НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Інтегровані інформаційні системи»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Онлайн-гра на базі Server-Authoritative архітектури»

Виконав:
студент IV курсу, групи ІА-92
Чикрій Андрій Олексійович _____

Керівник:
доцент кафедри ІСТ, кандидат технічних наук,
Галушко Дмитро Олександрович _____

Рецензент:
доцент кафедри РТС РТФ, кандидат технічних наук,
Катін Павло Юрійович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2023 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інтегровані інформаційні системи»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту

Чикрію Андрію Олексійовичу

1. Тема проєкту «Онлайн-гра на базі Server-Authoritative архітектури», керівник проєкту Галушко Дмитро Олександрович, старший викладач кафедри ІСТ, к.т.н., затверджені наказом по університету від «31» травня 2023 р. №2101-с

2. Термін подання студентом проєкту: 12 червня 2023

3. Вихідні дані до проєкту: *платформа розробки та цільова платформа – ПК, відсутність можливості клієнта отримати перевагу у нечесний спосіб.*

4. Зміст пояснювальної записки:

1. *Аналіз проблем сучаних онлайн-ігор: поняття онлайн-гри, Server-Authoritative архітектура, аналіз актуальних онлайн-ігор.*
2. *Проєктування гри: дизайн гри, ігровий рушій Unity, вибір мережевої бібліотеки, мова програмування C#, середовище розробки VSCode, с истема контролю версій Git.*

3. *Реалізація гри: загальна структура ігрової сцени, Remote Procedure Call (RPC), реалізація ігрових механік*
4. *Технологічний розділ: керівництво користувача, перевірка авторитетності сервера, подальша розробка та інтеграція Unity Game Services*

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо):
1. Схема ініціалізації гравців
2. Схема отримання значень показників для відображення
3. Схема серверних та клієнтських меж арени
4. Схема синхронізації переміщення юніта
5. Схема активації модуля
6. Діаграма потоків даних
7. Структура ігрової сцени

7. Дата видачі завдання: 1 березня 2022 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Затвердження теми роботи	22.05.23	
2.	Аналіз предметної області та існуючих проєктів	24.05.23 - 26.05.23	
3.	Проєктування гри	26.05.23 - 28.05.23	
4.	Дослідження засобів реалізації	28.05.23 - 01.06.23	
5.	Реалізація гри	01.06.23 - 08.06.23	
6.	Тестування застосунку	08.06.23 - 09.06.23	
7.	Оформлення текстової документації	09.06.23 - 12.06.23	

Студент Андрій ЧИКРІЙ

Керівник Дмитро ГАЛУШКО

АНОТАЦІЯ

Ключові слова: гра, онлайн-гра, архітектура, авторитетний сервер, мережа, клієнт, сервер, користувач, гравець, чит, сесія, Unity, Netcode, C#.

Пояснювальна записка дипломного проєкту містить 4 розділи, 22 рисунки, 6 таблиць, 2 додатки, 20 джерел.

Метою розробки онлайн-гри, що базуватиметься на Server-Authoritative архітектурі, є підвищення якості ігрового процесу та зменшення вразливості щодо можливості ігрового шахрайства.

Для реалізації системи було використано: ігровий рушій Unity Engine, бібліотека Netcode for GameObjects, мова програмування C#, середовище розробки Visual Studio Code, система контролю версій Git.

У дипломному проєкті було реалізувано програмний продукт, що демонструє основні елементи змагальної онлайн-гри і має мінімальну вразливість щодо можливості ігрового шахрайства

Розроблений застосунок дозволяє грати у локальній мережі, а за продовженої розробки можуть бути інтегровані хмарні сервіси, що зробить гру доступною гравцям з усього світу.

ANNOTATION

Keywords: game, online game, architecture, authoritative server, network, client, server, user, player, cheating, session, Unity, Netcode, C#. The explanatory note of the diploma project contains 4 chapters, 22 figures, 6 tables, 2 appendices, 20 sources.

The purpose of developing an online game based on Server-Authoritative architecture is to improve the quality of the game process and reduce the vulnerability to the possibility of game cheating. The following were used to implement the system: Unity Engine game engine, Netcode for GameObjects library, C# programming language, Visual Studio Code development environment, Git version control system.

The diploma project implemented a software product that demonstrates the main elements of a competitive online game and has minimal vulnerability to the possibility of cheating.

The developed application allows you to play in a local network, and with continued development, cloud services can be integrated, which will make the game available to players from all over the world.

Номер рядка	Формат	Позначення	Найменування	Кільк. аркушів	Номер екзем.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІА92.240БАК.004 ПЗ	Пояснювальна записка	72		
6	А3	ІА92.240БАК.004 Д1	Онлайн-гра на базі Server- Authoritative	1		
7			архітектури. Схема ініціалізації гравців			
8	А3	ІА92.240БАК.004 Д2	Онлайн-гра на базі Server- Authoritative	1		
9			архітектури. Схема отримання значень			
10			показників для відображення			
11	А3	ІА92.240БАК.004 Д3	Онлайн-гра на базі Server- Authoritative	1		
12			архітектури. Схема серверних та			
13			клієнтських меж арени			
14	А3	ІА92.240БАК.004 Д4	Онлайн-гра на базі Server- Authoritative	1		
15			архітектури. Схема синхронізації			
16			переміщення юніта			
17	А3	ІА92.240БАК.004 Д5	Онлайн-гра на базі Server- Authoritative	1		
18			архітектури. Схема активації модуля			
19	А3	ІА92.240БАК.004 Д6	Онлайн-гра на базі Server- Authoritative	1		
20			архітектури. Діаграма потоків даних			
21	А3	ІА92.240БАК.004 Д7	Онлайн-гра на базі Server- Authoritative	1		
22			архітектури. Структура ігрової сцени			
23						
24						
25						
26						
27						
28						

Зм.	Аркуш	№ докум.	Підпис	Дата	ІА92.240БАК.004 ТП		
Розроб.		Чикрій А.О.			Онлайн-гра на базі Server-Authoritative архітектури. Відомість проєкту	Літ.	Аркуш 1 / Аркушів 1
Керівн.		Галушко Д.О.				Т	
						КПІ ім. Ігоря Сікорського	
						Група ІА-92	
Затв.							

Пояснювальна записка до дипломного проєкту на тему: «Онлайн-гра на базі Server-Authoritative архітектури»

Київ – 2023 року

ЗМІСТ

ВСТУП

У сучасному світі індустрія цифрових розваг є однією з найбільш швидко зростаючих галузей, яка має величезний вплив на економіку та культуру суспільства. Індустрія цифрових розваг включає в себе виробництво відеоігор, веб-сайтів, мобільних додатків, анімаційних фільмів, музики, книг та інших видів цифрового контенту.

Індустрія цифрових розваг також має значний вплив на розвиток технологій та інновацій. Через постійний пошук нових ідей та технологій, ця галузь стимулює розвиток інформаційних технологій та допомагає у вирішенні складних технічних завдань.

Комп’ютерні ігри є однією з найпопулярніших форм розваг у сучасному світі, а онлайн-ігри стають все більш популярними. Ці ігри мають великий вплив на життя людей та суспільства, сприяючи розвитку креативності, критичного мислення, соціальних навичок та навичок співпраці.

Онлайн-ігри, зокрема, є важливим засобом взаємодії між людьми з різних країн та культур. Вони дозволяють користувачам об’єднуватися в онлайн-спільноти та взаємодіяти між собою у віртуальному світі. Це дає можливість людям розширювати соціальне коло знайомств, вчитися комунікувати та співпрацювати з іншими людьми, навчатися вирішувати конфлікти та розвивати навички лідерства.

Крім того, онлайн-ігри можуть стимулювати розвиток економіки. Багато онлайн-ігор є комерційними продуктами, які залучають велику кількість гравців та споживачів. Це сприяє розвитку цифрової економіки, створенню нових робочих місць та підтримує інновації в галузі інформаційних технологій.

Проблема читів в онлайн-іграх є однією з найбільш серйозних в ігрової індустрії. Чити (англ. cheats) або чит-програми - це програми або компоненти програм, що надають гравцеві незаконну перевагу в грі. Це можуть бути різноманітні функції, такі як нескінченне здоров'я, невидимість, збільшення швидкості чи точності стрільби, та інші. Використання читів є найпоширенішим видом ігрового шахрайства [1].

Одним з найбільших наслідків використання читів є те, що вони порушують баланс гри, знижуючи її рівень складності та справедливості. Гравці, які використовують чити, мають непорівнянну перевагу перед тими, хто грають чесно, що веде до негативного досвіду для останніх.

Крім того, використання читів може мати серйозні наслідки для онлайн-громади та ігрової індустрії в цілому. Наприклад, велика кількість гравців, які використовують чити, може вплинути на популярність гри, що призведе до її невдачі та закриття. До того ж, використання читів може негативно вплинути на імідж гравців, які відомі своїми здібностями та навичками в грі, а також на імідж самої гри.

Щоб запобігти використанню читів, розробники онлайн-ігор використовують різні заходи безпеки, такі як античит-системи, що можуть виявляти та блокувати використання читів у грі. Також можуть бути введені санкції проти гравців, які використовують чити, такі як блокування акаунту або тимчасова заборона на гру. Але базовою запорукою мінімальної або відсутньої можливості гравців отримати незаконну перевагу є правильні архітектурні підходи на етапі проєктування ігрової програми.

Узагальнюючи, проблема ігрового шахрайства в онлайн-іграх є досить складною та актуальною, оскільки вона підриває рівень рівноправності в грі, порушує баланс та впливає на імідж гри та індустрії в цілому. Важливо розробляти та вдосконалювати рішення для забезпечення максимальної безпеки та справедливості для гравців.

Мета і завдання дослідження**.** Метою розробки онлайн-гри, що базуватиметься на Server-Authoritative архітектурі, є підвищення якості ігрового процесу та зменшення вразливості щодо можливості ігрового шахрайства.

Для досягнення мети необхідно вирішити наступні завдання:

– дослідити існуючі архітектурні підходи щодо проєктування онлайн-ігор, визначити їх переваги та недоліки;

– проаналізувати існуючі технології для створення онлайн-ігор;

– розробити ігровий процес, що складатиметься з ігрових механік реалізованих із дотриманням Server-Authoritative архітектури;

– на основі розробленого ігрового процесу реалізувати програмний продукт, що демонструватиме основні елементи спроєктованої гри і матиме мінімальну вразливість щодо можливості ігрового шахрайства.

Об’єктом дослідження є онлайн-гра.

Предметом дослідження є Server-Authoritative архітектура для онлайн-ігор.

Дипломний проєкт складається з наступних розділів: вступ, основні розділи, висновки, список використаних джерел із 20 найменувань, 2 додатки. Графічна частина включає 7 креслеників формату А3. Загальний обсяг 72 сторінки.

1 АНАЛІЗ ПРОБЛЕМ СУЧАСНИХ ОНЛАЙН-ІГОР

1.1 Поняття онлайн-гри

Багатокористувацька гра або мультиплеєр (англ. multiplayer – «множина гравців») – режим комп'ютерної гри, під час якого грає більше однієї людини.

Першою багатокористувацькою грою вважається Tennis for Two (1958 рік). Гра була зроблена для аналогової ЕОМ та виводила ігрове поле на осцилографі [2].

Перші мережеві багатокористувацькі мережеві ігри (онлайн-ігри) вийшли в 1973 році. Приблизно одночасно створюються ігри Maze War, Empire і Spasim. Оскільки точна дата створення ігор не відома, вони досі претендують на звання першої онлайн-гри. Maze War – тривимірний шутер від першої особи з мережевою грою в режимі Deathmatch. Empire – розрахований на багато користувачів мережевий 2D-шутер (до 50 гравців). Spasim – багатокористувацький мережевий космічний симулятор (до 32 гравців) [2].

Під час гри в мережі кілька комп'ютерів з'єднані у мережу для обчислень. Зазвичай це може бути фізична локальна мережа, віртуальна локальна мережа або мережа Інтернет. Онлайн-ігри розподіляються за принципом організації зв'язку між цими комп'ютерами.

Peer-to-peer (P2P) (з англ. – рівний до рівного, однорангова мережа) є мережею, в якій відсутній виокремлений центральний комп'ютер; кожен комп'ютер передає інформацію іншим комп'ютерам. Усі комп'ютери мають достовірну інформацію про гру. Зазвичай один з комп'ютерів виступає як хост або сервер, його функція полягає у встановленні темпу гри та керуванні грою (зміна рівня, налаштувань тощо). Якщо хост виходить, будь-який інший комп'ютер може взяти на себе роль хоста. Цей режим часто використовується в стратегіях реального часу [2]. Схема моделі мережі Peer-to-peer зображена на рисунку 1.1.

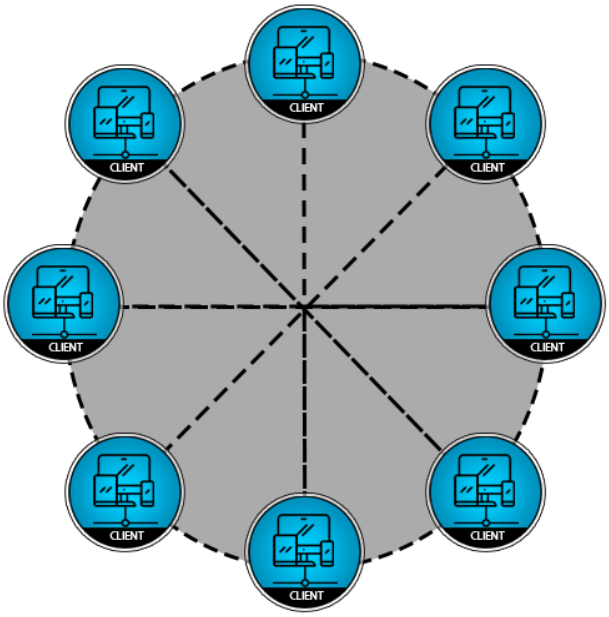

Рисунок 1.1 – Модель мережі Peer-to-peer [3]

Переваги: простота реалізації; у цьому режимі не потрібно мати центральний комп'ютер, оскільки кожен комп'ютер у мережі може передавати інформацію іншим комп'ютерам, що забезпечує гнучкість і масштабованість гри, оскільки ігровий світ може бути складним, але трафік залежить лише від кількості гравців. Одним із комп'ютерів, зазвичай, відводиться роль хоста, який відповідає за керування грою і встановлення темпу гри. Проте, якщо хост виходить з гри, будь-який інший комп'ютер може прийняти його обов'язки. Крім того, P2P режим дозволяє знизити затримки передачі даних і мінімізувати навантаження на хоста, оскільки навантаження розподіляється між комп'ютерами.

Недоліки: великі ризики при зламі гри, можливо повністю підкорити ігрову логіку; велика кількість гравців призводить до надвеликого трафіку; практично неможлива протидія неякісному зв'язку; проблемний вхід гравця в розпочату гру; необхідний канал зв'язку кожного з кожним; високе навантаження на керовані машини.

Застосування: перші мережеві багатокористувацькі ігри (на кшталт Doom), ігри для малої кількості гравців, ігри для телефонів або кишенькових приставок, а також стратегії в реальному часі.

Host-based (зіркоподібний зв'язок) мережа використовує архітектуру, яка подібна до "peer-to-peer", проте весь зв'язок здійснюється через один центральний комп'ютер, який виступає в ролі хоста. Цей режим є перехідним між "peer-to-peer" і "server-based" архітектурами, оскільки він поєднує деякі особливості обох підходів [2]. Схема моделі Host-based мережі зображена на рисунку 1.2.

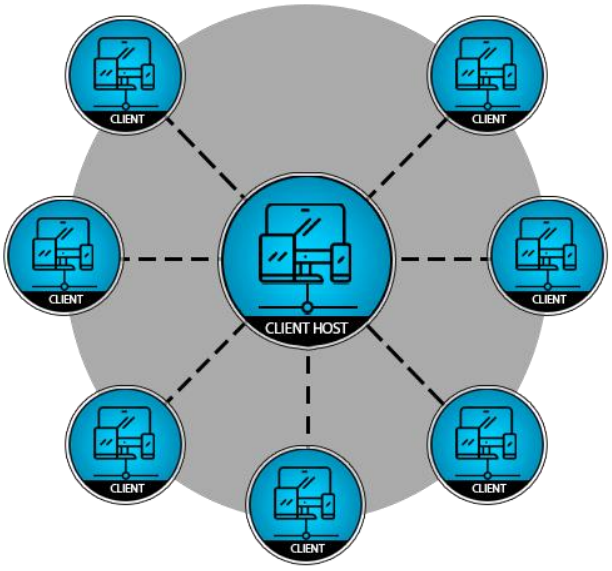

Рисунок 1.2 – Модель Host-based мережі [3]

Переваги: простота реалізації; низьке навантаження на мережу; може бути реалізована протидя затримкам, передача хоста (ведучого) іншому комп'ютеру та вхід нового гравця в уже розпочату гру; підходить для реалізації кооперативних ігор.

Недоліки: все ще великі ризики при зламі гри; високе навантаження на ведені машини; передача даних для ведених має високі затримки.

Застосування: те ж саме, що і в однорангових мережах.

Server-based (або клієнт-сервер). Один комп'ютер виступає в ролі сервера і містить повну та достовірну інформацію про ігровий світ. Решті комп'ютерів, які виступають у ролі клієнтів, передається лише необхідна частина інформації, яка дозволяє їм активно брати участь у грі та відтворювати ігровий світ з відповідною точністю [2]. Схема моделі Server-based мережі зображена на рисунку 1.3.

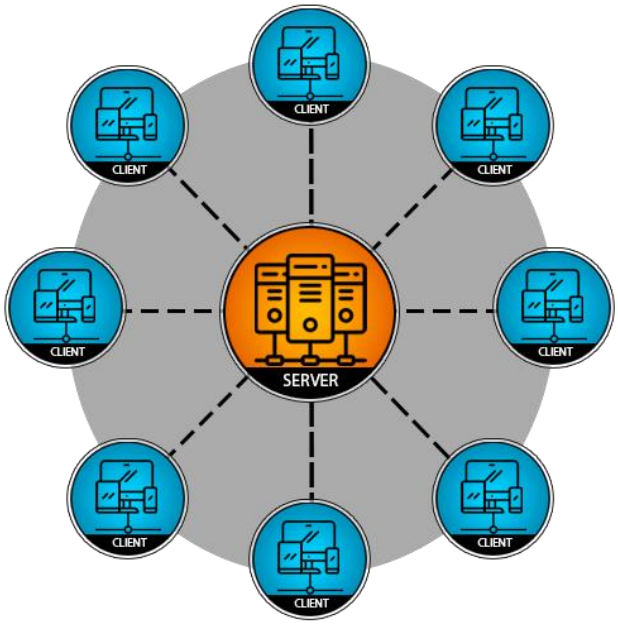

Рисунок 1.3 – Модель Server-based мережі [3]

Переваги: ключові ігрові обчислення майже не навантажують машини клієнтів; можна використовувати виділений сервер, тобто окрему машину, зазвичай на хостингу; найвища стійкість зламу, особливо у комбінації з підходом авторитетності сервера; можливість незалежної підтримки та оновлення серверної та клієнтської частин.

Недоліки: необхідність організовувати хостинг для сервера; насиченість ігрового світу динамічними об’єктами спричиняє велике навантаження на мережу; неможлива міграція сервера на іншу машину; висока затримка.

Застосування: переважна більшість сучасних онлайн-ігор, майже всі змагальні онлайн-ігри.

Багатосерверна модель. Сучасні локальні мережі мало схожі на мережі часів Doom — сучасні мережі мають більше колізій та, відповідно, вищу затримку. Багатосерверна модель була розроблена з метою зменшити проблеми, пов'язані з затримками, що властиві звичайній клієнт-серверній моделі. У цьому режимі кожен комп'ютер виступає як клієнт і сервер одночасно. Наприклад, коли гравець Б з'являється на екрані гравця А, гравець А може отримати дані безпосередньо від комп'ютера Б, отримуючи оновлену інформацію про його положення, обходячи центральний сервер.[2]

Переваги: мінімальні затримки на достатньо потужних машинах.

Недоліки: надскладна реалізація; низька стійкість до зламу.

Застосування: орієнтована на ігри, де долі секунди мають вирішальне значення, застосовується рідко. Вона знаходить своє використання в таких іграх, як Splinter Cell та Soldier of Fortune.

Станом на сьогодні, стандартами індустрії розробки онлайн-ігор є Host-based модель (зіркоподібний зв'язок) та Server-based модель (клієнт-сервер) [3].

1.2 Server-Authoritative архітектура

Авторитетний сервер (англ. – Authoritative server) – це підхід, за якого клієнт надсилає інструкції або інформацію на сервер. Сервер перевіряє, підтверджує дані та відповідно оновлює проксі та/або клієнта [4].

Наприклад, клієнт намагається рухатися 1000 м/с; однак сервер знає, що максимальна швидкість для цього клієнта становить лише 10 м/с. Хоча на мить може здатися, що клієнт пересувається дуже далеко, сервер змушує його повернутися в правильне положення. Сервер володіє над клієнтами. Навіть якщо гравець використовує зламаний клієнт, щоб не дозволити серверу повернути його назад у правильну позицію, його місцезнаходження на сервері, отже і у всіх інших клієнтів, залишатиметься коректним. Зламаний клієнт просто буде розсинхронізований із сервером і таким чином не буде досягнуто переваги над іншими гравцями. Авторитетний сервер не виникає природним шляхом, фактично це лише сукупність підходів до проєктування системи.

Ця модель добре підходить для змагальних ігор, де клієнт грає проти інших гравців, вона може не знадобитися для ігор з кооперативним сюжетним режимом або деяких інших.

Переваги**:**

– шахрайство на стороні клієнта набагато складніше (майже неможливе у більшості сценаріїв);

– чітко зрозуміло, хто має повноваження, і що існує лише один «стан» ігрового світу, який використовується для всіх клієнтів.

Недоліки**:**

– для уникнення затримки вводу, в іграх із швидким ігровим процесом може знадобитися передбачення на стороні клієнта (англ. – client-side prediction);

– серверу потрібно буде виконати додаткові обчислення (йому потрібно буде обробити всі дані, щоб отримати вихідні дані та підтвердити їх клієнту); це може зробити серверне обладнання дорожчим [4].

Використовуючи авторитетний сервер, можна запобігти найбільш кричущим зламам, таким як польоти гравців, проходження крізь стіни, невразливість тощо, які псують досвід для всіх гравців. Але, унеможливити всі види ігрового шахрайства не є реальною задачею. Навіть в онлайн-іграх із авторитетним сервером можуть існувати скрипти-макроси, що автоматизують складні послідовності вводу та чит-програми, що автоматизують частину ігрового процесу використовуючи дані, наявні на клієнтській стороні. Хоча сервер зазвичай не надсилає такі дані, що клієнти не бачать, існують усілякі сповіщення та інформація, які можуть бути зчитані, котрі зазвичай неочевидні.

Якщо у грі немає змагального багатокористувацького режиму гри, сервер може бути неавторитетним. Кожен клієнт створює екземпляр свого власного гравця, копіює значення, які мають бачити інші гравці, як-от механізми, здоров’я, використання зброї/здібностей тощо. В такому випадку не обов’язково піклуватися про перевірку колізій та обчислення фізики, команди чи призначення контролю.

1.3 Аналіз актуальних онлайн-ігор

Далі оглянуті деякі актуальні онлайн-ігри побудовані на різних архітектурах та різних жанрів. Різноманітність наведених прикладів дозволить отримати всебічну картину про проблеми сучасних онлайн-ігор.

1.3.1 Minecraft

Minecraft - це гра-пісочниця, що розроблена Mojang Studios. Її творцем є Маркус "Нотч" Перссон (англ. Markus "Notch" Persson), який використав мову програмування Java для створення цієї гри. Перші версії Minecraft були доступні для приватного тестування у травні 2009 року, а повна версія гри була випущена в листопаді 2011 року [5].

У Minecraft гравці мають можливість досліджувати блоковий тривимірний світ, який генерується процедурно і майже безкінечний. Вони можуть знаходити ресурси, добувати їх, створювати інструменти та предмети, а також будувати різноманітні споруди, роботи з землею та механізми. Гра надає гравцям різні режими, включаючи режим виживання, де гравці повинні здобувати ресурси, будувати світ і зберігати своє здоров'я, а також творчий режим, де гравці мають необмежені ресурси і можуть літати [5].

Граючи в Minecraft, можна грати в одиночну або багатокористувацьку гру, якщо є бажання грати з іншими людьми. Не дивлячись на те, що в сучасних версіях гри є можливість під’єднуватись до серверів і грати з великою кількістю людей, основним способом гри невеликою компанією людей залишається гра по локальній мережі. Гравці можуть підключатися до однієї ігрової сесії з використанням одного мережевого підключення. Це дає можливість гравцям грати разом на тому ж самому світі, обмінюючись ресурсами, будуючи спільні будівлі та борючись зі спільними ворогами.

Для того, щоб грати в Minecraft по локальній мережі, потрібно налаштувати локальну мережу між комп'ютерами, які будуть грати, і забезпечити, щоб всі комп'ютери підключалися до одного сервера. Зазвичай це виконується за допомогою програмного забезпечення, такого як Hamachi або Tunngle.

У такому випадку, один гравець завжди є хостом ігрової сесії, тобто він створює світ, робить налаштування, а під час гри саме його ігровий клієнт обробляє та синхронізує всі дії інших гравців. Таким чином, хост-гравець має повний контроль на ігровим процесом, при бажанні може отримати перевагу над іншими

гравцями нечесним способом, а також стабільність ігрової сесії повність залежить від стану хост-гравця. Описані проблеми притаманні всім іграм Host-based моделі.

1.3.2 EVE Online

EVE Online – масова багатокористувацька рольова (MMORPG) [7] онлайн-гра з науково-фантастичним сюжетом, дія якої розгортається в космосі. Гра розроблена ісландською компанією CCP Games та оприлюднена у 2003 році [6].

EVE Online часто називають найбільш масовою MMO через те, що всі її підписники з усіх країн світу, крім гравців з Китаю, використовують для гри один і той самий ігровий сервер. У Китаї був запущений окремий сервер із-за особливостей законодавства. Аудиторія ігри становить понад 330 000 активних гравців, при цьому одночасно в грі в різний час доби знаходиться, як правило, від 15 до 50 тисяч гравців (загальний пік активності склав 65 303 гравця одночасно за даними на 5 травня 2013 року) [6].

Гра реалізована на Server-based моделі із авторитетним сервером та пропонує гравцям під’єднуватись до єдиного, офіційного серверу. Використання єдиного ігрового сервера і, як наслідок, спільної ігрової сесії всіма гравцями відкриває унікальні ігрові ситуації та надає майже необмежені можливості великим групам гравців, що і є візитною карткою цієї гри. Однак, це є і основним джерелом проблем.

Задля оптимізації роботи сервера такт синхронізації ігрового всесвіту між клієнтом та сервером становить 1 на секунду. Для порівняння, в онлайн іграх з активним ігровим процесом це значення зазвичай в 20-30 разів більше. Такий такт синхронізації сильно впливає на можливості гравців, обмежує мінімальну швидкість ігрових дій до 1 секунди, хоча і сприяє економії інтернет-трафіку. Також, не дивлячись на чітку сегментованість ігрового всесвіту, при раптовому скупченні великої кількості гравців в одному місці, сервер штучно сповільнює темп ігрового процесу для уникнення аварійного вимикання і це впливає на інших гравців, що не приймають участі у скупченні.

1.3.3 Clash Royale

Clash Royale – це багатокористувацька онлайн-гра у жанрі Real-Time Strategy 1 на 1 [8].

У Clash Royale два гравці змагаються один з одним, намагаючись знищити базу супротивника та захистити свою від ворожих атак до закінчення відведеного часу. Гральне поле, відоме як арена, поділене навпіл, з нижньої частини якої починає гравець з трьома баштами і двома шляхами, а верхня частина містить аналогічні башти та шляхи противника [8].

Кожен гравець має свою колоду карт, із навмання розкриваються чотири карти. Розміщуючи карти на полі бою, гравець призиває воїнів, будує споруди або накладає закляття. Воїни з'являються в певному місці і самостійно рухаються по шляхах в напрямку ворожих сил, атакуючи найближчих ворогів. Деякі війська прямують безпосередньо до ворожих башт, ігноруючи інших ворогів.

Використання карт обмежене наявним "еліксиром", який поступово поповнюється з часом. Кожна карта також має час "перезарядки", після використання якої вона повинна відновитись перед наступним використанням. Максимальний запас еліксиру для гравця становить 10 одиниць.

Зважаючи на змагальну компоненту гри, вона має Server-based модель із авторитетним сервером, де і сервер, і клієнт симулюють гру синхронно. Сервер відповідає за перевірку дій гравців і відповідає за визначення реального стану гри у разі десинхронізації [8].

Загалом, ця гра реалізована за сучасними стандартами онлайн-ігор, вона достатньо оптимізована і її архітектура не допускає прямого втручання в перебіг ігрового процесу нечесними гравцями. Але, вона все одно має лазівки для отримання незаконної переваги у грі, оскільки деякі дані гравця-опонента фігурують на клієнтській стороні. Використовуючи модифікований ігровий клієнт або стороннє програмне забезпечення нечесний гравець може побачити карти

опонента та його накопичену кількість «еліксиру». Фактично, це надає таку саму перевагу, як можливість заглянути в карти опонентів у класичній грі в карти [8].

1.3.4 GTA Online

Grand Theft Auto Online (скорочено GTA Online) – це багатокористувацька пригодницька екшн-гра, розроблена Rockstar North та видана Rockstar Games. Вона вийшла 1 жовтня 2013 року для PlayStation 3 та Xbox 360, 18 листопада 2014 року для PlayStation 4 та Xbox One та 14 квітня 2015 року для Microsoft Windows [9].

Гра пропонує гравцям відкритий світ, де вони можуть вільно рухатись по штату, який включає в себе відкриту сільську місцевість і вигадане місто. Гравці контролюють головного героя, який має за мету стати відомим злочинцем, будуючи свій власний злочинний синдикат і виконуючи все більш складніші завдання.

Окрім одиночної кампанії, гра також пропонує Grand Theft Auto Online – багатокористувацький режим. В цьому режимі гравці можуть спільно виконувати місії для розкриття історії.

Grand Theft Auto Online розроблено разом з режимом для одного гравця, але його було задумано як самостійний та постійно оновлюваний досвід. У період запуску гра стикалася зі значними технічними проблемами, які призвели до неможливості виконання місій та втрати даних персонажів. Перші рецензії були суперечливими, але визнавали масштабність та відкритий геймплей.

У відритому доступі майже немає даних про технічні деталі реалізації GTA Online. Однак, ця гра вочевидь має Server-based модель, оскільки ігрові сесії запущені весь час, а гравці лише під’єднуються до них. Але, вже в перші тижні після виходу гри для персональних комп’ютерів, нечесні гравці за допомогою стороннього програмного забезпечення могли повноцінно втручатися в ігровий процес як свій, так і інших гравців. Зокрема, вони могли отримати нескінченну кількість здоров’я, набоїв, могли переміщувати звичайних гравців і т.д. Всі

перелічені дії зазвичай повинні виконуватись на сервері, отже ні за яких махінацій на клієнтській стороні не повинні бути досяжними.

Розробники доволі швидко оновили гру, заблокувавши всі лазівки, але факт того, що подібні втручання взагалі були можливі, свідчить про не повну авторитетність серверу в ігровій логіці.

Висновки до розділу

В цьому розділі було розглянуто поняття онлайн-гри, коротко оглянуто історію виникнення перших багатокористувацьких ігор, досліджено актуальні принципи організації мережі між гравцями, а саме: Peer-to-peer, Host-based, Server-based та багатосерверну модель. Кожна модель детально проаналізована, досліджено їх технологічні особливості, виокремлено основні переваги і недоліки, як з точки зору розробника, так і з точки зору гравця. Проведено паралелі між кожною з моделей та актуальними видами (жанрами) онлайн-ігор, де доречне їх використання.

Досліджено архітектурний підхід авторитетності серверу над клієнтами. Визначено доречність його використання в залежності від характеру та жанру гри.

Далі було проведено аналіз декількох популярних та актуальних онлайн-ігор принципово різних жанрів. Представлені проєкти були як на Host-based так і на Server-based моделях, за характером організації ігрового процесу як сесійні, так і з відкритим ігровим світом. Розглянувши ці приклади, було підтверджено наявність переваг і недоліків, властивих тій чи іншій моделі організації мережі між гравцями.

Також було проаналізовано проблеми розглянутих проєктів та лазівок щодо можливого ігрового шахрайства при тій чи іншій архітектурі програми. Доведено актуальність дослідження. На зроблених в цьому розділі висновках і базується онлайн-гра, розроблена в даному дипломному проєкті.

2 ПРОЄКТУВАННЯ ГРИ

У даній роботі розроблено онлайн-гру, що базуватиметься на Server-Authoritative архітектурі для підвищення якості ігрового процесу та зменшення вразливості щодо можливості ігрового шахрайства.

У попередньому розділі оглянуто проблеми сучасних проєктів схожого типу, що доводить актуальність цього дослідження. А саме виділено основні проблеми:

– погіршення якості ігрового процесу за великої кількості гравців в одній ігровій сесії або на одному сервері:

– вразливість гри до прямого втручання в ігровий процес клієнтами за допомогою стороннього ПЗ;

– наявність на клієнтській стороні даних, що не повинні бути видимі, та доступ до яких може надати перевагу;

– проблема оптимізації трафіку.

Головною ціллю розробки власної онлайн-гри є вирішення наведених вище проблем. Для цього сформульовано наступні критерії:

– кожна ігрова сесія повинна обслуговуватися окремим сервером і повинна містити невелику кількість гравців;

– архітектура програми повинна прагнути до повної авторитетності серверу щодо критичних для ігрового процесу розрахунків;

– всі важливі для обчислення ігрового процесу дані повинні зберігатись на сервері, а клієнтам передаватися лише в тому обсязі, що призначений для відображення;

– усі неважливі для ігрового процесу обчислення не повинні синхронізуватись із сервером для економії трафіку.

2.1 Дизайн гри

Розроблювана гра буде екшн-грою сесійного виду, формату один проти одного. Сесійна гра – це така, в якій гравці зустрічаються в короткочасних сесіях,

які, зазвичай, автоматично створюються для кожного матчу. Ігри такого виду ще називають MOBA (Multiplayer Online Battle Arena, англ. – Багатокористувацька онлайн битва в арені)[10]. Такий формат на сьогодні є найбільш актуальним та поширеним, оскільки дозволяє гравцю самостійно визначати кількість часу, проведеного в грі. Під час розробки гра орієнтуватиметься на платформу ПК зі збереженням можливості подальшої міграції на мобільні пристрої.

Кожен з двох гравців керуватиме своєю групою юнітів (бойова одиниця, англ. unit), почергово переключаючись між ними. Кожен юніт матиме певні початкові характеристики, такі як швидкість руху, запас одиниць здоров'я, енергії тощо. Також на всі юніти будуть встановлюватися модулі, такі як озброєння (віднімає одиниці здоров'я юнітам опонента), модулі ремонту (відновлює одиниці здоров'я іншим юнітам) тощо; та модулі, що модифікуватимуть характеристики одного або декількох юнітів. Модулі поділятимуться на активні і пасивні. Пасивні надаватимуть свої ефекти весь час, в той час як активні потрібно буде активувати задля їх дії, а їх активація потребуватиме витрат енергії та матиме циклічний характер. Енергія відновлюватиметься з часом.

Кожна ігрова сесія (матч) буде обмежена за часом та проходитиме в обмеженій ділянці ігрової сцени (арені). Після вичерпання часу або за межами арени юнітам поступово будуть змінюватись деякі характеристики, що прискорить перемогу одного з гравців та закінчення матчу.

Ціллю та завершенням ігрового матчу буде знищення всіх юнітів опонента. Задачею кожного гравця буде позиціонування своїх юнітів на арені відносно опонента, контроль їх модулів, нанесення якомога більшого ураження юнітам опонента та збереження своїх юнітів шляхом їх своєчасного ремонту. Гравець, що перший знищив всі юніти опонента, вважатиметься переможцем матчу.

Склад групи юнітів, їх початкові характеристики та перелік модулів кожного будуть налаштовуватися гравцем перед початком пошуку матчу. Така система забезпечить складову стратегічного мислення для гравців, вони зможуть застосовувати різні підходи та креатив у формуванні своєї групи юнітів та тактиці гри безпосередньо в матчі.

Застосунок складатиметься з наступних сцен:

– головне меню: надає можливість користувачу переходити до інших сцен, натискаючи відповідну кнопку;

– ігрова сцена: сцена, синхронізована із ігровим сервером і на якій безпосередньо відбувається матч;

– меню налаштування групи юнітів (оснащення);

– меню статистики гравця.

2.2 Ігровий рушій Unity

Unity, що означає "єдність" у перекладі з англійської та вимовляється як "юніті", є кросплатформовим середовищем розробки комп'ютерних ігор, розробленим американською компанією Unity Technologies. Це програмне забезпечення дозволяє створювати додатки, які працюють на понад 25 різних платформах, включаючи персональні комп'ютери, ігрові консолі, мобільні пристрої, веб-додатки та інші [11].

Unity Engine був випущений в 2005 році і з того часу продовжує активно розвиватися. Ця платформа використовується для створення тисяч ігор, програм і візуалізацій математичних моделей, охоплюючи безліч платформ і жанрів. Unity є популярним інструментом як для великих розробників, так і для незалежних студій [11].

Зазвичай, ігровий рушій надає розмаїття функцій, які можуть бути використані в різних іграх, включаючи моделювання фізичних середовищ, картування нормалей, динамічні тіні та багато іншого.

Редактор Unity має простий інтерфейс Drag&Drop, складається з різних вікон, що дозволяє проводити налаштування гри безпосередньо в редакторі. Для написання скриптів Unity використовує мову програмування C#. Раніше також підтримувалися Boo (діалект Python, підтримку припинено з 5-ої версії) та модифікація JavaScript, відома як UnityScript (підтримка припинена з версії 2017.1). Розрахунки фізики здійснюються за допомогою фізичного рушія PhysX від

NVIDIA для 3D фізики та Box2D для 2D фізики. Графічний API, яким користується Unity, - DirectX (наразі DX 11, а також підтримується DX 12).

У проекті Unity існує поняття сцен (рівнів), які представляють собою окремі файли і містять власні геймплейні світи з набором об'єктів, скриптів і налаштувань. Сцени можуть включати як об'єкти (моделі), так і порожні геймплейні об'єкти, які не мають графічного представлення (так звані "пустушки"). Об'єкти, у свою чергу, складаються з набору компонентів, з якими взаємодіють скрипти. Кожен об'єкт має свою назву (в Unity допускається наявність декількох об'єктів з однаковою назвою в одній сцені), може мати тег (мітку) і шар, на якому він повинен бути відображений. Крім того, кожен об'єкт на сцені обов'язково має компонент Transform, який зберігає координати розташування, обертання і розмірів об'єкта в тривимірному просторі [12].

У редакторі є система наслідування об'єктів; дочірні об'єкти будуть повторювати всі зміни позиції, повороту та масштабу батьківського об'єкта [12].

Компонентом в Unity називається клас (як окремий файл), який наслідується від одного із спеціальних вбудованих класів Unity, що визначає «поведінку» ігрового об’єкта, і є прикріплений до нього у сцені [12].

Unity Engine використовує концепцію циклу оновлення (update loop) для виконання ігрових обчислень. Основним тактом циклу оновлення є функція Update(), яка викликається кожен кадр гри. У цій функції розміщується код, який потрібно виконати кожен кадр для обробки введення користувача, оновлення стану гри та взаємодії об'єктів [12].

У Unity 3D підтримується система рівнів деталізації (LOD - Level Of Detail), яка дозволяє замінювати високодеталізовані моделі на менш деталізовані, коли вони знаходяться на великій відстані від гравця, і навпаки. Це дозволяє знизити навантаження на графічний процесор і оптимізувати продуктивність гри.

Також Unity 3D підтримує систему Occlusion Culling, яка дозволяє не відображати геометрію та колізії об'єктів, які знаходяться поза полем зору камери. Це допомагає знизити навантаження на центральний процесор та покращує

продуктивність гри, оскільки зменшується кількість обчислень, пов'язаних з візуалізацією та обробкою фізики об'єктів, які гравець не бачить.

При компіляції проекту в Unity 3D створюється виконуваний файл гри (наприклад, .exe для Windows), який містить всі необхідні дані гри. Також створюється окрема папка, яка містить всі ігрові рівні та бібліотеки, що динамічно підключаються до гри. Це дозволяє зручно розповсюджувати готову гру та забезпечує портативність проекту.

Unity має дві ключові переваги порівняно з іншими ігровими рушіями. Перша перевага - це наявність візуального середовища розробки, що включає інструменти для візуального моделювання. Воно також містить інтегроване середовище і ланцюжок збірки, що спрямовані на підвищення продуктивності розробників, особливо на етапах прототипування та тестування.

Друга перевага полягає у міжплатформовій підтримці. Unity надає можливість розгортання гри на різних платформах, таких як персональні комп'ютери, мобільні пристрої, ігрові консолі та інші. Крім того, середовище розробки Unity доступне для використання на операційних системах Windows і Mac OS.

Третя перевага - це модульна система компонентів Unity, яка дозволяє конструювати ігрові об'єкти шляхом комбінування функціональних елементів. Замість успадкування об'єктів відбувається об'єднання функціональних блоків, що спрощує створення прототипів ігор. Цей підхід дозволяє зручно комбінувати функціональність та прискорює розробку проектів.

Серед недоліків Unity згадуються деякі обмеження візуального редактора при роботі з складними багатокомпонентними сценами. У випадку, коли сцена стає дуже складною, візуальна робота може стати ускладненою, і це може вимагати додаткового зусилля для навігації та редагування.

Ще одним недоліком є відсутність вбудованої підтримки посилань на зовнішні бібліотеки в Unity. Робота з такими бібліотеками вимагає налаштувань, які програмістам доводиться виконувати самостійно. Це може призвести до

ускладнення командної роботи, особливо якщо різні члени команди використовують різні бібліотеки або версії.

Ще одним недоліком є складність редагування шаблонів екземплярів (prefabs) в Unity. Хоча концепція шаблонів екземплярів надає гнучкість для візуального редагування об'єктів, сам процес редагування шаблонів може бути складним, особливо якщо шаблони мають складну структуру або використовуються в багатьох місцях.

2.3 Вибір мережевої бібліотеки

Для Unity існує кілька мережевих бібліотек для створення мультиплеєрних ігор: Netcode for GameObjects, Mirror та Photon. Кожна з цих бібліотек має свої відмінності.

Netcode for GameObjects: Netcode for GameObjects є новою бібліотекою для мережевої гри в Unity, випущеною в 2020 році. Вона створена для забезпечення легкої розробки мережевих ігор в Unity з використанням Unity DOTS (Data-Oriented Technology Stack). Основними перевагами є висока продуктивність, автоматична синхронізація мережевих об'єктів та можливість використання серверного авторитету. Однак, ця бібліотека є дуже новою і не має такої популярності, як інші бібліотеки.

Mirror: Mirror є однією з найбільш популярних мережевих бібліотек для Unity, яка була розроблена на основі бібліотеки UNET, яка була вилучена з Unity. Mirror пропонує простий API, який дозволяє легко розробляти мультиплеєрні ігри, а також має підтримку серверного авторитету та інші корисні функції, такі як динамічні регістри мережевих об'єктів. Однак, Mirror може вимагати більшої кількості кодування для використання деяких функцій.

Photon: Photon є ще однією з популярних мережевих бібліотек для Unity, яка має широкий спектр функцій для мережевої гри. Вона пропонує підтримку серверного авторитету, мультиплатформності, високої продуктивності та

можливості розширення. Photon також має різні плани цін, що можуть впливати на доступність для розробників з обмеженим бюджетом.

У кожної мережевої бібліотеки є свої переваги та недоліки, і вибір конкретної бібліотеки залежить від потреб проєкту. Наприклад, якщо потрібна максимальна продуктивність, можна обрати Netcode for GameObjects, якщо потрібно легко розробляти мобільні мультиплеєрні ігри, Mirror може бути кращим варіантом, а якщо потрібна широка функціональність та можливості розширення, то Photon може бути кращим варіантом.

Окрім того, бібліотеки можуть відрізнятися в плані підтримки платформ, витривалості та безпеки. Наприклад, Photon за замовчуванням має підтримку платформ Android та iOS, а Mirror має вбудований механізм підтримки IPv6, що забезпечує витривалість мережевого з'єднання.

Враховуючи потреби даного проєкту, основною і єдиною мережевою бібліотекою для використання була обрана Netcode for GameObjects. Головним фактором, що зумовлює цей вибір, є те, що ця бібліотека є розробкою команди Unity Technologies, що і забезпечує її високу продуктивність. Бібліотека Netcode створена на заміну застарілій бібліотеці UNet, яка була видалена із Unity у 2017.

Netcode працює на версіях Unity починаючи з 2020.3 і підтримує такі платформи:

– Windows, macOS і Linux;

– iOS і Android;

– платформи XR (Extended Reality), що працюють на операційних системах Windows, Android та iOS

– більшість закритих платформ, таких як консолі [3].

Netcode за замовчуванням використовує транспортний рівень Unity (Unity Transport). Це дозволяє розробнику вести розробку на високому рівні не концентруючись на задачах встановлення зв’язку, доставки пакетів даних тощо [3].

Також важливо зазначити, що Netcode є частиною великого інструментарію Multiplayer в Unity Game Services. Окрім рішень Netcode для безпосереднього створення програмного коду онлайн-гри, він також включає сервіс для спрощення

налагодження зв’язку між гравцями при використанні Host-based архітектури – Relay; сервіс хостингу виділених ігрових серверів –Multiplay (для проєктів Server-based архітектури); сервіс Matchmaker, що автоматизує формування, організацію та пошук ігрових сесій гравцями та багато інших.

2.4 Мова програмування C#

C# (читається як "Сі шарп") – це об'єктно-орієнтована мова програмування, створена компанією Microsoft, яка широко використовується для розробки програмного забезпечення для платформи Windows та додатків для середовища .NET [13].

Основні особливості C#:

– об'єктно-орієнтована парадигма: C# базується на об'єктно-орієнтованій парадигмі програмування, що дозволяє зберігати дані та функції, які їх оброблюють, в одному об'єкті;

– підтримка платформи .NET: C# розроблялась з урахуванням платформи .NET, що забезпечує зручний доступ до стандартних бібліотек та API для розробки програмного забезпечення під Windows;

– статична типізація: C# підтримує статичну типізацію, тобто тип змінної визначається на етапі компіляції, що дозволяє виявити більшу кількість помилок на етапі розробки;

– можливості асинхронного програмування: C# підтримує асинхронне програмування, що дозволяє виконувати декілька операцій одночасно та зменшує час відгуку програми;

– розширення методів: C# підтримує розширення методів, що дозволяє розширювати функціональні можливості класів без модифікації їх коду [13].

C# є потужною мовою програмування, яка широко використовується в розробці програмного забезпечення та додатків для платформи Windows, включаючи мобільні та веб-додатки. Її основні переваги – це зручна розробка за допомогою стандартних бібліотек та API, підтримка асинхронного програму вання,

статична типізація та об'єктно-орієнтована парадигма. Крім того, C# є однією з основних мов програмування для розробки ігор в Unity, що дозволяє розробникам зручно створювати ігрові проекти та забезпечувати високу продуктивність та ефективність гри.

Синтаксис C# схожий на синтаксис інших мов програмування, таких як Java та C++, що дозволяє розробникам швидко навчитися програмуванню на C#. Крім того, C# підтримує різноманітні конструкції, такі як об'єктно-орієнтоване програмування, делегати, інтерфейси та збірку сміття, що дозволяє писати більш зручний та структурований код.

Однією з переваг C# є також те, що вона підтримує різні типи додатків, включаючи консольні, десктопні, мобільні та веб-додатки. Крім того, C# є платформонезалежною мовою програмування, що дозволяє розробляти додатки для різних платформ, таких як Windows, Android, iOS та Linux [13].

У загальному, C# є потужною та гнучкою мовою програмування, яка дозволяє розробникам створювати різноманітні програмні додатки, включаючи ігри, веб-додатки та мобільні додатки. Її основні переваги - це підтримка платформи .NET, статична типізація, об'єктно-орієнтована парадигма та можливості асинхронного програмування.

2.5 Середовище розробки VSCode

Visual Studio Code (VSCode) - це безкоштовний, відкритий і легкий редактор коду, розроблений компанією Microsoft. Він підтримує багато мов програмування, включаючи C++, C#, Java, JavaScript, Python та багато інших. VSCode є популярним вибором серед розробників з різних областей, включаючи веб-розробку, розробку мобільних додатків та розробку ігор [14].

Основні переваги VSCode полягають в його легкості, швидкості та розширюваності. Він має інтуїтивний та зручний інтерфейс, що дозволяє розробникам легко редагувати та відлагоджувати свій код. Крім того, VSCode

підтримує автодоповнення коду, а також можливість встановлення різноманітних розширень та плагінів, що значно полегшує роботу розробників.

За допомогою деяких розширень, таких як Unity Debugger, VSCode може гибоко інтегруватися з Unity, надаючи засоби для полягоження ігрового коду в середовищі розробки. Це полегшує відстеження помилок і забезпечує зручність під час розробки.

VSCode також має вбудовані функції для контролю версій, такі як Git, що дозволяють розробникам вести свій код відповідно до найкращих практик розробки програмного забезпечення. Крім того, VSCode підтримує роботу з віддаленими серверами, що дозволяє розробникам працювати на віддаленому сервері, не збільшуючи при цьому обсяг використаної пам'яті на локальному комп'ютері [14].

Загалом, VSCode є потужним та зручним редактором коду, який підтримує багато мов програмування та має безліч корисних функцій. У поєднанні із розширеннями для Unity він є оптимальним вибором для даного проєкту.

2.6 Система контролю версій Git

Системи контролю версій (СКВ) є інструментами, що дозволяють розробникам відстежувати зміни в файловій системі проекту, зберігати і керувати різними версіями файлів і співпрацювати з іншими розробниками. Однією з найпопулярніших систем контролю версій є Git.

Git - це розподілена система керування версіями, розроблена Лінусом Торвальдсом. Вона стала де-факто стандартом у світі розробки програмного забезпечення завдяки своїй потужності, швидкості та гнучкості. Git зберігає повну копію репозиторію на кожному робочому станції розробника, що дозволяє незалежно працювати над проектом навіть без з'єднання з центральним сервером.

Git надає розробникам багато корисних функцій, таких як створення гілок (branches) для розробки функціональності відокремлено від основної гілки, злиття

(merge) гілок для об'єднання змін, відстежування історії змін (commit history), відміну (revert) змін до попереднього стану та багато іншого.

Git працює з репозиторіями, які містять всю історію і файли проекту. Розробники можуть працювати з центральним репозиторієм або створювати власні локальні репозиторії для незалежної роботи. Зміни в файловій системі відстежуються, і розробники можуть зберігати свої зміни у комітах (commits), які створюють нові версії проекту. Коміти можуть бути прив'язані до певних гілок і об'єднуватися з іншими комітами, що дозволяє керувати розвитком проекту.

Git також підтримує спільну роботу над проектом. Розробники можуть спільно працювати над одним і тим же репозиторієм, обмінюючись змінами через віддалені репозиторії. Git надає зручні інструменти для об'єднання змін, вирішення конфліктів і відстежування роботи різних розробників.

Використання Git допомагає зберегти історію змін проекту, полегшує співпрацю розробників, дозволяє експериментувати з новими функціями відокремлено від основного коду та відновлювати попередні версії проекту. Він є потужним інструментом для ефективного керування версіями в розробці програмного забезпечення.

Висновки до розділу

В цьому розділі було підведено підсумок щодо загальних проблем онлайн ігор і на основі цього сформульовано конкретні критерії щодо розроблюваної в цьому проєкті онлайн-гри. З урахуванням цих критеріїв було розроблено дизайн гри. Обрано цільову платформу, під яку розробляється гра, а саме це персональні комп’ютери зі збереженням міграції на мобільні пристрої. Обрано формат та жанр гри, а саме сесійна екшн-гра один проти одного. Також наведено детальний опис ігрового процесу та застосунку в цілому, що включає в себе: перелік правил за якими проходитиме гра; дії, що виконуватимуть гравці; умови перемоги у матчі; а також, перелік ігрових сцен, що міститиме гра.

Далі було досліджено обраний ігровий рушій Unity Engine. Коротко оглянуто історію його створення. Було описано технології, що використовує Unity, а також перелічено основний інструментарій, що дозволить легко створювати гру. Визначено головні переваги і недоліки Unity Engine. Було обґрунтовано вибір даного ігрового рушія для гри, розроблюваної в цьому дипломному проєкті.

Наступним кроком був вибір мережевої бібліотеки для Unity. Було проаналізовано основні існуючі рішення, оцінено їх переваги і недоліки. Враховуючи результати проведеного аналізу, була обрана бібліотека Netcode for GameObjects, оскільки вона є найсучаснішою та була створена самими Unity Technologies, що забезпечить її високу ефективність з цим ігровим рушієм. Бібліотека Netcode for GameObjects допоможе швидко та ефективно розробляти онлайн-гру.

Потім було обрано мову програмування C#, оскільки вона є основною для рушія Unity, а також середовище розробки Visual Studio Code, через її легкість, розширюваність та вбудовану підтримку систем контролю версій.

На останок розглянуто систему контролю версій Git. Перелічено її функції, що допоможуть швидше та ефективніше розроблювати проєкт, надійно зберігати сам проєкт та його зміни в хмарному сховищі. Доведено необхідність використання системи контролю версій для успішної розробки.

3 РЕАЛІЗАЦІЯ ГРИ

В межах даного проєкту реалізовано демонстраційну версію гри, що містить повноцінний ігровий процес та достатню кількість і різноманіття ігрових механік, реалізованих із дотриманням Server-Authoritative архітектури.

Основна увага приділена ігровій сцені, на якій проводиться матч, оскільки саме вона взаємодіє із ігровим сервером і демонструє Server-Authoritative архітектуру. Інші функції застосунку, такі як налаштування і збереження складу групи юнітів та їх модулів, а також збереження і відображення статистики (досягнень) гравця, будуть розроблені, але не будуть реалізовані в повному обсязі. Застосунок функціонуватиме у локальній мережі.

В цій роботі досліджується в першу чергу архітектурний підхід, принципи взаємодії компонентів гри. Деталі реалізації тих чи інших функціональностей здебільшого опущені. Устрій компонентів, що відповідають за візуальні ефекти, інтерфейс користувача і т. н. – не описується.

Посилання на повний програмний код розроблених компонентів розміщено в Додатку А.

3.1 Загальна структура ігрової сцени

За умови авторитетності серверу на стороні клієнта відбувається здебільшого не сама гра, а лише її симуляція. Всі важливі для ігрового процесу обчислення відбуваються на сцені, що відтворюється в серверному екземплярі застосунку. З програмної точки зору, серверна сцена є такою ж самою сценою як і клієнтські, але саме на ній відбуваються «дійсні» процеси, в той час коли клієнтські сцени лише відтворюють результат, отриманий під час синхронізації із сервером [15].

Загальна схема мережевої ігрової сцени зображена на рисунку 3.1.

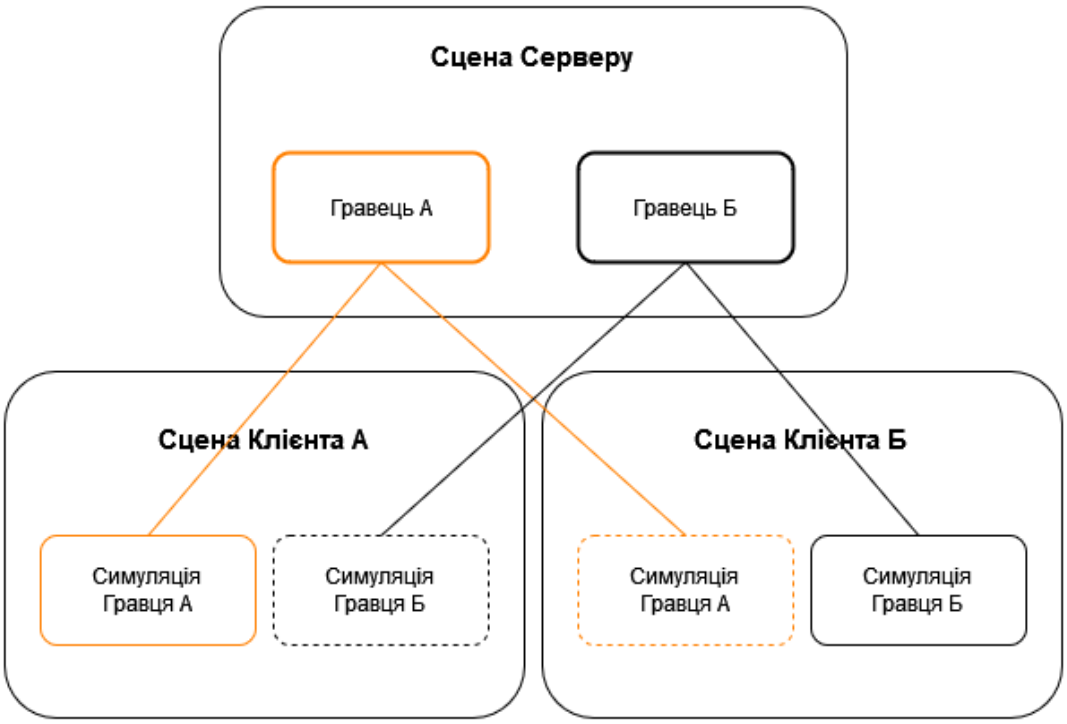

Рисунок 3.1 – Загальна схема мережевої ігрової сцени

Як зазначено на схемі вище, «дійсні» екземпляри гравців існують лише у сцені сервера (жирна рамка), в той час коли клієнти отримують тільки симуляції, що не виконують обчислень, а тільки приймають ввід. Те саме стосується всіх мережевих об’єктів на сцені. В даному контексті, «гравцем» називається ігровий об’єкт, що належить певному клієнту та створюється автоматично із під’єднанням цього клієнта до сервера. Слід зазначити, що Netcode синхронізує лише мережеві об’єкти, тобто такі, у яких наявний компонент NetworkObject. Всі інші об’єкти, такі як статичне середовище сцени, ситуаційні візуальні ефекти, інтерфейс гравця тощо – не синхронізуються та існують автомномно на одному чи кількох клієнтах.

Також компонент NetworkObject, зокрема, зберігає дані про володаря кожного мережевого об’єкта, що дає змогу перевіряти приналежність об’єкту під час обчислень. Це дозволяє виконувати операції над об’єктами, що належать певному клієнту, або забороняти виконання серверних операцій на клієнтських екземплярах об’єктів. За замовчуванням клієнт володіє лише об’єктом свого гравця (на схемі вище – тонка рамка), за виключенням випадків, коли об’єкт створений одразу із вказанням володаря. Володіння об’єктами не забезпечує повноваження на

операції над ними, а лише вказує приналежність цих об’єктів певним клієнтам або серверу.

Головним компонентом мережевої сцени є NetworkManager(Netcode). Він є центральним вузлом мережевої системи і відповідає за всі утилитарні функції: зберігає мережеві налаштування (IP-адреса та порт серверу, транспортний рівень тощо), налаштування синхронізації (щільність тактів синхронізації тощо), контроль існуючих та створення нових мережевих об’єктів і т.д. [3]

3.2 Remote Procedure Call (RPC)

Remote Procedure Call (англ. Дистанційний виклик процедури, далі - RPC) є механізмом, який дозволяє викликати функції або методи на віддаленому комп'ютері або віддаленому процесі через мережу. Він дозволяє розподіленому застосунку взаємодіяти з розподіленими ресурсами, ніби вони знаходяться локально [16].

RPC реалізується шляхом передачі повідомлень між віддаленими системами. Зазвичай використовується модель клієнт-сервер, де клієнт викликає віддалений метод і отримує результат виконання [16].

Основна ідея RPC полягає в тому, що виклик функції на віддаленому комп'ютері або віддаленому процесі виглядає аналогічно виклику локальної функції. Реалізація приховує деталі віддаленого виклику, щоб розробник міг працювати з віддаленими об'єктами так само, як з локальними [16].

В межах Unity Netcode існують такі вбудовані види RPC:

– ServerRPC: Клієнт робить виклик дистанційного виконання процедури, що отримується та виконується сервером на своїй стороні. ServerRPC є основним інструментом для створення ігрової логіки, саме через нього проходять всі важливі для ігрового процесу дії, такі як стрільба, рух, взаємодія із всесвітом тощо [3]. Схема виконання ServerRPC зображена на рисунку 3.2.

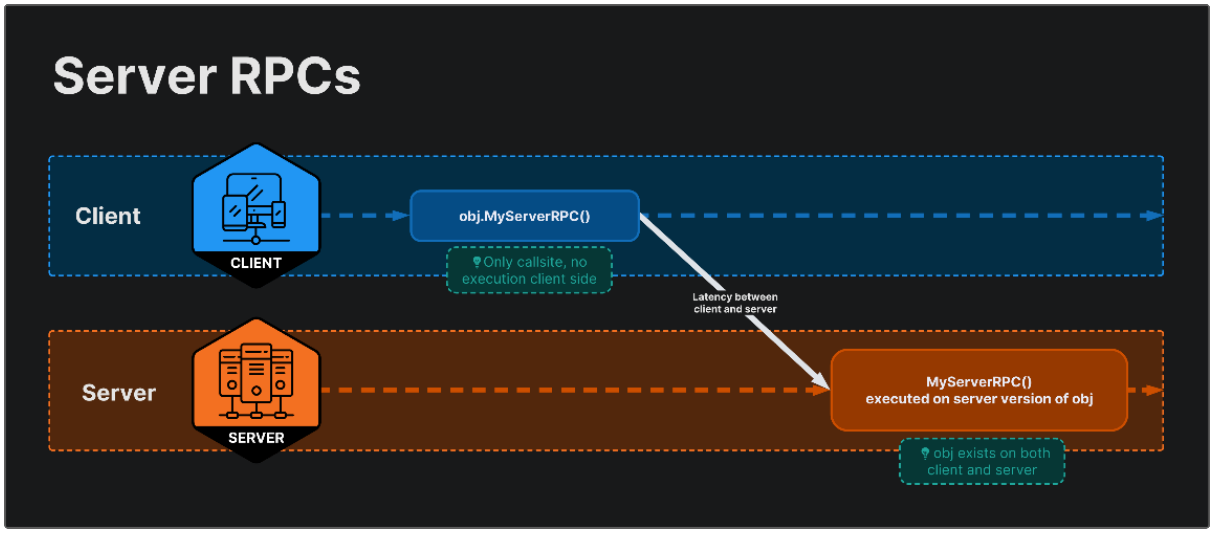

Рисунок 3.2 – Схема виконання ServerRPC [3]

– ClientRPC: Сервер робить виклик дистанційного виконання процедури, що отримується та виконується одним або декількома клієнтами на своїй стороні. ClientRPC дозволяє, наприклад, переносити дані на клієнтську сторону для відображення або відтворювати ефекти, що супроводжують ігрові події (постріли, вибухи тощо). За замовчуванням ClientRPC виконується на всіх клієнтах [3]. Схема виконання ClientRPC зображена на рисунку 3.3.

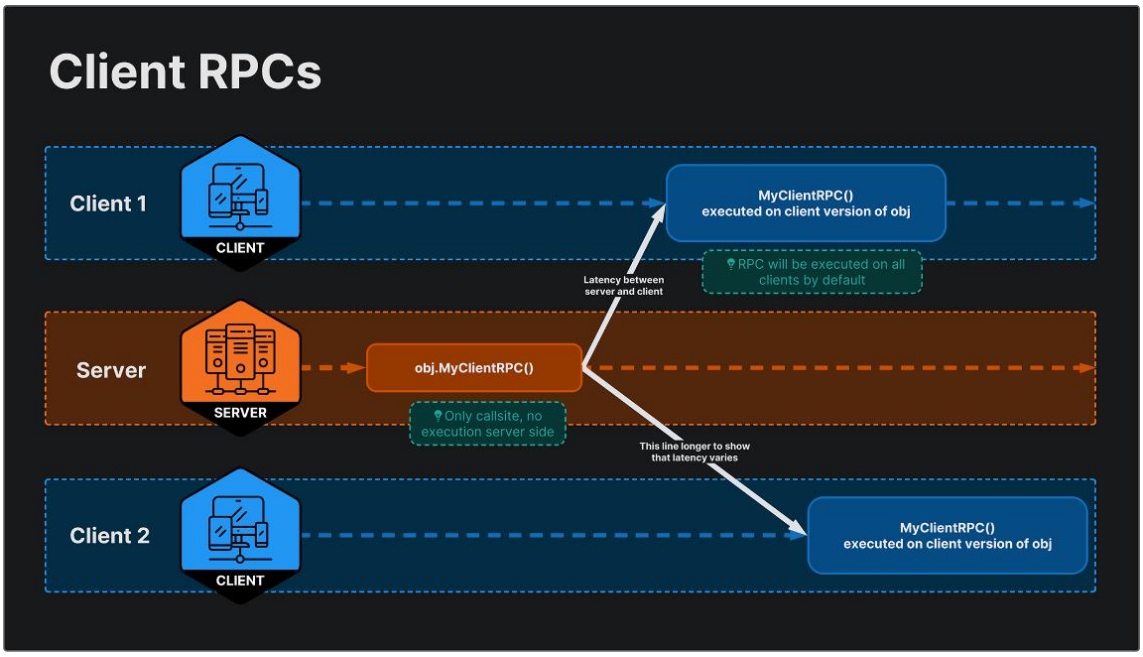

Рисунок 3.3 – Схема виконання ClientRPC [3]

3.3 Реалізація ігрових механік

Ігрова механіка (англ. game mechanics) — це набір правил і способів, який реалізує певним чином деяку частину інтерактивної взаємодії гравця і гри. Вся множина ігрових механік гри формує конкретну реалізацію її ігрового процесу [17].

Кожна реалізована ігрова механіка включає один або декілька компонентів (класів), що взаємодіють. Далі описана робота основних компонентів та їх ключових фрагментів, що забезпечують ігровий процес на сервері та клієнтах. Результуюча структурна схема ігрової сцени наведена в Додатку Б та на кресленику ІА92.240БАК.004 Д7.

В даному проєкті ігровий процес було умовно поділено на механіки, що об’єднані в логічні групи: організація матчу, керування, збереження даних.

Матч – це обмежена по часу ігрова сесія, на час якої клієнти під’єднуються до основної ігрової сцени, на якій відбувається основний мережевий ігровий процес. В одному матчі зустрічаються двоє гравців один навпроти одного, із рівною кількістю юнітів.

Для приєднання до матчу, гравець, знаходячись в сцені головного меню, натискає кнопку, що завантажує ігрову сцену. В даній демонстраційній версії гри, першим кроком клієнта на ігровій сцені є вибір режиму: сервер чи клієнт (відповідна сутність ініціалізується в мережі).

Керування включає в себе низку механік, що дозволяють гравцеві безпосередньо грати, тобто виконувати певні дії у грі. Ці механіки потрібні для вводу, передачі даних вводу на сервер та виконання відповідних ігрових дій на сервері.

Збереження даних передбачає механізми введення і збереження даних в зовнішньому сховищі (базі даних). Ці механізми були розроблені теоретично, але не реалізовані в програмному продукті. Вони є необхідними для онлайн-гри як такої, але не є ключовими в контексті теми даного дипломного проєкту.

3.3.1 Ініціалізація гравців

При встановленні зв’язку нового клієнта з сервером, на сервері створюється об’єкт гравця цього клієнта. Бібліотека Netcode робить це автоматично, якщо такий об’єкт був заздалегідь призначений.

В даному випадку, в грі немає єдиного «персонажа» гравця, тому початковий об’єкт-гравець є «пустишкою», що містить лише стандартний компонент NetworkObject(Netcode) (далі вбудовані компоненти Netcode позначено <компонент>(Netcode), всі інші компоненти реалізовані власноруч), що забезпечує синхронізацію в мережі (далі об’єкти що синхронізуються в мережі називаються скорочено «мережеві об’єкти»), а також компонент PlayerNetController.

PlayerNetController – основний компонент, що представляє гравця. Він функціонує лише на сервері. На початку матчу, він створює та розташовує юнітів, що належать цьому клієнту із налаштованими цим клієнтом показниками і модулями (див. пункт 3.3.2). Також він зберігає дані гравця, наприклад його рахунок, за яким визначається перемога чи поразка. Окрім цього, на стороні свого клієнта він створює окремий об’єкт – локальний об’єкт гравця, що включає в себе: камеру, через яку клієнт отримує зображення сцени, основний користувацький інтерфейс та компоненти керування (див. пункт 3.3.4). Тут і далі комунікація сутностей серверної із сутностями клієнтської та навпаки реалізована за допомогою відповідно ServerRPC та ClientRPC, що вже описані раніше. Схема ініціалізації гравців зображена на рисунку 3.4 та на кресленику ІА92.240БАК.004 Д1.

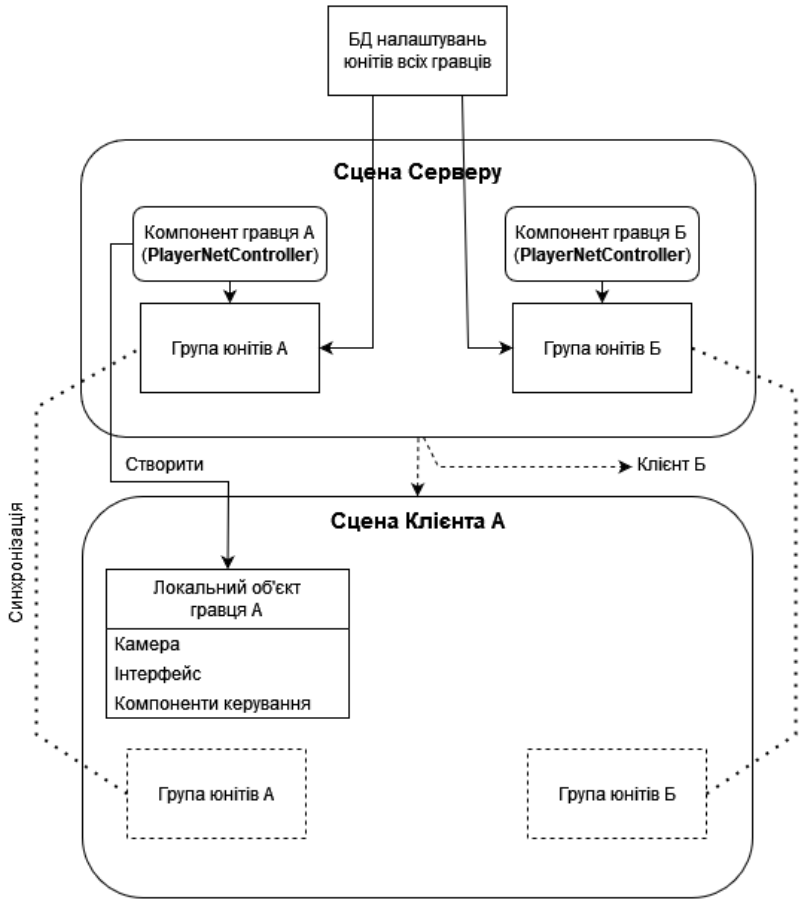

Рисунок 3.4 – Схема ініціалізації гравців

Пунктиром на схемі вище, а також на наступних схемах цього розділу, позначені «об’єкти-симуляції», тобто такі, значуща функціональність яких обмежена серверною стороною, а на клієнтській стороні вони лише синхронізують свій стан. Іншими словами, сервер зберігає авторитетність над цими об’єктами. Значуща функціональність, в даному випадку, це та, що безпосередньо впливає на перебіг ігрового процесу. Клієнтські екземпляри цих об’єктів все одно можуть мати певні компоненти, що виконують обчислення на клієнтській стороні, наприклад ті, що пов’язані з інтерфейсом користувача або візуальними ефектами.

Таким чином, в ініціалізації гравців та створенні їх об’єктів та груп юнітів повністю дотримані принципи авторитетності серверу. Клієнт не має доступу до «дійсних» даних гравця або групи юнітів як своєї, так і опонента. Об’єктом «гравця» для клієнта є локальний об’єкт, що забезпечує ввід та вивід. Юніти на стороні клієнта представлені локальними об’єктами з компонентами, що діють

виключно на клієнтській стороні, та мережевими компонентами, що не зберігають «дійсні» дані, а лише копіюють їх з сервера.

3.3.2 Показники і модулі юнітів

Призначення, збереження та доступ до операцій над показниками юніта забезпечує компонент UnitStatsHolder, що призначений кожному юніту.

Кожен юніт має базові характеристики, що призначаються при створенні екземпляра юніта. В фінальній версії гри ці характеристики завантажуватимуться з бази даних (див. пункт 3.3.7), але в демонстраційній версії, що розробляється в даному дипломному проєкті, вони призначаються кодом у штучний спосіб.

Базові характеристики можуть бути модифіковані ефектами пасивних модулів, встановлених на цей юніт. Результуючі характеристики називаються показниками.

Показники кожного юніта містять такі значення:

– рівень здоров’я (броні) юніта. Зниження нижче 0 призводить до знищення юніта;

– спротив броні юніта вхідному ураженню. Оригінальне значення та поточне значення, останнє може тимчасово знижуватись;

– швидкість переміщення юніта, поточне значення та максимальне;

– поточна кількість енергії юніта та максимальна накопичувана кількість енергії;

– коефіцієнт регенерації енергії.

Значення показників, що мають змінну максимального значення, обмежені від 0 до максимального значення. Показники рівня броні, спротиву броні та швидкості можуть бути змінені лише зовнішнім впливом.

Регенерація енергії відбувається із часом і залежить від відношення поточної кількості енергії до максимальної за значенням кривої. Крива регенерації енергії юнітів зображена на рисунку 3.5.

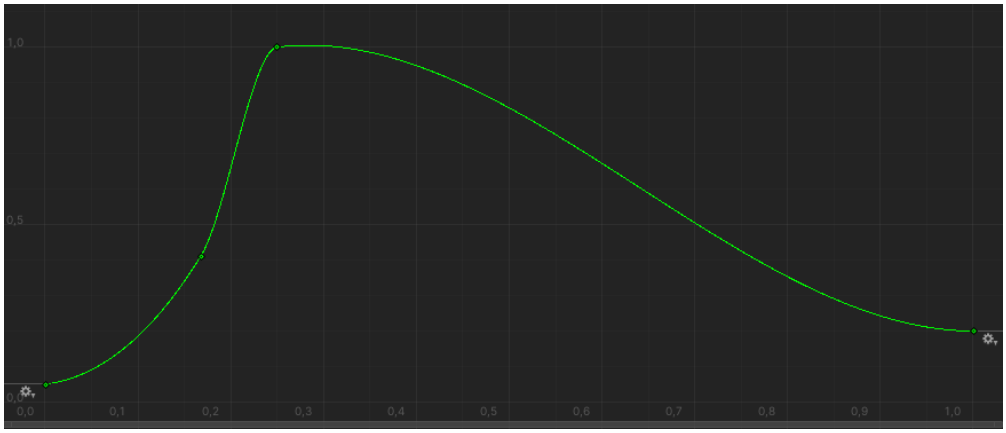

Рисунок 3.5 – Крива регенерації енергії юнітів

Як видно з рисунка, крива забезпечує неоднорідну регенерацію енергії: максимальне значення досягається лише на 25% від маскимальної заповненості енергії. В той же час, на 0% темп регенерації становить 10%, а на 100% - 20% від максимального темпу регенерації. Отримане на даному етапі значення додатково множиться на коефіцієнт регенерації енергії.

Така механіка змусить гравців докладати більше зусиль для контролю витрат енергії своїх юнітів, а також підвищить значення злагодженої гри своїми юнітами для нейтралізації енергії юнітів опонента (див. пункт 3.3.6).

Компонент UnitStatsHolder функціонує на серверній стороні. На клієнтських екземплярах юнітів він існує, але не зберігає «дійсні» значення показників. Наявність цього компонента на клієнтській стороні пов’язана з тим, що тільки мережевий компонент може викликати RPC.

На клієнтській стороні діє компонент StatsHolderLocal, який забезпечує синхронізацію та зберігання частини показників, що передбачають відображення на клієнтській стороні. Для цього він викликає ServerRPC на локальному екземплярі UnitStatsHolder, ця операція виконується на серверному екземплярі UnitStatsHolder (який містить «дійсні» значення показників) і останній через ClientRPC повертає їх на клієнтську сторону.

Схема отримання значень показників для відображення зображена на рисунку 3.6 та на кресленику ІА92.240БАК.004 Д2.

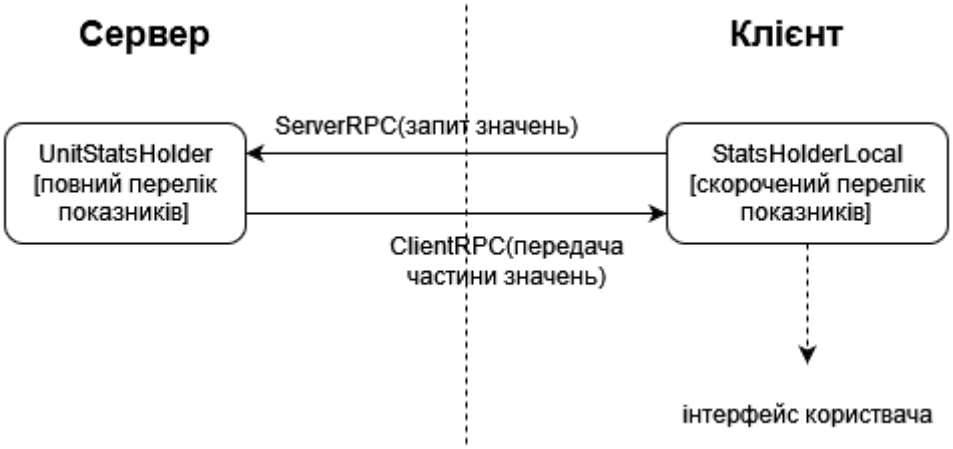

Рисунок 3.6 – Схема отримання значень показників для відображення

На етапі обміну RPC на сервері відбувається перевірка id юніта-відправника, бо StatsHolderLocal кожного юніта повинен мати доступ до серверного UnitStatsHolder виключно того самого юніта, але його серверного екземпляру. Id кожного мережевого об’єкта зберігається полем NetworkObjectId компонента NetworkObject(Netcode).

Значення ключових показників, що отримані на клієнтській стороні, відображаються елементами інтерфейсу користувача. Також вони визначають деякі візуальні ефекти.

Запит на синхронізацію локальних та серверних значень на клієнтській стороні відбувається в кожному такті обчислень, і, відповідно, синхронізація відбувається в кожному такті синхронізації з сервером (налаштований в NetworkManager).

Перелік всіх модулів кожного юніта зберігається його компонентом UnitStatsHolder на сервері. Як і показники юнітів, перелік модулів передбачає налаштування користувачем, тому в фінальній версії гри він зчитуватиметься з БД (див. пункт 3.3.7). Ініціалізація модулів відбувається одразу зі створенням юніта. UnitStatsHolder також містить функції застосування ефектів модулів до цього юніта, які визначають тип модуля і змінюють відповідні показники юніта. Модулі мають перелік значень, таких як: тип (id), сила ефекту, ціна активації, оптиммальна дистанція роботи, час циклу, ранг тощо.

Пасивні модулі мають лише поля тип та сила ефекту, та застосовуються один раз одразу при створенні. Активні модулі поділяються на ті, що діють точково на обрану ціль, та ті, що діють на всі юніти по певній площі. Реалізовані модулі першого типу: зброя, ремонт броні, нейтралізація енергії. Реалізований модуль другого типу – сфера сповільнення.

Таким чином, в ініціалізації та синхронізації показників повністю дотримані принципи авторитетності серверу. Клієнт не має доступу до «дійсних» показників або модулів юнітів як своїх, так і опонента. Показники юнітів на стороні клієнта представлені локальними компонентами, що діють на клієнтській стороні, та мережевими компонентами, що не зберігають «дійсні» дані, а лише копіюють деякі з них з сервера. «Дійсні» показники юнітів та переліки модулів зберігаються відповідним компонентом виключно на серверній стороні та операції над ними виконує виключно сервер. Доступ до даних на сервері обмежений, виконується валідація запитів.

3.3.3 Контроль перебігу матчу

Кожен матч обмежений по часу, проходить в обмеженій зоні (арені) та продовжується до повного знищення групи юнітів одного з гравців.

Контроль перебігу матчу здійснюється компонентом GameManager, який існує як на серверній, так і на клієнтській стороні. Функціонал клієнтського екземпляру цього компонента обмежений лише візуальним відображенням часу та меж арени.

Матч вважається розпочатим в момент під’єднання останнього (другого) гравця до сереверу. В той же момент гравці отримують можливість керувати своїми юнітами, а також починається відлік часу та ініціалізується рахунок.

Початковий рахунок кожного з гравців дорівнює кількості юнітів в їх групі. Зі знищенням кожного юніта певного гравця, в останнього рахунок зменшується на 1. Після досягнення значення 0, цей гравець вважається таким, що програв, а протилежний, відповідно, переможцем.

Відлік часу починається з початком матчу. Після вичерпання основного часу починається додатковий і продовжується до самого завершення матчу.

Центр арени розміщений в центрі сцени. Юніти гравців створюються симетрично відносно центру. На сервері компонент GameManager контролює вихід будь якого юніта за межі арени і застосовує до нього певні санкції. В основний час, в юніта, що знаходиться за межами арени, вимикається спротив броні, а максимальна кількість броні знижується вдвічі. Після повернення в межі арени показники відновлюються. В додатковий час описані санкціїї діються на всі юніти, а юніти, що знаходяться за межами арени, миттєво знищуються. Також, в додатковий час арена зменшується.

Налаштування умов матчу зберігаються окремо. В реалізованій в даному проєкті демонстраційній версії гри – в окремому файлі GameData. В готовій грі може використовуватися база даних. Схема реалізації серверного та клієнтського віріантів меж арени зображена на рисунку 3.7 та на кресленику ІА92.240БАК.004 ДЗ.

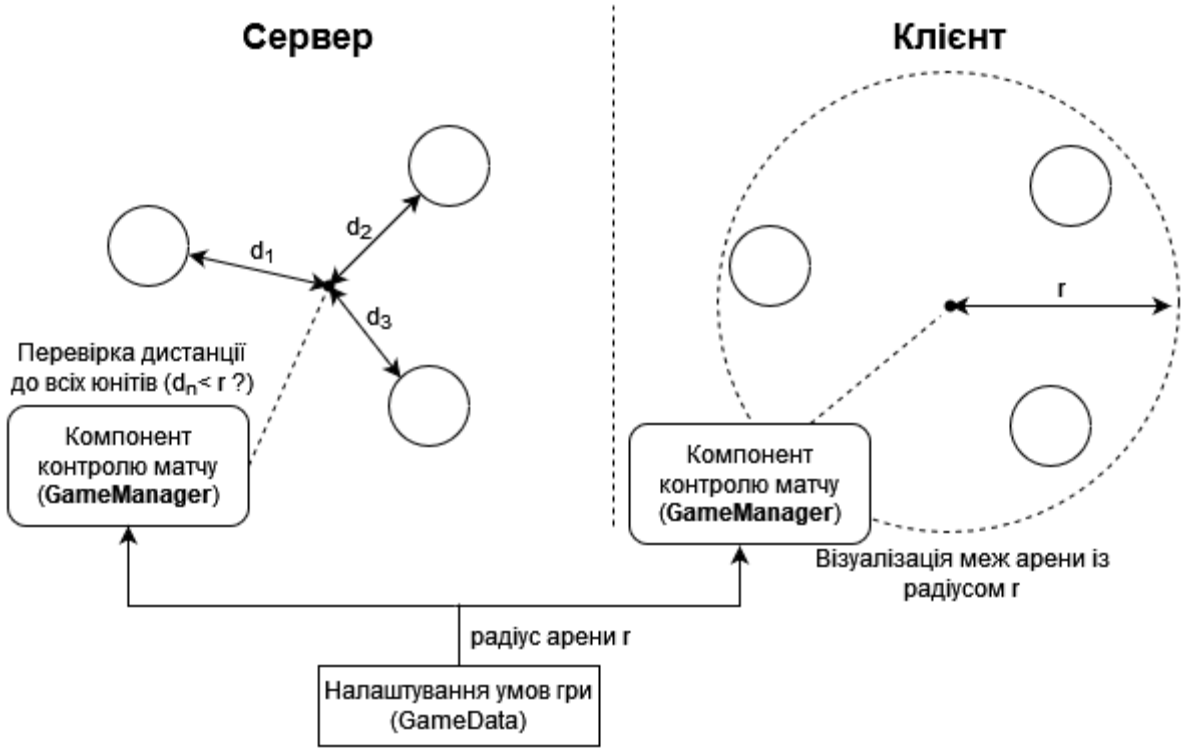

Рисунок 3.7 – Схема серверних та клієнтських меж арени

Наведена вище схема демонструє відмінності в роботі компонента GameManager на серверній та клієнтській сторонах. Перевірка дистанції (d) кожного юніта від центру арени, порівняння із заданим радіусом (r) арени та застосування санкцій відбувається виключно на сервері. На клієнтських сценах лише створюється візуалізація меж арени із урахуванням заданого радіуса. Як вже зазначено вище, дані про умови гри GameData в готовій грі можуть зберігатись в БД, але навіть у випадку локального зберігання цих значень, клієнт теоретично зможе вплинути лише на візуалізацію на своїй сцені, а не на логіку гри.

Відлік часу не синхронізується, на клієнтах він йде паралельно для економії обміну даними. Ігрова логіка, що зав’язана на час матчу, орієнтується на серверний відлік часу. Втручання клієнта у свій відлік часу вплине лише на візуалізацію на його сцені, а не на логіку гри.

Після завершення матчу здійснюється невелика пауза, під час якої клієнтам виводиться ім’я (id) переможця. В той же час на сервері виконується фіксація id переможця (вивід в консоль, в повноцінній версії гри – запис в БД).

Після паузи на сервері і клієнтах виконується від’єднання від мережі та знищуються екземпляри компонента NetworkManager(Netcode) для очищення мережевих даних цієї сесії. На сервері і клієнтах виклікається завантаження початкової сцени (головного меню). Знищення та повторне створення NetworkManager(Netcode) необхідно для того, щоб в наступних матчах на тому ж сервері не були наявні сліди минулих сесій. Від’єднання клієнтів санкціонується зі сторони сервера.

Таким чином, контроль перебігу матчу відбувається із дотриманням авторитетності серверу. Обчислення відстані юнітів від центру арени та відлік часу відбуваються на серверній стороні, у клієнтів ці процеси відбуваються паралельно та від них залежить лише візуалізація, що не впливає на ігровий процес. Рахунок знищених юнітів ведеться виключно у серверних екземплярах об’єктів гравців, клієнти не мають доступу до нього.

3.3.4 Ввід клієнта

Єдиним компонентом, що забезпечує ввід від клієнта, є компонент Selector. Він існує лише на клієнтській стороні і призначений локальному об’єкту гравця.Тобто, на сцені кожного клієнта існує лише один компонент Selector, який цей клієнт і використовує.

Для вводу фіксується натискання кнопок миші, натискання клавіш на клавіатурі та комбінації натискання кнопки миші із затиснутими клавішами клавіатури.

Компонент Selector також контролює локальне візуальне відображення статусу кожного юніта, а саме:

– свій;

– ворожий;

– ціль;

– активний.

Візуальне розділення на «своїх» та «ворожих» відбувається локально на клієнті, але серверні компоненти, що безпосередньо виконують ігрові обчислення, перевіряють факт володіння над об’єктом того клієнта, що здійснив ввід. Команда на призначення певних юнітів «своїми» надходить з сервера одразу після їх створення на сцені.

При натисканні на клавіатурі однієї з двох визначених клавіш здійснюється переключення по своїм юнітам в одну чи іншу сторону по списку циклічно. Обраний юніт є «активним». Ввід щодо переміщення або активації модулів цього юніта здійснюється виключно для цього юніта.

Натискання лівою кнопкою миші по будь якому юніту, окрім «активного», робить його «ціллю». Обрана ціль є загальною для всіх своїх юнітів.

Натискання клавіш цифр надсилає на сервер команду активації або деактивації відповідного цифрі модуля, встановленого на активний юніт. Команда обробляється серверним компонентом ModuleActivator (див. пункт 3.3.6).

Натискання лівої кнопки миші із певною затиснутою клавішею клавіатури надсилає на сервер команду стосовно переміщення активного юніта. Команда обробляється серверним компонентом Navigator (див. пункт 3.3.5).

Отже, основне керування своєю групою юнітів та їх модулями забезпечується локальним компонентом Selector, що контролює локальний візуальний супровід дій щодо керування у кожного клієнта, а також надсилає запити на серверні процедури відповідним серверним компонентам.

3.3.5 Переміщення юнітів

Всі переміщення певного юніта забезпечує компонент Navigator, що приєднаний до нього. Цей компонент діє на серверній стороні.

Клієнтський компонент Selector подає команду на активацію руху певного характеру викликом відповідного ServerRPC у Navigator. Той перевіряє відправника, і якщо він володіє юнітом, якому призначений цей Navigator, задає відповідний характер руху серверному екземпляру Navigator. На сервері, при заданому характері руху, Navigator у кожному такті обчислень виконує відповідне зміщення юніта. На кожний такт синхронізації клієнтів із сервером, зміщення серверного юніта застосовується до клієнтських екземплярів юнітів компонентом NetworkTransform(Netcode). Той самий компонент здійснює інтерполяцію зміщення, оскільки частота такту синхронізації значно нижча за частоту кадрів клієнта, а рух юнітів на клієнтській стороні повинен здійснюватись плавно. Схема синхронізації переміщення юніта зображена на рисунку 3.8 та на кресленику ІА92.240БАК.004 Д4.

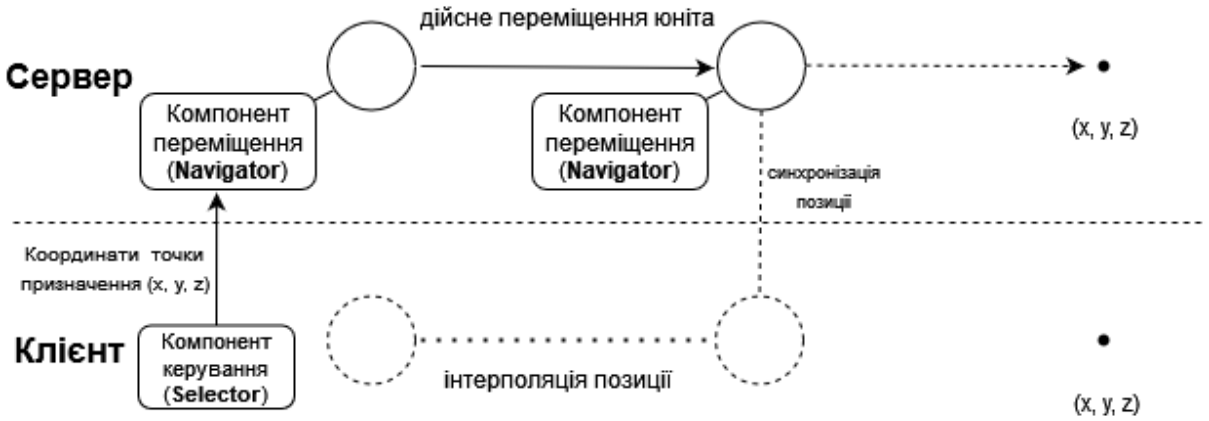

Рисунок 3.8 – Схема синхронізації переміщення юніта

Для гри розроблено два характери руху: переміщення в точку та рух по орбіті навколо юніта-цілі.

Для переміщення в точку, Selector викликає відповідний ServerRPC у Navigator із передачею координат точки призначення. Координати зберігаються у Navigator доти, поки не буде призначено нову точку призначення. Якщо ці координати визначені, Navigator рухатиме юніт в цю точку в кожному такті обчислень.

Рух по орбіті навколо юніта-цілі реалізовано схожим чином, але замість кінцевих координат передається id юніта, навколо якого виконуватиметься рух. Також передається введений клієнтом радіус орбіти. Координати точки призначення обчислюються на серверній стороні всередині Navigator у кожному такті обчислень і залежать від координат юніта-цілі.

Якщо відстань до юніта-цілі більша за радіус орбіти, керований юніт прямує по дотичній до кола орбіти. Після цього починає рух по орбіті навколо юніта-цілі.

Для реалізації руху по орбіті була використана формула, що описує рух точки (x, y) навколо точки (x_0, y_0) по колу, радіус якого дорівнює R:

$$x = x_0 + Rcost, \quad (3.1)$$

$$y = y_0 + Rsint, \quad (3.2)$$

де t – параметр, що лінійно змінюється (кут).

Схема етапів руху юніта по орбіті навколо юніта-цілі зображена на рисунку 3.9.

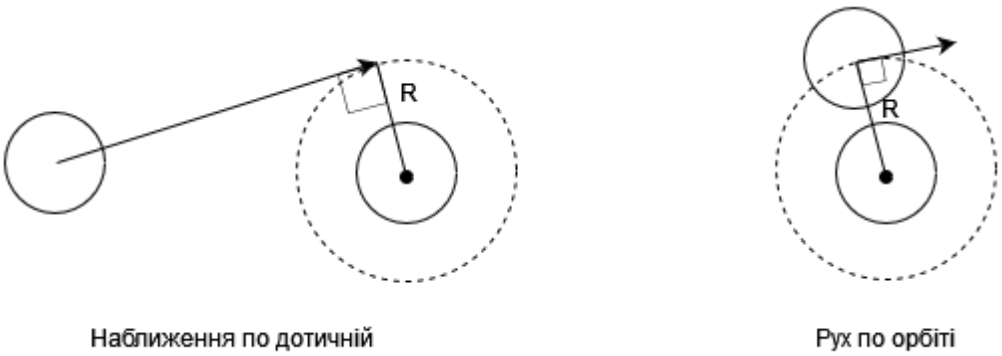

Рисунок 3.9 – Схема етапів руху юніта по орбіті навколо юніта-цілі

Таким чином, реалізована підсистема переміщення юнітів повністю відповідає принципам авторитетності серверу. Клієнт за допомогою локального компонета керування тільки робить запит на активацію певного руху із передачею параметрів в залежності від обраного характеру руху: координати точки призначення для переміщення в цю точку або id юніта-цілі та радіус для руху по орбіті навколо цього юніта-цілі. Валідація та обрахунки зміщення юніта виконуються виключно на сервері. Незалежно від переміщення, поточна позиція кожного юніта синхронізується клієнтами.

3.3.6 Активація модулів

Активацію, деактивацію та перебіг роботи модуля контролює компонент ModuleActivator. Він функціонує на серверній стороні, призначений кожному юніту, на клієнтській слугує лише для виклику ServerRPC, а також для виклику відтворення візуальних ефектів тих чи інших модулів.

Як вже зазначалось раніше, модуль має циклічний характер роботи. Він активується, після чого вважається запущеним. Після деактивації він працює до кінця поточного циклу і вимикається. Причиною вимкнення також може бути

знищення юніта, якому належить цей модуль, або нестача енергії для початку нового циклу.

Запит на активацію модуля надсилає компонент Selector, коли отримує відповідний ввід від клієнта. В запиті передається id юніта-цілі та id модуля в переліку активних модулів поточного активного юніта. Сам перелік існує тільки у компоненті UnitStatsHolder на сервері.

При активації модуля на сервері перевіряються умови для його запуску. У разі успішної перевірки запускається тривалий процес роботи модуля. Поки модуль активний, він не може бути запущений знову.

Тривалі процеси в Unity представлені механізмом Coroutine, що дозволяє виконувати певні обчислення паралельно до основних обчислень компонента, а також дає можливість призупиняти обчислення на певний час та потім їх відновлювати. Слід зазначити, що Coroutines не є потоками обчислень, синхронні операції виконуються в основному потоці [11].

Coroutine роботи модуля запускається із його активацією, приймає аргументами сам об’єкт модуля (що містить всі його дані) та id юніта-цілі. Coroutine запускає роботу модуля на ціль, викликаючи відповідні функції компонента UnitStatsHolder із вказанням модуля, веде відлік часу циклу модуля. При вичерпанні часу циклу модуля, за збереження статусу активованого, та за виконання всіх умов, модуль запускається на наступний цикл. Якщо модуль деактивований, то із завершенням циклу робота модуля закінчується, після чого він може бути активований знову.

Перевірка умов запуску модуля (чи активний модуль, чи достатньо енергії у юніта, чи не знищений юніт, чи не знищена ціль) виконується перед початком кожного циклу. Якщо умови не виконуються – модуль завершує роботу і не починає наступний цикл. На початку кожного циклу виконується виклик ClientRPC, що створює візуальні ефекти модуля у клієнтів. Перед кожним застосуванням ефекту модуля обчислюється дистанція до цілі, якщо вона більша за оптимальну дистанцію роботи модуля – ефект зменшується, як і інтенсивність візуального ефекту.

Застосування ефекту модуля до UnitStatsHolder юніта-цілі (або цілей) відбувається на початку циклу для модулів, що діють на ціль, та в кожний такт обчислень для модулів, що діють по площі. Така різниця створена для того, щоб вплив ефекту дії по площі закінчувався одразу після виходу цілі із цієї площі. Схема активації модуля зображена на рисунку 3.10 та на кресленику ІА92.240БАК.004 Д5.

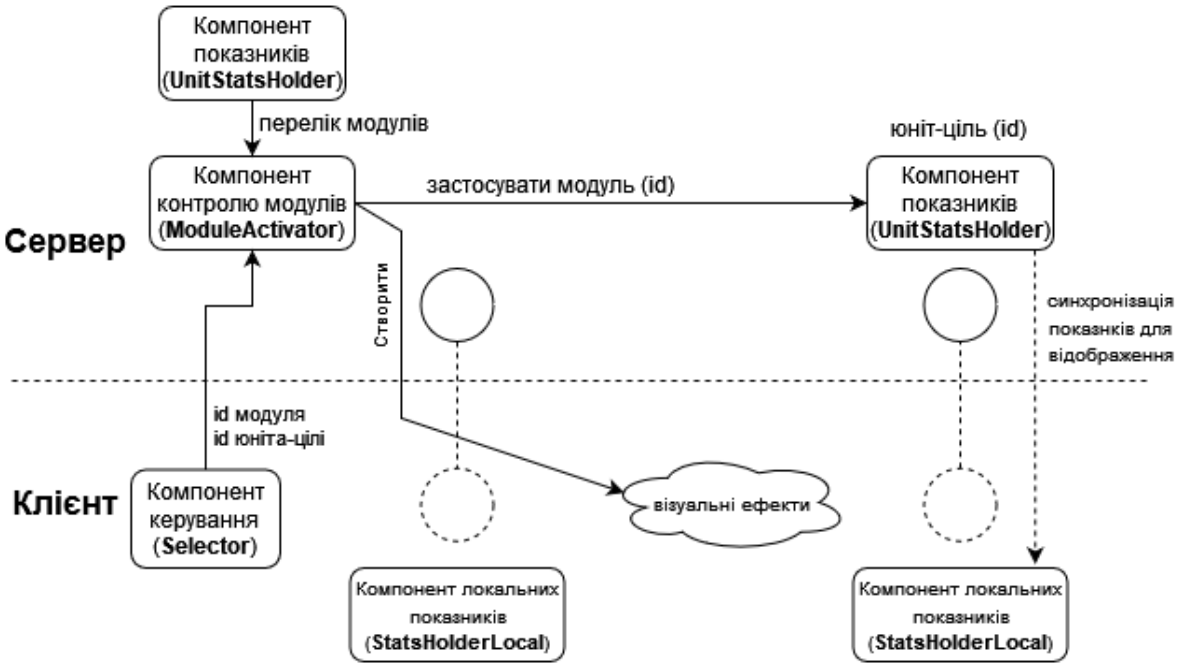

Рисунок 3.10 – Схема активації модуля

Таким чином, реалізована підсистема активації модулів юніта повністю відповідає принципам авторитетності серверу. Клієнт за допомогою локального компонета керування тільки робить запит на активацію певного модуля із вказанням індекса модуля в переліку модулів цього юніта, що зберігається на сервері, та вказанням id юніта-цілі. Серверний компонент, відповідальний за логіку активації, деактивації, перебігу роботи модулів і застосування ефектів модулів до цілей та функціонує виключно на сервері, активує модуль, що відповідає надісланому клієнтом індексу на відповідний юніт-ціль. Клієнт не має доступу до «дійсних» даних модулів або перебігу циклу модуля.

3.3.7 Збереження даних

В цьому пункті описані механізми введення і збереження даних в зовнішньому сховищі (базі даних). Ці механізми були розроблені теоретично, але не реалізовані в програмному продукті. Вони є необхідними для онлайн-гри як такої, але не є ключовими в контексті теми даного дипломного проєкту.

Бази даних (БД) використовуються з метою ефективного зберігання, керування та доступу до великих обсягів інформації. Вони є основою для багатьох сучасних програмних застосунків і систем. БД дозволяє ефективно зберігати великі обсяги структурованих даних, організовує швидкий та зручний доступ до них.

Як зазначено в п. 3.3.1 та 3.3.2, початкові показники юнітів та перелік їх модулів повинні зберігатися в БД. Вважатимемо, що користувач вже авторизований в системі і має свій унікальний ідентифікатор (питання авторизації буде розглянуте в наступному розділі). Відповідно, БД зберігатиме дані про юнітів та модулі цього користувача та зможе організувати роботу з ними використовуючи його ідентифікатор. Запис в цю БД виконуватиметься зі сцени налаштування оснащень. Ця ж сцена зчитуватиме дані з БД для відображення користувачу поточного оснащення.

В п. 3.3.3 вказано, що після завершення матчу виконується запис переможця в БД. Це окрема БД, що зберіає статистику (досягнення) кожного гравця. Дані з цієї БД можна буде переглянути на окремій сцені, що дозволить гравцям оцінити рівень своєї гри відносно інших гравців. Також ці дані використовуватимуться в майбутньому при підборі матчу для вибору рівного по навичках опонента (підбір матчів розглянуто в наступному розділі).

Також застосунок гри повинен зберігати користувацькі налаштування. Користувацькі налаштування в іграх відіграють важливу роль у створенні зручного та приємного геймплею. Вони дозволяють гравцям налаштовувати різні аспекти гри відповідно до їхніх особистих уподобань та потреб.

Одним з найпоширеніших користувацьких налаштувань є графічні налаштування. Гравці повинні мати можливість налаштовувати роздільну здатність

екрану, рівень деталізації графіки, наявність антиаліасингу, тіней, текстур та інших аспектів візуальної частини застосунку. Це дозволяє гравцям знайти баланс між візуальною якістю та продуктивністю гри в залежності від характеристик їхнього комп'ютера та особистих уподобань.

Гравці повинні мати можливість налаштовувати клавіатуру, мишу, геймпад або інші пристрої для керування у грі. Це дозволяє гравцям зручно взаємодіяти з грою та та вибрати для цього найефективніший спосіб.

Звукові налаштування також є важливою частиною налаштувань в іграх. Гравці повинні мати можливість налаштовувати гучність звуку, рівень ефектів, музики тощо.

Збереження користувацьких налаштувань зазвичай виконується локально на комп’ютері користувача та зберігається в окремих файлах. Виключенням може бути хмарне збереження налаштувань, наприклад якщо гра працює через ігровий сервіс, на кшталт Steam. Користувацькі налаштування застосунку виконуватимуться в додатковому меню, що відкриватиметься після натискання кнопки в головному меню.

В загальному вигляді, робота застосунку зі збереженими даними виглядатиме так, як зображено на рисунку 3.11 та на кресленику ІА92.240БАК.004 Д6.

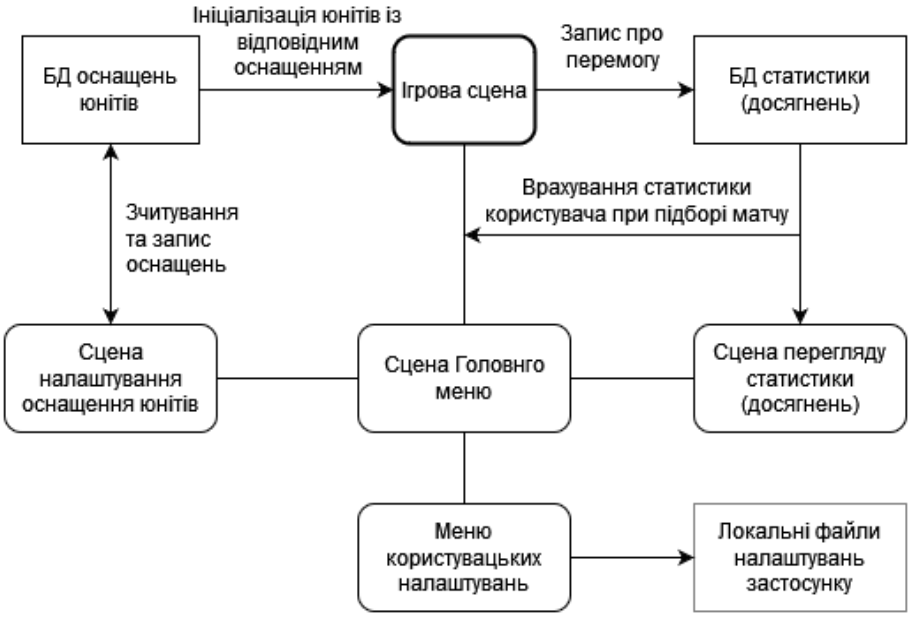

Рисунок 3.11 – Діаграма потоків даних

Користувацькі налаштування програми можуть бути збережені у файлах формату JSON або XML. Обидва ці формати дозволяють структуровано зберігати дані у зручному для читання та обробки вигляді. JSON має простий синтаксис, що дозволяє легко представляти об'єкти та масиви даних. Водночас, XML є багатофункціональним форматом, який може використовуватися для представлення структурованих даних з різних джерел. Обидва формати є платформонезалежними, що робить їх універсальними для зберігання та передачі налаштувань між різними системами та платформами.

Висновки до розділу

В цьому розділі було описано процес розробки онлайн-гри на базі архітектури з авторитетним сервером.

В першу чергу, було визначено пріорітет розробки, а саме ігрова сцена, на якій проходитиме матч. Була виокремлена та досліджена загальна структура ігрової мережевої сесії. Визначено, що мережеві ігрові сцени (всесвіти) на сервері та клієнтах існують паралельно, синхронізуючи стани. Досліджено відповідні компоненти Netcode, що забезпечують синхронізацію сцени мережею.

Далі описаний основний механізм, що використовувася для написання клієнт-серверної логіки ігрового процесу – Remote Procedure Call (RPC, дистаційний виклик процедури). Обґрунтовано використання його імплементацій в Netcode (ServerRPC та ClientRPC) для тих чи інших завдань при розробці.

На наступному кроці було розглянуто поняття ігрової механіки та позначено основні з них для реалізації в даному проєкті. Перелічені механіки являють собою основу застосунку, імплементують ключові елементи дизайну гри, описаного в попередньому розділі та відповідають визначеним раніше критеріям для проєкту.

Описано архітектуру механіки створення гравців в новій ігровій сесії, створення груп їх юнітів. Описано архітектуру механіки ініціалізації показників юнітів і їх модулів в новій ігровій сесії. Описано архітектуру підсистеми контролю перебігу матчу. Описано архітектуру підсистеми забезпечення вводу користувача

для виклику відповідних дій на сервері. Описано архітектури підсистем переміщення юнітів та активації модулів кожного юніта.

Для всіх описаних механік схематично зображено взаємодію відповідних компонентів клієнта та сервера. Доведено відповідність архітектури всіх описаних механік принципам авторитетності сервера та критеріям проєкту, а саме: клієнтські екземпляри застосунку не містять даних, відображення яких не передбачено; клієнти не мають змоги вплинути втучатися в основні ігрові обчислення, що відбуваються на сервері.

На останок, описано теоретично розроблені, але нереалізовані системи роботи із даними, зокрема: систему налаштування і збереження оснащення юнітів користувача, збереження і відображення статистики (досягнень) користувача, локальне збереження користувацьких налаштувань.

4 ТЕХНОГОЛІЧНИЙ РОЗДІЛ

В даному проєкті була реалізована демонстрайційна версія онлайн-гри, відповідно, керівництво користувача описує роботу із існуючим застосунком.

За допомогою інструментів Unity Editor, скомпільовано збірку програми (англ. build) для платформи ПК (Windows). Вона представлена папкою файлів із .exe файлом в голові. Загальна вага файлів скомпільованої гри – 136 МБ.

4.1 Керівництво користувача

Оскільки на даному етапі не використовується хостинг серверу, для функціонування системи всі екземпляри застосунку, і клієнти, і сервери, повинні бути запущені на машинах, що об’єднані однією локальною мережею. При цьому в усіх екземплярах застосунку повинні бути налаштовані IP-адреса та порт, що відповідають машині, на якій в межах цієї мережі запускається сервер. Для роботи всіх клієнтів та сервера на одній машині використовується localhost із будь яким вільним портом.

Для початку гри користувач запускає виконуваний файл (.exe) з папки збірки.

Початковою сценю гри є сцена головного меню. Скріншот сцени головного меню зображено на рисунку 4.1.

Рисунок. 4.1 – Скріншот сцени головного меню

У головному меню користувачу доступні чотири кнопки: PLAY (з англ. грати), settings (з англ. налаштування), кнопки вліво та вправо. Перша кнопка завантажує основну ігрову сцену. Друга – відкриває меню користувацьких налаштувань. Третя і четверта – відповідно, завантажують сцену оснащення юнітів та сцену перегляду статистики (досягнень). З перелічених функціонує тільки перша, основна.

Після натискання кнопки PLAY завантажується основна ігрова сцена. Для встановлення зв’язку клієнта із сервером користувач повинент обрати свою мережеву роль: клієнт або сервер; а також IP-адреса та порт. В одній мережі та за однією IP-адресою може бути запущений лише один сервер. Скріншот меню вибору мережевої ролі зображено на рисунку 4.2.

Рисунок 4.2 – Скріншот меню вибору мережевої ролі

Якщо користувач обере опцію «Server», його сцена ініціалізується як серверна в поточній мережі, якщо на даний момент не запущений інший сервер із тією самою адресою.

Якщо користувач обере опцію «Client», його сцена ініціалізується як клієнтська в поточній мережі, і як тільки сервер запущений – починається синхронізація станів між цим клієнтом та сервером.

Сервер буде запущений за вказаною адресою та портом. Відповідно клієнт повинен ввести таку ж адресу та порт для під’єднання до цього серверу. За замовчуванням використосується localhost (127.0.0.1) з портом 7777.

Можливість самостійно вводити адресу дозволяє грати в гру з різних машин у будь-якій локальній мережі. Це також може бути віртуальна мережа між машинами на відстані, створена емулятором на кшталт Hamachi.

Скріншот застосунку на початку гри зображено на рисунку. 4.3.

Рисунок 4.3 – Скріншот ігрового процесу, початок гри

За приєднання клієнта до серверу, на сцені ініціалізуються його юніти зі своїми кінцевими показниками. Камера клієнта автоматично регулює свою позицію та віддалення відносно положення всіх юнітів в арені, а саме, орієнтується на середні арифметичні координати між всіма юнітами. Червоний бар'єр представляє візуалізацію поточного краю арени, за межами якої до юнітів застосовуватимуться санкції. Гравці розміщуються на сцені один навпроти одного.

Користувацький інтерфейс включає в себе: основні показники поточного активного юніта (броня, рівень енергії, швидкість), слайдер для регулювання радіусу орбіти, панель модулів та їх статусів (не реалізована), відлік часу матчу.

В готовій грі панель модулів та їх статусів виводитиме іконки модулів активного юніта з їх ідентифікаторами, а також візуалізуватиме статус кожного модуля (активований, запущений, деактивований, вимкнений) для зручності їх контролю гравцем.

Також, інтерфейс гри включає консоль, в яку виводяться деякі повідомлення про ігрові події на сцені.

Скріншот серверного екземпляру застосунку під час гри зображено на рисунку. 4.4.

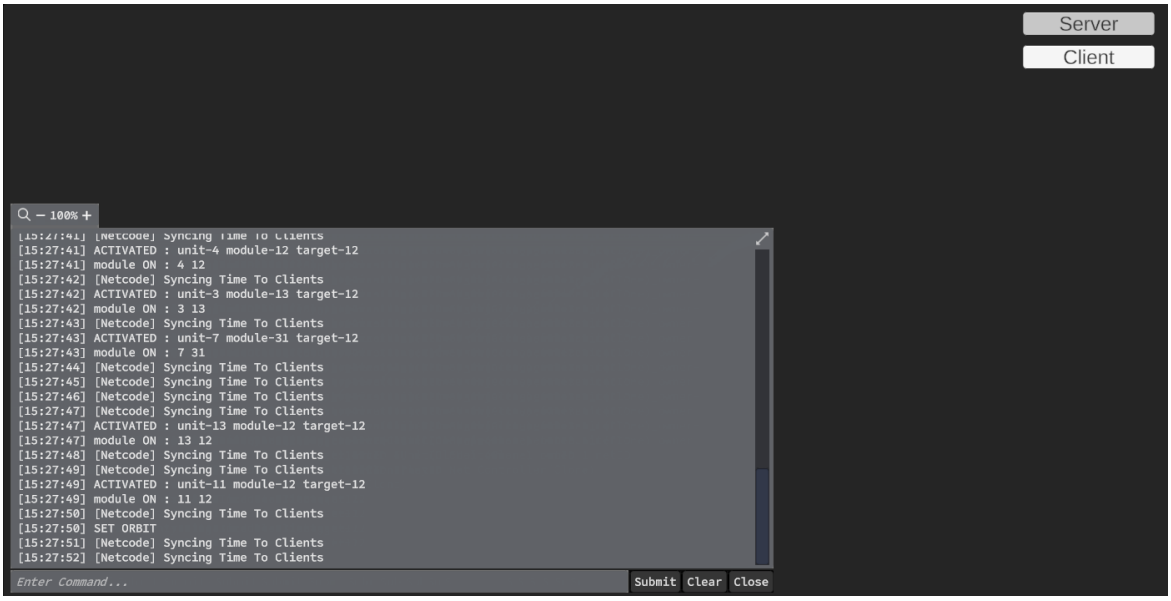

Рисунок 4.4 – Скріншот ігрового процесу, вид з сервера

На сервері відсутня візуалізація сцени для зменшення обчислювального навантаження. В консоль виводяться технічні повідомлення компонентів Netcode, що забезпечують мережве з’єднання. Також основні серверні компоненти виводять в консоль стислі повідомлення про виконувані дії.

Вивід в консоль є потужним інструментом полагодження програми. Завдяки йому можна зрозуміти які операції виконуються на клієнтах, а які на сервері, виводячи відповідні повідомлення в консоль.

Після початку матчу користувач може здійснювати керування: переміщувати юнітів задаючи їм кінцеву точку або орбіту, обирати свого «активного» юніта, та обирати юніт-ціль.

Переміщення юнітів зображено на скріншоті застосунку під час гри на рисунку 4.3.

Рисунок 4.5 – Скріншот ігрового процесу, переміщення юнітів

Зміщувати статус активного в своїй групі юнітів потрібно за допомогою клавіш A та D.

Для переміщення активного юніта в кінцеву точку необхідно затиснути W і натиснути лівою кнопкою миші на поверхню сцени. Для переміщення активного юніта по орбіті навколо іншого юніта необхідно затиснути Q і натиснути лівою кнопкою миші на юніт, навколо якого здійснюватиметься рух. При затисканні W або Q на екрані з’являється напис про обраний характер руху.

Натискання лівої кнопки миші по будь-якому юніту, окрім активного, робить його «обраним». До обраного юніта застосовуватимуться модулі при активації.

Візуальні відмінності статусів юніта зображено на скріншоті на рисунку 4.6.

Рисунок 4.6 – Візуальні статуси юнітів (зліва направо): свій, свій-актвний, ворожий, ворожий-ціль

Модулі, що передбачають активацію, поточного активного юніта активуються та деактивуються натисканням клавіш-цифр 1, 2, 3 і т.д. Цифра, в даному випадку, відповідає позиції цього модуля в оснащенні юніта. Порядок модулів в оснащенні в майбутньому буде визначатися користувачем при налаштуванні оснащення юніта. В реалізованій демонстраційній версії гри всі юніти мають набір модулів за замовчуванням, а саме – всі існуючі модулі в одному екземплярі.

Вплив активованих модулів візуалізується відповідними візуальними ефектами, що зображені на скриншоті нижче. Візуальні ефекти модулів, що діють на ціль, поєднують юніт, якому належить цей модуль, із юнітом-ціллю, та напрямлені на останнього. На скріншоті це (згори донизу): нейтралізація енергії, зброя, ремонт. Ефект модуля, що діє по площі, не прив’язаний до цілей. На скріншоті це сфера сповільнення.

Візуалізацію запущених модулів зображено на скріншоті на рисунку 4.7.

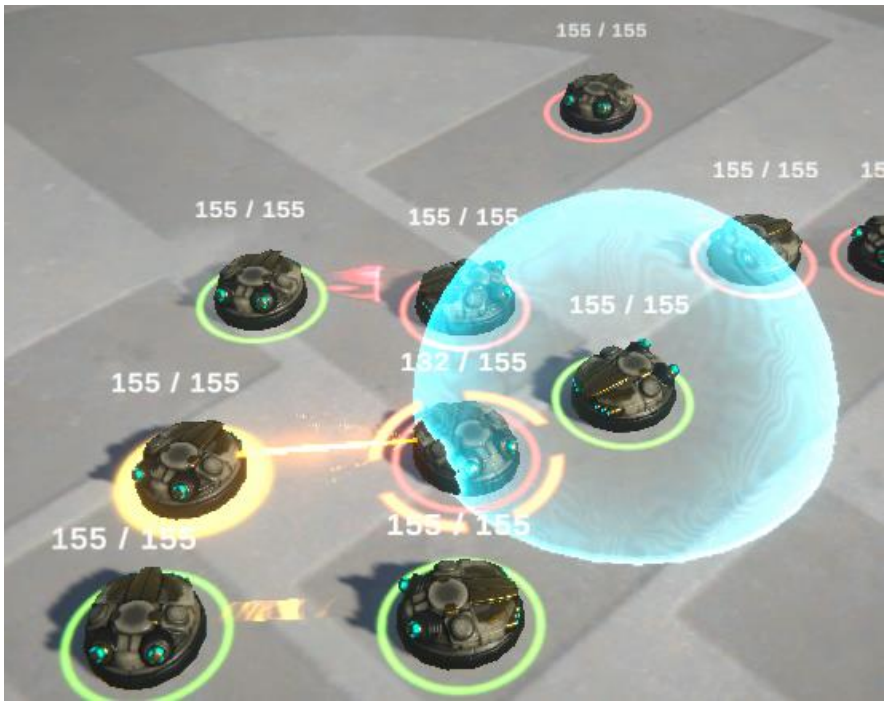

Рисунок 4.7 – Скріншот ігрового процесу, активовані модулі

Над кожним юнітом виводиться його рівень броні. Після знищення юніта керування ним заблоковано, а його модель змінюється на модель «взірваного» юніта.

Після знищення останнього юніта одного з гравців, на екран виводиться ім’я переможця (в демонстративній версії гри – «player» з додаванням його тимчасового id в цій сесії). За декілька секунд всі клієнти та сервер повертаються в головне меню. Після користувач може знову натиснути кнопку PLAY та увійти в новий матч.

Знищення останнього юніта та вивід імені переможця зображено на скріншоті на рисунку 4.8.

Рисунок 4.8 – Скріншот ігрового процесу, закінчення гри

4.2 Перевірка авторитетності сервера

Зважаючи на мережеву будову ігрової сцени, що описана на початку попереднього розділу, в своїй більшості мережеві компоненти існують у всіх екземплярах застосунку. Ті мережеві компоненти, що не виконують жодної функції на клієнтській стороні – вимкнені. Інші, здебільшого, лише спрямовують виклик RPC на свої серверні копії.

Тому, важливо остаточно переконатися, що ігрова логіка виконується на правильній стороні (клієнті чи сервері), а «дійсні» ігрові дані зберігаються виключно на сервері. Перевірка базуватиметься на сценаріях, що є найбільш вірогідними щодо спроб втручання нечесними гравцями.

Для перевірки реакції системи на спроби клієнта втрутитися в перебіг ігрового процесу будуть виконуватися певні дії в редакторі Unity від ролі клієнта. Це дозволить імітувати вплив чит-програм на клієнтській стороні.

Для перевірки наявності даних у компоненті, відповідні поля були серіалізовані для відображення та ручного редагування в редакторі Unity.

Для перевірки місця виконання обчислень, в програмному коді додано повідомлення, що виводитимуться в консоль в тому екземплярі застосунку, де виконується цей код.

Зміст перевірок та отримані результати внесено в таблиці 4.1 – 4.6.

Таблиця 4.1 – Перевірка переміщення юнітів

Мета перевірки	Перевірити авторитетність сервера над переміщенням юнітів.
Зміст перевірки	Зайти в гру як клієнт в редакторі Unity, намагатися вручну змінити координати юнітів в сцені.
Очікуваний результат	Юніти не можуть бути переміщені на клієнтській стороні
Отриманий результат	Юніти не можуть бути переміщені на клієнтській стороні

Отже, з таблиці 4.1 можна зробити висновок, що сервер має повну авторитетність над переміщенням юнітів.

Таблиця 4.2 – Перевірка наявності показників юнітів

Мета перевірки	Перевірити відсутність на клієнтській стороні «дійсних» показників юнітів
Зміст перевірки	Зайти в гру як клієнт в редакторі Unity, дослідити серіалізовані поля показників у юніта.
Очікуваний результат	Серверний компонент показників (UnitStatsHolder) містить порожні значення. Локальний компонент показників (StatsHolderLocal) містить скорочений перелік показників.
Отриманий результат	Серверний компонент показників (UnitStatsHolder) містить порожні значення. Локальний компонент показників (StatsHolderLocal) містить скорочений перелік показників.

Отже, з таблиці 4.2 можна зробити висновок, що «дійсні» показники юнітів не доступні для перегляду та впливу з клієнтської сторони.

Таблиця 4.3 – Перевірка можливості втручання у показники юнітів

Мета перевірки	Перевірити авторитетність сервера щодо зберігання «дійсних» показників юнітів.
Зміст перевірки	Зайти в гру як клієнт в редакторі Unity, вручну редагувати серіалізовані поля показників у юніта.

Очікуваний результат	Локальний компонент показників (StatsHolderLocal) приймає та зберігає модифіковані значення до наступного такту синхронізації з сервером. На ігровий процес це не вплинуло.
Отриманий результат	Локальний компонент показників (StatsHolderLocal) приймає та зберігає модифіковані значення до наступного такту синхронізації з сервером. На ігровий процес це не вплинуло.

Отже, з таблиці 4.3 можна зробити висновок, що сервер має повну авторитетність щодо зберігання та доступу до «дійсних» показників юнітів.

Таблиця 4.4 – Перевірка сторони обробки клієнтського вводу для переміщення юнітів

Мета перевірки	Перевірити авторитетність сервера в обробці клієнтського вводу для переміщення юнітів.
Зміст перевірки	Зайти в гру як клієнт, виконати ввід переміщення юнітів. Відстежити появу в консолі повідомлення, що надилається безпосередньо з коду, що здійснює переміщення.
Очікуваний результат	Повідомлення про початок руху з’являється в консолі на сервері. За відсутності зв’язку к сервером – рух не здійснюється.

Отриманий результат	Повідомлення про початок руху з’являється в консолі на сервері. За відсутності зв’язку к сервером – рух не здійснюється.

Отже, з таблиці 4.4 можна зробити висновок, що сервер має повну авторитетність щодо обробки клієнтського вводу для переміщення юнітів.

Таблиця 4.5 – Перевірка сторони обробки клієнтського вводу для активації та деактивації модулів

Мета перевірки	Перевірити авторитетність сервера в обробці клієнтського вводу для активації та деактивації модулів.
Зміст перевірки	Зайти в гру як клієнт, виконати ввід активації та деактивації довільного модуля будь-якого юніта. Відстежити появу в консолі повідомлення, що надилається безпосередньо з коду, що здійснює роботу модуля.
Очікуваний результат	Повідомлення про активацію, початок роботи, деактивацію та закінчення роботи модуля з’являється в консолі на сервері. За відсутності зв’язку к сервером – активація або ручна деактивація не спрацьовують.
Отриманий результат	Повідомлення про активацію, початок роботи, деактивацію та закінчення роботи модуля з’являється в консолі на сервері. За

	відсутності зв’язку к сервером – активація або ручна деактивація не спрацьовують.

Отже, з таблиці 4.5 можна зробити висновок, що сервер має повну авторитетність щодо обробки клієнтського вводу для активації та деактивації модулів.

Таблиця 4.6 – Перевірка фіксації переможця в матчі

Мета перевірки	Перевірити авторитетність сервера у фіксації переможця в матчі
Зміст перевірки	Зайти в гру як клієнт, грати до закінчення матчу. Відстежити появу в консолі повідомлення, що надилається безпосередньо з коду, що здійснює фіксацію переможця.
Очікуваний результат	Повідомлення з id переможця з’являється в консолі на сервері
Отриманий результат	Повідомлення з id переможця з’являється в консолі на сервері

Отже, з таблиці 4.6 можна зробити висновок, що сервер має повну авторитетність щодо фіксації переможця в матчі.

4.3 Подальша розробка та інтеграція Unity Game Services

В цьому дипломному проєкті було створено демонстраційну (або «розробницьку») версію онлайн-гри. Як зазначалося в попередньому розділі, реалізовано здебільшого тільки основні ігрові механіки, що забезпечують функціонування базового ігрового процесу. Реалізувавши систему налаштування

користувачем своєї групи юнітів та їх модулів, вже можна буде вважати гру такою, в яку можна комфортно грати з друзями або колегами по локальній мережі. Звісно, ні про яку змагальну компоненту в такому випадку не йдеться.

Повноцінна онлайн-гра повинна мати виділені сервери, що організовуватимуть матчі для гравців по всьому світу. Також, повинна бути система автентифікації користувачів, на створений гравцем обліковий запис записуватимуться його досягнення, оснащення юнітів і т.д.

Оскільки кожна ігрова сесія (матч) організовується одним сервером (мається на увазі екземпляр програми, а не фізичний сервер), потрібен сервіс, що контролюватиме запуск та вимкнення серверу під кожний матч. Цей же сервіс повинен буде під'єднувати гравців з загальної черги до конкретних сесій.

На щастя, описані вище складні завдання можуть бути повністю або частково вирішені за допомогою готових сервісів розміщення онлайн-ігор. В цьому проєкті передбачається майбутнє використання Unity Game Services, що був створений пліч-о-пліч з бібліотекою Netcode for GameObjects.

4.3.1 Автентифікація з Unity Authentication

Додаткам зазвичай потрібно знати особу гравця, щоб надавати різноманітні функції та послуги як розробникам ігор, так і гравцям, щоб забезпечити безпеку, послідовність і безпеку під час кожної взаємодії. Unity Authentication надає анонімні рішення автентифікації для певної платформи для підтримуваних платформ, включаючи мобільні пристрої та ПК [18].

Токени та ідентифікатор, створені цим сервісом, дозволяють продуктам Unity та іграм на Unity отримувати інформацію про гравця під час автентифікації [18].

Вони є засобом для ідентифікації гравця та зберігання даних гри гравця (наприклад, збереження стану гри та запис покупок у програмі. Вони забезпечують послідовний ігровий досвід для гравця (наприклад, бали для таблиць лідерів і запропоновані покупки в програмі). Вони надають звіт про ігрову поведінку гравця

на кількох пристроях, зберігають багатокористувацькі функції на різних платформах.

Автентифікація через певну платформі (також називається сторонньою автентифікацією або зовнішньою автентифікацією) використовує зовнішніх постачальників ідентифікаційних даних. Як правило, процес починається, коли гравець входить через зовнішнього постачальника ідентифікаційної інформації за допомогою своєї адреси електронної пошти або імені користувача та пароля. Коли гравець здійснює вхід, токен надсилається до Unity Authentication для перевірки. Якщо токен успішно перевірено зовнішнім постачальником ідентифікаційної інформації, токен буде пов’язано з ідентифікатором гравця.

4.3.2 Матчмейкінг з Unity Matchmaker

Матчмейкінг (англ. matchmaking) - це процес формування ігрових команд або пар на основі різних факторів для забезпечення рівного та конкурентного геймплею в онлайн-іграх. Це поняття широко застосовується в багатьох жанрах, таких як шутери, стратегії, MOBA і багато інших [19].

Мета матчмейкінгу полягає в тому, щоб скласти рівні та вигідні команди для кожного гравця. Це дозволяє створити балансовану гру, де кожен учасник має реальні шанси на перемогу, і де відбувається стимулююча конкуренція.

Unity Matchmaker надає функціональність автоматичного знаходження та об'єднання гравців у групи або команди. Він враховує різні фактори, такі як рівень навичок, ранг, преференції та доступність гравців, щоб забезпечити балансовані ігрові сесії [19].

4.3.3 Хостинг сервера на Unity Multiplay

Unity Game Server Hosting, відомий також як Multiplay, є сервісом, створеним Unity Technologies, який надає розробникам ігор можливість розмістити та керувати своїми ігровими серверами в хмарному середовищі. Цей сервіс дозволяє

легко масштабувати та керувати онлайн-ігровими серверами, забезпечуючи стабільну та високоякісну гру для гравців [20].

Multiplay використовує хмарну інфраструктуру, що дозволяє розробникам легко розмістити свої ігрові сервери без необхідності власної апаратної інфраструктури. Це забезпечує гнучкість, масштабованість та надійність роботи серверів.

Multiplay надає розробникам можливість контролювати та аналізувати працездатність своїх серверів. Вони можуть отримувати статистику щодо навантаження серверів, затримки, продуктивності та інших параметрів, що допомагає виявляти проблеми та вдосконалювати ігровий досвід.

Multiplay забезпечує можливість резервного копіювання та відновлення ігрових даних. Розробники можуть забезпечити безпеку та збереження прогресу гравців, уникнути втрати даних та забезпечити швидке відновлення після можливих збоїв.

Також, підтримуютья різні платформи, включаючи ПК, консолі та мобільні пристрої. Розробники можуть розміщувати свої сервери та надавати доступ до гри для гравців з різних пристроїв.

Висновки до розділу

В цьому розділі було підведено підсумки розробки. Оцінено ступінь готовності розроблюваної онлайн-гри.

Було наведено керівництво по використанню застосунку користувачем. Детально описано повний цикл використання застосунку, а саме: дії в головному меню, початок гри, ігрові дії в матчі, повернення в головне меню. Також, наведено використання всіх розроблених ігрових механік з точки зору користувача (гравця). Всі описані дії з використання застосунку в достатній мірі супроводжувалися іллюстраціями (скріншотами).

Далі було проведено перевірку авторитетності сервера в обчисленнях ігрової логіки. Було імітовано найімовірніші впливи чит-програм з клієнтської сторони:

спроби змінити позиції юнітів, модифікувати їх показники. Було перевірено відсутність наявності на клієнтській стороні «дійсних» даних, до яких клієнт не повинен мати доступу. Було переконливо доведено, що ключова ігрова логіка обчислюється виключно на сервері, шляхом виводу відповідних повідомлень в консоль. Всі перевірки успішно пройдені в повному обсязі, результати внесено у відповідні таблиці.

На останок було оцінено мабутнє проєкту. Визначено підсистеми, яких не вистачає грі на даний момент. А головне – розглянуті сервіси Unity Game Services, використання яких дозволить зробити онлайн-гру доступною для гравців з усього світу. Вони включатимуть засоби автентифікації, хмарний хостинг серверів та сервіс підбору матчів.

ВИСНОВКИ

В пояснювальній записці було оглянуто існуючі технології реалізації онлайн-ігор, досліджено різні моделі мережевого з’вязку, доведено необхідність використання Server-Authoritative архітектури в змагальних онлайн-іграх, детально описано реалізацію компонентів гри, проведено випробування системи, оцінено перспективи подальшої розробки.

В першому розділі було розглянуто існуючі моделі мережевого з’вязку в контексті онлайн-ігор, виділено їх переваги і недоліки як з точки зору як розробника, так і гравця. Виділені особливоті підтверджені на прикладі декількох онлайн-ігор. Досліджено підхід авторитетності сервера.

В другому розділі сформовано чіткі критерії для розроблюваної онлайн-гри. З урахуванням цих критеріїв було розроблено дизайн гри. Було обрано та досліджено технічні засоби для реалізації, що включили: ігровий рушій, мережеву бібліотеку, мову програмування, середовище розробки, систему контролю версій.

В третьому розділі була досліджена концепція мережевої ігрової сцени, описано механізм Remote Procedure Call (RPC). Детально описано розробку основних ігрових механік, що представляють ігровий процес.

В останньому розіділі наведено керівництво користувача, а також проведено ряд перевірок, що підтверджують відповідність проєкту визначеним раніше критеріям. На останок, оцінено перспективи проєкту.

Розроблена в даному проєкті онлайн-гра є повноцінною змагальною грою, із високим ступенем захисту від ігрового шахрайства. В поточному стані грати можна у локальній мережі, за продовженої розробки можуть бути інтегровані сервіси Unity Game Services, що зробить гру доступною гравцям з усього світу.

ПЕРЕЛІК ДЖЕРЕЛ

1. "Cheating". URL: https://www.dictionary.com/browse/cheating (дата звернення 24.05.23)
2. Snajay Madhav, Josh Glazer Multiplayer Game Programming: Architecting Networked Games. New York: Addison-Wesley Professional, 2015 – 847 p.
3. Unity Netcode for GameObjects. URL: https://docs-multiplayer.unity3d.com/netcode/current/terms-concepts/network-topologies (дата звернення 24.05.23)
4. Photon Engine Authoritative Server. URL: https://doc.photonengine.com/bolt/current/troubleshooting/authoritative-server-faq (дата звернення 24.05.23)
5. How to Play Minecraft Multiplayer. URL: https://help.minecraft.net/hc/en-us/articles/4410316619533-How-to-Play-Minecraft-Multiplayer (дата звернення 25.05.23)
6. Andrew Groen Empires of Eve: A History of the Great Wars of Eve. Lightburn Industries, 2016 – 188 p.
7. MMO Game. URL: https://www.dictionary.com/browse/mmo (дата звернення 25.05.23)
8. A basic analysis of Clash Royale multiplayer solution. URL: https://blog.gemserk.com/2016/09/05/analyzing-clash-royale-multiplayer-solution/ (дата звернення 25.05.23)
9. GTA Online guide with everything you need for your criminal empire. URL: https://www.gamesradar.com/gta-online-guide/ (дата звернення 25.05.23)
10. MOBA Game. URL: https://www.dictionary.com/browse/moba (дата звернення 29.05.23)
11. Unity Engine Documentation. URL: https://docs.unity3d.com/Manual/index.html (дата звернення 01.06.23)
12. Joseph Hocking Unity in Action: Multiplatform Game Development in C#. Shelter Island, NY: Manning Publications; 1st edition, 2015 – 352 p.

13. C# Documentation. URL: https://learn.microsoft.com/en-us/dotnet/csharp/ (дата звернення 02.06.23)
14. Visual Studio Code. URL: https://code.visualstudio.com/ (дата звернення 02.06.23)
15. Source Engine Multiplayer Networking. URL: https://developer.valvesoftware.com/wiki/Source_Multiplayer_Networking (дата звернення 03.06.23)
16. Remote Procedure Call (RPC). URL: https://www.techtarget.com/searchapparchitecture/definition/Remote-Procedure-Call-RPC (дата звернення 04.06.23)
17. Richard Rouse Game Design: Theory & Practice. Plano, TX: Wordware Publishing, 2004 — 698 p.
18. Unity Authentication. URL: https://docs.unity.com/authentication/en-us/manual/intro-unity-authentication (дата звернення 06.06.23)
19. Unity Matchmaker. URL: https://docs.unity.com/matchmaker/en/manual/matchmaker-overview (дата звернення 06.06.23)
20. Unity Game Server Hosting (Multiplay). URL: https://docs.unity.com/game-server-hosting/en/manual/welcome (дата звернення 07.06.23)

ДОДАТОК А

Посилання на репозиторій проєкту:

https://github.com/AndrewChikrii/Diplom

(/Assets/Scripts – шлях до директорії із програмним кодом)

ДОДАТОК Б

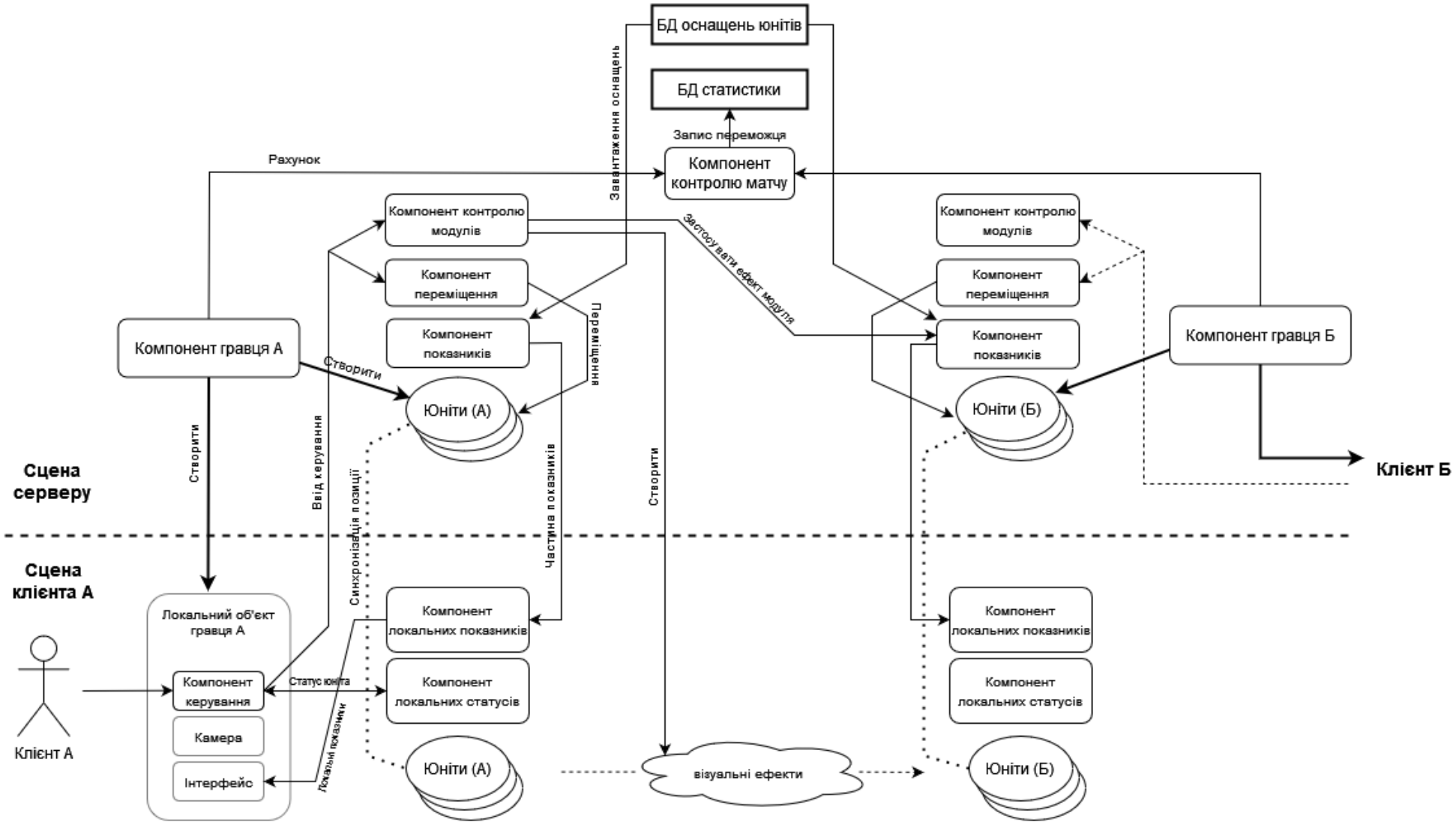

Рисунок Б.1 – Структура ігрової сцени