

Optimization of a Next-Generation 3D Zombie Survival FPS Game Architecture Based on Unreal Engine 5

Jaehoon Ahn¹, Kwangsik Park², Daekyung Lee³, Seungjae Lee^{4*}

^{1,2,3,4} Computer Engineering Department, Sunmoon University, Korea

ABSTRACT: This paper proposes the design, technical implementation process, and optimization techniques of a 3D zombie disaster survival simulation game developed using the latest game engine, Unreal Engine 5. To maximize the player's visual and experiential immersion, the system constructs an intelligent tracking AI system combining the AI Perception Component and Behavior Tree. It also implements a precise first-person shooting (FPS) mechanism utilizing Unreal Engine's physical Collision system. Furthermore, it maximizes the horrific atmosphere by introducing wide-angle distortion and camera shaking techniques—physical characteristics of action cameras—into the camera direction, and integrates the next-generation Lumen rendering technology. This study includes a practical troubleshooting methodology to resolve complex physical calculation errors and large asset synchronization conflicts that inevitably occur in a 3D game development environment with multiple developers. Through this, it proves that the system developed in this study simultaneously provides outstanding visual direction and a stable play environment.

KEYWORDS: Unreal Engine 5, Behavior Tree, First-Person Shooter (FPS), Collision Optimization.

INTRODUCTION

Recently, games in the 3D horror and First-Person Shooter (FPS) genres have gone beyond mere visual graphic advancements, inducing player immersion through precise interactions between intelligent Non-Player Character (NPC) AI and realistic physics engines [6, 7]. As game engine technologies evolve, players now expect highly responsive game worlds where enemies react logically to their surroundings rather than following pre-scripted paths. In particular, breaking away from the standardized, rigid first-person perspective, active research is being conducted to induce a deeper emotional identification between the player and the game protagonist [9]. To maximize this emotional resonance and provide extreme fear, developers are introducing the physical characteristics of an action camera into the game's perspective. These immersive techniques include applying wide-angle lens distortion to intentionally disorient spatial perception, and implementing dynamic camera shake that occurs naturally during character movement or combat. Additionally, various elements deliberately designed to convey fear, such as claustrophobic, closed map designs and the relentless pursuit of monsters, are organically combined to impose constant psychological pressure on the player.

Despite these industry-wide advancements, significant technical hurdles remain for smaller development teams. In small-scale indie game development environments or undergraduate-level projects, there are inherent technical limitations where the overall play experience is significantly degraded in an attempt to avoid system computation overhead [8]. For example, to save processing power, developers often rely on one-dimensional AI tracking methods based on simple, frame-by-frame distance calculations (Tick) [8]. This approach not only causes severe performance bottlenecks when multiple enemies are spawned but also results in monotonous and predictable enemy behavior. Furthermore,

the use of simplified and inaccurate collision detection to reduce physics calculations frequently breaks the game's realism, leading to immersion-breaking issues such as objects overlapping or projectiles missing their targets.

To overcome these developmental challenges, this study actively adopts the latest technical trends and optimization guidelines provided in the official Unreal Engine documentation [1, 2]. Based on these guidelines, this paper presents a comprehensive architecture and implementation methodology for a zombie disaster survival game that maximizes visual and psychological immersion. Specifically, this research aims to secure both optimal system processing performance and high-level gameplay. This objective is achieved through the implementation of a visual cognition-based AI logic that eliminates unnecessary tracking overhead, meticulous physical collision and projectile optimization to prevent clipping errors, dynamic camera direction for heightened realism, and a highly efficient Event-Driven UI design that significantly reduces CPU usage.

I. SYSTEM ARCHITECTURE

This game system consists of four core modules (intelligent AI control, player combat, rendering and camera, and UI optimization) that operate independently yet organically exchange data [8].

2.1. Behavior Tree-Based Intelligent Zombie AI System

To break away from the method of unconditionally tracking the player's position every frame, which causes computation overload, the official AI framework provided by Epic Games, UAIPerceptionComponent, was combined with the Behavior Tree system [1, 3]. A sight radius (1,000cm) and a peripheral vision angle of 45 degrees are set, and callback events are restrictively triggered only when a player is detected within this cone-shaped field of view [7]. The detected player's position information is immediately recorded in the Blackboard memory, and based on this data, the Behavior

Tree asynchronously processes the Finite State Machine (FSM) logic by independently determining the Idle, Patrol, Chase, and Attack states. Through this, an intelligent enemy behavior pattern that actively utilizes the terrain was implemented while minimizing system load.

2.2. Projectile Shooting and Physical Collision Optimization

The combat system adopted a first-person shooter (FPS) method that dynamically generates (SpawnActor) independent bullet actors in the world upon Left Mouse Button (LMB) input. To ensure precise judgment upon hitting objects in the game, Unreal Engine's Collision channels and Collision Responses system were meticulously adjusted [2]. In particular, to fundamentally prevent the tunneling phenomenon, where fast-moving bullet actors pass straight through thin walls or enemy models without physical collision during frame drops, the Continuous Collision Detection (CCD) feature was activated.

2.3. Action Camera Perspective and Lumen Rendering Direction

To maximize the player's fear and sense of presence, the

physical characteristics of an action camera were actively applied to the first-person perspective direction [9]. The player's spatial sense was intentionally distorted by combining perspective distortion using a wide-angle lens and Camera Shaking that occurs during character movement and when taking damage. In addition, Lumen global illumination, the core next-generation rendering technology of Unreal Engine 5, was applied to create realistic dynamic reflections and real-time shadows caused by the player's flashlight or muzzle flash, thereby visually completing the dark and closed atmosphere typical of the horror genre [4, 10].

2.4. UMG-Based Event-Driven UI

The UMG system was utilized to display status information such as the player's HP, stamina, and ammo count on the screen in real-time [5]. Moving away from the inefficient structure of existing games that update UI text and gauges every frame (Tick), an Event-Driven synchronization method was adopted. This method drastically saves CPU computation resources by sending a signal to the UI widget via a Delegate only when the values actually change due to the player taking damage or shooting.

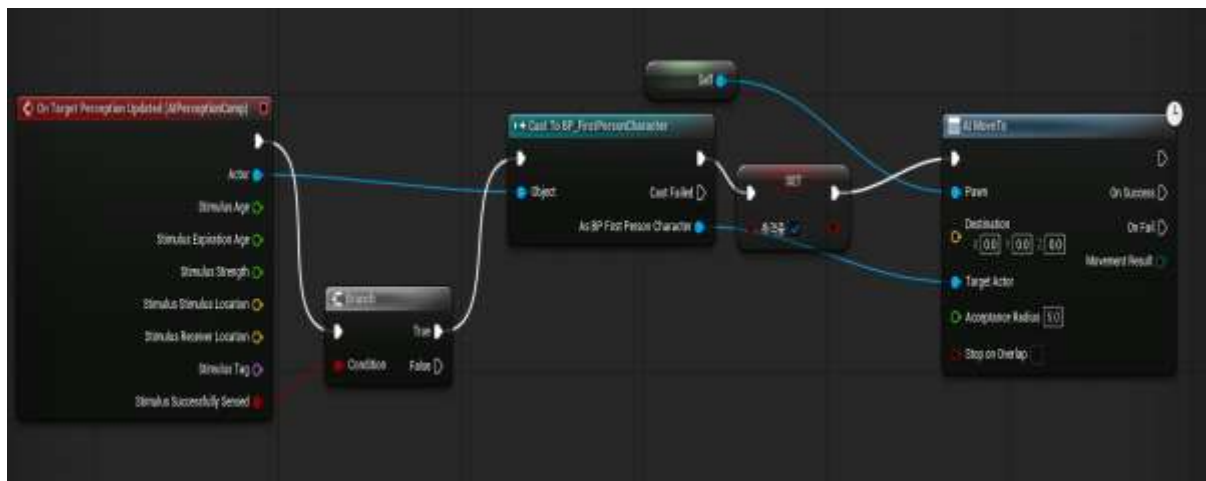


Fig. 1. System Flow

II. RESULT AND DISCUSSION

In a 3D game project development environment where multiple assets and systems are integrated, various engineering problems and structural issues inevitably occur. This section details the core problems encountered during the collaborative development process and the technical solutions applied to resolve them efficiently.

3.1. Dynamic Spawn Terrain Collision and Overlap Issues

A major technical issue was identified when dynamically spawning monsters at random coordinates across the map. Specifically, if the capsule component of the newly generated actor overlapped with the bottom of the terrain or uneven floor curves, the spawn request was frequently rejected by the game engine's internal collision protection mechanism. In other severe cases, this physical overlap caused the actor to clip through the geometry and fall out of the map entirely, disrupting the game's progression. To fundamentally resolve this issue, the default collision rules within the FActorSpawnParameters were forcefully overridden to the AlwaysSpawn setting. Furthermore, the derived Z-axis coordinate for the spawn location was deliberately adjusted upwards to avoid ground intersection. By having the engine safely push the actor into an empty space above the terrain before applying gravity, the game's spawning logic was

significantly strengthened to ensure successful and stable entity generation under any terrain condition.

3.2. Physical Errors in Complex Static Meshes

Another critical challenge involved high-resolution environment assets, such as collapsed debris, furniture, and pillars, which are essential for a survival horror atmosphere. In these complex static meshes, a collision ignore phenomenon frequently occurred, allowing the player character and projectiles to pass straight through the objects. This immersion-breaking phenomenon occurred primarily because the detailed visual modeling of the asset and its simplified physical collision body did not accurately match. To address this structural discrepancy, the Collision Complexity property of these problematic assets was collectively changed to the 'Use Complex Collision As Simple' setting in the engine. As a result of this adjustment, the physics engine was forced to accurately recognize physical collisions at the precise polygon level of the mesh. This targeted optimization successfully secured movement stability and reliable projectile interactions for the player traversing complex indoor and outdoor environments.

3.3. Large Asset Synchronization Conflicts

Collaborative development introduced significant version control challenges, particularly when multiple team members

worked on the same levels simultaneously or attempted to synchronize large binary texture files via GitHub. These overlapping actions frequently resulted in severe Git merge conflicts and serialization runtime errors, often crashing the engine with a 'Fatal Error' message upon loading. To solve these collaboration bottlenecks and ensure a stable version control environment, the target tracking of Git LFS (Large File Storage) was carefully subdivided and fully introduced into the repository. Additionally, a mandatory Clean Build

pipeline was established for the development team. In the event of a synchronization conflict, this pipeline instructed developers to manually delete and recreate their local Binaries and Intermediate cache folders to force a fresh compilation. The implementation of this pipeline effectively increased the recovery rate from synchronization errors, minimizing downtime and stabilizing the collaborative workflow.

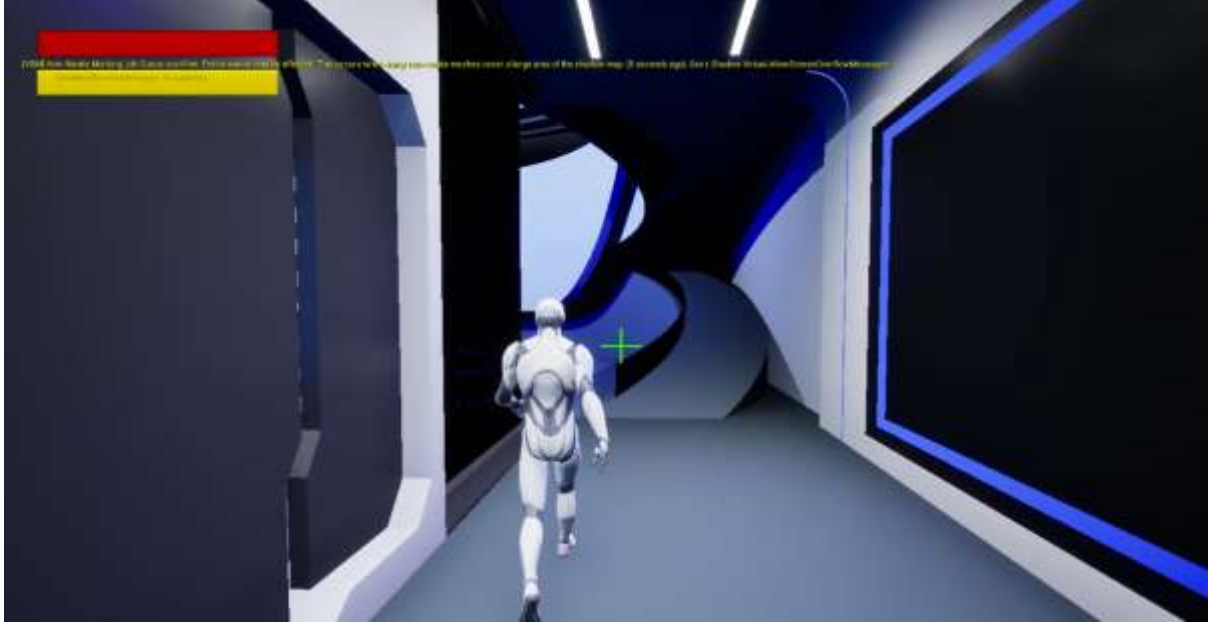


Fig. 2. Sample Screen of the Game

III. CONCLUSIONS

This study successfully implemented an FPS-based zombie survival game with high immersion at the level of a commercial game by technically integrating Unreal Engine 5's official AI framework, collision physics system, and action camera perspective direction. By moving away from resource-intensive tracking methods, the implementation of a sight-based intelligent AI utilizing the UAIPerception Component and Behavior Tree significantly reduced system overhead while maintaining unpredictable and threatening enemy behaviors. Furthermore, the meticulous optimization of projectile mechanics using Continuous Collision Detection (CCD) ensured precise hit registrations, fundamentally preventing tunneling effects during rapid combat scenarios. The visual atmosphere, a critical component of the survival horror genre, was brought to life by combining the physical characteristics of an action camera—such as wide-angle lens distortion and dynamic camera shaking—with the advanced real-time rendering capabilities of Lumen global illumination. Beyond the game's core mechanics, this research offers meaningful contributions to the collaborative development process. In particular, the AI computation optimization through the Behavior Tree and the experience of troubleshooting large-scale configuration management provide practical guidelines for the future development of large-scale 3D spatial information-based convergent content and metaverses. During the project, the team systematically resolved critical engineering challenges, including dynamic spawn overlap issues on uneven terrain, physical collision errors within complex high-resolution static meshes, and severe Git merge conflicts involving large binary assets. By establishing a robust troubleshooting methodology—such as enforcing Git LFS tracking and Clean Build pipelines—this

paper provides a valuable, actionable blueprint for small-scale indie developers and academic teams aiming to stabilize their version control and physical rendering environments.

While the current system provides a highly immersive single-player experience, several avenues remain for subsequent research. Future work will proceed with the introduction of a dynamic difficulty adjustment system utilizing Deep Learning and Replication optimization to build a network-based multiplayer environment. Implementing a Deep Learning-based director AI would allow the game to analyze player performance in real-time, dynamically adjusting item spawn rates and zombie aggression to maintain an optimal state of tension and flow. Additionally, to expand the survival experience into a cooperative mode, further studies will focus on engine-level network Replication techniques to ensure that AI behaviors, projectile physics, and Event-Driven UI states are seamlessly synchronized across multiple clients without severe latency.

ACKNOWLEDGEMENT

"This research was supported by the MSIT(Ministry of Science ICT), Korea, under the National Program for Excellence in SW, supervised by the IITP(Institute of Information & Communications Technology Planning & Evaluation) in 2026"(No. 2024-0-00023).

REFERENCES

1. Public Data Portal, "Government24 Public Service Open API." [Online]. Available: https://www.data.go.kr/data/15113968/openapi.do#/tab_layer_detail_function
2. Spring, "Spring Boot Reference Documentation." [Online]. Available: <https://docs.spring.io/spring->

- boot/
3. Spring, “Spring Security Reference Documentation.” [Online]. Available: <https://docs.spring.io/spring-security/>
 4. Meta Open Source, “React Documentation.” [Online]. Available: <https://react.dev/>
 5. OpenAI, “OpenAI API Documentation.” [Online]. Available: <https://platform.openai.com/docs/>
 6. PostgreSQL Global Development Group, “PostgreSQL Documentation.” [Online]. Available: <https://www.postgresql.org/docs/>
 7. Y.-J. Ahn, H.-Y. Cho, and S.-J. Kang, “Customized Recipe Recommendation System Implemented in the Form of a Chatbot,” *Journal of the Korea Academia-Industrial Cooperation Society*, vol. 21, no. 5, pp. 543-550, 2020.
 8. J.-H. Jung, W.-J. Lee, H.-W. Choi, D.-Y. Seo, J.-H. Lee, H.-B. Park, and H.-Y. Lee, “Development of a Customized Webtoon Recommendation Platform Using the GPT Model,” *Proceedings of KIIT Conference*, pp. 1361-1364, 2025.
 9. J.-W. Son, H.-B. Kim, D. Yoon, S.-H. Park, K.-I. Lee, and S.-J. Kim, “Development of an AI Chatbot-Based Personalized Restaurant Recommendation System,” *Proceedings of KIIT Conference*, pp. 1273-1277, 2025.
 10. S.-R. Jung and J. Kim, “Sentiment-Fused Hybrid Recommender System for Movie Domain: Integrating Aspect-Based Emotion Analysis with Collaborative-Content Filtering,” *Journal of the Korea Big Data Society*, vol. 10, no. 1, pp. 209-219, 2025.