УДК 004.514:004.946

ОПТИМІЗАЦІЯ СЕРВЕРНИХ РІШЕНЬ В UNREAL ENGINE 5

Яковенко Д. О.
Науковий керівник – ст. викл. Новіков Ю. С.
Харківський національний університет радіоелектроніки
61166, Харків, просп. Науки, 14, каф. ПІ,
e-mail: dmytro.iakovenko1@nure.ua

This paper explores the process of server optimization on the Unreal Engine 5 platform. The article covers strategies and methods aimed at improving the performance of server systems in the context of using this game engine. In particular, techniques for optimizing network interaction, data processing, and resource-intensive operations are considered. Practical examples of implementing solutions to improve performance and ensure stable server operation during the development of large game projects on Unreal Engine 5 are considered. This paper provides valuable guidance and recommendations for developers who seek to optimize server performance on this platform.

Unreal Engine 5 (UE5) – це потужний інструментарій, який розкриває безмежні можливості для розробки ігор нового покоління. Він пропонує широкий спектр інструментів, що робить його ідеальним вибором як для створення одиночних, так і для багатокористувацьких проектів.

Одне з ключових особливостей UE5 – це революційна система реплікації, яка забезпечує безперебійну синхронізацію даних між клієнтами в реальному часі. Завдяки цьому, UE5 гарантує плавний ігровий процес без затримок та лагів.

Для оптимізації серверних рішень в UE5, необхідно зрозуміти одне: чим менше даних треба реплікувати, тим краще. Тому уся оптимізаця побудована на ігноруванні непотрібних для реплікації даних, та зменшення об’єму реплікованих даних.

Для початку розглянемо випадок, як можна зменшити об’єм реплікованих даних на прикладі обертання об’єктів у просторі. В UE5 є два варіанти точності реплікації обертання у просторі: Byte(8біт) Та Short(16 біт). В рамках розроблюваного проекту, є 4 актора, які мають обертатися, та синхронізувати своє полодження між клієнтами. Результати тестування цих акторів в обох режимах представлено у таблиці 1.

Таблиця 1 – тестування різних режимів передачі полодження в просторі

	Short		Byte	
Об'єкти	Avg Bytes	Avg ms	Avg Bytes	Avg ms
1	14,3	0,009	11,2	0,007
2	12,1	0,009	8,7	0,007
3	11,9	0,009	8,7	0,007
4	1,5	0,009	1,5	0,007

Проведене дослідження свідчить про те, що синхронізація положення Short потребує на 25-30% більше часу, порівняно з Byte. З огляду на цю характеристику, Byte виступає більш економним рішенням у більшості випадків. Однак, важливо зазначити, що Byte має вдвічі меншу точність, ніж Short. В деяких ситуаціях це може призвести до візуального покращення плавності анімації обертання.

В рамках розроблюваного проекту, зброя має механізм перегрівання, синхронізація якого між клієнтами є необхідною. Зважаючи на динамічний характер даної характеристики, що змінюється кожен ігровий такт, просте реплікування змінних стає нерентабельним (див. рис. 1).

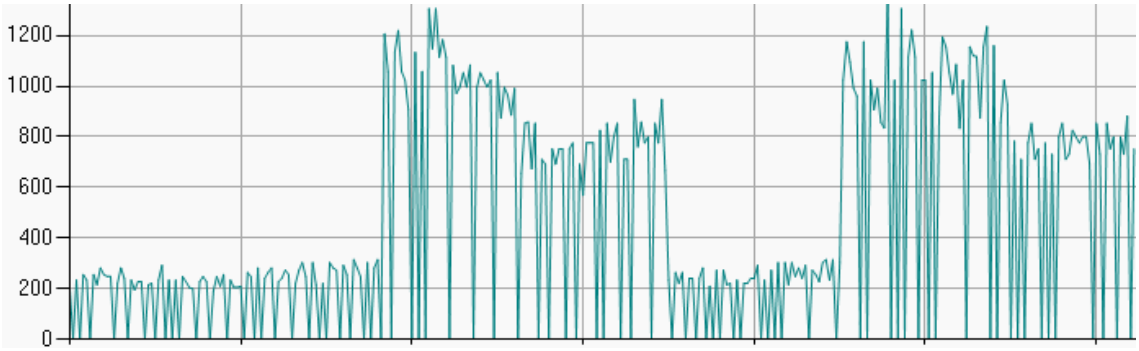

Рисунок 1 – Затрати, які спричиняє використання зброї до оптимізації

Як видно на рисунку 1, при використанні зброї, кожен такт сервер передає значення нагріву зброї, що навантажує його на додаткові 250 Байтів на секунду. Цього можна уникнути, якщо передавати нагрівання зброї тільки в момент коли гравець починає або закінчує її використання.

Оскільки інформація про перегрівання зброї необхідна лише в момент її використання, та вона необхідна лише тому клієнту, що її використовує, то можна перенести розрахунок перегріву з сервера, на клієнт, що тримає цю зброю у руках, а у момент, коли ніхто не тримає зброю, обчисленням нагрівання займеться сервер.

Передачу інформації про поточний стан від клієнта до сервера зробимо в момент зупинки використання зброї клієнтом, а передачу інформації від сервера до клієнта, в момент початку використання зброї клієнтом.

Таким чином, можна знизити навантаження на сервер, та на мережу, що можна побачити на рисунку 2.

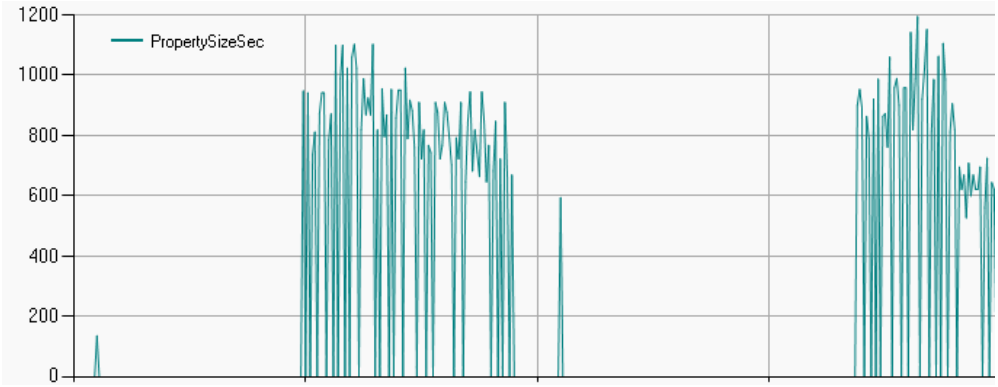

Рисунок 2 – Затрати, які спричиняє використання зброї після оптимізації

Як видно, при порівнянні графіків до та після оптимізації, в результаті ми отримали зменшення навантаження на мережу, що спричиняє використання зброї, на 20%. Також, це покращило швидкість відповіді сервера, шо можна побачити на таблиці 2.

Таблиця 2 – Порівняння середнього розміру та часу на отримання пакету для зброї, під час її використання, до та після оптимізації

	Avg Bytes	Avg ms
До оптимізації	8,5	0,008
Після оптимізації	3,7	0,004

Як видно з таблиці 2, оптимізація знизила час на відправку пакетів на 50%, що можна інтерпретувати як те, що тепер використання двох зброй одночасно всередньому спричиняє таку ж затримку, як використання однієї до оптимізації.

Список використаних джерел:

1. Документація Unreal Engine 5.0 Multiplayer and Networking. URL: https://docs.unrealengine.com/5.0/en-US/networking-and-multiplayer-in-unreal-engine/ (дата звернення 03.03.2024).