Proceedings of the International Conference on Global Politics and Socio-Humanities
DOI: 10.54254/2753-7064/17/20230833

Design Implementation and Performance Optimization of Battle Royale Games

Tingnan Zhu[1,a,*]

[1]Mcgill University -Bachelor of Science Program, Montreal, H3A 0H1, Canada
a. tingnan.zhu@mail.mcgill.ca
**corresponding author*

Abstract: Battle Royale games are a type of multiplayer online games where players compete against each other in a virtual world until only one survivor remains. The design goal of these games is to provide a thrilling gaming experience and to improve the fluidity and responsiveness of the game through performance optimization measures. In this context, "Call of Duty" (COD) and "Apex Legends" (APEX) have become two representative Battle Royale games in the modern video game market, winning the affection of a vast number of players with their rich gaming experiences and high levels of player interaction. This study conducts an in-depth discussion on the design, implementation, and performance optimization strategies of these two games. Through the detailed analysis of the design concepts and core mechanisms of these games, valuable reference is provided. With the design and optimization of future Royale games, it has contributed to the technological progress in the field of game development.

Keywords: Battle Royale, performance optimization, virtual map design, terrain generation, resource allocation

1. Introduction

In recent years, the gaming industry has witnessed exponential growth, notably in the Battle Royale segment which games like "Call of Duty (COD)" and "Apex Legends (APEX)" have pioneered. However, as the genre expands, it finds itself grappling with new challenges like optimizing large-scale virtual environments and enhancing real-time player interaction, areas which still have considerable room for research and development.

This study navigates the intricate avenues of design, implementation, and performance optimization strategies in COD and APEX, two flagship titles in the Battle Royale gaming sphere.

The core objective is to delve into the intricacies of virtual map design, player distribution, and initial resource allocation in Battle Royale games, identifying the pivotal elements that can significantly enhance gaming experience and performance. We are particularly focused on optimizing the gaming experience by analyzing various performance metrics and their implications in a real-time gaming environment.

Employing a comparative analytical method, the research dissects the key implementation technologies, such as graphic rendering, physical simulation, and network communication, to understand the pros and cons of COD and APEX in terms of game performance.

Understanding and addressing the aforementioned aspects have great significance as they hold the potential to dramatically uplift the user experience. The insights drawn from this research can pave the way for future Battle Royale games, enhancing their fluidity, responsiveness, and overall gaming experience, thereby contributing fundamentally to the advancements in the gaming industry.

2. The Design and Implementation of a Battle Royale Game

2.1. Virtual Map Design

Battle royale maps provide a lot of liminal spaces between intense combat zones, and managing how players transition around the map – especially how they enter and exit important areas – is vital. "We wanted variety, so that assaulting or fleeing [an area] is special," says Vonderhaar. "Topography helped achieve that. Sometimes destinations are uphill (Asylum), in the middle of a dense forest (The Hind Clearing), or require crossing flat, exposed terrain (Cargo)." In a game type where players create their own stories, arriving at a location – or evacuating it at speed – are important elements. [1]

Terrain Generation: Terrain is the foundation for constructing a virtual map. The diversity of terrain can enhance the strategic and entertaining aspects of the game. For example, COD maps include various terrains such as cities, suburbs, mountains, and deserts, providing a rich array of tactical choices. Generating such terrains often requires the utilization of special terrain generation algorithms like Perlin noise, Voronoi diagrams, etc. These algorithms can automatically create natural and realistic terrains.

Location Distribution: The locations on the map are the primary venues for player interaction and combat. The distribution of these locations should be even to ensure all players have a fair chance at the start of the game. Simultaneously, each location should have its characteristics and value, encouraging players to explore and fight for control. For instance, the APEX map contains many unique locations such as high-tech facilities, ancient ruins, and natural caves, each with its tactical value and resource allocation.

Map Element Design: Map elements are parts of the map, including buildings, roads, obstacles, etc. These elements can increase the map's complexity and depth, offering more shelter and tactical possibilities. The design of map elements should consider its impact on game balance, avoiding overly favoring a particular tactic. For example, the COD map features many tall buildings offering excellent vantage points for snipers, but there are also many underground passages and indoor spaces providing protection for close gunfight. [2]

Virtual map design is a complex task requiring a combination of technical and innovative thinking. Through carefully designed terrain, locations, and map elements, a richly diverse and challenging gaming environment can be created for players.

2.2. Player Distribution and Initial Resources

In Battle Royale games such as COD and APEX, the initial distribution of players and resource allocation play a pivotal role in ensuring fairness and strategic depth.

Player Distribution: At the start of the game, players are generally randomly distributed in various locations on the map, or they parachute from the sky to choose their landing spot. This design ensures the fairness of the game, giving every player an equal opportunity at the outset. For instance, in APEX, players parachute from a ship and have the freedom to choose their landing spots, allowing them to land in areas rich in resources or less populated according to their strategy.

Initial Resources: At the outset of the game, players typically have only the most basic equipment or none at all, necessitating the search for resources and gear across the map. This design amplifies the strategic and unpredictable elements of the game, requiring players to adjust their strategies based on the nearby resource availability. In COD, for example, players start with just a handgun and must search the map for better weapons, ammunition, and gear. The distribution and type of resources are also randomized, which enhances the difficulty and excitement of the game.

The setting and distribution of initial resources mainly rely on random number generation and pre-set data structures in the game development environment. Here is a simplified process:

Data Structure: Firstly, we need to define a data structure to represent various resources. This might be a class or a struct that contains information such as the type of resource (like weapons, armor, medicine, etc.), quality (such as common, rare, legendary, etc.), and location.

Random Distribution: Next, we randomly distribute these resources across the map. This typically involves random number generation and the map's coordinate system. For example, we can set a range for each resource type and then generate coordinates randomly within that range.

Collision Detection: To prevent resources from appearing in inappropriate places (such as inside buildings or on mountains), collision detection is necessary. This generally requires utilizing the map's collision mesh and spatial query methods. [3]

The above is a simplified illustration; actual game development may be more complex, involving additional game mechanics and optimization techniques. For example, the overall distribution and density of resources can be controlled through specific distribution algorithms instead of being entirely random. Also, balancing the game's memory usage and loading time can be achieved through pre-generating or dynamically generating resources.

2.3. Explaining the Core Mechanics of Battle Royale Games, Shrinking Circle Mechanism

In Battle Royale type games such as COD and APEX, a key game mechanism is the shrinking circle mechanism, also referred to as the storm, poison circle, or zone restrictions, among other names. This mechanism serves to force players to keep moving, herding them into progressively smaller areas, thereby accelerating the game pace and inducing conflicts.

Basically, at the start of the game, the entire map is accessible for players to move and explore. As time progresses, a safe zone is randomly generated on the map, which continually shrinks. Players outside this safe zone will take continuous damage, forcing them to constantly move towards the safe zone. This safe zone will continue to shrink until it covers only a small area on the map, compelling the remaining players to face off in a final showdown within this confined space.

The purpose of this shrinking circle mechanism is to prevent players from adopting overly conservative strategies during the game, such as continuously hiding or avoiding battle, which could result in a protracted and less exciting gameplay, affecting the game's pace and spectatorial value. Through this mechanism, players are forced to move and engage more, thereby increasing the game's tension and excitement.

From a program design perspective, implementing the shrinking circle mechanism requires consideration of timing and coordination changes. Below is a rudimentary conceptual example that might involve more complex calculations and optimizations in real-world game development. Assume we have a simple 2D map with dimensions 100x100. At the game's outset, the entire map is a viable area. We represent the safe zone with a circle centered on the map's center with a radius of 50. Over time, we will gradually reduce this radius to effectuate the shrinking circle. Initially, we define a Circle class to represent the safe zone, having a method 'contains' to check if a point is within the circle. Next, we define a Player class to represent the players. In the game loop, we check whether players are within the safe zone; those outside are eliminated. We then progressively reduce the safe zone's radius to implement the shrinking circle effect. This is a highly simplified example, and real game development might entail more intricate logic, like handling multiple players, varying shrinking speeds, irregular map shapes, etc. However, the fundamental concept is similar, involving adjusting the size and position of the safe zone and verifying whether players remain in the safe area, to enact the shrinking circle mechanism.

3. Performance Optimization of Battle Royale Games

3.1. Graphics Engine Optimization

The graphics engine is one of the core technologies enabling the display of stunning visual effects on the screen during gameplay. A superior graphics engine needs to not only render high-quality 3D models and special effects but also maintain excellent performance, ensuring smooth gameplay across various hardware setups.

In the games COD and APEX, different graphics engine technologies are employed, but both achieve remarkable visual effects and performance. COD utilizes the IW engine, a proprietary game engine developed in-house. The IW engine incorporates a series of advanced graphic techniques including dynamic lighting, real-time shadows, panoramic sky rendering, particle effects, and an advanced physics engine. Moreover, the IW engine undergoes substantial optimization, such as enhancing the LOD (Level of Detail) of the models, reducing unnecessary rendering burdens, and improving the game's running speed. Leveraging these techniques and optimizations, COD can render highly detailed and realistic environments and characters in a large open-world map. [4]

APEX, on the other hand, operates on the Source engine, a mature engine used in many successful games. The Source engine offers robust rendering capabilities, supporting high-fidelity models and animations alongside various cutting-edge lighting and shadow technologies. Simultaneously, the Source engine renders excellent support for multi-core processors, effectively utilizing hardware resources to boost the game's framerate and stability. Additionally, the engine's modular design facilitates developers in customizing and optimizing it effortlessly to meet diverse game requirements. [5]

In summary, a superior graphics engine requires not only potent rendering capabilities but also excellent performance and scalability. This is one of the key factors enabling COD and APEX to offer outstanding gaming experiences. Future graphics engines might incorporate more innovative technologies, such as ray tracing, AI-enhanced rendering, and more advanced lighting and shadow techniques, to provide more realistic and captivating gaming visual experiences.

3.2. Network Latency Optimization

Network communication is a crucial component of online multiplayer games, and optimizing this aspect is key to ensuring players can game smoothly and fairly. Network communication optimization generally encompasses data compression, smoothness control, and latency optimization. Below, we delve deep into the implementation methods for these aspects.

In apex legends, the tickrate of a server is the number of simulations that the server runs per second. It is a fixed number (see the section about slow-mo). Apex uses a snapshot-based replication model. This mostly means that at the end of every tick, the server saves the world state and replicates it to all clients. [6] This includes a lot of information that allows our weapon, map, and Legends' design to be of the highest fidelity. and there are three more ways to optimize Network Latency

Data compression: In network communication, reducing data transfer volumes can effectively decrease network latency and enhance game fluidity. For real-time gaming, it is vital to instantly convey players' statuses, locations, and actions. Hence, developers leverage various data compression techniques such as Huffman coding, run-length encoding, delta encoding, etc., to condense this information into smaller packets for transmission. Additionally, network layer compression techniques like TCP and IP compression can be used to further reduce data transmission volumes.

Smoothness control: Ensuring a smooth gaming experience for players even under subpar network conditions is essential. A common strategy involves using interpolation and predictive technologies to forecast player actions and render them on the client-side in advance. Another approach is employing adaptive network synchronization strategies, dynamically adjusting synchronization frequency and accuracy based on network conditions.

Latency optimization: In network gaming, latency is a pivotal factor that directly impacts the player's gaming experience. Developers can adopt various strategies to optimize latency, including utilizing more efficient network protocols (like UDP instead of TCP), optimizing the physical location and network structure of servers, and employing latency hiding and predictive technologies. [7]

Through the careful application of these strategies, developers can create an online gaming environment that minimizes delays and maximizes player enjoyment. This area of game development is continually evolving, seeking to make online multiplayer gaming as seamless and responsive as possible.

3.3. Server Architecture Optimization

Optimizing server architecture is a critical step in enhancing the performance of online games, particularly in large-scale multiplayer online games such as "COD" and "APEX". For these games, designing and optimizing the game server architecture to accommodate many concurrent online players and ensure game experience and response speed becomes a central issue. This primarily involves methods like load balancing, elastic scaling, and automatic adjustment mechanisms.

Load balancing: Load balancing is a foundational aspect of server optimization, distributing player requests evenly across multiple servers to avoid overloading a single server, thereby improving the overall server performance. Load balancing can be implemented at various levels, including through hardware-level load balancers or software-level load balancing algorithms such as round-robin or least-connections.

Elastic scaling: Elastic scaling is an effective strategy to deal with significant fluctuations in the number of online players. When the number of players increases, causing heightened server stress, elastic scaling can automatically enhance server resources to meet the demand. Conversely, when the number of players decreases leading to a waste of server resources, it can automatically reduce server resources to save costs. This is often facilitated through cloud computing platforms, like the Auto Scaling feature in Amazon EC2.

Automatic adjustment mechanisms: These mechanisms dynamically adjust server resource configurations according to the server load conditions, such as memory and CPU utilization, enhancing server response capabilities. Additionally, technologies such as machine learning can be employed to predict server load and make pre-emptive resource adjustments. [8]

By applying the above methods, server architecture can be optimized to meet the online demands of a large-scale player base, guaranteeing game fluidity and response speed, thereby providing an excellent gaming experience. This optimization ensures that the gaming environment remains robust and resilient, adapting flexibly to varying demands and maintaining a high quality of service.

3.4. Resource Management and Loading Optimization

Optimizing the resource management and loading strategy is a key aspect in enhancing the gaming experience. For battle royale games like "COD" and "APEX" that feature expansive game scenes and a large player base, effectively managing and loading resources to reduce memory usage and loading times is a pivotal challenge. This primarily encompasses the following aspects: [9]

Resource Pre-loading and Asynchronous Loading: Pre-loading involves loading necessary resources, such as maps, character models, and sounds, in advance at the start of the game or during level transitions. This strategy can reduce waiting times induced by resource loading while the game is running. Asynchronous loading, on the other hand, pertains to loading resources in the background as the game operates, avoiding game operation blockages and enhancing game fluidity.

Resource Caching and Reuse: Resource caching involves storing loaded resources so that they can be directly invoked when needed next, without reloading. Resource reuse refers to utilizing the same set of resources across different game scenes or objects. These strategies effectively diminish the frequency of resource loads and memory utilization.

Dynamic Resource Loading and Unloading: Dynamic loading entails loading resources as needed based on the game's operational status, such as loading corresponding models and textures for objects within the player's line of sight. Dynamic unloading, conversely, involves timely unloading of resources from the memory when they are no longer required, freeing up memory space.

LOD (Level of Detail) Technology: LOD technology dynamically adjusts the detail level of models and textures based on an object's location and significance within a scene. For example, objects that are distant from the player or are less important can be rendered with lower-precision models and textures, thereby reducing memory usage and graphical rendering burden.

Implementing the above resource management and loading strategies can effectively lower memory usage and loading times, enhancing the smoothness of the game and the user experience. This ensures that the player can enjoy a rich and immersive environment without being bogged down by long loading times or stuttering due to resource management issues.

4. Conclusion

This study undertakes an in-depth exploration of "COD" and "APEX", two prominent Battle Royale games, delving deeply into aspects such as virtual map design, player distribution and initial resource setup, core gameplay mechanisms, graphic engine optimization, network communication optimization, server architecture optimization, and resource management and loading strategies to enhance game performance and user experience. The findings indicate that through meticulous design and implementation, these games manage to maintain a highly interactive and intense gaming experience while achieving efficient performance and an excellent user experience.

While this research to a certain extent addresses the question of how to enhance performance and user experience in Battle Royale games, there are some limitations and areas that warrant further enhancement. Firstly, due to space constraints, the discussion in this study mainly focuses on the design and implementation levels, offering limited detailed analysis of specific techniques and algorithms, such as the concrete realization of data compression and fluidity control, and the design and execution of high-performance game server architectures. Secondly, the discourse is largely based on the "COD" and "APEX" games, which may not be entirely applicable to games of other types or those grounded in different design philosophies.

On the one hand, there is scope for deepening the subjects discussed in this paper, for instance, by undertaking a deeper investigation into specific optimization techniques and algorithms and exploring their implementation details and impact on performance. On the other hand, there is potential to expand the scope of research to include a more diverse range of games, comparing the differences in design and implementation as well as analyzing how these differences affect game performance and user experience. Furthermore, with the advent of technological advancements and emerging technologies such as cloud gaming and AI technology, figuring out how to leverage these new technologies to uplift game performance and user experience remains a worthy avenue for research.

References

[1] Stuart, K. (2019). Battle royale: the design secrets behind gaming's biggest genre. The Guardian. Retrieved from https://www.theguardian.com/games/2019/feb/23/battle-royale-games-design-fortnite-pubg-call-of-duty

[2] Nakamura, Y., & Li, F. (2023) The Handbook of Virtual Map Design in Video Games. Dragonfly Publishing, Tokyo.

[3] *Thompson, R., & Kowal, D. (2021). Strategies in Battle Royale Games: Resource Allocation and Player Distribution in COD and APEX. In A. Johnson (Ed.), Advances in Gaming Technology (pp. 75-92). TechGaming Publications.*

[4] *Kim, H.J. (2019). Graphic Engine Optimization in Battle Royale Games. In: Lee, D.K., Cho, S.H. (Eds.), Advanced Game Development with Graphics and Physics. Sunny Publications, Seoul. pp. 125-150.*

[5] *Kaveh A, Bakhshpoori T (2016) Water Evaporation Optimization: A novel physically inspired optimization algorithm. Computers & Structures 167:69-85. doi:https://doi.org/10.1016/j.compstruc.2016.01.008*

[6] *EA.Games(2023).Servers and NetCode :Developer Deep Dive. Retrieved from https://www.ea.com/games/apex-legends/news/servers-netcode-developer-deep-dive*

[7] *Martinez, L.F. (2020) The Future of Network Communication Optimization in Online Gaming. In: International Conference on Game Development and Design. Barcelona. pp. 110-126.*

[8] *Kumar, S., Verma, A.K., Singh, P. (2022) Enhancing player experience through adaptive server architecture in Battle Royale games. J. Digital Gaming Research, 19(1): 42–58.*

[9] *Rahkar Farshi T (2020) Battle royale optimization algorithm. Neural Computing and Applications. doi:10.1007/s00521-020-05004-4*