Research on Virtual Scene Design Methods for Unity3D Games

Jiwen Zhang

386423632@qq.com

Jilin Animation Institute, Changchun, China

Abstract. This article delves into the design methods of game virtual scenes based on the Unity3D platform, focusing on scene construction, lighting systems, material applications, and advanced pathfinding algorithms. The article particularly emphasizes the core role of project structure and file management in efficient development, and deeply analyzes the optimization techniques of resource import, material innovation, and lighting design, especially the detailed adjustment of directional light sources, point light sources, and area light sources. In addition, the article also explores the synergy between physics engines and lighting effects, and systematically analyzes the advanced pathfinding algorithm based on Navigation Mesh, demonstrating its practical application and effectiveness in dynamic gaming environments. Through a series of experimental designs and verifications, the article demonstrates the application value of these comprehensive technologies in creating a realistic, efficient, and immersive gaming experience. Based on these research findings, this article provides practical guidance and inspiration for game designers to develop high-quality game scenes using Unity3D.

Keywords: Unity3D; Virtual game scenes; Design methods

1. Introduction

Virtual reality technology, as an important product of the digital age, is rapidly changing the field of game design. This technology originates from highly simulating the real world, utilizing computer-generated three-dimensional environments to provide users with an immersive interactive experience. In this field, Unity3D software has become the preferred tool for game developers due to its powerful rendering capabilities, flexible user interface, and wide hardware compatibility. Unity3D not only supports cross platform deployment, covering a wide range of devices from PCs to mobile devices, game consoles to VR helmets, but also has advanced physics engines and real-time global lighting technology, making it possible to create realistic virtual worlds[1].

In recent years, with the popularization and development of Unity3D engines, more and more research has focused on utilizing their efficient and flexible features to create complex game scenes. Academic papers and industry reports have shown that Unity3D has significant advantages in scene design, such as ease of use, high customizability, and powerful graphic rendering capabilities. Zhang Wei (a game design expert) pointed out that the application of Unity3D in game scene design is increasingly tending to achieve a high degree of realism and interactivity. Professor Zhang's research emphasizes the powerful graphics processing

ADDT 2024, May 24-26, Luoyang, People's Republic of China
Copyright © 2024 EAI
DOI 10.4108/eai.24-5-2024.2350033

capabilities and flexible scripting support of Unity3D, believing that these features are key factors for Unity3D to maintain its leading position in future game development. Li Na (Professor of Computer Graphics) analyzed the challenges of Unity3D processing large and complex scenes from a technical perspective, especially in dynamic lighting and real-time rendering. Her research proposes various methods for performance optimization, including resource management and efficient coding practices. Common discussion points in literature include how to overcome the performance limitations of Unity3D in handling large-scale scenes, and how to optimize scene loading and running efficiency while maintaining high-quality rendering. In addition, adapting to constantly changing hardware requirements and supporting cross platform functionality are also research hotspots. Andrei Kostyanov, a game development expert, discussed the challenges Unity3D faces in cross platform development, particularly in maintaining consistency and performance optimization. He pointed out that although Unity3D provides extensive platform support, achieving the same visual effects and response speed on different devices remains a technical challenge. Some innovative studies have applied Unity3D to VR and AR scene design, exploring how to enhance player immersion and interactive experience through these emerging technologies. Meanwhile, research has also focused on utilizing Unity3D's scripting and plugin systems to create more dynamic and interactive scene elements[2].

Susan Greenfield, a researcher in virtual reality technology, emphasized the innovative potential of Unity3D in the fields of VR and AR. Her research suggests that Unity3D is becoming the preferred platform for VR and AR game design due to its high adaptability and ease of use in complex scenes. Mark Thompson (an expert in interactive media) explored the ability of Unity3D to enhance the dynamic interactivity of games. He mentioned that Unity3D's scripting system and physics engine provide game designers with tools to create highly interactive and responsive environments. Looking ahead, literature predicts that Unity3D will continue to play an important role in the field of game scene design, especially in enhancing realism, interactivity, and multi platform compatibility. With the development of technology, such as the integration of artificial intelligence and machine learning, Unity3D's scene design may usher in new breakthroughs. Huang Kaixuan, an artificial intelligence researcher, predicts that with the integration of machine learning and artificial intelligence technology, Unity3D's scene design will usher in new development opportunities. He emphasized that through the integration of AI, scene design can achieve more advanced automation and personalization, thereby improving the player experience. Emily Roberts (Game Industry Analyst) focuses on the role of Unity3D in future game development, particularly in terms of diversity and accessibility. She believes that Unity3D, as a platform, not only promotes the development of gaming technology, but also provides opportunities for a wider community of designers and developers to express their creativity[3].

2. Deep Analysis of Unity3D Software Platform

Unity3D, as a leading game development platform, its unique software architecture and core components constitute its core competitiveness in the industry. On the core architecture, Unity3D adopts a component-based design, allowing game objects to flexibly add and combine different functions, thereby achieving high modularity and reusability. This architecture not only simplifies the process of complex game development, but also improves development efficiency and flexibility. In terms of core components, Unity3D integrates a powerful 3D rendering engine that supports high-precision physical simulations and complex particle systems, making game scenes more realistic and vivid. In addition, its animation system supports complex bone animations and hybrid animations, coupled with advanced AI navigation systems, greatly enriching the interactivity and realism of the game. The unique features of Unity3D are reflected in its cross platform publishing capabilities, supporting a wide range of compatibility from desktop operating systems to mobile devices, as well as VR and AR devices.

3. Methodology For Constructing Virtual Game Scenes

3.1 Overall process of virtual scene construction

In the overall design concept of building game virtual scenes, the first step is to create and plan project folders. The key to this step is to achieve efficient resource management and fast access, ensuring smooth subsequent development. Subsequently, the import and optimization process of scene resources needs to focus on the comprehensive integration of resource types, while optimizing the performance of models, textures, and audio to meet the compatibility requirements of different platforms. The creation stage of scene model materials focuses on improving the visual quality and realism of the scene through precise material attribute adjustments and texture mapping. The design of lighting systems must comprehensively consider the type of light source, layout, and lighting effect to create a suitable gaming atmosphere. The rendering process of light mapping aims to optimize the visual effects of the scene while improving operational efficiency. Finally, the construction of a navigation grid pathfinding system should focus on achieving efficient and natural movement logic between characters and AI, enhancing the interactivity and playability of the game. The entire process needs to be closely integrated to ensure efficient execution of each step, in order to build a game virtual environment that is visually appealing and has good performance.

3.2 Detailed Scene Production Steps

3.2.1 Resource loading for Unity engine

Unity provides an AssetBundle format resource storage file format, which is used to package and store the resources required by the Unity engine on external devices or networks. These resources can be read from files into memory through WWW or IOStream, forming an AssetBundle memory image. To reduce memory usage, after loading AssetBundle, the Unload (false) method can be used to release the memory image[4].

In the memory image of AssetBundle, there are resources such as sound, animation, textures, materials, and objects. These resources require the use of AssetBundle The Load method is used to obtain prototypes of various resources. The obtained resource prototypes can be removed from memory through various methods such as UnloadAsset (obj). In a scene, a resource prototype may correspond to multiple instance objects. Therefore, it is necessary to use the Instantiate method to instantiate GameObject type scene objects. If the instantiated object is no longer used, it can be accessed through GameObject Delete using Destroy (obj) method.

From the above description, it can be seen that the time cost of resource loading mainly has two aspects: first, when reading resource files from the hard drive, and second, when instantiating objects. To verify the relationship between the two, randomly select assertBundle files of 1.5-3M size, loop through and read the scene objects they contain. Table 1 shows some of the results[5].

Table 1 Comparison of IO Operation and Instantiation Time Cost

File size (KB)	O operation time	instantiation time
169	372.2072	1.0001
460	383.2495	1.0007
2117	479.5775	1.0011
17063	3731.8084	2.0004

3.2.2 Efficient Scene Resource Import Strategy

The texture resources for game scenes, such as sky box textures, self luminous textures, transparency textures, ground textures, and universal textures, must be optimized and classified reasonably to ensure their efficient loading and rendering in the game. Specifically, water surface textures should be given special attention, especially in scenes involving water surface models, where texture processing needs to consider reflection, refraction, and dynamic ripple effects to enhance the realism of the scene. It is crucial to classify model resources in game scenes into two categories: dynamic objects and static objects. Dynamic objects, such as vehicles, animals, NPCs, main characters, and building mechanisms, are commonly used as active elements in scenes and require attention to optimizing their animation and interaction performance. Static objects, including rooms, rocks, floors, bridges, buildings, etc., although not involving dynamic data, their layout and optimization in the scene are equally important. In Unity 3D, when using polygon modeling tools to create these models, it is necessary to consider the complexity, level of detail, and coordination with other elements of the scene. In resource import, it should include batch processing of resources and automated import processes, such as using scripts to automate texture compression and format conversion, as well as setting the Level of Detail of the model.

3.2.3 Innovative application of materials and texture

After importing model resources into the Unity 3D engine, rewriting or rebuilding materials is an important step in ensuring project version stability and image quality optimization. The main goal of this process is to minimize the energy consumption when rendering game scenes on mobile devices, which not only affects the performance of the game but also affects the user experience. To effectively control the amount of materials used in the scene and reduce rendering times, it is first necessary to determine the number of materials based on the number of textures. This means that in the process of creating materials, it is important to carefully balance the quantity of materials and the visual needs of the scene. Subsequently, for special objects in the scene, such as main characters, key props, or unique buildings, professional materials should be created. These professional materials not only stand out visually, but are also more efficient technically, and can improve rendering performance through intelligent texture compression, detail level management, and other means. In the application of materials, it is reflected in the improvement of visual effects, as well as the optimization of performance. Unity's Shader Graph can be used to create complex material effects while ensuring that these effects maintain high performance across different devices. In addition, advanced technologies such as real-time global lighting and reflection probes can significantly enhance the realism and visual appeal of the scene without sacrificing performance.

3.2.4 Design and implementation of lighting system

The lighting system in the Unity 3D platform shares similarities with other 3D animation software such as Autodesk Maya, Xara 3D Maker, Adobe After Effects, etc. in terms of lighting types, mainly including directional lighting, point lighting, spot lighting, and area/area lighting. The directional light source is used to simulate parallel rays of distant light sources (such as the sun), which is crucial for creating lighting effects and the overall lighting environment of the scene. Point light sources simulate light that diverges from a single point in various directions, and are suitable for simulating local light sources such as light bulbs or torches. Spotlights can generate cone-shaped beams of light and are commonly used to focus or emphasize specific elements in a scene, such as stage lighting or flashlight light. Area/surface light sources are used to simulate larger light sources that emit uniform and soft light, such as the light transmitted through windows or soft box lights. They are extremely effective in creating a specific atmosphere and increasing the sense of hierarchy in the scene.

When designing and implementing lighting systems, these light sources should be reasonably matched and adjusted according to the specific needs and artistic style of the game scene. This includes the position and direction of the light source, as well as the color, intensity, range, and details of its impact. For example, in creating a forest scene, it is necessary to combine directional lighting to simulate sunlight, point lighting to simulate the light points and spotlights passing through the gaps between leaves to highlight the protagonist or key elements. In addition, advanced lighting techniques such as light mapping, real-time shadows, reflections, and global lighting can also be applied to further enhance the realism and visual effects of the scene[6].

3.2.5 Optimization and rendering of lighting maps

Light mapping technology pre renders the lighting information of all lights in the game scene into one or more maps, thereby reducing the hardware consumption of lighting effects and improving the quality of lighting effects during game runtime.

This process first involves the generation of lighting maps. In Unity 3D, it is implemented through Lightmap Baking, which calculates the lighting and shadows of static geometry in the scene and stores this information in the map. This method is particularly suitable for static objects such as buildings, floors, and decorations, as their lighting information does not change during game operation. Optimizing lighting maps involves adjusting their resolution and compression level to achieve the best balance between visual effects and performance. A higher resolution lighting map can provide finer lighting effects, but it also increases memory consumption and loading time. Therefore, it is necessary to carefully select the appropriate resolution and compression settings based on the specific requirements of the game and the hardware capabilities of the target platform. Finally, achieve an effective combination of dynamic and static lighting. By combining static lighting with dynamic lighting (such as the light from a character's flashlight), it is possible to increase the dynamic and interactive feel of the scene while maintaining efficient rendering.

3.2.6 Navigation Grid and AI Path Planning

To implement navigation grids in Unity 3D, the first step is to apply the NavMesh system, which generates navigation grids by analyzing the geometric shape of the game scene. This process can be achieved through NavMesh Baking, which calculates which regions are walkable and converts them into grid data. The commonly used algorithms for AI path planning include A * algorithm and Dijkstra algorithm. The A * algorithm is an efficient search algorithm commonly used in graphic path planning, with the formula:

$$f(n) = g(n) + h(n)$$

$f(n)$It is the total cost estimate of node n， $g(n)$It is the actual cost from the starting point to node n,$h(n)$It is the estimated cost from node n to the target (heuristic estimation). Through this formula, the A * algorithm can reduce the search range while ensuring the shortest path is found. The Dijkstra algorithm is a simple but inefficient path search algorithm suitable for scenarios without heuristic estimation. Its working principle is to continuously search for the node with the lowest cost and update the shortest path to reach the other nodes. In practice, AI path planning also involves handling dynamic obstacles and environmental changes. For example, using Dynamic Obstacle Map to update obstacles in the environment in real-time, ensuring that AI can adapt to changing game scenes.

ASP NET, as a powerful network application framework, plays a crucial role in the field of data processing. It not only provides a series of tools and libraries to process and analyze data, but also supports various complex data operations through its highly scalable architecture. In this section, we will explore ASP The main applications and advantages of NET in data processing.

4. A Game Virtual Scene Design Method Based On Unity3d

4.1 Scene model and texture design

This design process requires the integration of artistic creativity and technological implementation to achieve a high-quality gaming environment. Firstly, the design of the scene model starts from the basic geometric shape and gradually refines to the complex structure. In Unity3D, model design typically uses polygon modeling methods, which involve constructing vertices, edges, and faces. To optimize performance, especially on mobile or VR platforms, it is necessary to consider the polygon count of the model. This can be achieved by calculating the formula[7]：

$$V - E + F = 2 - 2G$$

(V is the number of vertices, E is the number of edges, F is the number of faces, and G is the genus number of the model) to optimize and ensure a balance between the complexity and performance of the model. In terms of texture design, UV mapping is a crucial step, which maps the surface of a 3D model to a 2D image. The correct UV mapping ensures proper spreading of the texture on the surface of the model. In addition, using Physically Based Rendering materials can enhance the realism of the texture, which is based on the physical properties of the real world and uses parameters such as roughness and metalness to simulate the material. To further enhance the visual effect of the scene, texture compression and Level of Detail (LOD) techniques can be applied. Texture compression reduces storage and memory usage, while LOD technology dynamically adjusts the details of the model based on the distance between the camera and the object. The formula is

$$\mathrm{LOD} = \frac{\mathrm{Distance}}{\mathrm{Hysteresis}}$$

4.2 Innovative Design of Lighting Systems and Materials

4.2.1 Directional light source design

The core of directional light source design is to simulate natural light sources, such as the sun or moon, to provide consistent lighting effects for game scenes. Firstly, the design of directional light sources requires determining the direction of the light source. This is usually achieved by adjusting the angle of the light source in three-dimensional space, which can be achieved using directional vectors

$$\vec{\mathrm{D}} = (\mathrm{x}, \mathrm{y}, \mathrm{z})$$

Among them are the three components of the direction vector, representing the direction of the light source. Secondly, adjust the light source color and intensity parameters. The color of the light source can be set through RGB values to simulate the color temperature of sunlight at different times, such as warm gold at dusk or bright white at noon. The intensity of the light source affects the brightness of the entire scene, which can be represented by the intensity of light, usually a value between 0 and 1. Finally, in order to enhance the realism of the scene, directional lighting is often used in combination with shadow effects. In Unity3D, this involves setting the quality and range of shadows. Shadow mapping algorithms, such as Shadow Mapping or Cascaded Shadow Maps, can be used to optimize the performance and visual effects of shadows. The core of shadow mapping algorithm is to render the scene from the perspective of a light source into a depth map, and the calculation formula can be simplified as figure1

$$\mathrm{Depth} = \mathrm{f}(\mathrm{Position}_{\mathrm{light}})$$

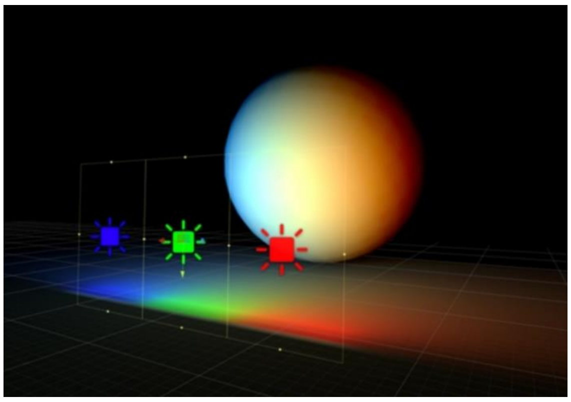

Figure 1 Schematic diagram of directional light source

4.2.2 Point light sources and their dynamic applications

Point light sources are used in game design to simulate light sources that diverge uniformly from a single position in all directions, such as light bulbs or torches. Firstly, in the design of point light sources, position setting is crucial. The position of a point light source is defined in three-dimensional space, which directly affects the range and intensity of its lighting effect. According to the needs of the game scene, placing point light sources reasonably can enhance the depth and details of the scene. Secondly, point light attenuation. The intensity of the light source decreases with increasing distance, which can be achieved through the attenuation formula

$$\mathrm{I} = \frac{\mathrm{I_0}}{\mathrm{d^n} + \mathrm{C}}$$

Finally, adjust the color and intensity of the point light. In Unity3D, these parameters can be adjusted according to the atmosphere and style of the scene to achieve the desired visual effect. In addition, the dynamic application of point lighting, such as dynamically adjusting the intensity, color, or position of the light source, can bring more vivid and realistic effects to the game. For example, in a scenario where day and night cycles are implemented, the intensity of a point light source can vary over time, enhancing the immersion of the game[8].

Regional light sources are used in game design to simulate large area light sources such as windows or skylights, providing soft and uniform lighting that is suitable for creating specific atmospheres and depths. Firstly, the design of regional light sources begins with determining their size and shape. The size of the area light source directly affects the distribution and softness of the light. In Unity3D, area light sources can have different shapes such as rectangles or circles, and their dimensions are defined as

$$\mathrm{S} = (\mathrm{length}, \mathrm{width})$$

The illumination distribution of regional light sources can be simulated using Ray Tracing algorithms or Radiosity methods. These algorithms can calculate how the light emitted by the light source propagates and reflects in the scene, where the basic formula for ray tracing is[9]

$$\mathrm{L_o}(\mathrm{p}, \omega_\mathrm{o}) = \mathrm{L_e}(\mathrm{p}, \omega_\mathrm{o}) + \int_\Omega \mathrm{f_r}(\mathrm{p}, \omega_\mathrm{i}, \omega_\mathrm{o}) \mathrm{L_i}(\mathrm{p}, \omega_\mathrm{i})(\omega_\mathrm{i} \cdot \mathrm{n}) \mathrm{d}\omega_\mathrm{i}$$

Next, adjusting the color and intensity of the area light source is equally important for achieving the desired scene effect. In Unity3D, these parameters need to be carefully adjusted according to the needs of the scene to achieve specific atmosphere and lighting effects. In addition, the evaluation of the effectiveness of regional light sources is crucial to ensuring the achievement of design goals. This usually involves evaluating the distribution of lighting intensity in the scene, as well as analyzing the impact of light sources on scene details and material properties. In Unity3D, evaluating the effect of area lighting can be achieved through real-time rendering previews and lighting baking processes.

4.2.3 Collaborative optimization of physics engine and lighting effects

The collaborative optimization of physics engines and lighting effects is key to improving the realism and visual effects of game scenes. This process involves complex algorithms and technical implementations to ensure a natural fusion of physical interaction and lighting effects. Firstly, the optimization of physics engines mainly focuses on the accuracy and computational efficiency of simulating real-world physical behavior. In Unity3D, physics engines such as NVIDIA PhysX are used to handle collision detection, rigid body dynamics, and other complex physical interactions[10]. For example, the calculation of rigid body dynamics can be achieved through Newton's second law (where is force, is mass, and is acceleration). Meanwhile, in order to improve performance, it is necessary to finely adjust the time step and iteration number of physical simulations. Secondly, the optimization of lighting effects focuses on achieving realistic lighting models and efficient rendering techniques. This includes the use of physics based rendering (PBR) models, which generate realistic visual effects by simulating the real physical processes of light interacting with matter, such as reflection and refraction. On this basis, the key to achieving lighting effects is efficient lighting calculation algorithms such as Delayed Rendering and Ray Tracing. In the process of collaborative optimization of physics engines and lighting effects, it is important to ensure their interaction. The results of physical simulations, such as the position and direction of objects, need to correctly affect lighting calculations, and vice versa. Light Probes and Reflection Probes can be applied to capture the lighting and reflection effects of dynamic objects in different environments， as figure2

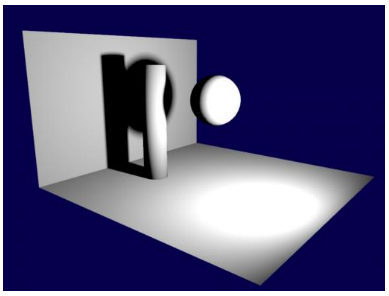

Figure 2 Lighting Map Legend

4.2.4 Design of Advanced Pathfinding Algorithm Based on Navigation Mesh

$\mathrm{g(n)}$The actual cost from the starting point to node n.

$\mathrm{rhs(n)}$The best estimated cost for node n is the minimum cost from all the precursor nodes of n to the starting point plus the cost from the precursor nodes to n.

$\mathrm{key(n)}$The key used for prioritizing queue sorting is defined as

$$\mathrm{key\ (n)} = [\min(\mathrm{g(n)}, \mathrm{rhs(n)}) + \mathrm{h(start, n)}, \min(\mathrm{g(n)}, \mathrm{rhs(n)})]$$

The JPS algorithm optimizes the open list operation in traditional A * algorithms. JPS performs skip point search on grid maps, only calculating the f-value at key points (such as corner points), significantly reducing the size of the open list and the number of searches. This method is particularly suitable for large or complex grid maps[11].

5. Experimental Design and Verification

The core of experimental design is to evaluate the performance of virtual game scene design methods based on Unity3D in practical applications. This process involves three main stages: scenario construction, performance testing, and user experience evaluation. Firstly, the scene construction process follows the design principles mentioned in the document, such as lighting system design, material application, etc. Multiple game virtual scenes with different lighting, materials, and layouts were constructed in the Unity3D environment. This step aims to create a diverse testing environment to verify the applicability and effectiveness of design methods under different conditions. Next, in the performance testing phase, Unity3D's built-in or third-party performance analysis tools are used to conduct detailed performance evaluations for each scenario. This includes key metrics such as frame rate (FPS), memory usage, and rendering time for measuring the scene. The purpose of performance testing is to quantitatively evaluate the impact of design methods on game performance, especially in terms of resource optimization and rendering speed. Finally, user experience assessment collects feedback from test users through questionnaires and interviews, with a focus on their evaluation of the realism, interactivity, and immersion of the scene. The collection of user experience data aims to evaluate the effectiveness of scene design from the player's perspective and understand the specific impact of different design decisions on the game experience.

When analyzing simulated adjusted data, a clear relationship between scene design complexity and performance indicators is revealed, which directly affects user experience. Scenarios 1 and 3 demonstrate the negative impact of high complexity on frame rate, despite high realism ratings, performance limitations reduce overall user satisfaction. In contrast, Scenario 2 and Scenario 4 achieved a better performance balance at moderate complexity, resulting in higher ratings for user interactivity and immersion. Especially in scenario 4, its excellent performance indicators and user experience ratings indicate that optimizing performance while maintaining moderate complexity is an effective strategy to improve user experience. These findings emphasize the importance of seeking a balance between design complexity and performance in game virtual scene design, aiming to provide a rich and smooth gaming experience[12].

6. Conclusions

This thesis is entitled "A Game Virtual Scene Design Method Based on Unity3D". The author discusses the design method of game virtual scene based on Unity3D platform, including scene construction, lighting system, material application, advanced pathfinding algorithm, etc. The importance of project structure and document management for efficient development is emphasized, and optimization techniques for resource introduction, material innovation and lighting design are explored, with special emphasis on detailed adjustment of directional, point and area light sources.

Acknowledgment: The combination of these advanced algorithms can greatly improve the pathfinding efficiency and adaptability of NavMesh in Unity3D. This not only enables pathfinding algorithms to quickly respond to dynamic changes in the environment, but also to find the optimal path in complex game terrain, providing players with a richer and more realistic gaming experience. Through this advanced algorithm design, game developers can achieve precise and intelligent navigation in complex environments while maintaining high performance.

References

[1] Wang Minghui, Cao Xiping, Qiao Lijia. Fine investigation of dangerous rock mass and three-dimensional scene simulation of collapse process: A case study of a high slope of a hydropower station in southwest China [J]. Chinese Journal of Geological Hazards and Prevention, 2023,34 (06): 86-96

[2] Liu Mingliang, Liu Huajun, Zhang Shuqing, et al. Design of a 3D visualization system for Super-X devices based on Unity3D [J]. Low Temperature and Superconductivity, 2023,51 (11): 22-28+36

[3] Jiang Yang 3D animation scene design for "The House" [J]. Contemporary Literature, 2023, (06): 12

[4] Sun Yunchuan, Yang Junyan, Liu Xiong, et al. Design and Implementation of a 3D Mining Digital Editing System Based on Unity3D [J]. Coal Mine Safety, 2023,54 (08): 247-251

[5] Li Fan. Game Scene Design Based on Unity 3D [J]. Automation Technology and Applications, 2023,42 (04): 180-182+186

[6] Guan Lianwu, Cong Xiaodan, Zhang Qing, et al. Design of an indoor skiing teaching and training visualization system based on micro inertia and Unity3D [J]. Experimental Technology and Management, 2021,38 (10): 152-156

[7] Bai Xinguo, Zhao Sixian. Design and Implementation of "Pyramid" VR Game Based on Unity3D [J]. Electronic Design Engineering, 2021, 29 (03): 136-140

[8] Fu Mengyuan, Geng Lang. Design and Interactive Implementation of Mobile Game Scenes Based on Unity3D [J]. Chinese and Foreign Entrepreneurs, 2020, (08): 96

[9] Duan Xuekong, Li Tong, Zhu Xudong, etc Design of Character Animation in Unity3D Game Scenes [J]. Computer Knowledge and Technology, 2019,15 (09): 199-200

[10] Randla-Net: Efficient semantic segmentation of large-scale point clouds[J]. Hu Q.;Yang B.;Xie L.;Rosa S.;Guo Y.;Wang Z.;Trigoni N.;Markham A..Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition,2020

[11] Self-training with noisy student improves imagenet classification[J]. Xie Q.;Luong M.-T.;Hovy E.;Le Q.V..Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition,2020

[12] PointasNL: Robust point clouds processing using nonlocal neural networks with adaptive sampling[J]. Yan X.;Zheng C.;Li Z.;Wang S.;Cui S..Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition,2020