut, FCULL, FLOD). This enhancement is accompanied by a 47.7% reduction in vertex counts (17.4M → 9.1M) and a 71.4% decrease in triangular facets (9.8M → 2.8M).Concurrently, loading response time and CPU utilization are reduced by 35% and 28%, respectively, demonstrating that hierarchical culling and mesh simplification synergistically address computational bottlenecks in large-scale maritime simulations.

IV. Summary

This study successfully implements a virtual simulation system for large passenger ships based on Unity3D, demonstrating the feasibility of a comprehensive optimization framework. Key contributions are as follows: (1) A hybrid modeling methodology integrating NURBS curve modeling, polygonal modeling, and primitive modeling to construct high-precision 3D ship models with 47.7% fewer vertices (17.4M→9.1M) and 69.2% fewer triangular facets (9.8M →2.4M); (2) A unified system architecture enabling virtual scene roaming, dynamic environment simulation (e.g., weather conditions), and an interactive user interface (UI); (3) Performance optimizations including hierarchical visibility culling and asynchronous resource loading, reducing memory consumption by 50.2% (2.81GB→1.40GB) and achieving a dramatic 8× improvement in rendering efficiency (15 FPS → 120 FPS). These interdependent strategies collectively enhance real-time rendering performance while maintaining geometric fidelity. The framework provides a scalable solution for maritime simulations, aligning with industrial

demands for high-performance training and emergency response systems.

REFERENCES

[1] Mahfuzulhoq Chowdhury, "Liberty: An Economy, Three Fold Collaboration, and Virtualization-aware Resource Management Approach for Multiverse Based Application Execution Over 6G Enhanced Networks," *IAENG International Journal of Computer Science*, vol. 51, no. 11, pp1804–1844, 2024.

[2] I. Wohlgenannt, A. Simons, and S. Stieglitz. "Virtual reality," *Business & Information Systems Engineering*, vol. 62, pp455–461, 2020.

[3] X. Zhou, "Intangible Cultural Heritage Art Exhibition System Based on Mobile Virtual Reality Technology," *International Conference on Innovative Computing, Singapore: Springer Nature Singapore*, vol. 1215, pp224–235, 2024.

[4] X. Li, J. Zhang, J. Cheng, and Q. Luo, "Simulation of switching operation based on virtual reality method," *Journal of Physics: Conference Series*, vol. 1650, no. 3, pp032121, 2020.

[5] S. Yu, J. Han, "Virtual Reality Platform-Based Conceptual Design and Simulation of a Hot Cell Facility," *The International Journal of Advanced Manufacturing Technology*, vol. 116, no. 1, pp487–505, 2021.

[6] K. Zhu, D. Xiao, and X. Xu, "Investigation of Onboard Training Quality Based on STCW Manila Amendment," In *CICTP*, pp642–652, 2024.

[7] P. Yao, Z. Chen, T. Jing, and L. Qian, "Virtual simulation system of cutter suction dredger based on Unity3D," *Journal of system simulation*, vol. 28, no. 9, pp2069–2075, 2020.

[8] G. Shi, Y. Zhou, M. Li, H. Zhang, and Y. Wang, "LNG ship simulation system based on virtual reality technology," *Shipbuilding and Sea Engineering*, vol. 50, no. 03, pp25–28 +33, 2021.

[9] Z. Li, "Virtual simulation training system for large engineering ships," *Ship Science and Technology*, vol. 45, no. 11, pp147–150, 2023.

[10] J. Yin, H. Ren, and Y. Zhou, "The whole ship simulation training platform based on virtual realiy," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 2, pp207–215, 2021.

[11] M. Dong, Z. Gao, and L. Liu, "Model optimization method based on Rhino," *Advanced Manufacturing and Automation IX 9th, Springer Singapore*, vol. 634, pp267–274, 2020.

[12] N. A. S. Abdullah, N. I. A. Rusli, and M. F. Ibrahim, "Mobile game size estimation: Cosmic fsm rules, uml mapping model and unity3d game engine," *IEEE Conference on Open Systems (ICOS), IEEE*, pp42–47, 2014.

[13] D. Huang, H. Yan, "NURBS curve controlled modelling for facial animation," *Computers & Graphics*, vol. 27, no. 3, pp373–385, 2003.

[14] Zhangfang Hu, Shanshan Tang, Yuan Luo, Fang Jian, and Xingtong Si, "3DACRNN Model Based on Residual Network for Speech Emotion Classification," *Engineering Letters*, vol. 29, no.2, pp400–407, 2021.

[15] P.-P. Sloan, J. Kautz, and J. Snyder, "Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments," *Seminal Graphics Papers: Pushing the Boundaries,* vol. 2, pp339–348, 2023.

[16] K. Tanaka, "Simulation Analysis of Optimal Camera Viewpoints for Glossy-Surface Inspection," *Engineering Letters,* vol. 32, no. 11, pp2083–2089, 2024.

[17] X. Yang, H. Ren, J. Lian, and D. Wang, "VR interactive three-dimensional virtual ship modeling and simulation," *China Navigation*, vol. 45, no. 01, pp37–42+49, 2022.

[18] B. Zhang, W. Hu, "Game special effect simulation based on particle system of Unity3D," *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), IEEE*, pp595–598, 2017.

[19] F. Biljecki, H. Ledoux, J. Stoter, and G. Vosselman, "The variants of an LOD of a 3D building model and their influence on spatial analyses,"*ISPRS Journal of Photogrammetry and Remote Sensing,* vol. 116, pp42–54, 2016.

[20] Y. Li, X. Luo, "Rendering optimization algorithm based on virtual reality," *Computer System Applications*，vol. 28, no. 06, pp178–182, 2019.

[21] Xiao Wei, Lei Sun, and Wei Zheng, "Packing [2,8] Coloring of Planar Graphs with Maximum Degree 4," *IAENG International Journal of Applied Mathematics,* vol. 55, no. 1, pp147–153, 2025.

[22] O. Liaskos, S. Mitsigkola, and A. Arapakopoulos, "Development of the virtual reality Application:"The ships of Navarino," *Applied Sciences*, vol. 12, no. 7, pp3541, 2022.