Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ГРИ-ШУТЕРА З ВИКОРИСТАННЯМ UNREAL ENGINE 4

DEVELOPMENT OF A SHOOTER GAME USING UNREAL ENGINE 4

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
Групи ІПЗ-42
Яницький Сергій Андрійович

Керівник:
Повстяна Юлія Славомирівна

Кваліфікаційну роботу
допущено до захисту
«10» 06 2025 р.
к.т.н., доцент
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна
_____
(підпис)

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: *«Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«20» 12 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Яницькому Сергію Андрійовичу

1. Тема кваліфікаційної роботи: *«Розробка гри-шутера з використанням Unreal Engine 4»*
Керівник роботи: *к.т.н., доцент, Повстяна Юлія Славомирівна*
затверджені наказом закладу вищої освіти від «*19» грудня 2024 р.* № *474/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи *«10» червня 2025 р.*
3. Вихідні дані до роботи: *технічне та програмне забезпечення ЕОМ, вимоги до розробки програмного забезпечення, методичні вказівки до виконання кваліфікаційної роботи бакалавра.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення, опис функціонального наповнення об'єкта проектування, практична реалізація об'єкта проектування.*

5. Перелік графічного матеріалу: *14 рисунків*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Повстяна Ю. С.*		
Специфікації вимог до розробленої системи	*Повстяна Ю. С.*		
Розробка об'єкта проєктування	*Повстяна Ю. С.*		
Нормоконтроль	*Вознюк А. В.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	2,35 %		
Академічна доброчесність	*Повстяна Ю. С.*		

7. Дата видачі завдання *«20» грудня 2024 р.*

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	*до 04.02.2025 р.*	вик
2	Аналіз проблеми розробки та впровадження об'єкту проектування	*до 01.03.2025 р.*	вик
3	Обґрунтування вибору шляхів, технологій (алгоритмів) і засобів вирішення поставленого завдання	*до 15.03.2025 р.*	вик
4	Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних	*до 29.03.2025 р.*	вик
5	Практична реалізація об'єкта проектування та розробка бази даних	*до 26.04.2025 р.*	вик
6	Тестування та налагодження об'єкта проектування	*до 03.05.2025 р.*	вик
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру	*до 10.06.2025 р.*	вик

Здобувач вищої освіти _____ Сергій ЯНИЦЬКИЙ

Керівник кваліфікаційної роботи _____ Юлія ПОВСТЯНА

АНОТАЦІЯ

Яницький С. А. Розробка гри-шутера з використанням Unreal Engine 4. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел.

У першому розділі проведено аналіз сучасного стану відеоігрової індустрії, огляд рушіїв для створення ігор, зокрема Unreal Engine 4, і сформульовано технічне завдання для розробки гри-шутера. У другому розділі визначено вимоги до функціональності майбутньої гри, здійснено проектування архітектури гри, побудовано UML-діаграми, обґрунтовано вибір технологій та інструментів. У третьому розділі розроблено прототип гри-шутера за допомогою Unreal Engine 4 із використанням C++ і Blueprints: реалізовано управління персонажем, стрільбу, штучний інтелект супротивників, HUD та інші елементи геймплею. Здійснено тестування та оптимізацію гри. Результатом стала функціональна версія гри з базовим ігровим процесом.

Ключові слова: шутер, Unreal Engine 4, Blueprints, C++, геймплей, AI, HUD.

ABSTRACT

Yanitsky S. Development of a Shooter Game Using Unreal Engine 4. Manuscript. Qualification work of the bachelor's program "Software engineering" in the specialty "Software engineering". Lutsk National Technical University. Lutsk, 2025.

The bachelor's qualification thesis consists of an introduction, three chapters, conclusions and a list of references.

The first chapter analyzes the current state of the video game industry, reviews modern game engines with a focus on Unreal Engine 4, and formulates the technical requirements for developing a shooter game. The second chapter defines the functional requirements, designs the game's architecture, includes UML diagrams, and justifies the selection of technologies and tools. The third chapter presents the development of a prototype shooter game using Unreal Engine 4 with C++ and Blueprints: implementing player control, shooting mechanics, enemy AI, HUD, and other gameplay elements. Testing and optimization were carried out. The result is a functional game version with core gameplay features.

Keywords: shooter, Unreal Engine 4, Blueprints, C++, gameplay, AI, HUD.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі індустрія розробки відеоігор посідає одне з провідних місць серед напрямів цифрових технологій, що стрімко розвиваються. Ігри стали не лише формою розваги, а й важливим елементом культури, мистецтва та інженерії. Відеоігри охоплюють широкий спектр жанрів і тематик, пропонуючи гравцям інтерактивні історії, виклики, симуляції й соціальні платформи. З-поміж усіх жанрів особливу популярність мають шутери від першої або третьої особи, що забезпечують динамічний ігровий процес, високий рівень залучення гравців і гнучкість у реалізації механік. Створення таких ігор потребує глибоких знань в області програмування, дизайну, фізики, графіки та обробки подій у реальному часі.

Одним із найпотужніших інструментів для створення тривимірних ігор, зокрема шутерів, є ігровий рушій Unreal Engine 4 (UE4), розроблений компанією Epic Games. UE4 забезпечує розробників широким спектром функціональних можливостей, включно з фізичним рендерингом, підтримкою віртуальної реальності, інтуїтивним візуальним середовищем Blueprints, а також можливістю створення високоякісної графіки та розгалуженої логіки без надмірного навантаження на апаратне забезпечення. Враховуючи зростання попиту на сучасні, інтерактивні ігрові продукти, тема розробки гри-шутера з використанням Unreal Engine 4 є актуальною як з інженерної, так і з навчально-практичної точки зору.

Метою даної кваліфікаційної роботи бакалавра є розробка комп’ютерної гри в жанрі шутер з використанням рушія Unreal Engine 4, яка дозволяє користувачу взаємодіяти з ігровим світом у режимі реального часу, використовуючи сучасні принципи ігрового дизайну та архітектури програмного забезпечення. Розроблена гра повинна включати базові механіки переміщення персонажа, стрільби, взаємодії з об’єктами навколишнього середовища.

Об’єктом кваліфікаційної роботи бакалавра є процес розробки інтерактивних програмних продуктів у вигляді тривимірних відеоігор із використанням сучасних ігрових рушіїв.

Предметом кваліфікаційної роботи бакалавра є технічні, програмні та дизайнерські аспекти створення ігрового продукту типу шутер на базі рушія Unreal Engine 4, включаючи архітектуру проекту, структуру ігрових механік, оптимізацію та інтерфейс користувача.

Для досягнення поставленої мети були сформульовані такі завдання:

– проаналізувати сучасні тенденції в розробці відеоігор жанру шутер, визначити типові механіки та вимоги до таких ігор;

– дослідити функціональні можливості рушія Unreal Engine 4, інструменти програмування (Blueprints і C++), а також засоби реалізації 3D-графіки;

– розробити концепцію гри-шутера, включаючи сюжет, візуальний стиль, карту ігрового світу, механіки управління та систему озброєння;

– спроектувати архітектуру гри-шутера на базі шаблону «гравець-контролер-ігровий режим» із використанням UML-діаграм;

– реалізувати основні ігрові механіки (переміщення, стрільба, AI супротивника, HUD) із використанням Unreal Engine 4;

– виконати тестування, налагодження та оптимізацію продуктивності гри з урахуванням цільових системних вимог.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

Відеоігрова індустрія – одна з найбільш швидко розвиваються і динамічних галузей сучасної економіки. Від моменту виникнення першої комп'ютерної гри до теперішнього часу технології розробки відеоігор значно змінилися, дозволяючи створювати геймерські світи з дивовижною графікою, реалістичною фізикою та складними механіками. Всі ці досягнення стали можливими завдяки інноваційним технологіям, що постійно вдосконалюються. У цьому розділі розглядаються основні технології, які використовуються при створенні відеоігор, та їхній вплив на розвиток індустрії.

У контексті розробки тривимірних комп'ютерних ігор у жанрі шутер особливої уваги заслуговують результати сучасних фахових досліджень, що розглядають як технологічні, так і проектно-архітектурні аспекти створення подібного програмного забезпечення. Нижче представлено огляд трьох актуальних наукових публікацій, які безпосередньо стосуються даної теми.

У статті [1] детально описують процес створення шутера від етапу ініціалізації проєкту до реалізації ігрових механік. Особливу увагу приділено використанню Blueprints для створення логіки гри, налаштуванню геймплейних компонентів, а також роботі з матеріалами та освітленням. Хоча дослідження базується на Unreal Engine 5, більшість підходів та інструментів є релевантними для Unreal Engine 4, зокрема в аспектах побудови сцени, налаштування персонажів та інтеграції ігрової логіки. Ця робота є цінною для розуміння практичних аспектів розробки шутера на рушії Unreal.

У статті [2] представлено комплексний підхід до оптимізації продуктивності ігор, розроблених на Unreal Engine 4. Автори розглядають інструменти для профілювання та аналізу продуктивності, такі як використання Level of Detail (LOD), оптимізація частинок, зменшення кількості

емісійних джерел світла та повторне використання еміттерів. Ці рекомендації спрямовані на покращення частоти кадрів та загальної стабільності гри, що є критично важливим для забезпечення комфортного геймплею, особливо на системах із середніми характеристиками. Ця публікація є корисною для розробників, які прагнуть підвищити ефективність своїх проєктів на Unreal Engine 4.

У дослідженні [3] представлено інструмент EvolvingBehavior, який дозволяє еволюціонувати поведінкові дерева для неігрових персонажів (NPC) у Unreal Engine 4 за допомогою генетичного програмування. Автори порівнюють ефективність згенерованих дерев із ручно створеними, демонструючи потенціал автоматизованого підходу до розробки AI. Це дослідження підкреслює можливості креативного дизайну та адаптивної поведінки NPC, що є актуальним для розробників, які прагнуть створити більш реалістичну та динамічну ігрову поведінку.

Однією з ключових складових сучасних відеоігор є ігрові рушії – програмні платформи, які надають розробникам інструменти для створення графіки, фізики, звуку та інтерфейсу гри. Вони дозволяють значно спростити процес розробки та скоротити час, необхідний для створення гри. Найпопулярнішими рушіями є:

– Unreal Engine 4 (UE4) – це один з найпотужніших рушіїв, який дозволяє створювати високоякісні 3D-ігри з реалістичною графікою та фізикою (рис. 1.1). UE4 підтримує програмування на C++ та візуальне програмування через Blueprints, що дозволяє навіть тим, хто не має досвіду в програмуванні, створювати складні ігрові механіки. Цей рушій широко використовується для розробки шутерів, стратегій, рольових ігор та навіть віртуальної реальності. Завдяки своїй відкритій архітектурі, Unreal Engine 4 надає розробникам гнучкі можливості для модифікації рушія відповідно до потреб проєкту. Крім того, UE4 має потужну спільноту користувачів та велику кількість навчальних матеріалів, що значно полегшує процес освоєння платформи;

Рисунок 1.1 – Логотип Unreal Engine 4 [4]

– Unity (рис. 1.2) – ще один популярний рушій, що має велику підтримку інді-розробників завдяки своїй доступності та простоті використання. Він дозволяє створювати як 2D, так і 3D-ігри і підтримує численні платформи – від мобільних пристроїв до консолей та ПК. Unity також має великий набір інструментів для створення фізики, анімацій, штучного інтелекту та мережевих функцій;

Рисунок 1.2 – Логотип Unity [5]

– CryEngine (рис. 1.3) – цей рушій вважається одним із найпотужніших для створення графічно складних ігор з відкритими світом і великою кількістю динамічних елементів. CryEngine використовується для створення великих проектів, таких як Far Cry або Crysis. CryEngine забезпечує високий рівень деталізації навколишнього середовища, реалістичне освітлення та потужну систему частинок. Рушій також підтримує складну фізику, штучний інтелект та мультиплатформену розробку, що робить його привабливим для студій AAA-рівня.

Рисунок 1.3 – Логотип CryEngine [6]

Ігрові рушії також підтримують інструменти для роботи з фізичними симуляціями, що дозволяє розробникам досягати максимальної реалістичності. Системи, такі як Havok Physics або Chaos Physics, реалізують фізику, яка враховує рух об’єктів, взаємодію з навколишнім середовищем, руйнування та багато інших аспектів.

Графіка є одним із основних аспектів, що визначають успіх гри, і технології візуалізації значно впливають на досвід користувачів. З розвитком графічних процесорів (GPU) стали доступними нові технології, які дозволяють створювати вражаючі зображення з високим рівнем деталізації.

Однією з найбільших революцій у графіці відеоігор є використання технології трасування променів (ray tracing). Ця технологія дозволяє створювати реалістичне освітлення, тіні та відблиски. Завдяки трасуванню променів можна досягти надзвичайно реалістичних ефектів, що робить ігри більш захоплюючими і природними. Наприклад, в Cyberpunk 2077 ця технологія використовувалася для створення вражаючої графіки у нічних містах, де кожен промінь світла має своє відображення на поверхнях.

Фотореалістичні текстури (PBR – Physically-Based Rendering) – технологія фізично обґрунтованого рендерингу (PBR) стала стандартом для створення текстур та матеріалів. Вона дозволяє створювати більш реалістичні поверхні, враховуючи, як світло взаємодіє з матеріалами. Завдяки цьому можна

досягти ефектів, таких як дзеркальні відображення, матовість, вигоряння і багато інших.

Віртуальна реальність (VR) і доповнена реальність (AR) у відеоіграх все більше переходять у сферу віртуальної реальності, де гравці можуть не тільки бачити, але й взаємодіяти з ігровим світом у реальному часі через спеціальні гарнітури. VR дозволяє повністю зануритися в ігрову реальність, а AR – інтегрувати елементи гри в реальний світ за допомогою камери та екрану пристрою (як це реалізовано у таких іграх, як Pokémon Go).

Штучний інтелект (ШІ) займає важливу роль у створенні складних ігрових механік, зокрема в іграх, де взаємодія з неігровими персонажами (NPC) є важливою частиною гри. Завдяки ШІ ігри можуть пропонувати більш адаптивні і реалістичні сценарії.

У багатьох сучасних іграх вороги використовують складні алгоритми для того, щоб не бути передбачуваними. Вони можуть адаптувати свої стратегії в залежності від стилю гри гравця. У шутерах, таких як Halo або Call of Duty, штучний інтелект ворогів здатний на тактичні дії, наприклад, при укритті або взаємодії з іншими персонажами.

Розвиток машинного навчання дозволяє розробникам створювати системи, які можуть адаптуватися до дій гравця, покращуючи досвід. Наприклад, це може стосуватися рекомендацій ігрових механік або адаптації складності гри.

Мережеві технології стали основою для створення багатокористувацьких ігор, де гравці можуть змагатися або співпрацювати в реальному часі. У таких іграх важливими є стабільність з’єднання, синхронізація даних та низька затримка.

Для великих багатокористувацьких ігор необхідні потужні сервери, які обробляють величезний обсяг даних. Ігрові студії використовують різні технології для масштабування серверної інфраструктури, включаючи хмарні сервіси, що дозволяє обробляти тисячі одночасних з’єднань без втрати продуктивності.

Важливою складовою мережевих ігор є швидка та точна синхронізація даних, що забезпечує відсутність лагів та затримок, що критично для шутерів і інших ігор з динамічним ігровим процесом.

Завдяки технологіям просторового звуку та динамічного аудіо можна значно покращити занурення в ігровий процес. Використання технологій, таких як Dolby Atmos або 3D Audio, дозволяє гравцям почути звуки, які змінюються в залежності від їхнього місця у віртуальному світі.

Розробка тривимірних шутерів є однією з найбільш динамічно розвиваючихся галузей у відеоігровій індустрії. Завдяки технологічним досягненням, таким як високоякісна графіка, реалістична фізика та інтерактивний геймплей, цей жанр залишається популярним серед гравців та розробників. Одним із провідних інструментів для створення таких ігор є ігровий рушій Unreal Engine 4 (UE4), який надає потужні можливості для реалізації складних ігрових механік та візуальних ефектів.

Важливою частиною аналізу для розробки нової тривимірної комп’ютерної гри в жанрі шутер є розгляд існуючих аналогів, що вже стали частиною відеоігрової культури. Цей аналіз дозволяє виявити сильні сторони конкурентів, а також виявити недоліки та потенційні можливості для вдосконалення. У даному розділі буде розглянуто кілька популярних тривимірних шутерів, що використовують рушій Unreal Engine 4, а також окремо відзначимо їхні особливості та потенційні проблеми, з якими стикаються розробники.

Fortnite (рис. 1.4) є однією з найбільш відомих багатокористувацьких онлайн-ігор, що поєднує елементи шутера та будівництва. Гра має великий попит завдяки простому, але захоплюючому ігровому процесу, який дозволяє гравцям взаємодіяти з навколишнім світом у режимі реального часу. Особливість Fortnite полягає в її механіці будівництва: гравці можуть створювати укриття, конструкції та навіть пастки для своїх супротивників.

Перевагами Fortnite є те, що ігровий процес з акцентом на творчість «Механіка будівництва» дозволяє гравцям виражати свою креативність і

адаптувати стратегію під умови гри. Багатокористувацький режим гри підтримує великі сесії з великою кількістю гравців, що забезпечує динамічну атмосферу. Регулярні сезонні оновлення дозволяють тримати гравців у напрузі і підтримувати їхній інтерес до гри.

Недоліками є те, що ігровий процес часто критикують за перевантаження механік – поєднання шутера і будівництва може стати складним для новачків. Fortnite широко використовує систему мікротранзакцій для продажу косметичних предметів, що може негативно сприйматися гравцями, особливо в контексті «pay-to-win». Для того, щоб мати успіх у грі, необхідно володіти високими навичками будівництва, що може стати бар'єром для нових гравців.

Рисунок 1.4 – Гра Fortnite (Battle royale) [7]

Apex Legends – це безкоштовна королівська битва, яка використовує рушій Unreal Engine 4 і акцентує увагу на персонажах з унікальними здібностями (рис. 1.5). Гра має велику популярність завдяки швидкому темпу, тактичному геймплею та високій динаміці. Особливістю Apex Legends є її «підкласова» система, коли кожен персонаж має свої індивідуальні здібності, що змінюють стиль гри.

Рисунок 1.5 – Гра «Apex Legends» [8]

Перевагами Apex Legends є вибір персонажів з унікальними здібностями: Це дозволяє гравцям вибирати персонажів, що відповідають їхньому стилю гри. Кожен персонаж має власну унікальну роль і здібності, що дозволяє різноманітність тактичних підходів.

Важливу роль у грі відіграє співпраця з командою. Це допомагає підтримувати інтерактивність і мотивує гравців до командної роботи. Гра пропонує швидкий, але в той же час виважений і тактичний ігровий процес, що дозволяє швидко реагувати на зміни в обстановці.

Недоліками є те, що хоча гра має вражаючі графічні ефекти, деякі гравці повідомляли про проблеми з продуктивністю на менш потужних комп'ютерах або консолях. Так як гра має велику кількість різних видів зброї, вони не завжди збалансовані, що іноді може призводити до нерівних поєдинків. Як і у випадку з Fortnite, конкуренція в Apex Legends може бути дуже жорсткою, що створює бар'єри для новачків, які не можуть досягти успіху через досвідченіших супротивників.

Крім того, регулярні оновлення та зміни в ігровому балансі можуть викликати незадоволення серед частини гравців, особливо тих, хто звик до певного стилю гри. Також іноді виникають проблеми з підбором гравців у матчах, що може призводити до нерівних команд і впливати на загальне враження від гри.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Для досягнення поставленої мети були сформульовані такі завдання:

– дослідити функціональні можливості рушія Unreal Engine 4, інструменти програмування (Blueprints і C++), а також засоби реалізації 3D-графіки;

– розробити концепцію гри-шутера, включаючи сюжет, візуальний стиль, карту ігрового світу, механіки управління та систему озброєння;

– спроектувати архітектуру гри-шутера на базі шаблону «гравець-контролер-ігровий режим» із використанням UML-діаграм;

– реалізувати основні ігрові механіки (переміщення, стрільба, AI супротивника, HUD) із використанням Unreal Engine 4;

– виконати тестування, налагодження та оптимізацію продуктивності гри з урахуванням цільових системних вимог.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

У даному розділі розглядаються основні вимоги до програмного забезпечення, яке розробляється для проекту. Оскільки метою цієї роботи є створення комп'ютерної гри в жанрі шутер за допомогою рушія Unreal Engine 4, всі вимоги повинні враховувати специфіку ігрового процесу, архітектуру програмного забезпечення, взаємодію з користувачем та технічні аспекти, що дозволяють забезпечити високий рівень геймплейного досвіду.

Для моделювання програмної системи вибрано об'єктно-орієнтоване моделювання (ООМ), яке є ефективним підходом для розробки складних програмних продуктів, таких як відеоігри. Об'єктно-орієнтований підхід дозволяє структурувати програмний код, чітко розподіляти обов'язки між класами та компонентами системи, що робить проект масштабованим і зручним для подальшої розробки та вдосконалення. Цей підхід також дає змогу без зайвих труднощів розширювати гру новими функціями чи елементами без необхідності переписувати значну частину коду.

Моделювання елементів програмного забезпечення буде виконуватися за допомогою нотації UML (Unified Modeling Language), яка є стандартом для графічного зображення архітектури програмних систем. UML дозволяє зручно відобразити структуру класів, їх атрибути та операції, а також зв'язки між класами, що значно полегшує процес проектування і подальшої реалізації системи [8].

Основні вимоги до програмного забезпечення, що розробляється, зводяться до таких основних функціональних аспектів [9, 10]:

– графіка та анімація – необхідно забезпечити високу якість тривимірних моделей, а також ефективне управління анімацією персонажів і навколишнього

середовища. Механізм змін у навколишньому середовищі, а також освітлення повинні адаптуватися до дій гравця для створення динамічної атмосфери;

– інтерфейс користувача – ігровий інтерфейс повинен бути інтуїтивно зрозумілим, щоб гравець міг швидко освоїти управління персонажем, вибір зброї, налаштування гри, а також взаємодіяти з іншими гравцями в багатокористувацькому режимі. Окремо слід розробити механізми для зміни інтерфейсу в залежності від налаштувань гри;

– механіка геймплею – реалізація механік стрільби, руху, застосування спеціальних здібностей персонажа, а також взаємодії з навколишнім світом є одними з ключових складових успіху гри. Важливо, щоб механіка була динамічною і адаптувалася до вибору гравця;

– мережеві функції – для багатокористувацького режиму потрібно реалізувати стабільне підключення за допомогою протоколу TCP/IP, яке забезпечить синхронізацію даних між гравцями, зокрема в реальному часі, а також дозволить проводити онлайн-сесії без затримок;

– безпека та захист даних – для захисту персональних даних гравців буде реалізовано шифрування та заходи щодо запобігання несанкціонованому доступу. Також необхідно забезпечити захист від потенційних атак на сервери гри та підтримку цілісності даних користувачів.

UML-діаграма класів для проєкту 3D-гри-шутера на Unreal Engine 4 представлено на рисунку 2.1.

На представленій UML-діаграмі зображено структуру класів, що реалізують ключові компоненти тривимірної гри-шутера, створеної за допомогою Unreal Engine 4. У центрі архітектури знаходиться клас Player, який представляє ігрового персонажа – має атрибути, пов’язані зі станом здоров’я, позицією, боєприпасами, очками та ін. Клас Weapon пов’язаний із Player та відповідає за параметри зброї: шкоду, кількість набоїв, швидкість стрільби та дальність. Weapon має асоціацію з класом Bullet, що описує характеристики кулі (швидкість, напрямок, час життя) та моделює її політ і взаємодію.

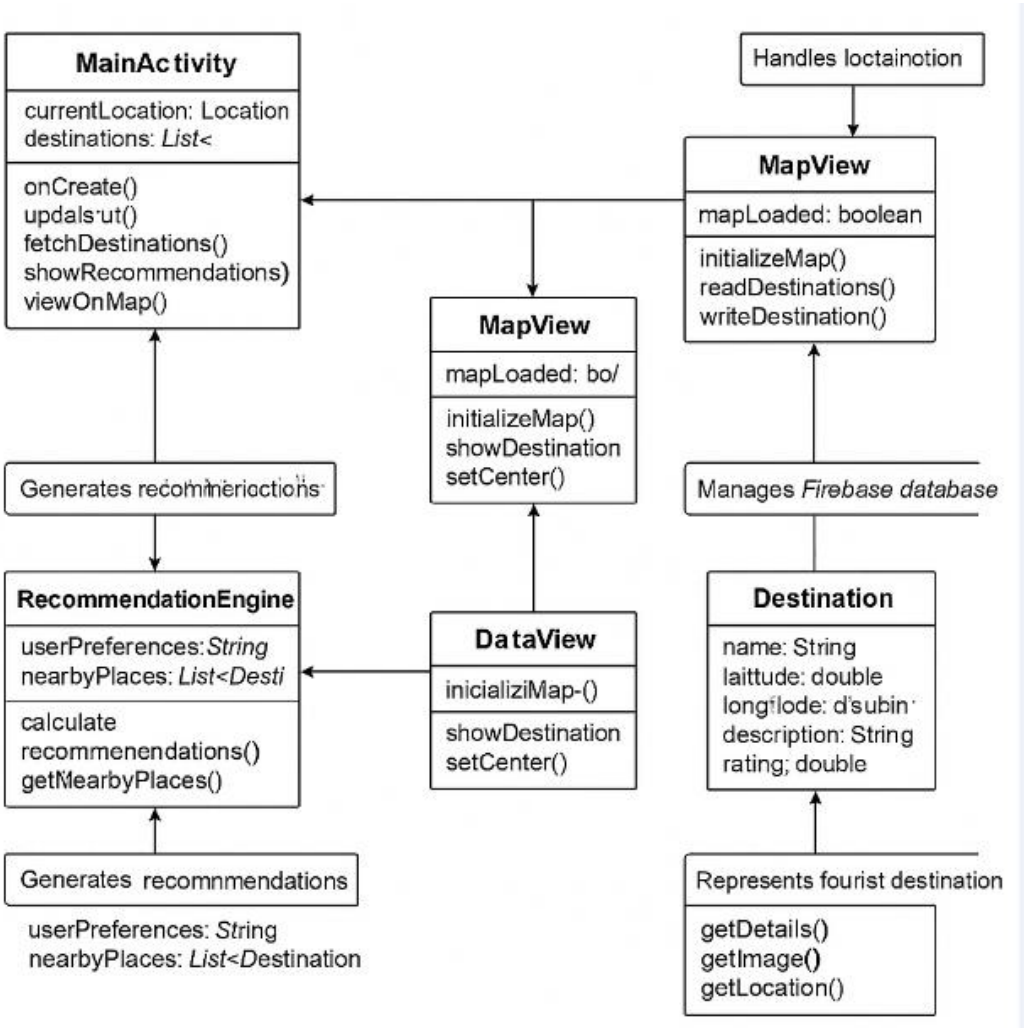

Рисунок 2.1 – UML-діаграма класів для проєкту 3D-гри-шутера

Також на діаграмі присутні класи Enemy, GameController, HUD та Environment. Enemy реалізує супротивників з параметрами здоров’я та поведінковими методами, що використовують штучний інтелект. GameController керує логікою гри, відстежує стани та запускає відповідні сценарії. Клас HUD відповідає за відображення інформації на екрані гравця, наприклад, здоров’я чи кількість боєприпасів, а Environment описує об’єкти навколишнього середовища та їхню взаємодію з гравцем і супротивниками. Усі класи взаємодіють через чітко визначені зв’язки, що дозволяє масштабувати проєкт, додаючи нові функції без порушення загальної архітектури.

Рисунок 2.2 демонструє UML-діаграму прецедентів, де відображено основні сценарії взаємодії користувачів з системою гри-шутера, побудованої на Unreal Engine 4. Центральним актором є «Гравець», який виконує основні дії:

рухається, атакує, взаємодіє з об’єктами гри та використовує предмети. Ці дії охоплюють ключові геймплейні механіки, що забезпечують активну участь користувача в ігровому процесі. Окрім гравця, також присутній актор «Глядач», який має доступ лише до дії спостерігати за ігровим процесом, що може бути корисним у багатокористувацьких або змагальних режимах.

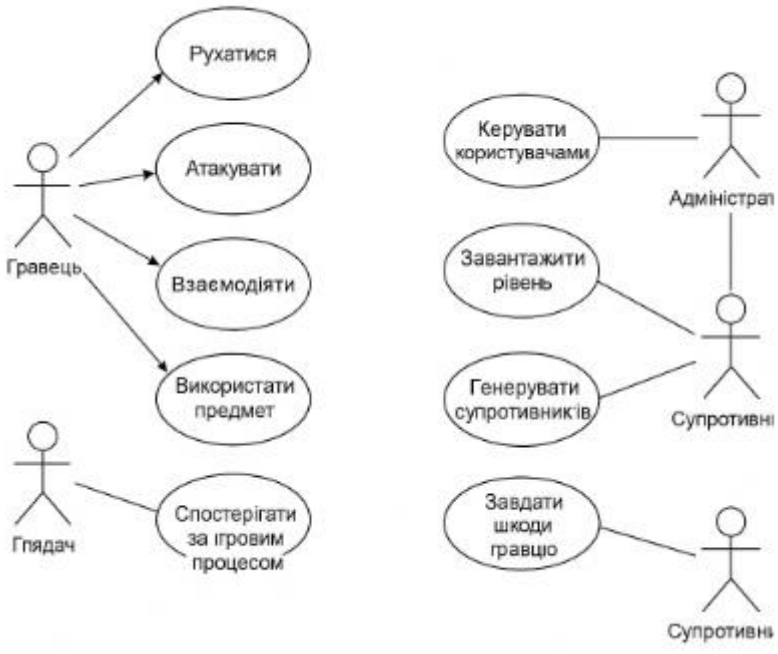

Рисунок 2.2 – UML-діаграма прецедентів для проєкту 3D-гри-шутера

Інші актори включають «Адміністратора», який має доступ до функції керувати користувачами, що передбачає створення, редагування чи блокування облікових записів. «Оточення» та «Супротивник» представляють системні компоненти, які автоматично ініціюють такі дії, як завантаження рівня, генерація супротивників та завдання шкоди гравцю. Ці взаємодії підкреслюють реалізацію елементів штучного інтелекту та автоматизованої поведінки у грі, що створює виклики для гравця й урізноманітнює ігровий процес. Діаграма демонструє узгоджену структуру ролей і функціоналу, необхідну для реалізації повноцінної інтерактивної системи гри.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

У цьому розділі проведемо огляд інструментів, методів та алгоритмів, що будуть використані для реалізації тривимірної комп’ютерної гри в жанрі шутер за допомогою рушія Unreal Engine 4. Вибір цих засобів базується на вимогах до високої якості графіки, інтерактивного геймплею, ефективності розробки та надійності програмного забезпечення.

Основним інструментом для розробки гри є рушій Unreal Engine 4, що забезпечує потужну платформу для створення тривимірних ігор з реалістичною графікою, фізикою та анімацією. Однією з основних переваг UE4 є його здатність забезпечувати високоякісне рендеринговане зображення та підтримку передових технологій, таких як трасування променів (Ray Tracing), фізично обґрунтований рендеринг (PBR), а також інтеграцію з сучасними графічними процесорами (GPU). Unreal Engine також дозволяє швидко прототипувати ігрові механіки завдяки візуальному програмуванню через систему Blueprints, що дозволяє створювати складні механізми без необхідності глибоких знань у програмуванні.

Особливістю UE4 є його підтримка роботи з фізичними симуляціями, що дозволяє розробляти реалістичну поведінку об'єктів і персонажів у реальному часі. Система фізики рушія включає підтримку для таких ефектів, як руйнування, взаємодія об’єктів, гравітація та інші, що важливо для жанру шутерів.

Для реалізації базових механік гри, таких як рух персонажа, стрільба, взаємодія з об’єктами та супротивниками, будуть використовуватися стандартні інструменти Unreal Engine 4, зокрема системи Character Movement Component для управління рухами персонажа та Aiming System для точності прицілювання. Стрільба буде реалізована через спеціалізовані компоненти для збереження боєзапасу, створення кулі та відстеження її траєкторії, а також

застосування звукових ефектів і візуальних ефектів для реалістичності пострілів.

Для реалізації штучного інтелекту супротивників будуть використані AI Controllers і Behavior Trees. Це дозволить створити динамічне та адаптивне поведінкове дерево для ворогів, що реагують на дії гравця, змінюють свою тактику в залежності від ситуації і можуть виявляти агресивну поведінку.

Інтерфейс користувача (HUD) буде розроблений з використанням вбудованих інструментів Unreal Engine для створення графічних елементів (віджетів). Це дозволить показувати гравцеві інформацію про стан здоров'я, боєзапас, поточні завдання та інші параметри. Для цього будуть використані системи UMG (Unreal Motion Graphics), які дозволяють швидко створювати інтерфейс за допомогою візуального редактора.

Для забезпечення складних і реалістичних реакцій супротивників на дії гравця, буде використано Behavior Trees – алгоритм, що дозволяє задавати складну поведінку NPC [11]. Алгоритм AI буде враховувати не лише прості умови, такі як стан здоров'я, наявність боєзапасу, але й складніші, як тактику пошуку укриттів, командні дії, спроби обходу і перехоплення гравця.

Також будуть застосовані методи Pathfinding для пошуку шляху через складне середовище, зокрема за допомогою NavMesh для створення навігаційних сіток. Це дозволить AI супротивників ефективно переміщатися по карті, уникати перепон і націлюватися на гравця.

Для досягнення високої продуктивності гри на різних конфігураціях комп'ютерів, будуть застосовуватися методи оптимізації, що дозволяють покращити швидкість обробки графіки та фізики, зокрема використання Level of Detail (LOD) для моделей, Occlusion Culling для виведення тільки видимих об'єктів, а також ефективні алгоритми для обробки колізій.

Окремо буде проведено тестування на різних апаратних платформах для виявлення потенційних «вузьких місць» у продуктивності, що дозволить налаштувати графічні параметри та забезпечити стабільну роботу гри навіть на середніх та низьких конфігураціях ПК.

Для налагодження та тестування гри будуть використані інструменти, вбудовані в Unreal Engine 4. Це включає систему Blueprint Debugger, яка дозволяє в реальному часі відслідковувати виконання коду і виявляти помилки, а також профайлери для моніторингу продуктивності. Будуть також використані Unit Tests та Automated Tests для перевірки стабільності основних функціональних частин гри, зокрема механік стрільби, поведінки AI та інтерфейсу користувача.

Процес розробки гри буде організовано за допомогою гнучкої методології Agile, з використанням циклів розробки (спринтів), що дозволяє швидко адаптуватися до змін і коригувати напрямок розробки на основі отриманих відгуків. Кожен спринт буде включати етапи планування, розробки, тестування та перегляду результатів. Використання гнучкої методології дозволить зберегти високий рівень контролю над проектом і ефективно реагувати на можливі труднощі.

Розробка буде здійснюватися за допомогою таких мов програмування та технологій:

– C++ для реалізації основних ігрових механік та оптимізації продуктивності;

– Blueprints для швидкої розробки та тестування ігрових функцій без необхідності писати великі обсяги коду, особливо для створення логіки поведінки персонажів та інтерфейсу;

– Unreal Engine 4 tools (NavMesh, Particle Systems, AI Controllers) для створення складних механік, таких як AI і навігація.

РОЗДІЛ 3
РОЗРОБКА ГРИ-ШУТЕРА З ВИКОРИСТАННЯМ UNREAL ENGINE 4

3.1 Практична реалізація об’єкта проектування

У цьому розділі розглянемо поетапне створення базового шутера в Unreal Engine 4, використовуючи C++ та Blueprints.

Початкове налаштування проєкту в Unreal Engine 4 розпочинається зі створення нового проєкту через Unreal Project Browser. На цьому етапі обирається шаблон «First Person», який уже містить базову реалізацію персонажа від першої особи з налаштованими компонентами – камерою, сіткою рук і прикріпленою зброєю (рис. 3.1). Такий вибір дозволяє швидко перейти до розробки шутера, оскільки основна логіка керування і стрільби вже реалізована у вигляді Blueprint, які при бажанні можна розширити або переписати на C++. У вікні створення також задається назва проєкту, його місце розташування на диску, тип графіки (Desktop/Console), якість рендерингу (Maximum Quality або Scalable), а також мова програмування (Blueprint або з підтримкою C++).

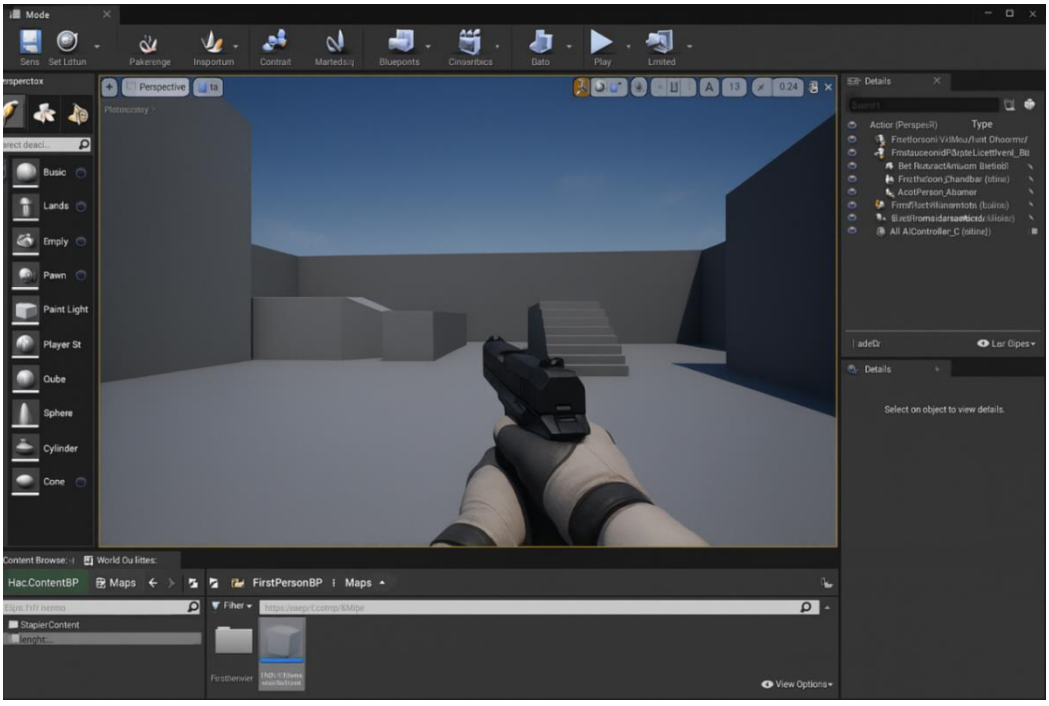

Рисунок 3.1 – Вигляд від першої особи з рушницею

На етапі налаштування гравця було використано шаблон First Person Character, який надає базову реалізацію керованого персонажа з виглядом від першої особи. Клас гравця AMyCharacter наслідується від ACharacter, що дозволяє скористатися стандартною системою пересування та анімацій у Unreal Engine 4. Основна логіка руху, повороту та стрибка реалізована мовою програмування C++, тоді як стрільба – через Blueprint, з викликом функції Fire(), визначеної у класі AMyCharacter. Це забезпечує зручне розділення логіки: обчислювально важливі процеси винесено до C++, а візуальні елементи – до Blueprint.

Компонування камери (FirstPersonCameraComponent), сітки рук (Mesh1P) та зброї здійснюється безпосередньо в редакторі, що формує вигляд гравця від першої особи. У методі SetupPlayerInputComponent() відбувається прив’язка введення клавіш до відповідних методів керування рухом і стрільбою. Логіка руху реалізована у функціях MoveForward() та MoveRight(), які відповідають за переміщення гравця відповідно до напрямку камери. Логіка стрільби в методі Fire() ініціює появу снаряду в певній точці простору перед гравцем, використовуючи визначений клас ProjectileClass. Детальну реалізацію цих методів подано у лістингу 3.1.

Лістинг 3.1 – Реалізація методів руху та стрільби в класі AMyCharacter

```
void AMyCharacter::SetupPlayerInputComponent(UInputComponent*
PlayerInputComponent)
{
    Super::SetupPlayerInputComponent(PlayerInputComponent);
    PlayerInputComponent->BindAxis("MoveForward", this,
&AMyCharacter::MoveForward);
    PlayerInputComponent->BindAxis("MoveRight", this,
&AMyCharacter::MoveRight);
    PlayerInputComponent->BindAction("Jump", IE_Pressed, this,
&ACharacter::Jump);
    PlayerInputComponent->BindAction("Fire", IE_Pressed, this,
&AMyCharacter::Fire);
}
void AMyCharacter::MoveForward(float Value)
{
    if (Value != 0.0f)
        AddMovementInput(GetActorForwardVector(), Value);
```

```
}
void AMyCharacter::MoveRight(float Value)
{
    if (Value != 0.0f)
        AddMovementInput(GetActorRightVector(), Value);
}
void AMyCharacter::Fire()
{
    // Blueprint реалізує spawn projectile, а C++ викликає функцію
    if (ProjectileClass)
    {
        FVector SpawnLocation = FirstPersonCameraComponent-
>GetComponentLocation() + FirstPersonCameraComponent->GetForwardVector()
* 100.0f;
        FRotator SpawnRotation = FirstPersonCameraComponent-
>GetComponentRotation();
        GetWorld()->SpawnActor<AMyProjectile>(ProjectileClass,
SpawnLocation, SpawnRotation);
    }
}
```

кінець лістингу 3.1

Після налаштування гравця наступним етапом є створення ігрового рівня (карти), який формує візуальне середовище для ігрового процесу. Для побудови сцени було використано інструменти Unreal Engine 4, які дозволяють швидко створювати примітивні структури за допомогою BSP-об’єктів (Binary Space Partitioning) або ж більш деталізованих статичних мешів. На початковому етапі побудова карти здійснювалася з використанням кубічних примітивів для формування стін, підлоги та інших елементів архітектури, що надалі можуть бути замінені на деталізовані 3D-моделі. Це забезпечує ефективне тестування логіки гри до фінального оформлення сцени [12].

До створеного простору було додано матеріали і текстури, які імітують бетон, метал чи інші поверхні, шляхом призначення стандартних або користувацьких матеріалів до відповідних об’єктів. Для коректного освітлення сцени використано джерела світла: Directional Light (основне джерело – сонце), Sky Light для наповнення м’яким світлом, а також Point Light для внутрішніх приміщень (рис. 3.2). Усі компоненти розміщено через панель «Place Actors», після чого освітлення було перебудовано через опцію Build Lighting, що

забезпечило реалістичну тінь і відображення в сцені. Наявність статичних та динамічних джерел світла дозволяє гравцеві краще орієнтуватися в просторі, а також надає візуальної глибини.

На етапі розробки реалізується стрілецька механіка гри шляхом створення окремих Blueprint-акторів для зброї (наприклад, пістолету) та снарядів (куль). Снаряд – це об’єкт, який спавниться при натисканні на клавішу стрільби. Йому задається швидкість руху, напрям, фізична колізія, а також логіка взаємодії при зіткненні з іншими об’єктами (наприклад, руйнування, частинки вибуху або зникнення).

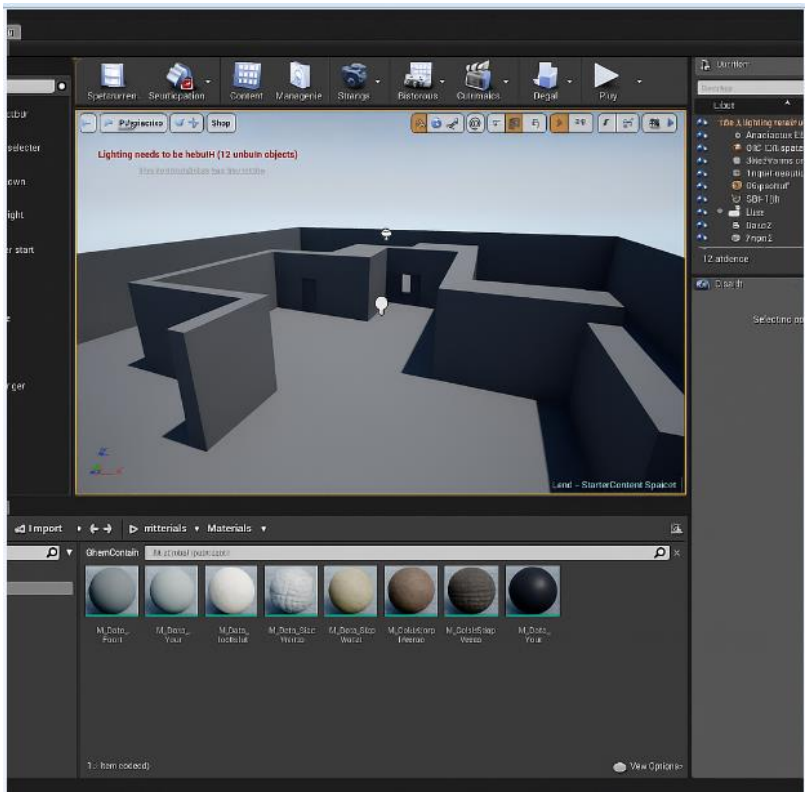

Рисунок 3.2 – Додавання матеріалів і текстури

Зброя, як правило, є дочірнім компонентом персонажа й прикріплена до руки або камери, що дозволяє забезпечити правильне положення та орієнтацію при стрільбі. У редакторі Unreal Engine це все реалізується з використанням Blueprint-логіки, а саме налаштування Projectile Movement, Collision (Sphere), Mesh, а також подій OnHit (рис. 3.3).

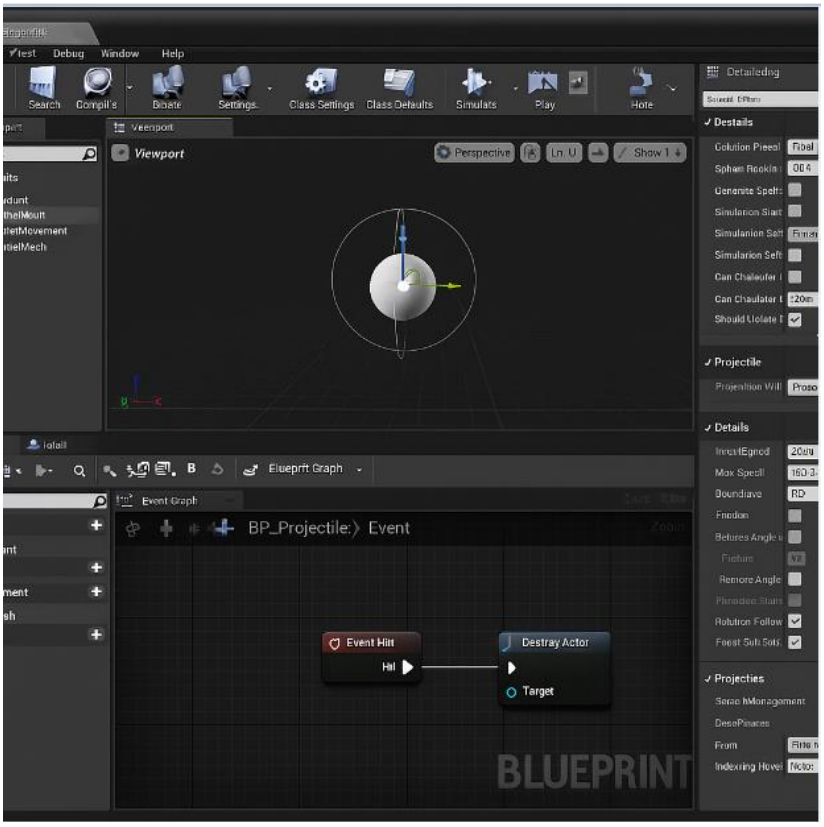

Рисунок 3.3 – Налаштування стрілецької механіки

У Blueprint-снаряді створюється подія Event Hit, яка викликається при зіткненні з іншим об’єктом. У ній реалізується логіка знищення кулі або нанесення шкоди іншому актору. Також налаштовується візуальне оформлення снаряда (матеріал, форма) та його поведінка (гравітація, швидкість, напрямок). Стрільба активується з Blueprint або C++ функції Fire(), яка спавнить снаряд в координатах, що відповідають положенню камери гравця. При потребі візуальний ефект або звук можна додати через компоненти Niagara або Audio Cue. Метод руху снаряда FireInDirection у класі Projectile наведено у лістингу 3.2.

Лістинг 3.2 – Метод руху снаряда FireInDirection у класі Projectile

```
void AMyProjectile::FireInDirection(const FVector& ShootDirection)
{
    if (ProjectileMovementComponent)
    {
```

```
        ProjectileMovementComponent->Velocity = ShootDirection *
ProjectileMovementComponent->InitialSpeed;
    }
}
```

кінець лістингу 3.2

Далі реалізується базова ворожа поведінка з використанням інструментів штучного інтелекту Unreal Engine 4. Реалізуємо базовий ШІ для ворогів у грі. Спочатку створюється новий Blueprint-клас ворога під назвою BP_EnemyCharacter, який наслідується від стандартного ACharacter. Цей клас представляє собою ворожого персонажа, здатного пересуватись і реагувати на гравця. Для керування діями ворога додається окремий AI-контролер – BP_EnemyAIController, у якому задається логіка поведінки через Behavior Tree (дерево поведінки). У Behavior Tree задається структура дій, яку виконуватиме ворог, зокрема: перевірка, чи бачить ворог гравця, і, якщо бачить – пересування до нього.

Для збереження та використання змінних, які використовує AI, створюється Blackboard – наприклад, ключ Target, у який записується гравець як об’єкт спостереження. Щоб ворог міг рухатися в просторі гри, на ігрову мапу додається NavMeshBoundsVolume – спеціальний об’єкт, що створює навігаційну сітку, по якій може переміщатися AI. У Behavior Tree реалізовано просту логіку: якщо ворог бачить гравця – рухається до нього; якщо не бачить – стоїть або патрулює. При зіткненні з гравцем (через подію OnHit) можна додатково реалізувати відповідну реакцію, наприклад, анімацію атаки або зменшення кількості здоров’я у гравця. Такий підхід дозволяє створити динамічну й логічну поведінку ворогів у грі (рис. 3.4).

Далі реалізується система здоров’я для гравця та ворогів, а також інтерфейс користувача (UI), який відображає рівень здоров’я гравця в реальному часі [13]. Спершу до класів гравця та ворога додається змінна Health (наприклад, типу float), яка відповідає за поточний рівень здоров’я. При

отриманні пошкодження (наприклад, при попаданні снаряду або під час атаки ворога) значення Health зменшується.

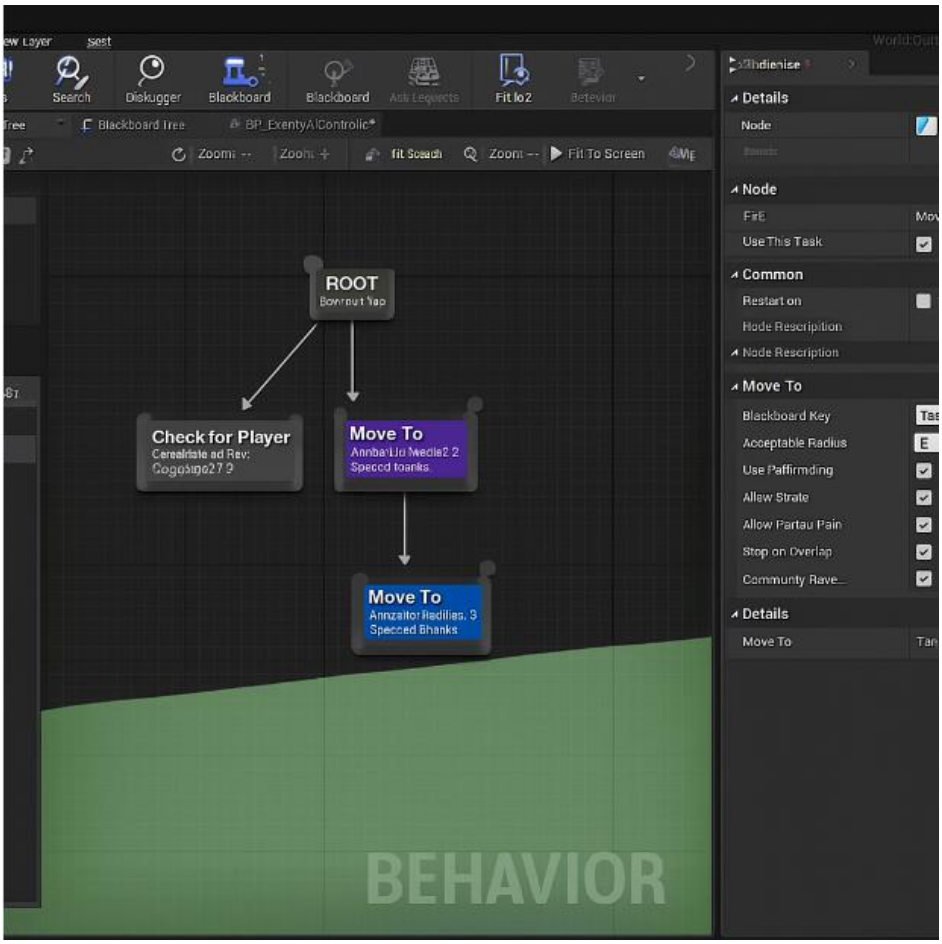

Рисунок 3.4 – Налаштування Behavior Tree для ворога

Якщо воно досягає нуля або менше – виконується дія «смерті» персонажа: для гравця це може бути перезапуск рівня, для ворога – знищення об’єкта або запуск анімації загибелі (рис. 3.5).

Для візуалізації здоров’я гравця створюється UI віджет (UserWidget) у вигляді панелі зі шкалою (Progress Bar). Цей віджет додається до HUD гравця при старті гри. Значення шкали зв’язується з поточним значенням Health, і змінюється динамічно під час гри.

Для цього у віджеті реалізується функція або binding, що звертається до актуального значення здоров’я гравця. Таким чином, коли гравець отримує пошкодження, зміна відображається на екрані у вигляді зменшення шкали HP.

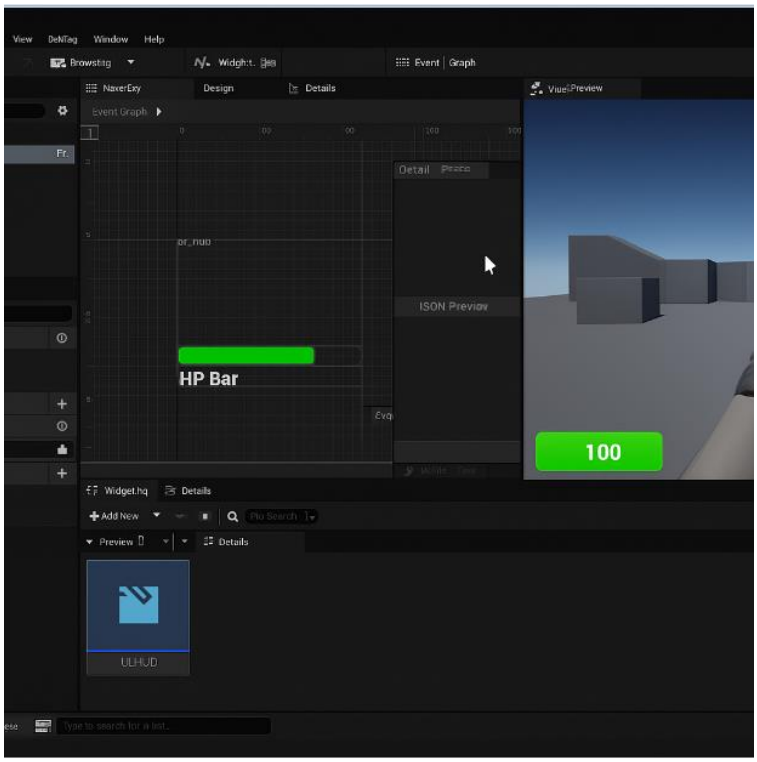

Рисунок 3.5 – Налаштованим віджетом для відображення здоров’я гравця (HP Bar)

Подібну систему можна реалізувати і для ворогів, якщо потрібно візуально показати їхній стан. Це додає грі динамічності та зручності для гравця (ліст. 3.3).

Лістинг 3.3 – Гравець отримує пошкодження

```
void AMyCharacter::TakeDamage(float DamageAmount)
{
   Health -= DamageAmount;
   if (Health <= 0.0f)
   {
      // Гравець помер
      Die();
   }
}
```

кінець лістингу 3.3

На фінальному етапі реалізації гри виконується збірка проєкту в повноцінний виконуваний файл (EXE) для платформи Windows. Спершу у налаштуваннях проєкту Unreal Engine 4 потрібно переконатися, що вказано стартовий рівень гри – для цього у меню Project Settings → Maps & Modes вказується стартова мапа у полі Game Default Map. Це гарантує, що при запуску зібраної гри одразу відкриється потрібний ігровий рівень.

Далі у меню File → Package Project → Windows (64-bit) запускається процес збірки. Unreal Engine компілює всі ресурси (Blueprint, C++ код, матеріали, звуки тощо) у готовий ігровий продукт. Після завершення збірки у вказаній директорії з'являється папка з грою, в якій міститься .exe файл та інші необхідні дані (Content, Config, тощо). Цей файл можна запускати як звичайну гру поза межами редактора.

3.2 Тестування та налагодження інформаційно-комп'ютерної системи

Після завершення етапу практичної реалізації гри-шутера на рушії Unreal Engine 4 було розпочато етап тестування та налагодження системи з метою виявлення недоліків, покращення продуктивності та забезпечення стабільної роботи застосунку на різних конфігураціях обладнання. У процесі тестування використовувалися як вбудовані інструменти рушія (Blueprint Debugger, Output Log, Session Frontend), так і ручні методи тестування сценаріїв користувача. Основну увагу приділено коректності обробки подій у режимі реального часу: стрільба, колізії, рухи персонажа, реакція ШІ-супротивника, обробка стану здоров'я та відображення HUD.

На початковому етапі було протестовано основні механіки гравця: переміщення, прицілювання, стрибки, стрільба та взаємодія з супротивниками. Було виявлено кілька недоліків, зокрема зависання снаряда в повітрі при неправильному розрахунку точки спавну. Для усунення проблеми було модифіковано метод Fire() у класі гравця AMyCharacter, додавши перевірку на

наявність активної камери та додатковий зсув від позиції гравця. Виправлена версія методу наведена у лістингу 3.4.

Лістинг 3.4 – Виправлена функція стрільби з урахуванням позиції камери

```
void AMyCharacter::Fire()
{
    if (ProjectileClass && FirstPersonCameraComponent)
    {
        FVector MuzzleLocation = FirstPersonCameraComponent-
>GetComponentLocation() + FirstPersonCameraComponent->GetForwardVector()
* 100.0f;
        FRotator MuzzleRotation = FirstPersonCameraComponent-
>GetComponentRotation();
        GetWorld()->SpawnActor<AMyProjectile>(ProjectileClass,
MuzzleLocation, MuzzleRotation);
    }
}
```

кінець лістингу 3.4

Наступним етапом стало тестування механіки взаємодії снарядів із середовищем. Під час випробувань виявлено, що при зіткненні кулі з деякими об’єктами (наприклад, статичними сітками) не відбувається ані візуального ефекту, ні знищення снаряда. Проблема полягала у відсутності коректно реалізованої події OnHit у Blueprint-снаряді. Виправлення включало додавання компонента «Collision (Sphere)» та обробника події зі знищенням об’єкта (рис. 3.6) [14].

Для супротивників було протестовано коректність логіки штучного інтелекту, побудованої на базі Behavior Tree. Основна помилка полягала в тому, що ворог не реагував на гравця, якщо гравець з’являвся в полі зору надто швидко. Аналіз Blackboard показав, що змінна «Target» не оновлювалася належним чином. Після налагодження логіки в AI Controller з використанням методу SetFocus(PlayerPawn) було досягнуто очікуваної поведінки – ворог починає переслідування гравця при виявленні його у радіусі дії.

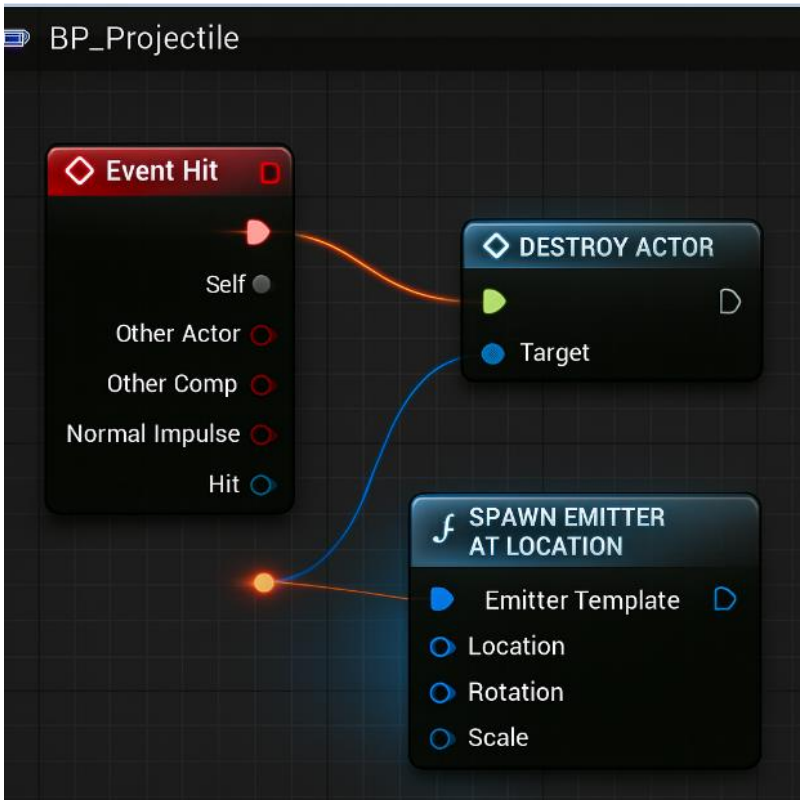

Рисунок 3.6 – Реалізація події OnHit у Blueprint снаряда

У ході тестування продуктивності було виявлено, що сцена втрачає продуктивність під час великої кількості динамічних джерел світла. Задля оптимізації було впроваджено метод Occlusion Culling, а також зменшено кількість активних Point Light на рівні, що дозволило підвищити частоту кадрів з 38 FPS до 59 FPS на середньому ПК (Core i5-8600, 8 ГБ ОЗП, GeForce GTX 1050Ti).

Окрему увагу було приділено інтерфейсу користувача. Під час тестування виявлено, що HUD не оновлює значення здоров’я при отриманні шкоди. Помилкою виявилося те, що Progress Bar не мав прив’язки до актуального значення змінної Health. Було реалізовано функцію GetHealthPercent(), яка повертала відсоткове значення, та оновлення віджету через Binding (ліст. 3.5).

Лістинг 3.5 – Метод для оновлення шкали здоров’я гравця

```
float AMyCharacter::GetHealthPercent() const
```

```
{
    return Health / MaxHealth;
}
```

кінець лістингу 3.5

На завершальному етапі було сформовано системні вимоги для стабільної роботи гри. Мінімальні системні вимоги:

- ОС: Windows 10 (64-bit);
- процесор: Intel Core i3-7100 або аналогічний AMD;
- оперативна пам’ять: 8 ГБ;
- відеокарта: NVIDIA GTX 750 Ti або AMD R7 360;
- вільне місце на диску: 5 ГБ.

В результаті проведеного тестування було усунено ряд технічних помилок, налагоджено ключові механіки гри, покращено продуктивність та UX. Таким чином, інформаційно-комп’ютерна система у вигляді шутера на Unreal Engine 4 готова до фінальної збірки, демонстрації та використання на презентаційних заходах.

3.3 Розробка бази даних

У межах розробки гри-шутера на Unreal Engine 4 було реалізовано прототипну модель бази даних, яка дозволяє зберігати базову інформацію про гравців, статистику проходження та результати. Для реалізації бази даних обрано популярне реляційне сховище MySQL, керування яким здійснюється через графічний веб-інтерфейс phpMyAdmin. Цей інструмент забезпечує зручне створення таблиць, зв’язків між ними та виконання SQL-запитів. База даних може бути використана у розширених версіях гри для зберігання користувацьких профілів, досягнень, історії сесій, таблиці лідерів та інших аналітичних даних.

Структура бази даних побудована з урахуванням принципів нормалізації, щоб уникнути дублювання інформації та забезпечити логічні зв’язки між

сутностями. Основними таблицями є: Players – містить дані користувачів (ID, ім’я, рівень); Scores – зберігає результати гри (гравець, кількість очок, дата сесії); Games – таблиця для інформації про сесії; GameItems – перелік предметів, які гравець може отримати в грі. Зв’язки між таблицями реалізовані за допомогою зовнішніх ключів (FOREIGN KEY), що дозволяє зберігати структуровані залежності між записами (рис. 3.7).

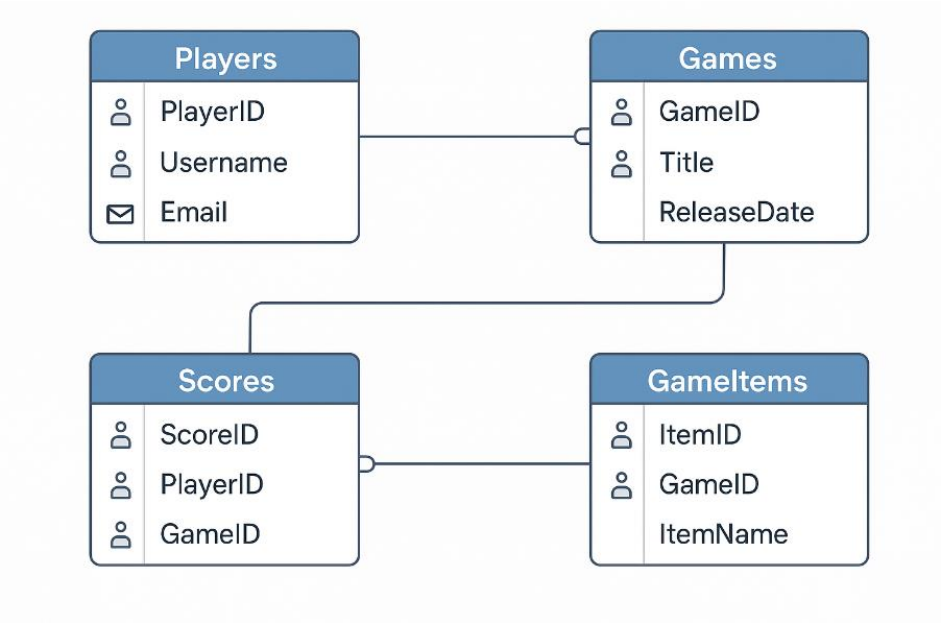

Рисунок 3.7 – Структура БД

Процес створення бази даних починається з ініціалізації нової схеми в phpMyAdmin та створення таблиць SQL-запитами. Наприклад, для виведення всіх результатів з таблиці Scores використовується стандартний запит: SELECT * FROM Scores; для вставлення нового запису про результат гравця – такий запит: INSERT INTO Scores(PlayerID, Points, Date) VALUES (1, 1500, NOW()).

Аналогічно, можна реалізувати механізми вибірки результатів за датою, гравцем або іншими параметрами. Завдяки використанню MySQL, дані можуть зберігатися локально або інтегруватися з ігровим рушієм за допомогою REST API, JSON або сторонніх плагінів (наприклад, VaRest для UE4). Такий підхід дозволяє у майбутньому розширити гру до мережевої, із збереженням даних гравців та прогресу.

Отже, розробка бази даних забезпечує основу для потенційного масштабування гри, додавання багатокористувацьких функцій, рейтингових таблиць, статистики та індивідуального прогресу гравця, що значно підвищує рівень функціональності ігрового проєкту.

ВИСНОВКИ

У ході виконання бакалаврської кваліфікаційної роботи на тему «Розробка гри-шутера з використанням Unreal Engine 4» було здійснено повний цикл створення ігрового програмного продукту, починаючи з дослідження теоретичних аспектів і закінчуючи практичною реалізацією ігрового прототипу. Кожне із поставлених завдань було виконано послідовно, з дотриманням сучасних методик проектування, моделювання та розробки програмного забезпечення.

Проведено аналіз сучасних тенденцій розробки ігор у жанрі шутер. Вивчено особливості популярних комерційних продуктів, таких як Fortnite та Apex Legends, визначено ключові характеристики, які впливають на ігровий досвід: якісна графіка, динамічний геймплей, система озброєння, інтерфейс та AI-супротивники. Отримані дані дозволили сформулювати вимоги до майбутнього проєкту.

Було досліджено функціональні можливості рушія Unreal Engine 4. Проаналізовано його архітектуру, систему візуального програмування Blueprints, механізми реалізації фізики, освітлення, анімації, підтримки AI та інструменти розробки користувацького інтерфейсу. Також було проведено порівняльний огляд інших сучасних рушіїв – Unity та CryEngine, що дозволило аргументовано обґрунтувати вибір саме Unreal Engine 4 для реалізації гри завдяки його потужним засобам рендерингу, підтримці C++ і високій якості графіки.

Розроблено концепцію гри-шутера, що включає опис базових механік, ігровий сценарій, дизайн рівня, візуальний стиль і карту світу. Була сформована система озброєння, описані ключові параметри персонажа, а також продумано взаємодію з оточенням і супротивниками, що дозволило закласти основу для реалізації повноцінної геймплейної структури.

Завдання проєктування архітектури гри на основі об’єктно-орієнтованого підходу та шаблону «гравець-контролер-ігровий режим» передбачало

створення UML-діаграми класів та прецедентів, де враховано взаємодію ключових об’єктів гри – гравця, зброї, супротивників, ігрового середовища, HUD тощо. Це забезпечило логічну організацію структури проєкту і стало фундаментом для реалізації функціональних компонентів.

Створено ігрового персонажа з можливістю переміщення, прицілювання, стрільби, шляхом програмної реалізації основних ігрових механік за допомогою C++ та Blueprints у середовищі Unreal Engine 4. Розроблено AI супротивників з адаптивною поведінкою, реалізовано HUD для відображення стану гравця, а також систему здоров’я й логіку уражень. Усі ці компоненти протестовані та налаштовані для взаємодії.

Проведено тестування, налагодження та оптимізацію гри. За допомогою профайлерів, дебагерів та unit-тестів було виявлено і виправлено типові помилки, пов’язані з колізіями, спавном об’єктів, оновленням інтерфейсу тощо. Оптимізовано роботу з джерелами світла, застосовано Occlusion Culling, що дало змогу досягти високої продуктивності на середніх ПК.

Отже, у результаті виконаної роботи розроблено повноцінний прототип гри-шутера на рушії Unreal Engine 4 з використанням сучасних підходів до розробки ігрових продуктів. Здобуто практичні навички з проєктування, моделювання та програмування інтерактивних додатків, що дозволяє зробити висновок про досягнення поставленої мети та успішне виконання всіх завдань. Створена гра може слугувати базовим зразком для подальшого розвитку в напрямку мультиплеєра, кастомізації персонажів та покращення графіки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Nihar Raut. 3D First-Person Shooter Game Development Using Unreal Engine 5. International Journal for Research in Applied Science and Engineering Technology 11(5). Pp. 4106-4115. DOI:10.22214/ijraset.2023.51039.

2. Unreal Engine 4 Optimization Tutorial, Part 1 UML:Unreal Engine 4 Optimization Tutorial, Part 1. URL: https://uk.wikipedia.org/wiki/Unreal_Engine (дата звернення: 16.03.2025).

3. Nathan Partlan. EvolvingBehavior: Towards Co-Creative Evolution of Behavior Trees for Game NPCs. URL: https://doi.org/ 10.48550/arXiv.2209.01020 (дата звернення: 16.03.2025).

4. Unity. URL: https://uk.wikipedia.org/wiki/Unity_(%D1%96%D0%B3%D1%80%D0%BE%D0%B2%D0%B8%D0%B9_%D1%80%D1%83%D1%88%D1%96%D0%B9) (дата звернення: 02.03.2025).

5. CryEngine. URL: https://uk.wikipedia.org/wiki/CryEngine (дата звернення: 12.03.2025).

6. Найкращі онлайн шутери на ПК. URL: https://uaplay.com.ua/najkrashchi-onlajn-shutery-na-pk/ (дата звернення: 02.03.2025).

7. Мобільна гра Apex Legends. URL: https://itc.ua/ua/tag/electronic-arts-ua/ (дата звернення: 02.03.2025).

8. Aghyad Albaghajati&Jameleddine Hassine. A Use Case Driven Approach to Game Modeling. Requirements Engineering, Springer. URL: https://doi.org/10.1007/s00766-021-00362-4 (дата звернення: 14.03.2025).

9. Kravchenko S. M., Suhonyak I. I., Marchuk H. V., Hryshkun Ye. O., and Venglovska Yu. M. UML Modeling of the Puzzle Game Design Process. URL: https://doi.org/10.32782/2663-5941/2023.3.1/25 (дата звернення: 16.03.2025).

10. Samchuk L., Povstiana Y. UML Diagrams of the Management System of Maintenance Stations. Informatyka, Automatyka, Pomiary W Gospodarce I Ochronie Środowiska, vol. 14, no. 4. 2024. Pp. 141-145.

11. Sharma S. P. V. Mastering Unreal Engine 4.X. Birmingham: Packt Publishing, 2020. 420 p.

12. Rusu M. Artificial Intelligence and Gameplay Programming with Unreal Engine. CRC Press, 2021. 312 p.

13. Cordone R. Unreal Engine Blueprints Visual Scripting Projects. Packt Publishing, 2021. 328 p.

14. Medeiros J. Building First-Person Shooter Games with Unreal Engine 4. Packt Publishing, 2023. 390 p.

15. Методичні вказівки до виконання кваліфікаційної роботи бакалавра: для здобувачів першого (бакалаврського) рівня вищої освіти освітньої програми «Інженерія програмного забезпечення» галузі знань 12 Інформаційні технології спеціальності 121 Інженерія програмного забезпечення денної та заочної форм навчання / уклад. Н. М. Ліщина, А. А. Ящук. Луцьк: ЛНТУ, 2025. 56 с.