МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Острозька академія»
Навчально-науковий інститут інформаційних технологій та бізнесу
Кафедра інформаційних технологій та аналітики даних

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавра

на тему: «**Розробка та проектування шутера від першої особи на Unreal Engine»**

Виконав: студент 4 курсу, групи КН-41
першого (бакалаврського) рівня вищої освіти
спеціальності 122 Комп'ютерні науки
освітньо-професійної програми «Комп'ютерні науки»
Максим Анатолійович Ніколайчук

Керівник: *викладач, фахівець практик Володимир Віталійович Місай*

Рецензент: *кандидат технічних наук, доцент, доцент кафедри прикладної математики Донецького національного університету імені Василя Стуса Загоруйко Любов Василівна*

РОБОТА ДОПУЩЕНА ДО ЗАХИСТУ

Завідувач кафедри інформаційних технологій та аналітики даних

_____ (проф., д.е.н. Кривицька О.Р.)

Протокол № 11 від «20» травня 2026 р.

Острог, 2026

АНОТАЦІЯ
кваліфікаційної роботи
на здобуття освітнього ступеня бакалавра

Тема: *Розробка та проектування шутера від першої особи на Unreal Engine*

Автор: *студент спеціальності 122 Комп'ютерні науки освітньо-професійної програми «Комп'ютерні науки»*
Ніколайчук Максим Анатолійович

Науковий керівник: *викладач, фахівець практик*
Володимир Віталійович Місай

Захищена «……»…… 20__ року.

Пояснювальна записка до кваліфікаційної роботи: ***44*** *(кількість сторінок роботи) с.,* ***27*** *(кількість рисунків) рис.,* ***0*** *(кількість таблиць) табл.,* ***0*** *(кількість додатків) додатків,* ***12*** *(кількість джерел) джерел.*

Ключові слова: *ВІДЕОГРА, ПРОТОТИП, ІГРОВИЙ РУШІЙ, UNREAL ENGINE 5, ВІЗУАЛЬНЕ ПРОГРАМУВАННЯ, BLUEPRINTS, ШУТЕР ВІД ПЕРШОЇ ОСОБИ, FPS, ВИЖИВАННЯ, КОНТРОЛЕР ПЕРСОНАЖА, ENHANCED INPUT, ІГРОВІ МЕХАНІКИ, ШТУЧНИЙ ІНТЕЛЕКТ, NAVMESH, СИСТЕМА ХВИЛЬ, ЛЕВЕЛ-ДИЗАЙН, ДИНАМІЧНІ БАРИКАДИ, КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС, UMG, HUD, ВІЗУАЛЬНІ ЕФЕКТИ, VFX, ПРОСТОРОВИЙ ЗВУК.*

Короткий зміст праці: *Кваліфікаційна робота присвячена проєктуванню та розробці прототипу відеогри в жанрі шутера від першої особи (FPS) з елементами виживання. Метою роботи є створення динамічного ігрового продукту, що об'єднує комплексні бойові механіки, систему хвиль супротивників та тактичну взаємодію з оточенням. У процесі виконання обґрунтовано вибір ігрового рушія Unreal Engine 5 як основного інструментарію та реалізовано повний цикл розробки: від створення базового контролера персонажа до фінального аудіовізуального полірування. Спроєктовано та деталізовано ігрову локацію в стилістиці підземного Sci-Fi комплексу, що включає унікальну геймплейну систему генерації динамічних барикад. Технічну реалізацію логіки виконано засобами візуального програмування Blueprints із застосуванням підсистеми Enhanced Input, методів просторового трасування (Line Trace) для обробки колізій зброї та навігаційної сітки (NavMesh) для штучного інтелекту. Для створення цілісного ігрового циклу інтегровано алгоритми сенсорного сприйняття ворогів (Pawn Sensing) та математичну модель ескалації складності хвиль на базі Game Mode. Інтерфейс користувача та HUD реалізовано через систему UMG, а імерсивність процесу посилено інтеграцією візуальних ефектів (VFX) та просторового звуку (Sound Attenuation). Програмний продукт є завершеним функціональним вертикальним зрізом відеогри, готовим до подальшого масштабування або використання як демонстраційного портфоліо.*

ABSTRACT

of the qualification

work for a Bachelor's degree

Theme: *Development and design of a first-person shooter on Unreal Engine*

Author: *Student of degree programme 122 Computer Science within the vocational education programme «Computer Science»*

Maksym Anatoliyovych Nikolaichuk

Supervisor: *Lecturer, Practising Specialist*
Volodymyr Vitaliyovych Misai

Defended on «……»…… 20__.

Explanatory note to the qualification work: ***44*** *(number of pages) pp.,* ***27*** *(number of figures) figs.,* ***0*** *(number of tables) tables,* ***0*** *(number of appendices) appendices,* ***12*** *(number of sources) sources.*

Keywords: *VIDEO GAME, PROTOTYPE, GAME ENGINE, UNREAL ENGINE 5, VISUAL PROGRAMMING, BLUEPRINTS, FIRST-PERSON SHOOTER, FPS, SURVIVAL, CHARACTER CONTROLLER, ENHANCED INPUT, GAME MECHANICS, ARTIFICIAL INTELLIGENCE, NAVMESH, WAVE SYSTEM, LEVEL DESIGN, DYNAMIC BARRICADES, USER INTERFACE, UMG, HUD, VISUAL EFFECTS, VFX, SPATIAL AUDIO.*

Abstract: *This qualification work is dedicated to the design and development of a video game prototype in the first-person shooter (FPS) genre with survival elements. The aim of the work is to create a dynamic gaming product that combines complex combat mechanics, an enemy wave system, and tactical interaction with the environment. In the course of the execution, the choice of the Unreal Engine 5 game engine as the primary toolkit was justified, and a full development cycle was implemented: from creating a basic character controller to the final audiovisual polishing. A game location was designed and detailed in the style of an underground Sci-Fi complex, which includes a unique gameplay system for generating dynamic barricades. The technical implementation of the logic was carried out using Blueprints visual programming tools, applying the Enhanced Input subsystem, spatial tracing methods (Line Trace) for weapon collision processing, and a navigation mesh (NavMesh) for artificial intelligence. To create a cohesive game loop, enemy sensory perception algorithms (Pawn Sensing) and a mathematical model of wave difficulty escalation based on the Game Mode were integrated. The user interface and HUD were implemented through the UMG system, while the immersiveness of the process was enhanced by integrating visual effects (VFX) and spatial audio (Sound Attenuation). The software product is a completed functional vertical slice of a video game, ready for further scaling or use as a demonstration portfolio.*

ЗМІСТ

ВСТУП

Актуальність теми. Сучасна ігрова індустрія переживає стрімкий розвиток інді-сегменту, де навіть невеликі команди або соло-розробники здатні створювати високоякісні проєкти. Жанр шутерів від першої особи традиційно залишається одним із найпопулярніших та найбажаніших серед гравців [13, с. 45]. Зомбі-шутери, зокрема, створюють унікальний ігровий досвід: постійна загроза, відчуття ізольованості та необхідність виживати на самоті викликають високу емоційну напругу та стимулюють гравця до активних дій.

Розробка таких ігор сьогодні стала значно доступнішою завдяки передовим інструментам, насамперед ігровому рушію Unreal Engine 5. Будучи провідним індустріальним стандартом, він пропонує потужний функціонал та систему візуального програмування Blueprints, що суттєво знижує поріг входу в розробку складної ігрової логіки [17, с. 22]. Наявність великої кількості навчальних матеріалів робить навичку створення ігор на цьому рушії надзвичайно затребуваною та корисною для сучасного розробника.

Мета і задачі дослідження. Метою кваліфікаційної роботи є розробка та проектування прототипу відеогри в жанрі шутера від першої особи (зомбі-шутера) з використанням інструментарію рушія Unreal Engine 5 для практичного дослідження та опанування сучасних ігрових механік.

Для досягнення поставленої мети було сформульовано такі **задачі**:

- Проаналізувати особливості жанру FPS та ключові можливості рушія Unreal Engine 5.
- Сформувати концепт гри та описати основний ігровий цикл.
- Спроектувати та реалізувати базові ігрові механіки за допомогою системи Blueprints: логіку керування персонажем, контролер зброї та систему регенерації здоров'я.
- Розробити систему штучного інтелекту для ворогів (зомбі), що включає алгоритми виявлення гравця (Pawn Sensing), динамічну зміну швидкості руху та систему реакцій на пошкодження.

- Реалізувати ігровий режим із системою генерації (спавну) хвиль ворогів.
- Створити дизайн ігрової локації з урахуванням геймплейних потреб (наявність укриттів та простору для маневрування).
- Розробити користувацький інтерфейс (UI) та інтегрувати аудіовізуальне оформлення (VFX ефекти влучань, звуковий супровід).

Об'єкт дослідження - процес створення та проектування відеогри у жанрі шутера від першої особи.

Предмет дослідження - ігрові механіки, архітектурні рішення, алгоритми штучного інтелекту та інструментальні засоби рушія Unreal Engine 5 (зокрема система Blueprints), що використовуються для розробки логіки та візуальної складової гри.

Практичне значення одержаних результатів. Робота має яскраво виражений дослідницький та прикладний характер. Її результати дозволяють глибоко "зазирнути під капот" розробки шутерів, детально розібрати логіку їх функціонування та на практиці порівняти підходи Unreal Engine з іншими ігровими рушіями. Створений проєкт є результатом успішного опанування нових навичок (робота з Blueprints, ШІ, левел-дизайном, UI) і може слугувати міцною базою для подальшого розширення або стати повноцінною частиною професійного портфоліо розробника.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПРОЕКТУВАННЯ КОНЦЕПТУ ГРИ

1.1. Огляд жанру шутерів (FPS) та аналіз рушія Unreal Engine 5

Шутери від першої особи (First-Person Shooter, FPS) є одним із найстаріших та найпопулярніших жанрів у відео ігровій індустрії. Його особливість полягає у максимальному зануренні гравця, оскільки огляд ігрового світу відбувається "очима" головного героя. Це висуває високі вимоги до технічного виконання та візуальної складової гри [14, с. 120].

При розробці даного проекту основними референсами виступили такі еталонні представники піджанру «зомбі-шутерів», як *Left 4 Dead* та режим Zombies у серії *Call of Duty*. Концепція гри зосереджена на динамічному виживанні проти хвиль ворогів у закритому просторі, що створює атмосферу постійної напруги. Аналіз жанру дозволяє виділити ключові чинники залучення гравця, які були покладені в основу розробки:

1. **«Відчуття зброї» (Gunplay):** імпакт від стрільби, реалістичність віддачі та візуальна реакція ворогів на влучання.
2. **Динаміка подій:** швидкий темп ігрового процесу та постійна зміна ситуації на полі бою.
3. **Анімаційна складова:** висока деталізація рухів персонажа та зброї від першої особи, що безпосередньо впливає на сприйняття реалістичності.
4. **Аудіовізуальний супровід:** звук кроків, стрільби та оточення, що доповнює візуальну картину і створює необхідну атмосферу [10].

Вибір інструментарію для розробки припав на рушій **Unreal Engine 5 (UE5)**. Проводячи порівняльний аналіз із рушієм *Unity*, на якому базувався попередній досвід автора, можна виділити кілька вагомих причин переходу на UE5 для розробки 3D-шутера:

- **Спеціалізація:** Unreal Engine історично створювався для розробки шутерів, тому його архітектура та базові класи (Character, Pawn, Controller) ідеально оптимізовані під цей жанр [1].

- **Візуальне програмування Blueprints:** на відміну від мови C# у Unity, система Blueprints у UE5 дозволяє створювати складну ігрову логіку швидше та інтуїтивніше, зберігаючи при цьому високу продуктивність. Вона є менш вимогливою до синтаксису, що дозволяє більше фокусуватися на алгоритмах, а не на написанні коду [5].
- **Інтегрований інструментарій:** Unreal Engine має потужні вбудовані засоби для роботи з анімаціями (Animation Blueprints) та звуком (MetaSounds), що покривають більшість потреб розробника без необхідності пошуку сторонніх плагінів.

Окрему увагу в роботі приділено новітнім технологіям UE5, які було використано для покращення якості та продуктивності гри:

1. **Lumen:** система глобального динамічного освітлення та відображень, яка дозволила досягти фотореалістичної картинки в реальному часі без необхідності тривалого процесу «запікання» світла.
2. **Nanite:** технологія віртуалізованої геометрії, що дозволяє використовувати високополігональні асети без значних втрат продуктивності, що значно розширює можливості левел-дизайну [15, с. 89].
3. **Animation State Caching:** використання кешування станів для окремих частин тіла дозволило гнучко поєднувати різні анімації (наприклад, біг та стрільбу), а інтеграція звукових повідомлень безпосередньо в анімаційні послідовності забезпечила точну синхронізацію кроків та перезарядки.

Отже, Unreal Engine 5 на сьогодні є найбільш прогресивним та доступним рішенням для інді-розробників, пропонуючи не лише високу якість графіки, а й гнучкий інструментарій для створення складних ігрових механік та оптимізації готового продукту.

1.2. Розробка геймдизайн-документу (GDD) та опис ігрового циклу

Геймдизайн-документ (GDD) є ключовим етапом проектування відеогри, який визначає її правила, механіки та загальне бачення. Для даного прототипу розроблено концепт фокусованого зомбі-шутера, де головним завданням гравця є виживання в умовах закритої локації та постійної загрози.

Основний ігровий цикл (Core Loop). Ігровий процес побудований на хвильовій системі генерації ворогів. Макроцикл гри складається з таких етапів [18, с. 210]:

1. **Активна фаза:** Гравець протистоїть хвилі ворогів. З кожним новим етапом кількість зомбі та тиск на гравця зростають.
2. **Фаза перепочинку:** Після знищення останнього ворога у хвилі настає коротка пауза (декілька секунд). Цей час дається гравцеві для тактичного переміщення, пошуку вигідної позиції для оборони та перезаряджання зброї.
3. **Сповіщення:** Початок нової хвилі супроводжується текстовим повідомленням на екрані, що сигналізує про нову загрозу та не дає гравцю втратити концентрацію.

Глобальною метою гри є успішне виживання протягом фіксованої кількості хвиль (від 10 до 15), після чого ігрова сесія завершується перемогою.

Бойова система та менеджмент ресурсів. Основним і єдиним інструментом захисту гравця є штурмова гвинтівка. Задля зміщення фокусу з пошуку ресурсів на динаміку бою, загальний боєзапас зроблено нескінченним, проте ємність магазину обмежена 30 набоями. Ключовим елементом балансу є час на перезаряджання зброї: під час цієї анімації гравець є найбільш вразливим. Це стимулює грамотно розпоряджатися набоями в магазині, контролювати дистанцію та використовувати геометрію рівня для укриття.

Додатковим обмеженням є неможливість ведення вогню під час бігу (спринту). Така механіка змушує гравця постійно маневрувати, розривати дистанцію з ворогами, зупинятися для прицільної стрільби і знову змінювати позицію, що формує класичний тактичний підхід «hit-and-run» (вдар та біжи).

Система здоров'я та виживання У грі реалізовано систему автоматичної регенерації здоров'я (HP), що дозволяє відновлювати сили без пошуку аптечок. Логіка системи працює наступним чином: якщо рівень здоров'я падає нижче 90%, після невеликої затримки починається поступове відновлення кроками по 5%.

Для інформування гравця про критичний стан використовується візуальний інтерфейс (UI): при зниженні HP до 50% і нижче на екрані з'являється спеціальний кривавий оверлей. Це створює психологічний тиск і чітко сигналізує про необхідність терміново знайти укриття та уникати контакту з ворогами до відновлення показників.

Баланс та поведінка ворогів. Вороги (зомбі) атакують гравця виключно в ближньому бою. Для збереження ігрового балансу та надання гравцеві шансу на виживання у випадку оточення натовпом, швидкість їхніх атак навмисно занижена.

Важливою механікою є локальна система шкоди (Locational Damage), реалізована за допомогою фізичних матеріалів (Physical Materials) рушія Unreal Engine 5. Різні частини тіла ворога мають власні фізичні матеріали, що дозволяє системі розпізнавати зону влучання та застосовувати відповідні множники шкоди:

- **Влучання в тіло:** потребує близько 5 пострілів для знищення цілі.
- **Влучання в голову (Headshot):** завдає критичної шкоди, вимагаючи лише 2 постріли.

Такий підхід винагороджує гравця за точну стрільбу та стимулює розвивати навички наведення на ціль (aiming), що є фундаментальною складовою залучення у жанрі FPS [19, с. 150].

1.3. Постановка задачі розробка

У рамках даної кваліфікаційної роботи необхідно розробити та спроектувати повноцінний ігровий прототип шутера від першої особи (FPS) з елементами виживання (зомбі-шутер).

Основна мета розробки полягає у створенні цілісного «вертикального зрізу» (Vertical Slice) гри, який на практиці демонструє роботу ключових ігрових механік, архітектуру взаємодії обєктів та аудіовізуальне оформлення. Відповідно до концепції

гри та геймдизайн-документу, визначено такі технічні та функціональні вимоги до розробки:

1. Вимоги до платформи та інструментарію:

- **Середовище розробки:** ігровий рушій Unreal Engine 5.
- **Метод програмування:** реалізація всієї ігрової логіки за допомогою системи візуального програмування Blueprints без прямого використання мови C++.
- **Цільова платформа:** персональні комп'ютери (ОС Windows).
- **Система управління:** класична схема для ПК (клавіатура та миша).

2. Функціональні вимоги (ігрові механіки):

- **Контролер гравця:** розробка системи пересування від першої особи, що включає ходьбу, спринт, стрибки та присідання. Інтеграція реакції зброї на наближення до перешкод (анімація відведення зброї біля стіни).
- **Бойова система:** реалізація механіки стрільби зі штурмової гвинтівки, системи перезаряджання, обмеження ведення вогню під час бігу та розрахунок локальної шкоди (Physical Materials для тіла та голови).
- **Штучний інтелект (AI):** створення логіки поведінки ворогів із використанням навігаційної сітки (NavMesh). Інтеграція компонента виявлення (Pawn Sensing) для переходу ворогів зі стану повільного патрулювання до швидкого переслідування гравця [16, с. 305].
- **Ігровий цикл:** розробка ігрового режиму (Game Mode), що контролює систему спавну хвиль ворогів, паузи між ними та умови перемоги/поразки.

3. Вимоги до інтерфейсу та аудіовізуального супроводу:

- **Користувацький інтерфейс (UI):** створення головного меню, ігрового HUD (відображення стану здоров'я та набоїв), екрану смерті та системи текстових сповіщень про початок нової хвилі.
- **Система здоров'я:** налаштування логіки автоматичної регенерації здоров'я після отримання шкоди з виведенням попереджувального оверлею при критичному показнику HP (<= 50%).
- **Аудіо та VFX:** інтеграція звуків кроків, стрільби, перезаряджання, а також озвучення дій зомбі. Налаштування візуальних ефектів (VFX) для

відображення крові при влучанні у ворога та ефектів відскоку куль від елементів оточення.

4. Вимоги до левел-дизайну та оптимізації:

- Створення просторої ігрової локації з продуманою геометрією та укриттями, що дозволяють гравцеві тактично розривати дистанцію з ворогами.
- Використання сучасних підсистем рушія (Nanite для оптимізації високополігональних моделей та Lumen для глобального освітлення) з метою досягнення оптимального балансу між якістю графіки та продуктивністю.

Виконання поставлених задач дозволить створити завершений ігровий продукт, що наочно демонструє процес розробки 3D-шутерів та ефективність використання інструментарію Unreal Engine 5.

Висновок до розділу 1

У першому розділі було проведено комплексний аналіз предметної області, сформовано концептуальне бачення майбутньої гри та визначено технічні вимоги для її реалізації. На основі проведеного дослідження можна зробити такі висновки:

1. **Обґрунтовано вибір інструментарію:** Проаналізовано специфіку жанру шутерів від першої особи (FPS) та доведено доцільність використання ігрового рушія Unreal Engine 5 для розробки даного прототипу. Встановлено, що архітектура UE5, система візуального програмування Blueprints та вбудовані технології (Lumen, Nanite) забезпечують оптимальний баланс між високою якістю візуальної складової та швидкістю розробки складних ігрових механік порівняно з іншими рішеннями на ринку.
2. **Сформовано концепт та геймдизайн-документ (GDD):** Детально розроблено правила гри та основний ігровий цикл (Core Loop), що базується на хвильовій системі виживання. Визначено ключові механіки, які стимулюватимуть активність гравця та створюватимуть необхідну атмосферу: тактичний менеджмент набоїв, неможливість стрільби під час бігу, система автоматичної регенерації здоров'я та локальна система шкоди (вищий урон при влучанні в голову).

3. **Визначено технічні вимоги:** Сформовано чітку постановку задачі розробки, яка поділяє весь процес на логічні етапи: проектування контролера гравця, розробка штучного інтелекту ворогів на базі навігаційної сітки, програмування ігрового режиму, створення користувацького інтерфейсу та левел-дизайну.

Таким чином, перший розділ повністю формує теоретичне, концептуальне та технічне підґрунтя, необхідне для переходу до етапу безпосередньої практичної реалізації ігрового прототипу, що буде розглянуто в наступних розділах роботи.

РОЗДІЛ 2. ПРОЕКТУВАННЯ АРХІТЕКТУРИ ТА ІГРОВИХ МЕХАНІК

2.1. Розробка логіки персонажа та контролера зброї (пересування, анімації від стін, регенерація здоров'я)

Основою взаємодії гравця з ігровим світом є клас персонажа (Character). Для реалізації управління та обробки вводу було використано сучасну підсистему Unreal Engine 5 - **Enhanced Input System**. Цей підхід дозволяє гнучко налаштовувати дії (Input Actions) та прив'язувати їх до контекстів (Mapping Contexts).

Логіка пересування базово базується на компоненті Character Movement Component (див. рис. 2.1.). Система підтримує три стани переміщення: звичайну ходьбу (базове значення Max Walk Speed), спринт (зі збільшеним показником швидкості) та присідання.

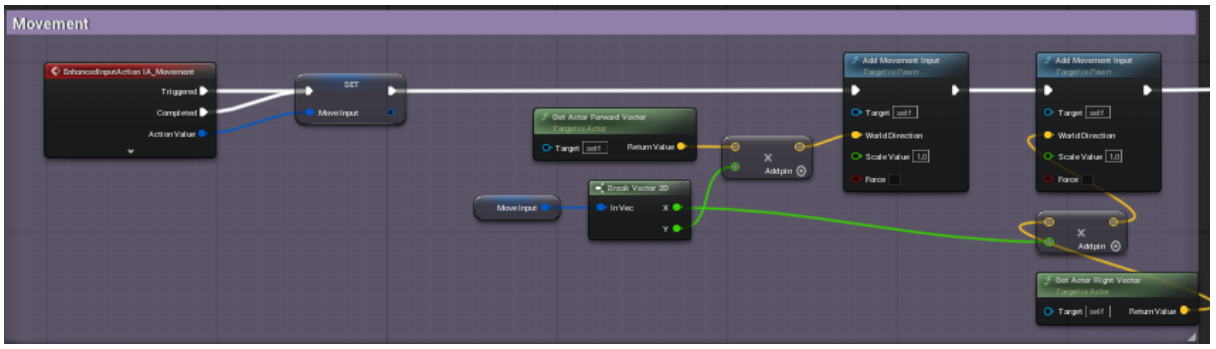

Рисунок 2.1. Реалізація базового пересування персонажа за допомогою Enhanced Input

Джерело: Розроблене автором

Механіка присідання має складну логіку обробки вводу: розроблено алгоритм, який підтримує як утримання клавіші (Hold), так і перемикання стану одноразовим натисканням (Toggle) (див. рис. 2.2.). Під час переходу в стан присідання (Custom Crouch) динамічно зменшується висота капсули колізії (Capsule Half Height), що дає змогу гравцю долати перешкоди та пролізати у важкодоступні місця рівня, при цьому швидкість пересування автоматично знижується.

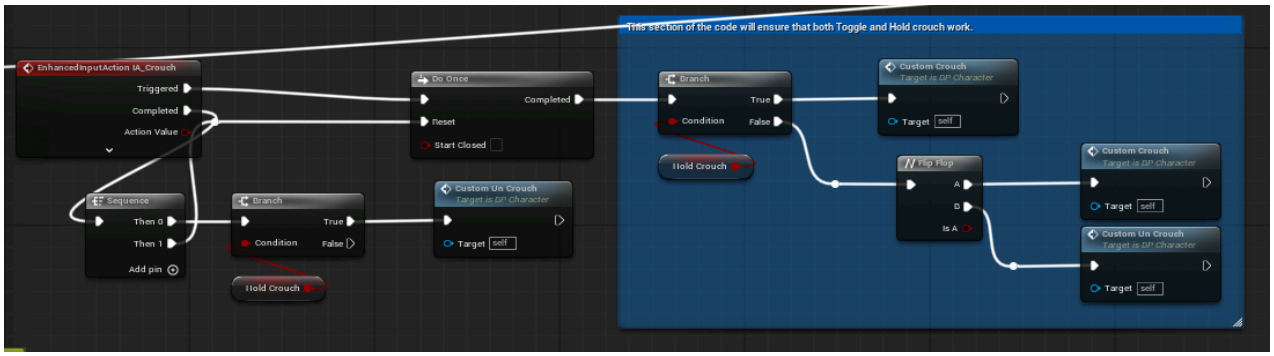

Рисунок 2.2. Алгоритм обробки присідання (Hold та Toggle)

Джерело: Розроблене автором

Важливою деталлю імпакту та реалізму шутера від першої особи є реакція зброї на оточення. Оскільки візуальна складова гравця представлена виключно моделлю рук (Arms Mesh), було реалізовано механіку тактичного опускання зброї при наближенні до стін чи укриттів. Для визначення перешкод використовується метод трасування променя - Line Trace By Channel (див. рис. 2.3.). Промінь випускається з позиції камери (Get World Location) у напрямку її погляду (Get Forward Vector) на задану дистанцію. Якщо промінь фіксує зіткнення зі статичним об'єктом, булева змінна Close_To_Wall набуває значення True.

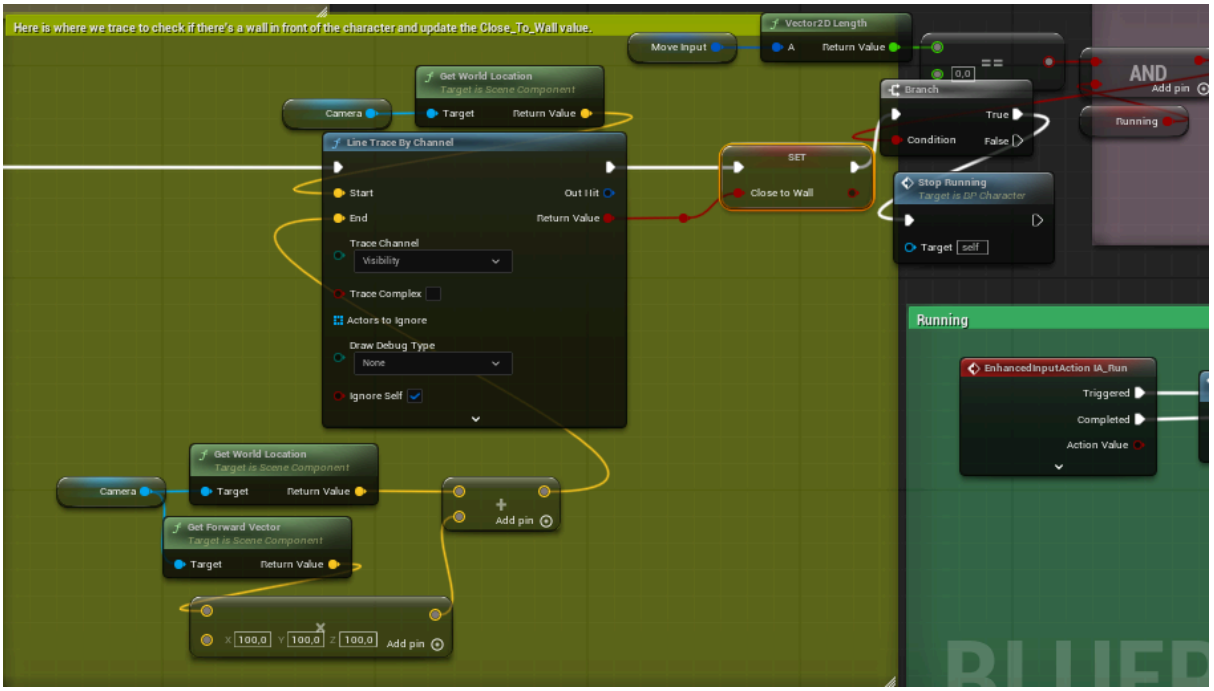

Рисунок 2.3. Визначення перешкоди перед гравцем методом трасування променя

Джерело: Розроблене автором

Ця змінна безпосередньо передається в анімаційне креслення (Animation Blueprint). За допомогою вузла Blend Poses by bool (див. рис. 2.4.) система плавно перемикає базову кешовану позу (Cached Pose) на позу зі згорнутою зброєю (CPose_Lowered_State), що візуально інформує гравця про неможливість ведення вогню впритул до перешкоди.

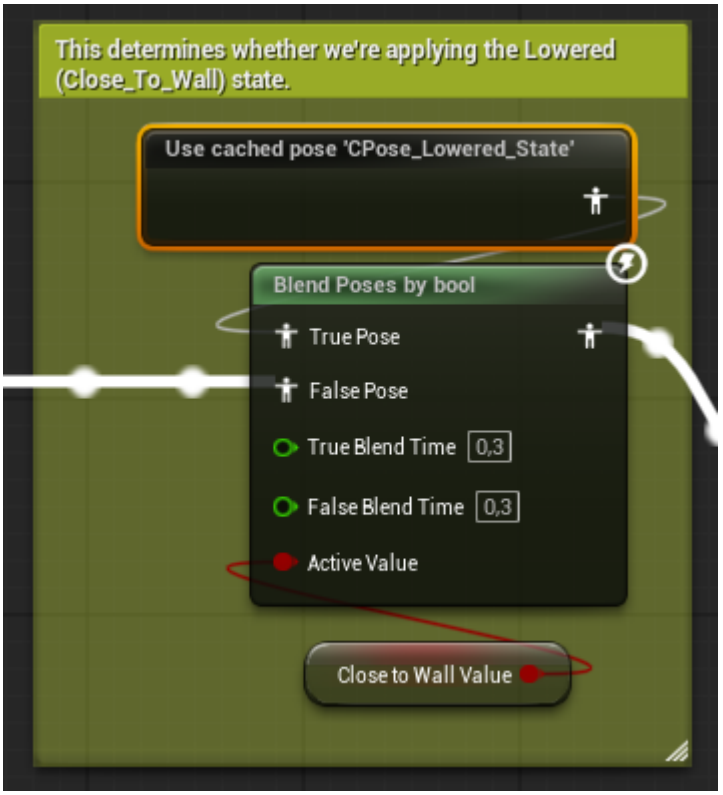

Рисунок 2.4. Перемикання анімаційних станів зброї в Animation Blueprint

Джерело: Розроблене автором

Контролер зброї та механіка стрільби побудовані на основі фізичних снарядів (Projectiles). При натисканні кнопки вогню відбувається генерація (Spawn) об'єкта кулі, який має власні налаштування колізії, розміру та використовує компонент Projectile Movement. Куля летить по прямій траєкторії від центру камери до цілі. Відмова від складної балістики на користь прямолінійної траєкторії зумовлена аркадною та динамічною природою зомбі-шутера.

Перезаряджання зброї ініціюється гравцем вручну. Процес супроводжується відповідною анімацією рук від першої особи та звуковими ефектами (SFX). Оновлення лічильника набоїв у логіці Blueprints не є миттєвим: воно виконується через вузол затримки (Delay). Час затримки скрупульозно підігнаний під таймінги

анімації таким чином, щоб додавання куль у змінні відбувалося рівно в той момент, коли персонаж візуально вставляє новий магазин у гвинтівку. Це забезпечує максимальну синхронізацію логіки та візуалу.

Система здоров'я та його регенерації інкапсульована в Blueprint персонажа. Відновлення очок здоров'я ініціюється автоматично після отримання шкоди, якщо гравець уникає нових атак. Логіка циклу регенерації реалізована за допомогою вузла Set Timer by Event. Таймер налаштовано на спрацьовування кожні 5 секунд, відновлюючи 5% здоров'я за ітерацію. Цикл триває доти, доки рівень здоров'я не досягне максимальної позначки, або доки гравець не отримає нову шкоду (що скидає таймер регенерації).

2.2. Проектування штучного інтелекту (Pawn Sensing, зміна швидкості зомбі, реакції на влучання

Штучний інтелект (ШІ) ворогів у грі жанру зомбі-шутер є ключовим елементом, що формує темп та складність ігрового процесу. Для уникнення ситуацій, коли гравець змушений витрачати час на пошук ворогів, що заблукали на локації, базову навігаційну логіку зомбі було побудовано на постійному відстеженні цілі. За замовчуванням вороги завжди знають приблизне розташування гравця і повільно рухаються в його напрямку за допомогою системи навігаційної сітки (NavMesh).

Однак для створення відчуття раптової небезпеки було імплементовано двоступеневу систему агресії на базі компонента Pawn Sensing (див. рис. 2.5.). Цей компонент дозволяє ворогу «бачити» та реагувати на об'єкти в ігровому світі. Налаштування сенсора визначають периферійний зір зомбі (Peripheral Vision Angle) на рівні 90 градусів та радіус зору (Sight Radius) на 1440 одиниць.

Рисунок 2.5. Налаштування параметрів компонента Pawn Sensing

Джерело: Розроблене автором

Коли персонаж гравця потрапляє у зону дії Pawn Sensing, у Blueprint-класі зомбі викликається подія On See Pawn (див. рис. 2.6.). Вона ініціює перехід ШІ у стан активного переслідування: змінна Max Walk Speed компонента Character Movement миттєво збільшується до значення 500, перемикаючи ворога з повільної ходьби на швидкий спринт. Анімаційний контролер автоматично згладжує перехід між цими станами руху.

Рисунок 2.6. Логіка активації спринту при виявленні гравця

Джерело: Розроблене автором

Логіка завдання шкоди гравцю базується на просторових перевірках. Коли зомбі досягає цілі (спрацьовує логіка Run to Player), ініціюється кастомна подія

AttackPlayer (див. рис. 2.7.). Вона запускає анімаційний монтаж удару (Play Montage) у супроводі відповідного звукового ефекту. Для точної реєстрації влучання в момент удару використовується вузол Sphere Overlap Actors, який генерує невидиму сферу радіусом 80 одиниць перед моделлю зомбі. Якщо гравець (об'єкт класу Character) знаходиться всередині цієї сфери в момент перевірки, вузол Apply Damage віднімає 20 одиниць здоров'я. Після завершення атаки перевіряється дистанція: якщо гравець відбіг, зомбі знову повертається до стану переслідування.

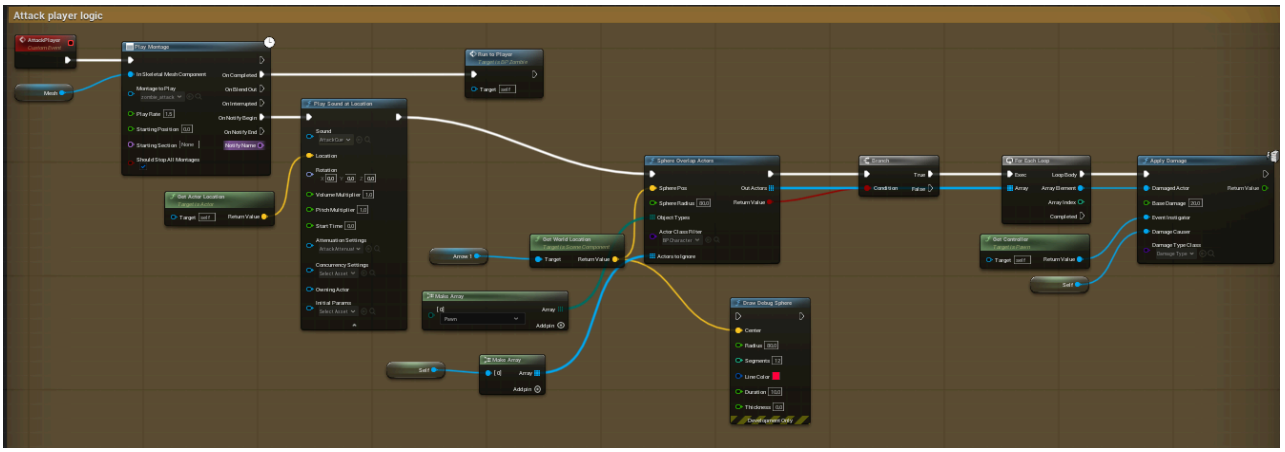

Рисунок 2.7. Реалізація системи ближнього бою через просторове перекриття (Sphere Overlap)

Джерело: Розроблене автором

Важливою складовою імпакту є реакція ШІ на отримання шкоди. Цей процес поділено на два етапи для оптимізації логіки. Визначення зони влучання (у голову чи в тіло) відбувається всередині Blueprint-класу самої кулі на основі фізичних матеріалів (Physical Materials). Після розрахунку множника шкоди куля передає підсумкове значення у клас зомбі.

Отримання шкоди самим зомбі обробляється через універсальний вузол Event AnyDamage (див. рис. 2.8.). Спочатку система перевіряє, чи живий ворог (булева змінна Is Dead?). Далі значення здоров'я (Health) зменшується на величину отриманої шкоди. Якщо залишок здоров'я стає меншим або дорівнює нулю, викликається подія смерті (Death).

Якщо ж ворог залишається живим, запускається анімація реакції на влучання (Zombie_Reactic). Візуальний зворотний зв'язок також доповнюється генерацією ефекту крові (VFX). Завдяки системі кешування поз (Cached Poses) в Animation

Blueprint, відтворення анімації відкидання застосовується лише до верхньої частини тіла (через Layered blend per bone), що дозволяє зомбі продовжувати рух ногами без зупинки чи конфлікту анімацій під час отримання шкоди.

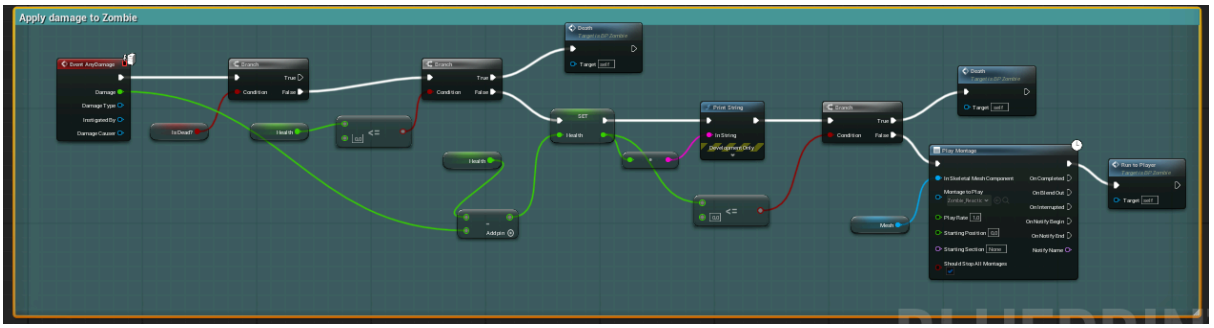

Рисунок 2.8. Обробка отриманої шкоди, перевірка стану смерті та відтворення реакції зомбі

Джерело: Розроблене автором

Загалом, розроблена архітектура штучного інтелекту забезпечує стабільну та передбачувану роботу ворогів у межах ігрового рівня. Використання компонента просторового виявлення у поєднанні з навігаційною сіткою та модульною системою обробки шкоди дозволило створити динамічних супротивників, які адекватно реагують на дії гравця. Запропонований підхід до програмування ШІ через Blueprints є гнучким і дозволяє легко масштабувати логіку в майбутньому, наприклад, додаючи нові типи атак або змінюючи параметри агресії для балансування складності гри.

2.3. Реалізація ігрового режиму та системи спавну хвиль ворогів

Керування глобальним станом гри, правилами та ігровим циклом (Core Loop) здійснюється через спеціальний клас ігрового режиму - Game Mode. У межах даного прототипу основним завданням цього класу є контроль системи генерації (спавну) хвиль ворогів, підрахунок вбитих супротивників та перевірка умов перемоги гравця.

Математична модель складності гри побудована на лінійному зростанні кількості ворогів із кожною новою хвилею. Для розрахунку загальної кількості зомбі на поточний раунд використовується формула, де номер поточної хвилі (Current Wave) множиться на базовий коефіцієнт (у даному випадку - на 4). Отримане

значення записується у змінні Zombie Total (загальна кількість для спавну) та Zombies Left (лічильник живих ворогів).

Щоб уникнути скупчення всіх ворогів в одній точці, загальна кількість зомбі ділиться на два рівні потоки. Генерація ворогів реалізована за допомогою циклів (For Loop), які ініціюють створення об'єктів через вузол SpawnActor BP_Zombie. Кожен цикл прив'язаний до окремої точки спавну на локації, координати якої заздалегідь збережені у змінних типу Transform (Spawn Loc 1 та Spawn Loc 2) (див. рис. 2.9.).

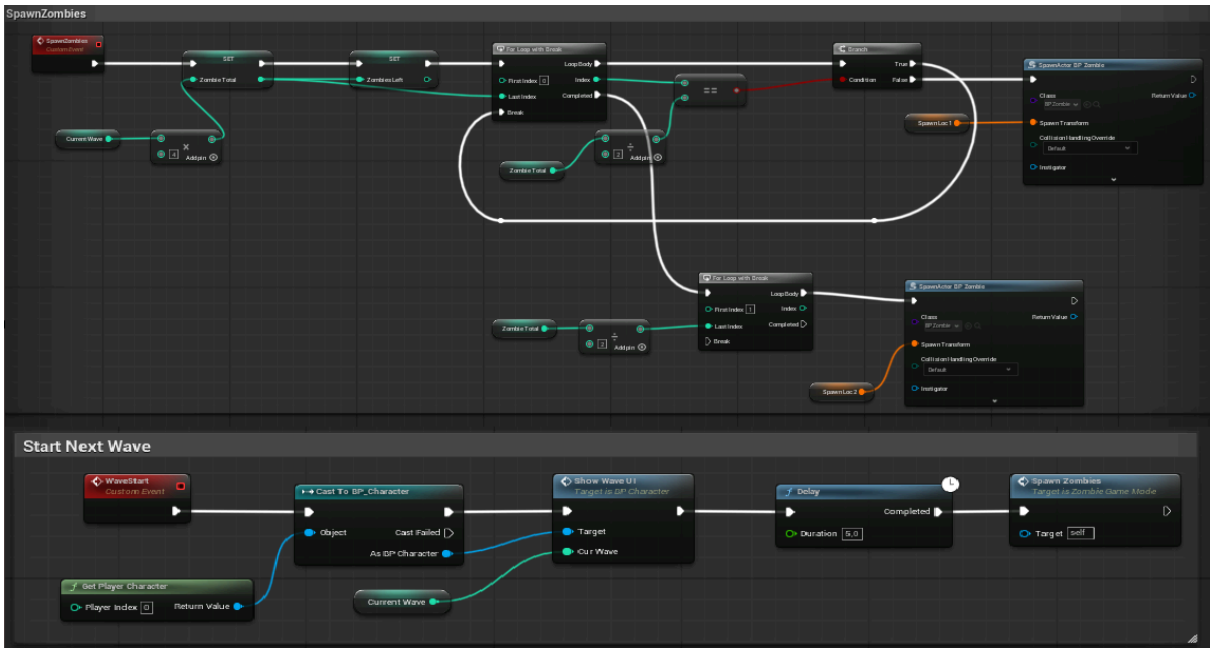

Рисунок 2.9. Алгоритм розрахунку кількості ворогів та генерації хвиль у Game Mode

Джерело: Розроблене автором

Важливим технічним нюансом при масовому спавні об'єктів в одній точці є запобігання колізіям, коли моделі ворогів можуть застрягти одна в одній. Для вирішення цієї проблеми у вузлі SpawnActor було налаштовано параметр Spawn Collision Handling Override. Алгоритм рушія автоматично намагається знайти вільне місце навколо заданої точки спавну. Навіть якщо ідеальної вільної позиції немає, система все одно примусово створює ворога, динамічно коригуючи його координати для уникнення критичних накладень фізичних капсул (Capsule Components).

Прогресія ігрової сесії залежить від успішності дій гравця. При кожному знищенні зомбі викликається кастомна подія ZombiesKilled, яка зменшує значення лічильника Zombies Left на одиницю. Після цього система перевіряє, чи дорівнює

залишок нулю. Якщо всі вороги у хвилі знищені, ініціюється подія оновлення хвилі (Wave Increment).

Алгоритм оновлення містить перевірку умови перемоги (Win Condition). Система порівнює поточну хвилю (Current Wave) з максимально передбаченою кількістю хвиль у грі (Max Waves). Якщо гравець успішно завершив останню хвилю, викликається подія перемоги гравця (Player Wins). У протилежному випадку номер поточної хвилі збільшується на одиницю, і запускається подія старту наступного раунду (Wave Start) (див. рис. 2.10.).

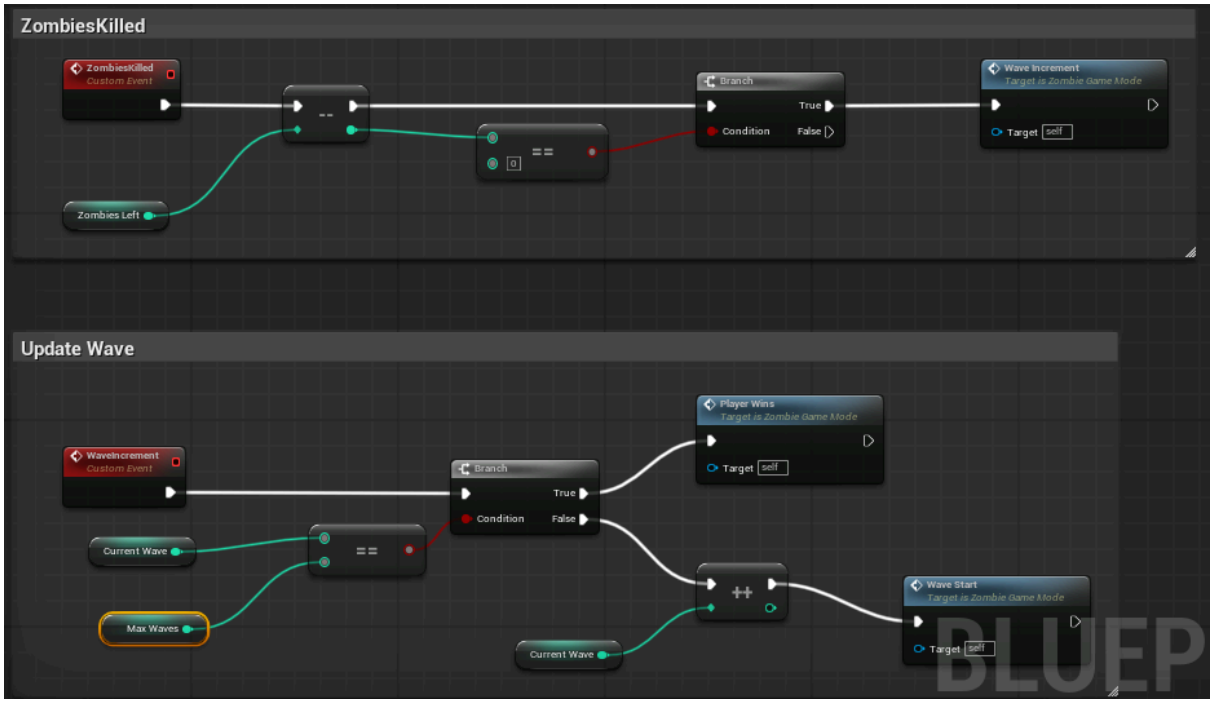

Рисунок 2.10. Логіка підрахунку вбитих ворогів та переходу до наступної хвилі

Джерело: Розроблене автором

Таким чином, розроблена архітектура ігрового режиму створює замкнений, циклічний та масштабований ігровий процес. Вона дозволяє легко балансувати складність гри, змінюючи лише математичні коефіцієнти розрахунку кількості ворогів або додаючи нові точки генерації супротивників, без необхідності переписувати базову логіку системи.

Висновок до розділу 2

У другому розділі було спроектовано та реалізовано базову архітектуру ігрового проєкту, а також розроблено ключові ігрові механіки з використанням

системи візуального програмування Blueprints в ігровому рушії Unreal Engine 5. Усі імплементовані системи утворюють цілісний програмний фундамент для динамічного шутера від першої особи.

У ході роботи було розроблено логіку ігрового персонажа на базі сучасної підсистеми Enhanced Input System, що забезпечило гнучке керування станами переміщення (ходьба, спринт, присідання). Створено комплексний контролер зброї, що використовує фізичні снаряди (Projectiles) для ведення вогню, імплементовано систему ручного перезаряджання з точною синхронізацією логіки та анімації. Окрему увагу приділено імпакту взаємодії з оточенням: розроблено механіку тактичного опускання зброї перед перешкодами за допомогою методу трасування променів (Line Trace) та систему автоматичної регенерації здоров’я.

Для забезпечення динамічного ігрового процесу було спроектовано штучний інтелект супротивників (зомбі). Використання навігаційної сітки (NavMesh) та компонента Pawn Sensing дозволило створити дворівневу систему агресії зі зміною швидкості руху при виявленні гравця. Логіку завдання шкоди гравцеві побудовано на основі просторових перевірок (Sphere Overlap), а система отримання шкоди самим ШІ підтримує реєстрацію влучань у різні частини тіла з відповідними візуальними реакціями та анімаційним кешуванням поз.

З метою об'єднання окремих механік у повноцінний ігровий цикл (Core Loop) було налаштовано клас ігрового режиму (Game Mode). Розроблено математичну модель масштабування складності гри через генерацію хвиль ворогів. Створено надійну систему спавну супротивників, яка автоматично запобігає колізійним накладенням моделей, а також імплементовано алгоритми підрахунку знищених цілей та перевірки умов перемоги гравця.

Таким чином, розроблена архітектура є стабільною, оптимізованою та легко масштабованою для подальшого розширення контенту.

РОЗДІЛ 3. ДИЗАЙН РІВНІВ, ІНТЕРФЕЙС ТА АУДІОВІЗУАЛЬНЕ ОФОРМЛЕННЯ

3.1. Створення ігрової локації та левел-дизайн (укриття, геометрія)

Левел-дизайн (проектування рівня) є критично важливим етапом розробки, оскільки саме геометрія простору диктує гравцеві правила поведінки та тактику ведення бою. Для даного прототипу було обрано сеттинг підземної науково-дослідної лабораторії (або промислової фабрики) у стилі Sci-Fi (див. рис. 3.1.). Для побудови оточення використовувався модульний набір оптимізованих 3D-моделей (Static Meshes) з відкритої бібліотеки асетів. Це дозволило наповнити рівень деталями (ящики, труби, панелі) і позбутися відчуття «порожнього простору» (див. рис. 3.2.).

Рисунок 3.1. Загальний вигляд головної зали ігрової локації у Sci-Fi сеттингу

Джерело: Розроблене автором

Архітектура рівня не є лінійною. Вона складається з великої центральної зали, яка з'єднана мережею вужчих коридорів та окремих кімнат. Проектування локації здійснювалося вручну з використанням ітеративного підходу: після кожного етапу розстановки об'єктів проводилося плейтестування (Playtest) для перевірки динаміки переміщення та контролю точок інтересу.

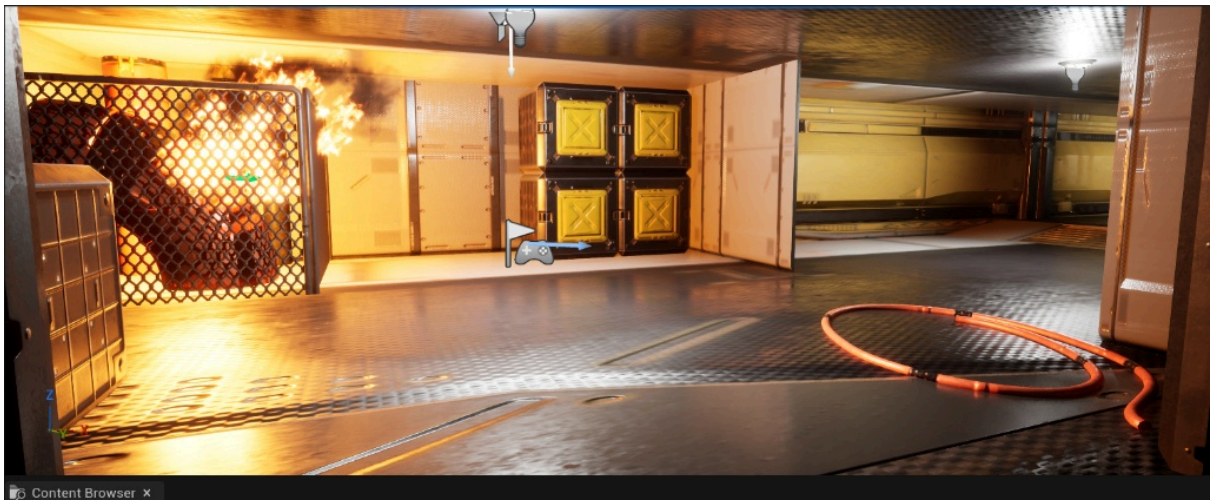

Рисунок 3.2. Стартова зона гравця та візуальне наповнення коридорів

Джерело: Розроблене автором

Оскільки ігровий цикл побудований на постійному русі та уникненні прямого контакту з натовпом ворогів, геометрія рівня містить спеціальні зони для тактичного маневрування. Для розриву дистанції з ШІ передбачені вузькі проходи, через які гравець може пролізти за допомогою механіки присідання (Crouch), а також вікна, через які можна перестрибнути. Крім того, наявність великої кількості стін та укриттів тісно взаємодіє з розробленою раніше механікою тактичного опускання зброї (Weapon Collision).

Щоб розширити тактичні можливості гравця, локацію було доповнено інтерактивною системою динамічних барикад. На рівні розміщено спеціальні тригер-зони (Blueprint-спавнери), які візуально позначені спрайтом у вигляді паркану. Це сигналізує гравцеві про можливість створити тимчасове укриття (див. рис. 3.3.).

Рисунок 3.3. Кімната для тактичного відступу з індикатором зони побудови барикади

Джерело: Розроблене автором

Процес зведення барикади потребує часу, що додає грі елемент ризику (див. рис. 3.4.). Коли гравець входить у зону колізії спавнера (On Component Begin Overlap), система перевіряє його клас (Cast to BP_Character). Якщо ідентифікація успішна, запускається таймер (Set Timer by Event) на 3 секунди. Важливою умовою є відсутність сторонніх об'єктів (наприклад, зомбі) у зоні забудови під час відліку. Перед генерацією об'єкта також виконується перевірка Is Valid, яка блокує спавн нової барикади, якщо попередня на цьому місці ще не зруйнована. Успішна генерація супроводжується відтворенням відповідного просторового звуку (SFX).

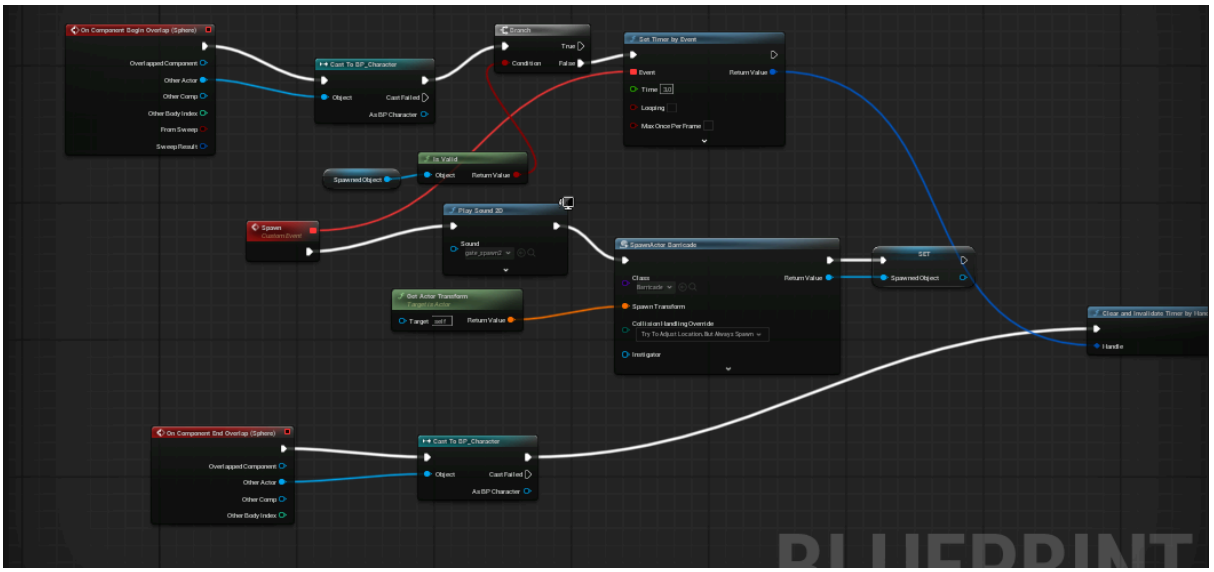

Рисунок 3.4. Програмна логіка перевірки умов та генерації барикади (Blueprint)

Джерело: Розроблене автором

Збудована барикада функціонує як фізична перешкода, що блокує прохід для ворогів (навігаційна сітка NavMesh динамічно оновлюється або ігнорується), дозволяючи гравцеві перевести подих та відновити здоров'я. Проте барикада не є безсмертною: вона має власний запас міцності (Health). Отримання шкоди обробляється через вузол Event AnyDamage (див. рис. 3.5.). Зруйнувати барикаду можуть як зомбі під час атаки, так і сам гравець за допомогою вогнепальної зброї. Останнє рішення (Friendly Fire по барикаді) є свідомим кроком геймдизайну, який гарантує, що гравець не заблокує сам себе в глухій кімнаті. Знищення об'єкта також ініціює звуковий ефект руйнування та видаляє його зі світу (Destroy Actor).

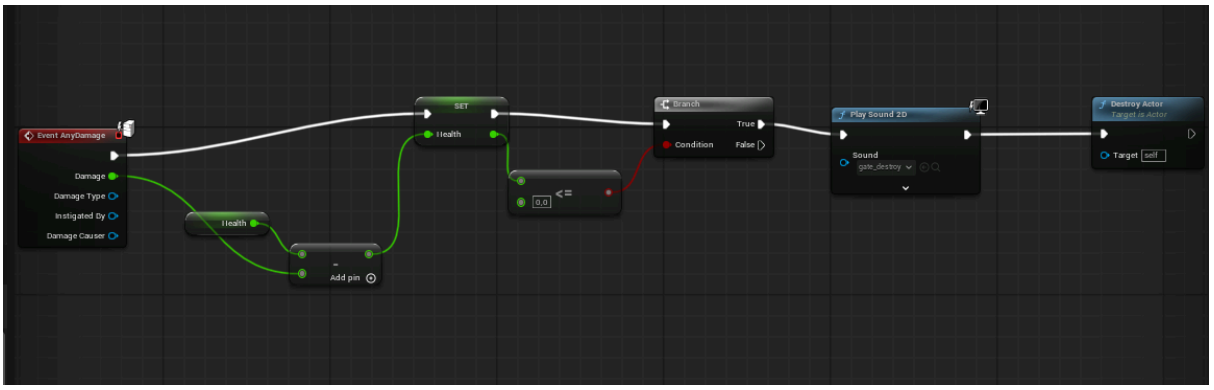

Рисунок 3.5. Логіка отримання шкоди та знищення об'єкта барикади

Джерело: Розроблене автором

Підсумовуючи, розроблений дизайн рівня не лише виконує естетичну функцію, створюючи атмосферу науково-фантастичного комплексу, але й безпосередньо впливає на геймплей. Продумане розташування укриттів, просторі зали для маневрування та інтеграція системи динамічних барикад формують глибокий тактичний досвід, стимулюючи гравця постійно аналізувати оточення та приймати швидкі рішення під час відбиття хвиль ворогів.

3.2. Розробка користувацького інтерфейсу (UI) (меню, HUD, оверлей HP, екран смерті

Користувацький інтерфейс (User Interface) є основним інструментом взаємодії між гравцем та ігровою системою. Для розробки інтерфейсу в межах проєкту було використано інструментарій Unreal Motion Graphics (UMG), який дозволяє

створювати візуально привабливі та функціональні віджети (Widgets) із власною логікою на Blueprints.

Розробка почалася зі створення Головного меню (див. рис. 3.6.). Воно реалізоване як окремий рівень із власним інтерфейсом та музичним супроводом. Меню містить функціональні кнопки для старту гри (завантаження основної карти) та виходу з додатка. Дизайн меню витриманий у загальній Sci-Fi стилістиці проєкту.

Рисунок 3.6. Візуальне оформлення Головного меню гри

Джерело: Розроблене автором

Під час безпосереднього ігрового процесу на екрані відображається **HUD (Heads-Up Display)** (див. рис. 3.7.). Він містить усі необхідні параметри для орієнтації гравця:

- **Індикатор прицілу (Crosshair):** динамічний елемент у центрі екрана, який автоматично приховується при переході гравця у режим прицілювання через мушку зброї (ADS) для кращого огляду.
- **Параметри здоров'я:** графічна шкала (Progress Bar), що відображає поточний стан персонажа.
- **Індикатор боєзапасу:** текстові блоки, що показують кількість набоїв у магазині та загальний резерв.

- **Інформація про хвилі:** дані про поточну хвилю та загальну кількість етапів. При завершенні поточної хвилі на екрані з’являється службове текстове повідомлення про початок наступного раунду.

Оновлення більшості елементів HUD реалізовано через механізм **Bind** (динамічна прив'язка до змінних), що забезпечує актуальність даних у реальному часі.

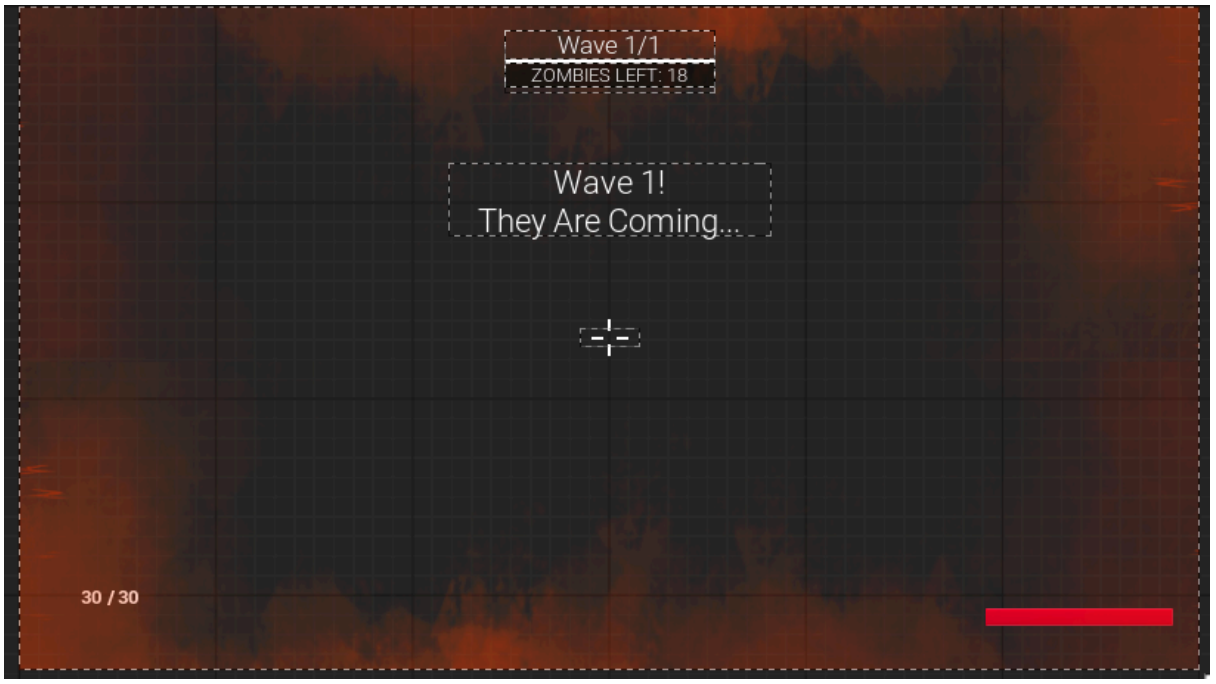

Рисунок 3.7. Компоненти ігрового інтерфейсу (HUD) під час ігрового процесу

Джерело: Розроблене автором

Для посилення візуального відгуку на отримання шкоди було розроблено **систему оверлея поранень**. Це графічний елемент (текстура крові по краях екрана), що входить до складу основного віджета. Програмна логіка через Animation Blueprint віджета змінює прозорість (Opacity) цієї картинки, створюючи ефект пульсації при критичному рівні здоров’я, що слугує інтуїтивним сигналом для гравця про небезпеку (див. рис. 3.8.).

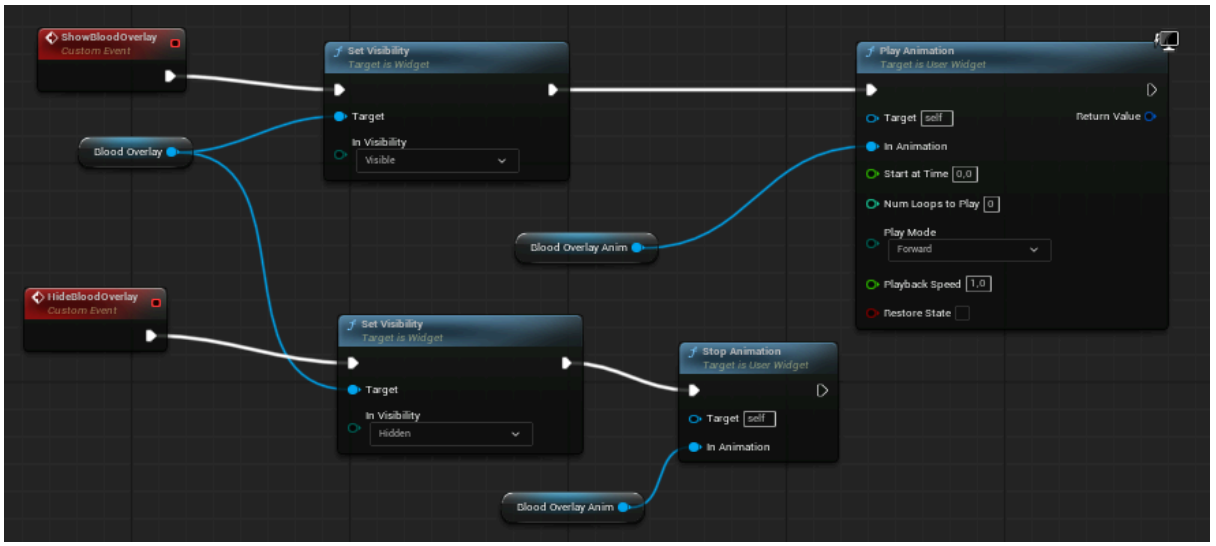

Рисунок 3.8. Логіка анімації прозорості оверлея поранення

Джерело: Розроблене автором

Особлива увага приділена екранам завершення ігрової сесії - **екранам Перемоги та Смерті** (див. рис. 3.9.). Технічно вони реалізовані через один шаблон віджета, де змінюється лише текстовий контент залежно від результату. При активації цих екранів ігрова логіка виконує ряд критичних дій: блокує рух персонажа, викликає вузол Set Input Mode UI Only (перемикання введення виключно на інтерфейс) та ініціює Show Mouse Cursor (див. рис. 3.10.). Це дозволяє гравцеві вільно користуватися кнопками для перезапуску рівня, виходу в головне меню або повного виходу з гри.

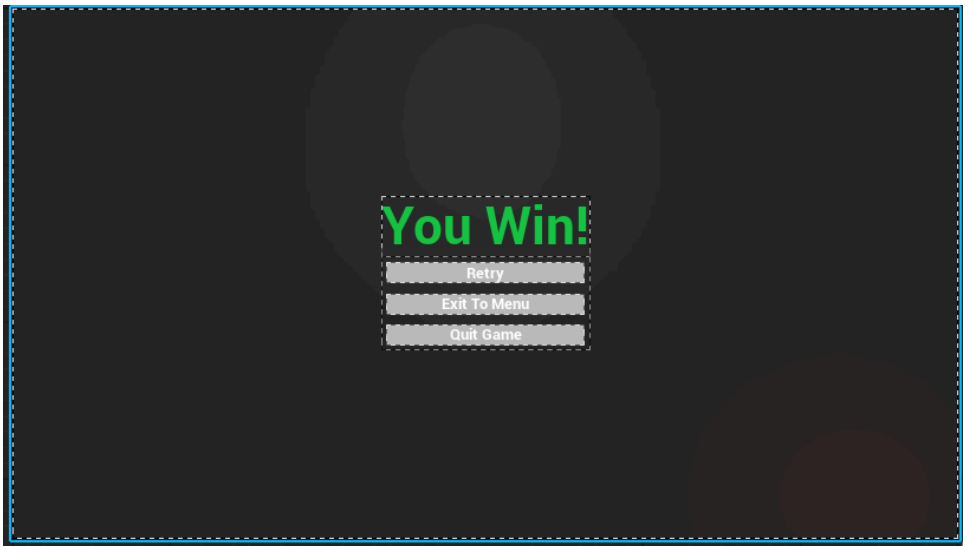

Рисунок 3.9. Інтерфейс екрана перемоги (Win Screen)

Джерело: Розроблене автором

Перехід між станами гри (гра-меню-рестарт) забезпечується через взаємодію віджетів із класом Player Controller та функціями керування рівнями (Open Level).

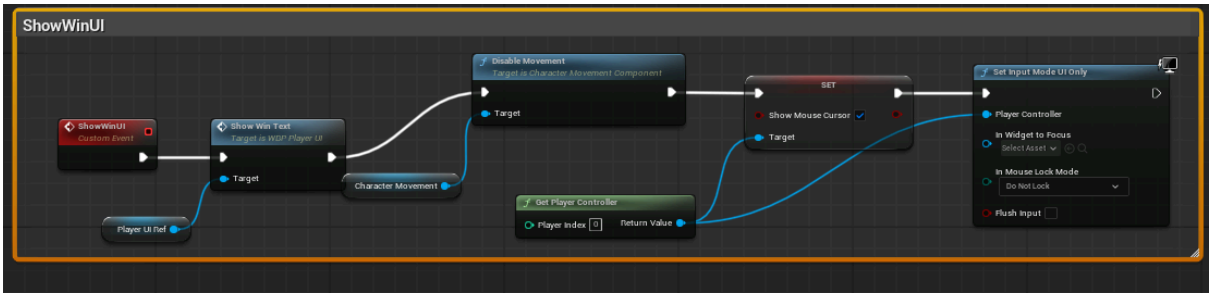

Рисунок 3.10. Програмна логіка керування станами введення та відображення курсора

Джерело: Розроблене автором

Створена система користувацького інтерфейсу забезпечує повний контроль над ігровим процесом та надає гравцеві необхідний обсяг інформації у зручній формі. Використання подійного підходу та динамічних прив'язок дозволило мінімізувати навантаження на систему, зберігаючи при цьому високу швидкість відгуку інтерфейсу на дії користувача або зміну ігрових параметрів.

3.3. Інтеграція візуальних ефектів (VFX) та звукового супроводу

Аудіовізуальне оформлення є невід'ємною складовою сучасних відеоігор, що відповідає за занурення гравця в атмосферу та забезпечення якісного зворотного зв'язку (Game Feel). У межах розробки даного прототипу було імплементовано комплекс візуальних ефектів (VFX) та систему просторового звуку.

Візуальні ефекти розділені на дві категорії: бойові та ефекти оточення. До бойових належать ефекти вогнепальної зброї та реакції на влучання. Під час пострілу з автомата генерується спалах полум'я та дим із дула (Muzzle Flash) (див. рис. 3.11.). Оскільки рівень виконаний у Sci-Fi стилістиці з перевагою металевих поверхонь, під час влучання кулі у геометрію рівня через вузол Spawn Emitter at Location викликається ефект розльоту іскор. Якщо ж куля влучає у ворога, система генерує ефект бризок крові, що слугує інтуїтивним підтвердженням успішного попадання.

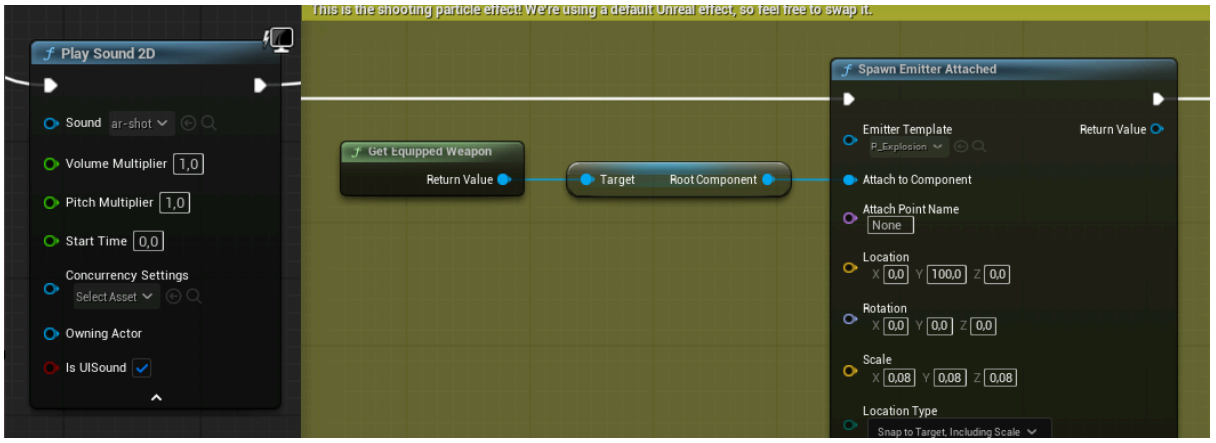

Рисунок 3.11. Логіка генерації візуальних ефектів пострілу та відтворення звуку зброї

Джерело: Розроблене автором

Для пожвавлення ігрової локації на карті було вручну розставлено ефекти оточення (див. рис. 3.12.). До них належать викиди полум'я з пошкоджених газових труб, дим із систем вентиляції та іскри з несправних електронних дверей. Ці елементи підкреслюють атмосферу закинутої та зруйнованої підземної лабораторії.

Рисунок 3.12. Візуальні ефекти оточення на ігровій локації

Джерело: Розроблене автором

Звуковий супровід проєкту складається з фонової музики (Ambient), звуків гравця та аудіосистеми ШІ. Дії самого гравця, такі як постріли та перезаряджання зброї, реалізовані через вузол Play Sound 2D. Це рішення зумовлене тим, що звуки власної зброї не повинні піддаватися просторовому спотворенню і завжди мають звучати чітко. Звуки кроків синхронізовані з анімаціями переміщення: їхній темп динамічно змінюється залежно від того, чи гравець іде, чи використовує спринт (див. рис. 3.13.).

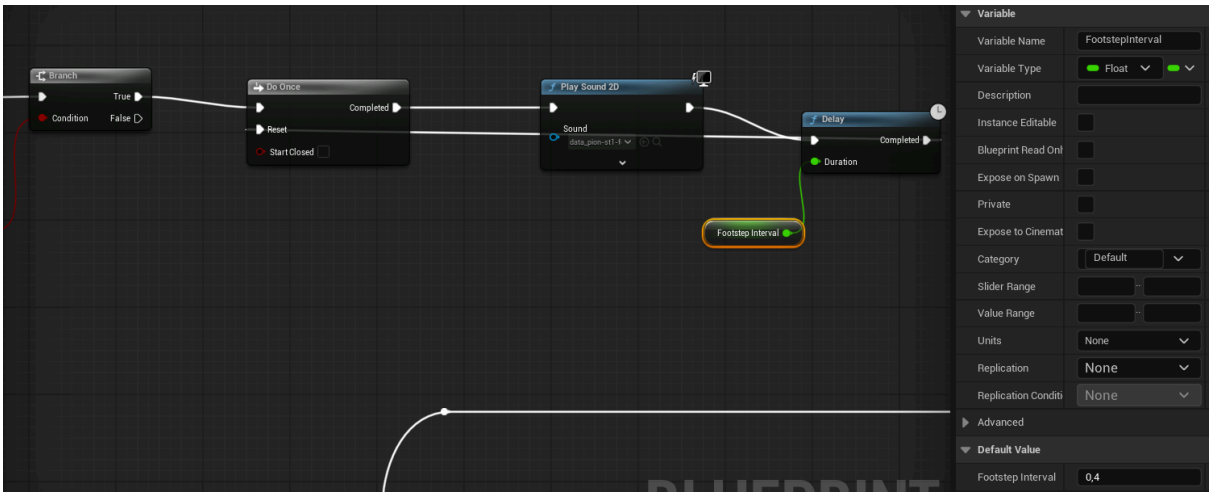

Рисунок 3.13. Синхронізація звуків кроків з анімацією руху гравця

Джерело: Розроблене автором

Особливу увагу приділено звуковому дизайну ворогів. Зомбі видають характерне гарчання під час переслідування та атаки, а також специфічні звуки при отриманні шкоди та смерті. Для уникнення ефекту аудіальної втоми (коли багаторазове повторення одного звуку дратує гравця), аудіофайли ворогів організовані у Sound Cues із використанням вузла Random (див. рис. 3.14.). Цей вузол при кожному виклику випадковим чином обирає один із кількох доступних варіантів звуку.

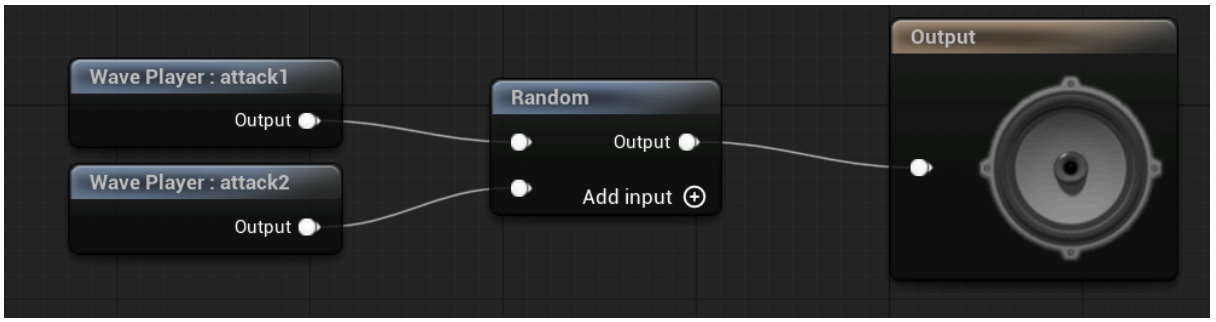

Рисунок 3.14. Налаштування випадкового відтворення аудіофайлів у Sound Cue

Джерело: Розроблене автором

Для тактичного орієнтування гравця звуки ворогів відтворюються через вузол Play Sound at Location. До кожного звукового об'єкта ШІ застосовано налаштування згасання звуку (Sound Attenuation) (див. рис. 3.15.). Це дозволяє рушію динамічно розраховувати гучність та позиціювання звуку в 3D-просторі залежно від відстані між гравцем та джерелом шуму. Завдяки цьому гравець може на слух визначати наближення небезпеки з темних коридорів.

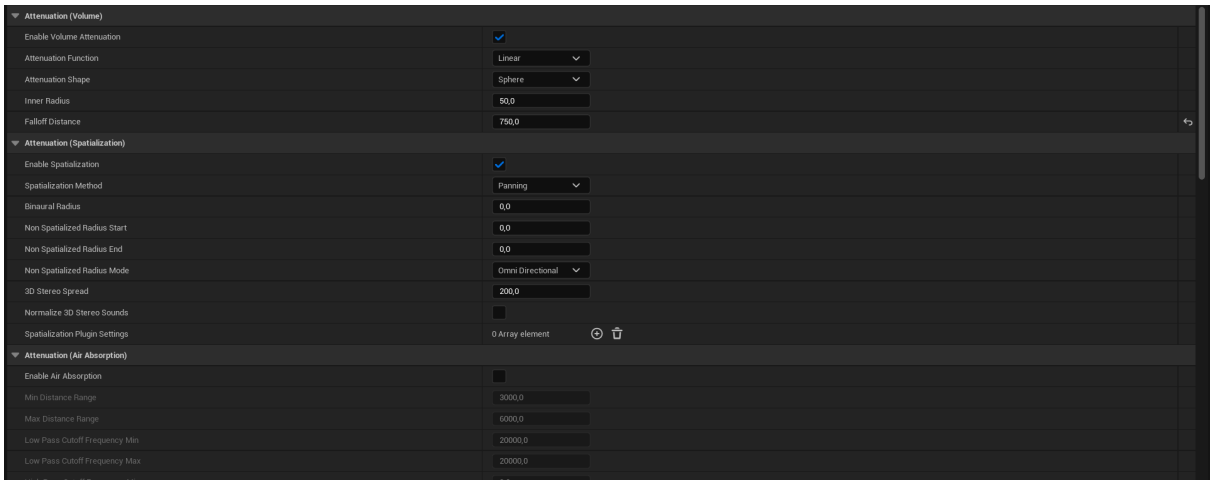

Рисунок 3.15. Параметри просторового згасання звуку (Sound Attenuation)

Джерело: Розроблене автором

Комплексна взаємодія візуальних ефектів та звукового супроводу критично важлива для реєстрації ігрових подій. Наприклад, подія смерті зомбі (див. рис. 3.16.) одночасно викликає зупинку логіки ШІ, відтворення звуку смерті та генерацію відповідних VFX, що робить сцену цілісною та реалістичною.

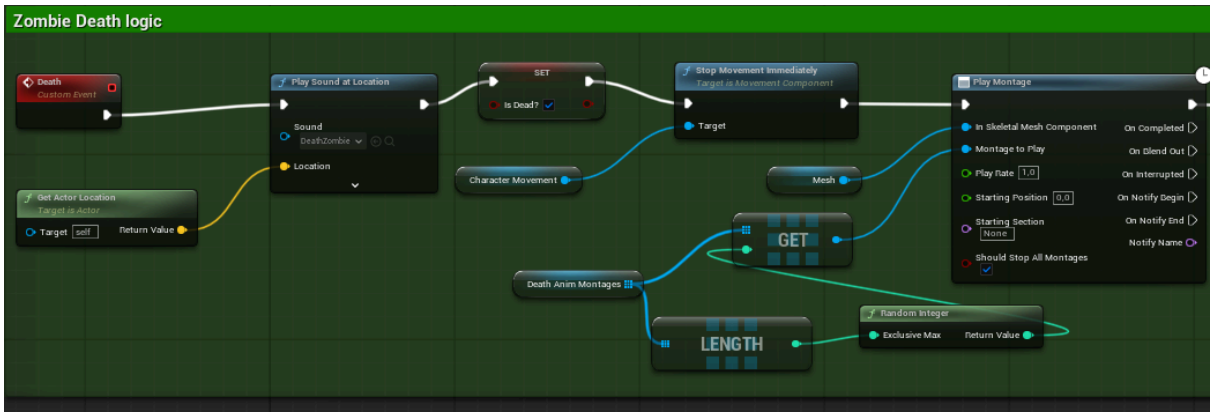

Рисунок 3.16. Комплексна обробка події смерті зомбі з аудіовізуальним супроводом

Джерело: Розроблене автором

Інтегрована система аудіовізуальних ефектів успішно виконує свої функції: вона надає гравцеві необхідний зворотний зв'язок про стан ігрового світу, підкреслює обраний Sci-Fi сеттинг та значно підвищує загальний рівень занурення в ігровий процес. Налаштування просторового звуку та випадкової генерації семплів роблять ігровий досвід більш природним та комфортним для тривалих ігрових сесій.

3.4. Опис готового продукту та керівництво користувача

Результатом виконання практичної частини кваліфікаційної роботи є повністю функціональний прототип (вертикальний зріз) відеогри в жанрі шутера від першої особи (FPS) із елементами виживання (див. рис. 3.17.). Розроблений програмний продукт об'єднує в собі всі раніше описані системи: контролер персонажа, логіку штучного інтелекту, архітектуру хвиль ворогів, аудіовізуальне оформлення та інтерфейс користувача.

Головна мета гри - вижити протягом заданої кількості хвиль, відбиваючи атаки зомбі у закритому просторі підземної Sci-Fi лабораторії. З кожною новою хвилею кількість супротивників збільшується, що вимагає від гравця постійного руху, влучного ведення вогню та тактичного використання оточення.

Початок роботи та інтерфейс меню При запуску гри користувач потрапляє на екран Головного меню, де за допомогою миші може обрати початок нової ігрової сесії або вихід з програми. Після натискання кнопки старту відбувається завантаження ігрового рівня.

Керівництво користувача (Керування) Керування персонажем здійснюється за допомогою стандартної для ПК розкладки клавіатури та миші (завдяки налаштованій системі Enhanced Input):

- **W, A, S, D** - переміщення персонажа у просторі.
- **Рух мишею** - огляд та прицілювання.
- **Ліва кнопка миші (LMB)** - ведення вогню зі зброї.
- **Права кнопка миші (RMB)** - точне прицілювання (режим ADS), яке автоматично приховує екранний приціл (Crosshair).
- **R** - перезаряджання зброї.
- **Shift (утримання)** - перехід у режим спринту для швидкого уникнення ворогів.
- **Ctrl** (або відповідна клавіша) - присідання для проходження крізь вузькі отвори.

Рисунок 3.17. Демонстрація ігрового процесу готового продукту

Джерело: Розроблене автором

Ігрові механіки та тактика Для успішного проходження гравцеві необхідно постійно слідкувати за інформацією на HUD-екрані: контролювати залишок набоїв у магазині та загальний резерв, а також слідкувати за шкалою здоров'я.

Якщо гравець отримує шкоду від зомбі, по краях екрана з'являється червоний візуальний ефект (оверлей крові). Оскільки в грі реалізована система автоматичної регенерації, користувачеві необхідно тактично відступати і уникати отримання нових пошкоджень протягом кількох секунд, щоб здоров'я відновилося. Важливо

враховувати механіку колізії зброї: при наближенні впритул до стін персонаж автоматично опускає зброю, що тимчасово унеможливлює ведення вогню.

Для створення безпечних зон гравець може використовувати систему барикад. Знайшовши на локації зону, позначену голограмою паркану, гравець повинен зайти в неї та протриматися всередині 3 секунди (за умови відсутності ворогів у зоні). Після цього згенерується фізична перешкода, яка заблокує шлях зомбі. Гравцеві слід пам'ятати, що барикаду можуть зруйнувати як вороги, так і він сам за допомогою вогнепальної зброї, що дозволяє звільнити прохід у разі необхідності.

Після знищення всіх ворогів у поточній хвилі на екрані з'являється відповідне текстове сповіщення, і дається короткий час на підготовку до наступного раунду. У разі втрати всього здоров'я або успішного проходження фінальної хвилі, гравець потрапляє на екран смерті або перемоги відповідно, де може перезапустити рівень або повернутися до Головного меню.

Розроблений ігровий продукт має інтуїтивно зрозуміле керування, плавну криву навчання та стабільну технічну базу. Інтеграція візуальних підказок, звукового супроводу та тактичних механік формує цілісний і захопливий користувацький досвід, що повністю відповідає поставленим завданням дипломного проєкту.

Висновок до розділу 3

У третьому розділі було виконано комплекс робіт із просторового, візуального та звукового оформлення ігрового проєкту, а також налаштовано систему взаємодії користувача з грою.

На етапі левел-дизайну було спроєктовано ігрову локацію в стилістиці підземного науково-дослідного комплексу (Sci-Fi). Завдяки використанню оптимізованих модульних 3D-моделей створено архітектуру рівня, яка стимулює гравця до постійного переміщення та тактичного маневрування. Важливим геймдизайнерським рішенням стала імплементація системи динамічних барикад, яка органічно поєднує геометрію простору з програмною логікою, дозволяючи гравцеві самостійно формувати тимчасові укриття.

Для забезпечення ефективної комунікації між гравцем та ігровою системою було розроблено повноцінний користувацький інтерфейс за допомогою інструментарію Unreal Motion Graphics (UMG). Створено інтерактивне Головне меню, екрани завершення сесії (перемоги та смерті) з правильним перехопленням контролю введення (Input Mode), а також інформативний ігровий HUD. Застосування динамічних прив'язок (Bind) забезпечило коректне відображення ключових параметрів (здоров'я, боєзапас, поточна хвиля) у реальному часі.

З метою посилення імерсивності ігрового процесу інтегровано систему візуальних ефектів (VFX) та просторового звуку. Налаштовано бойові візуальні ефекти (спалахи пострілів, іскри, бризки крові) та ефекти оточення. Аудіосистема проєкту використовує параметри згасання звуку (Sound Attenuation) для позиціювання ворогів у 3D-просторі та вузли рандомізації аудіофайлів (Sound Cues), що мінімізує аудіальну втому гравця і додає грі тактичної глибини.

У результаті об'єднання розроблених базових механік (контролер, ШІ, система хвиль) із створеним рівнем, інтерфейсом та аудіовізуальним супроводом було отримано цілісний функціональний прототип (вертикальний зріз) відеогри. Готовий програмний продукт має стабільний ігровий цикл, інтуїтивно зрозуміле керування та повністю задовольняє вимоги, поставлені до практичної частини кваліфікаційної роботи.

ВИСНОВОК

Кваліфікаційну роботу присвячено актуальному завданню сучасної індустрії інтерактивних розваг - розробці повноцінного прототипу тривимірної відеогри в жанрі шутера від першої особи (FPS) з елементами виживання (Survival). У ході виконання роботи було пройдено всі ключові етапи створення ігрового продукту: від концептуального проєктування та побудови базової програмної архітектури до дизайну рівнів, налаштування штучного інтелекту та інтеграції аудіовізуального супроводу. Як основний інструмент розробки було обрано передовий ігровий рушій Unreal Engine 5, а логіку реалізовано за допомогою системи візуального програмування Blueprints, що забезпечило високу швидкість ітерацій та гнучкість архітектури.

Підсумовуючи результати проведеного дослідження та практичної розробки, можна зробити наступні висновки:

Фундаментом ігрового процесу стала розробка комплексного контролера гравця на базі сучасної підсистеми Enhanced Input System. Це дозволило реалізувати гнучку та масштабовану систему керування, яка легко адаптується під різні пристрої введення. Було успішно імплементовано механіки переміщення (ходьба, спринт, присідання), які безпосередньо впливають на тактичні можливості гравця на рівні. Крім базових рухів, спроєктовано просунуту систему поводження зі зброєю: механіку стрільби на основі фізичних снарядів (Projectiles), логіку точного прицілювання (ADS), а також систему ручного перезарядження, синхронізовану з анімаційними монтажами. Інноваційним технічним рішенням стала реалізація системи тактичного опускання зброї при наближенні до перешкод (на основі методів трасування променів Line Trace), що значно підвищило реалізм взаємодії персонажа з оточенням. Для забезпечення балансу виживання впроваджено математичну модель автоматичної регенерації здоров’я після виходу з бою.

Для створення динамічного та непередбачуваного ігрового досвіду було розроблено алгоритми поведінки штучного інтелекту ворогів (зомбі). Використання навігаційної сітки (NavMesh) та сенсорних компонентів (Pawn Sensing) дозволило

наділити супротивників здатністю самостійно виявляти гравця в просторі та переслідувати його, оминаючи перешкоди. Впроваджено модульну систему обробки шкоди, яка коректно реагує на влучання та ініціює відповідні стани (атака, отримання пошкоджень, смерть). Глобальний контроль над ігровою сесією реалізовано через архітектуру класу Game Mode. У його межах розроблено математичну модель ескалації складності - систему генерації хвиль ворогів. Створена логіка автоматично розраховує кількість супротивників для кожного раунду, розподіляє їх між точками генерації та управляє умовами перемоги або поразки. Використання механізмів обробки колізій (Spawn Collision Handling) під час появи ворогів забезпечило технічну стабільність масового спавну ШІ без критичних помилок чи застрягань моделей.

Практична значущість роботи підкріплюється створенням унікальної ігрової локації у науково-фантастичному (Sci-Fi) сеттингу. Завдяки грамотному використанню модульних 3D-асетів побудовано нелінійний рівень з просторими залами для маневрування та вузькими коридорами для відступу. Геометрія рівня тісно інтегрована з ігровими механіками. Визначним досягненням геймдизайну в межах проєкту стала розробка системи динамічних барикад. Ця механіка дозволяє гравцеві взаємодіяти зі спеціальними тригер-зонами для створення тимчасових фізичних укриттів, які блокують шлях ворогам. Наявність у барикад власних параметрів міцності та можливості їх руйнування як гравцем, так і ворогами, формує глибокий рівень тактичного планування під час бою.

Для забезпечення комфортної комунікації між користувачем та грою створено повноцінний інтерфейс (UI) за допомогою інструментарію UMG. Розроблено багаторівневу структуру меню: Головне меню, інформативний ігровий екран (HUD) із динамічною прив'язкою даних (Bind) про стан здоров'я, боєзапас та прогрес хвиль, а також екрани завершення сесії. Для посилення імерсивності впроваджено оверлей поранень із динамічною зміною прозорості. Аудіовізуальна складова (Game Feel) була суттєво покращена шляхом налаштування візуальних ефектів (спалахи пострілів, іскри від влучань у метал, ефекти крові, навколишнє середовище) та глибокої роботи зі звуком. Імплементовано просторове згасання аудіо (Sound

Attenuation) для точного позиціювання ворогів у темряві та застосовано систему рандомізації звукових семплів (Random Sound Cues), що мінімізує ефект одноманітності при тривалій грі.

Практичне значення одержаних результатів полягає у тому, що розроблений програмний продукт є повністю функціональним вертикальним зрізом (Vertical Slice) відеогри. Проєкт може успішно використовуватися як готове портфоліо, яке демонструє високий рівень володіння інструментарієм Unreal Engine 5, розуміння принципів об'єктно-орієнтованого підходу в Blueprints, а також навички левел-дизайну та проєктування ігрових систем. Крім того, створена гра виступає надійним технічним фундаментом для подальшої комерційної розробки, оскільки її модульна архітектура дозволяє без критичних змін ядра додавати нові класи зброї, розширювати бестіарій ворогів та масштабувати ігровий світ.

Перспективи подальшого розвитку. Запропонована в дипломній роботі архітектура має значний потенціал для масштабування. У майбутньому проєкт може бути розширений шляхом додавання системи економіки (отримання валюти за вбитих ворогів та купівля зброї/патронів між хвилями), впровадження нових типів штучного інтелекту (міні-боси з унікальними патернами атак, вороги дальнього бою), реалізації механіки збереження прогресу (SaveGame), а також розробки розрахованого на багато користувачів кооперативного режиму (Multiplayer Networking) для спільного проходження ігрових рівнів.

У результаті виконання кваліфікаційної роботи всі поставлені завдання були успішно вирішені, а головну мету - створення функціонального прототипу гри в жанрі FPS-Survival за допомогою сучасних технологій розробки - повністю досягнуто.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Unreal Engine 5.7 Documentation URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5-7-documentation (дата звернення: 06.08.2025)
2. Learn Blueprints in Unreal Engine 5 - Beginner Tutorial URL: https://www.youtube.com/watch?v=opyV7rhKSFc (дата звернення: 25.08.2025)
3. Unreal Engine Online Learning URL: https://learn.unrealengine.com/ (дата звернення: 12.09.2025)
4. How to Create a Simple FPS in Unreal Engine 5 URL: https://www.kodeco.com/32435756-how-to-create-a-simple-fps-in-unreal-engine-5 (дата звернення: 03.10.2025)
5. Community Tutorial: Basics of Blueprints (UE5) URL: https://dev.epicgames.com/community/learning/tutorials/EW7d/unreal-engine-epic-games-store-basics-of-blueprints-ue5?locale=pl-pl&locale=fr-fr&locale=en-us&locale=en-us (дата звернення: 21.10.2025)
6. Unreal Engine Basics & Intro to UE5 Blueprints URL: https://www.youtube.com/watch?v=ITCWa3oLNAQ (дата звернення: 15.11.2025)
7. Enhanced Input in Unreal Engine URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/enhanced-input-in-unreal-engine (дата звернення: 05.12.2025)
8. UMG UI Designer for Unreal Engine URL: https://dev.epicgames.com/documentation/unreal-engine/umg-ui-designer-for-unreal-engine?application_version=5.2 (дата звернення: 20.12.2025)
9. Navigation System and NavMesh in Unreal Engine URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/navigation-system-in-unreal-engine (дата звернення: 14.01.2026)

10. Audio System Overview in Unreal Engine URL: https://dev.epicgames.com/documentation/unreal-engine/audio-system-overview?application_version=4.27 (дата звернення: 02.02.2026)
11. Level Design Fundamentals in Unreal Engine URL: https://dev.epicgames.com/community/learning/tutorials/3VKJ/unreal-engine-fortnite-level-design-fundamentals (дата звернення: 18.02.2026)
12. Artificial Intelligence in Unreal Engine URL: https://dev.epicgames.com/documentation/unreal-engine/artificial-intelligence-in-unreal-engine (дата звернення: 10.03.2026)
13. Schell J. The Art of Game Design: A Book of Lenses. (дата звернення: 18.03.2026)
14. Swink S. Game Feel: A Game Designer's Guide to Virtual Sensation. (дата звернення: 25.03.2026)
15. Totten C. W. An Architectural Approach to Level Design. (дата звернення: 02.04.2026)
16. Millington I. AI for Games. (дата звернення: 11.04.2026)
17. Romero M. Blueprints Visual Scripting for Unreal Engine 5. (дата звернення: 19.04.2026)
18. Sylvester T. Designing Games: A Guide to Engineering Experiences. (дата звернення: 26.04.2026)
19. Rogers S. Level Up! The Guide to Great Video Game Design. (дата звернення: 03.05.2026)
20. Nystrom R. Game Programming Patterns. (дата звернення: 10.05.2026)