

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

Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри

_____ (підпис)

Едуард ЖАРИКОВ
(ім'я прізвище)

“ _____

2025 р.

Дипломний проект
на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інженерія програмного забезпечення
інформаційних систем»
спеціальності «121 Інженерія програмного забезпечення»

на тему: Гра у жанрі Role-Playing-Game(RPG) на Unreal Engine з
використанням ІІІ

Виконав студент IV курсу, групи ІІ-11
(шифр групи)

Головня Олександр Ростиславович
(прізвище, ім'я, по батькові)

_____ (підпис)

Керівник ст.викл.,д-р філософії Головченко М. М.
(посада, науковий ступінь, вчене звання, прізвище та ініціали)

_____ (підпис)

Консультант _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали)

_____ (підпис)

Рецензент _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали)

_____ (підпис)

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

Студент _____
(підпис)

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії
Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 121 Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення
інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
Едуард ЖАРИКОВ
(підпис) (ім'я прізвище)
“ ____ ” 2025 р.

ЗАВДАННЯ
на дипломний проект студенту
Головні Олександру Ростиславовичу

(прізвище, ім'я, по батькові)

1. Тема проекту Гра у жанрі Role-Playing-Game(RPG) на Unreal Engine з використанням III
керівник проекту Головченко М. М., ст.вик., д-р філософії
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

- затверджені наказом по університету від «23.05.2025» травня 2025 р. №1705-с
2. Термін подання студентом проекту «16» червня 2025 року
3. Вихідні дані до проекту: технічне завдання
4. Зміст пояснівальної записки
- 1) Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі.
- 2) Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних.
- 3) Конструювання та розробка: змістовна та математична постановки задачі, обґрунтування та опис методу розв'язання.
- 4) Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів.
- 5) Технологічний розділ: керівництво користувача, методика випробувань програмного продукту.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Архітектурна модель системи – контейнерна(C4 Container Diagram)

3) Креслення вигляду екранних форм

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «15» березня 2025 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	15.03.2025	
2	Аналіз існуючих методів розв'язання задачі	04.04.2025	
3	Постановка та формалізація задачі	11.04.2025	
4	Розробка інформаційного забезпечення	14.04.2025	
5	Алгоритмізація задачі	22.04.2025	
6	Обґрунтування вибору використаних технічних засобів	23.04.2025	
7	Розробка програмного забезпечення	10.05.2025	
8	Налагодження програми	15.05.2025	
9	Виконання графічних документів	18.05.2025	
10	Оформлення пояснівальної записки	26.05.2025	
11	Подання ДП на попередній захист	16.05.2025	
12	Подання ДП рецензенту	10.06.2025	
13	Подання ДП на основний захист	16.06.2025	

Студент

(підпис)

Керівник

(підпис)

Олександр ГОЛОВНЯ

(ініціали, прізвище)

Максим ГОЛОВЧЕНКО

(ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з п'яти розділів, містить 37 таблиць, 51 рисунка та 19 джерел – загалом 99 сторінок.

Дипломний проєкт присвячений розробці гри у жанрі Role-Playing-Game(RPG) на Unreal Engine з використанням ШІ.

Метою розробки є підвищення якості ігрового досвіду користувачів шляхом реалізації штучного інтелекту для неігрових персонажів

У розділі 1 розглянуто передпроєктне обстеження предметної області, що включає в себе постановку завдання, аналіз предметної області, аналіз існуючих рішень та загальний аналіз і моделювання бізнес-процесів.

Розділ 2 присвячений розробці вимог до програмного забезпечення, а саме розробка функціональних та нефункціональних вимог, аналіз системних вимог аналіз економічних показників ПЗ та постановку завдання на розробку.

У розділі 3 розглянуто конструювання та розроблення ПЗ, розглянуто архітектуру та архітектурні рішення, обґрунтовано вибір засобів розробки.

У розділі 4 наведений аналіз якості та опис процесів тестування ПЗ, а також опис контрольного прикладу.

Програмне забезпечення впроваджено у розділі 5, який присвячений розгортанню та супроводу.

КЛЮЧОВІ СЛОВА: ГРА, RPG, UNREAL ENGINE, IT, ШТУЧНИЙ ІНТЕЛЕКТ, НІП.

ABSTRACT

The explanatory note of the diploma project consists of five sections, contains 37 tables, 51 figures and 19 sources – in total 99 pages.

The diploma project is dedicated to the development of a Role-Playing Game (RPG) using Unreal Engine with the implementation of artificial intelligence. The goal of the project is to enhance the user's gaming experience by integrating artificial intelligence for non-player characters (NPCs).

Chapter 1 presents the preliminary analysis of the subject area, including task definition, domain analysis, review of existing solutions, and overall analysis and modeling of business processes.

Chapter 2 is focused on the development of software requirements, covering functional and non-functional requirements, system requirements analysis, economic evaluation of the software, and task formulation for development.

Chapter 3 addresses the design and development of the software, discussing the architecture and architectural decisions, as well as the rationale behind the choice of development tools.

Chapter 4 includes quality analysis and a description of the software testing processes, along with an explanation of the test case. Chapter 5 covers the deployment and maintenance of the software.

KEYWORDS: GAME, RPG, UNREAL ENGINE, IT, ARTIFICIAL INTELLIGENCE, NPC.

					КПІ.ІП-1107.045440.00.90		
Змін.	Арк.	№ докум.	Підп.	Дата			
Розроб.	Головня О.Р.					Літ.	Аркуш
Перевір.	Головченко М.М					1	Аркушів
					Відомість дипломного проєкту		
Н.контр.	Головченко М.М				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІП-11		
Затв.	Жаріков Е.В.						

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРИКОВ

“___” _____ 2025 р.

ГРА У ЖАНРІ ROLE-PLAYING-GAME(RPG) НА UNREAL ENGINE 3

ВИКОРИСТАННЯМ ШІ

Технічне завдання

КП.ІП-1107.045440.01.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Максим ГОЛОВЧЕНКО

Нормоконтроль:

_____ Максим ГОЛОВЧЕНКО

Виконавець:

_____ Олександр ГОЛОВНЯ

Київ – 2025

ЗМІСТ

1	НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ	4
2	ПІДСТАВА ДЛЯ РОЗРОБКИ.....	5
3	ПРИЗНАЧЕННЯ РОЗРОБКИ	6
4	ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	7
4.1	Вимоги до функціональних характеристик	7
4.1.1	Користувацького інтерфейсу	7
4.1.2	Забезпечення функціонування головного меню гри:.....	9
4.1.3	Забезпечення функціонування ігрових механік:.....	10
4.1.4	Забезпечення функціонування ігрового інтерфейсу(компоненти HUD):	12
4.1.5	Забезпечення функціонування взаємодії з інвентарем:	13
4.1.6	Забезпечення функціонування штучного інтелекту неігрових персонажів гри:	14
4.1.7	Забезпечення взаємодії з колізією об'єктів:	15
4.2	Вимоги до надійності.....	17
4.3	Умови експлуатації	17
4.3.1	Вид обслуговування.....	18
4.3.2	Обслуговуючий персонал.....	18
4.4	Вимоги до складу і параметрів технічних засобів.....	18
4.5	Вимоги до інформаційної та програмної сумісності.....	19
4.5.1	Вимоги до входних даних	19
4.5.2	Вимоги до вихідних даних	19
4.5.3	Вимоги до мови розробки	20
4.5.4	Вимоги до середовища розробки	20
4.5.5	Вимоги до представленню вихідних кодів	20
4.6	Вимоги до маркування та пакування	20
4.7	Вимоги до транспортування та зберігання	20
4.8	Спеціальні вимоги	20
5	ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ	21

5.1	Попередній склад програмної документації	21
5.2	Спеціальні вимоги до програмної документації	21
6	СТАДІЇ І ЕТАПИ РОЗРОБКИ	22
7	ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ	23

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Гра у жанрі Role-Playing-Game(RPG) на Unreal Engine з використанням ШІ.

Галузь застосування: Індустрія відеоігор

Наведене технічне завдання поширюється на розробку програмного забезпечення «RPG-гра з використанням ШІ» [RPG-AI-001], котра використовується для інтерактивного рольового ігрового середовища, що забезпечує унікальний досвід користувача завдяки динамічній генерації контенту, сценаріїв і поведінки персонажів на основі алгоритмів штучного інтелекту та призначена для галузі цифрових розваг, а саме ігрової індустрії. Потенційними користувачами є гравці, гейм/левел дизайнери, новачки у сфері ШІ та розробники відеоігор.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки RPG-гри з використанням ШІ на рушії UE5 є завдання на дипломне проектування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для забезпечення функціонування ігрових механік, що дозволяють реалізацію цікавого штучного інтелекту гри у жанрі RPG і використовується у галузі розваг для широкого кола користувачів.

Метою розробки є підвищення якості ігрового досвіду користувачів шляхом реалізації цікавого штучного інтелекту для неігрових персонажів.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функцій:

4.1.1 Користувацького інтерфейсу

Меню головного екрану, можливість початку нової гри або продовження поточної, доступ до налаштувань гри, вихід з гри(Рис. 4.1).

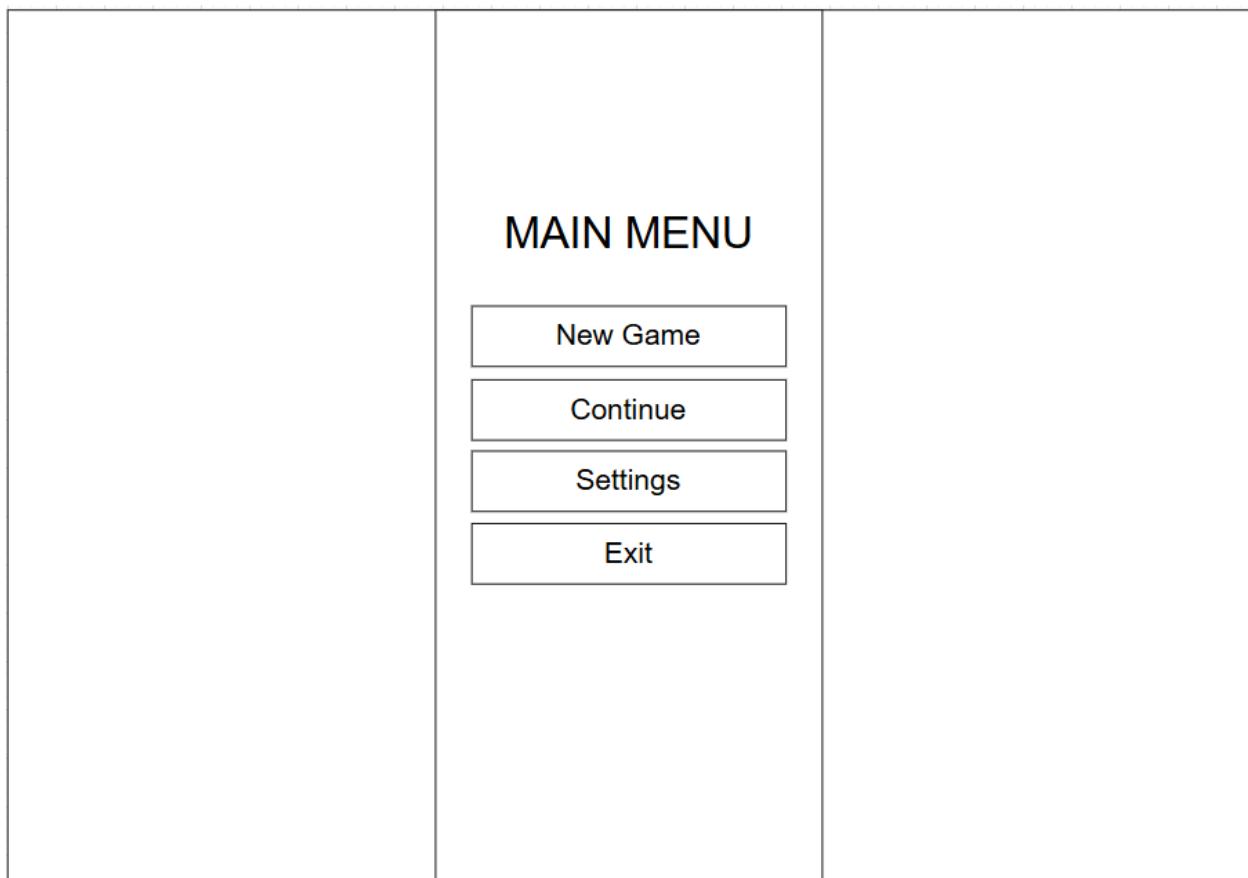


Рисунок 4.1 – Головна сторінка

Інтерфейс управління персонажем, показ основних характеристик персонажа: здоров'я, витривалість, досвід (Рис. 4.2).

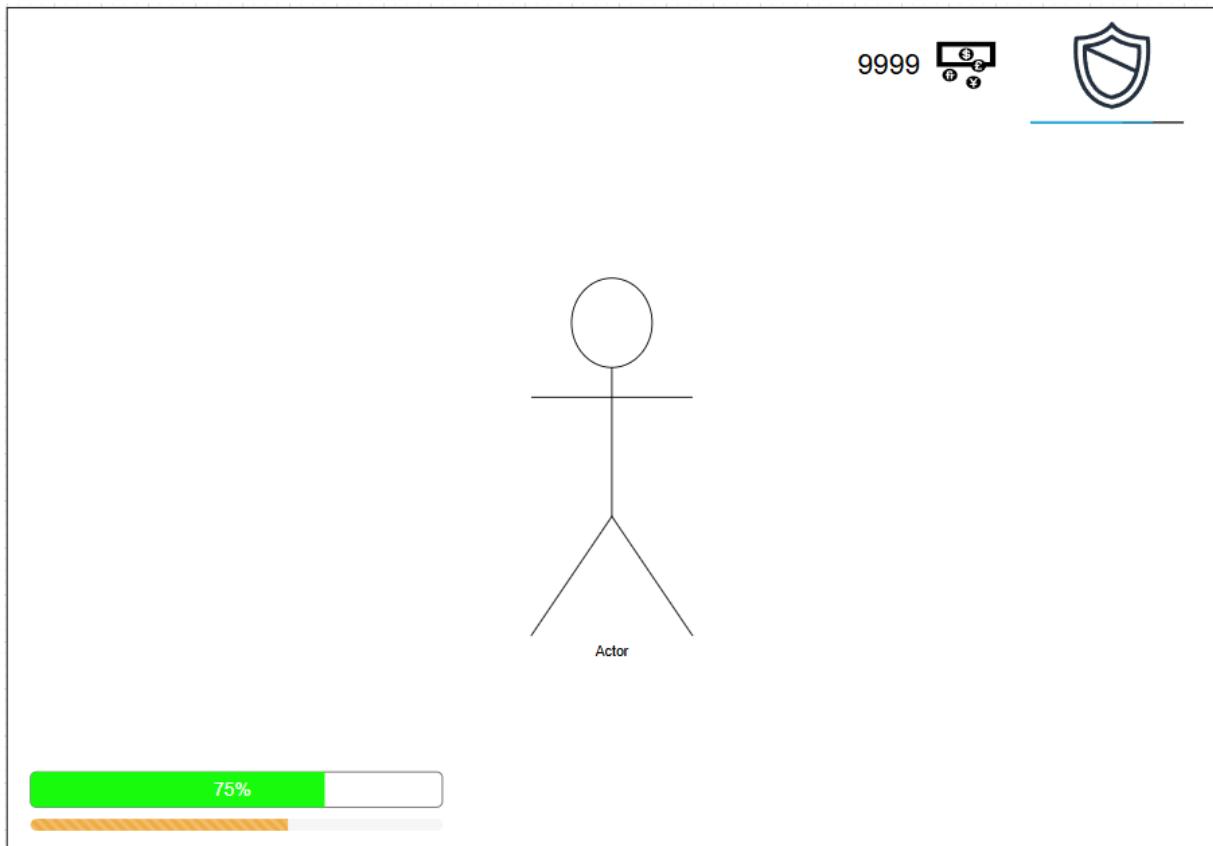


Рисунок 4.2 – Сторінка інтерфейсу управлінням персонажа

Інвентар персонажа, можливість перегляду та управління предметами в інвентарі, операції з предметами: equip, use, drop. Фільтрація предметів за категоріями: зброя, броня, ліки (Рис. 4.3).

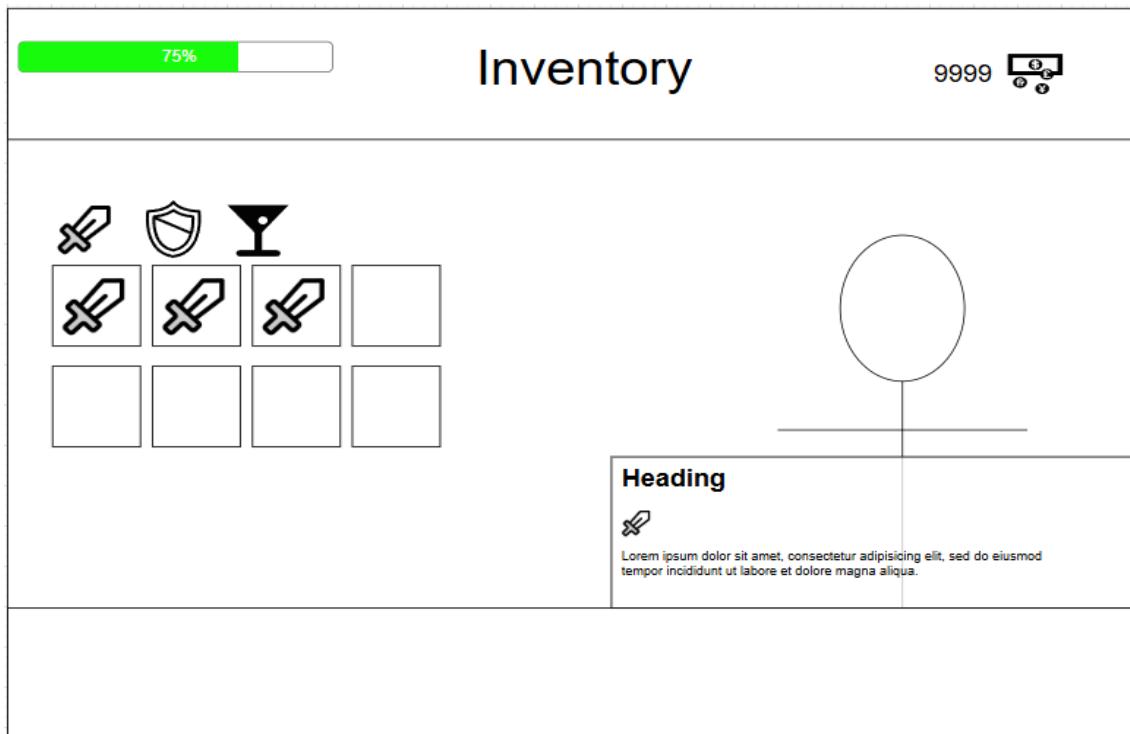


Рисунок 4.3 – Сторінка управління інвентарем гравця

Навігація та мінікарта, відображення локальної локації, позначення важливих об'єктів та NPC - неігрові персонажі(Рис. 4.4)

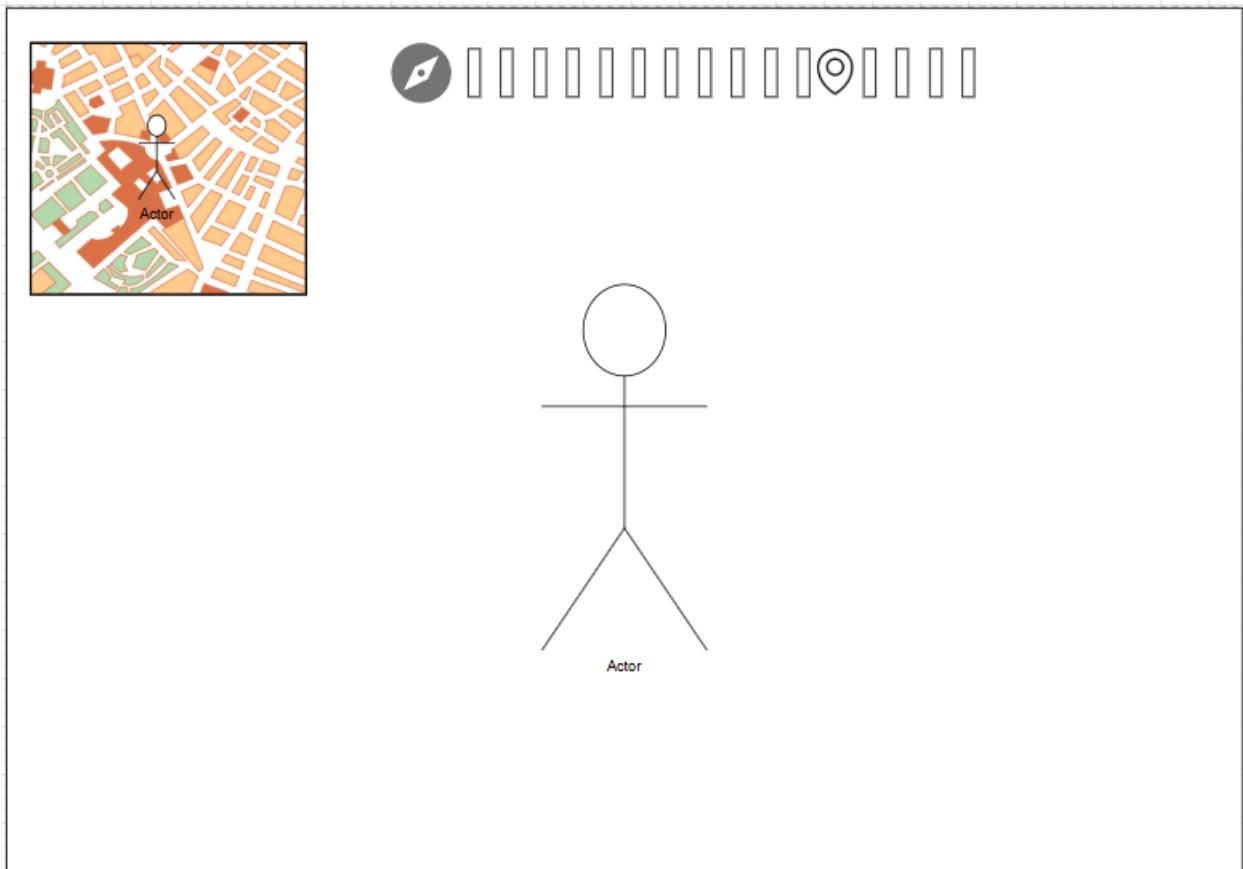


Рисунок 4.4 – Відображення інтерфейсу навігації

4.1.2 Забезпечення функціонування головного меню гри:

- 4.1.2.1 Запуск нової сесії гри.
- 4.1.2.2 Завантаження останнього збереження.
- 4.1.2.3 Доступ до тренувальної арени.
- 4.1.2.4 Налаштування гри.
 - 4.1.2.4.1 Налаштування режиму вікна.
 - 4.1.2.4.2 Налаштування роздільної здатності.
 - 4.1.2.4.3 Якість графіки гри.
 - 4.1.2.4.4 Вертикальна синхронізація.

4.1.2.5 Завершення гри та повернення на робочий стіл.

4.1.3 Забезпечення функціонування ігрових механік:

4.1.3.1 Керування камерою.

4.1.3.1.1 Вільне пересування камери по XY 2D-axis.

4.1.3.1.2 Захоплення цілі.

4.1.3.2 Пересування гравця.

4.1.3.2.1 Ходьба: вліво/право, вперед/назад.

4.1.3.2.2 Біг: вліво/право, вперед/назад.

4.1.3.2.3 Стрибок.

4.1.3.2.4 Присідання.

4.1.3.2.5 Перестрибування.

4.1.3.2.6 Скелелазіння.

4.1.3.2.7 Ухиляння.

4.1.3.3 Можливості персонажа.

4.1.3.3.1 Створення звуку для відволікання.

4.1.3.3.2 Вибір бойового стану.

4.1.3.3.3 Телепорт гравця.

4.1.3.3.4 Взаємодія з об'єктами.

4.1.3.4 Бойова система персонажа.

4.1.3.4.1 Атака магічними ударами.

4.1.3.4.2 Атака ближнього бою.

4.1.3.4.3 Проведення комбо-ударів.

4.1.3.4.4 Блокування.

4.1.3.4.5 Парирування.

4.1.3.5 Система характеристик персонажа.

4.1.3.5.1 Здоров'я.

4.1.3.5.2 Атака.

4.1.3.5.3 Досвід та рівень.

4.1.3.5.4 Витривалість.

- 4.1.3.5.5 Золото.
- 4.1.3.6 Квестова система.
 - 4.1.3.6.1 Відкриття діалогу при взаємодії з NPC.
 - 4.1.3.6.2 Отримання завдання.
 - 4.1.3.6.2.1 Квест має назву, ціль, опис.
 - 4.1.3.6.3 Відстеження прогресу.
 - 4.1.3.6.4 Повідомлення про завершення.
 - 4.1.3.6.5 Отримання винагороди.
- 4.1.3.7 Інтерактивне середовище.
 - 4.1.3.7.1 Взаємодія з об'єктами.
 - 4.1.3.7.1.1 Об'єкти, які можна досліджувати: скрині, двері, механізми.
 - 4.1.3.7.1.2 Взаємодія через натискання клавіші E в радіусі досяжності.
 - 4.1.3.7.2 Фізичні об'єкти (переміщення, піднімання).
 - 4.1.3.7.2.1 Піднімання предметів (зброї, ресурсів, предметів квесту).
 - 4.1.3.7.2.2 Можливість перетягувати об'єкти (наприклад, ящики для вирішення головоломок).
- 4.1.3.8 Навігація та міні-карта.
 - 4.1.3.8.1 Міні-карта.
 - 4.1.3.8.1.1 Відображає поточне розташування гравця.
 - 4.1.3.8.1.2 Динамічно обертається відносно напрямку камери.
 - 4.1.3.8.1.3 Використання Scene Capture 2D для стилізованого вигляду.
 - 4.1.3.8.2 Компас.
 - 4.1.3.8.2.1 Відображення основних напрямків (N, E, S, W).
 - 4.1.3.8.2.2 Динамічна адаптація до орієнтації камери.

4.1.4 Забезпечення функціонування ігрового інтерфейсу(компоненти HUD):

4.1.4.1 WB_HUD.

- 4.1.4.1.1 Рівень та досвід гравця.
- 4.1.4.1.2 Ігрова валюта.
- 4.1.4.1.3 Події/повідомлення.

4.1.4.2 WBP_HealthBar.

- 4.1.4.2.1 Динамічна шкала здоров'я.
- 4.1.4.2.2 Стилістичний підрахунок здоров'я у вигляді сердець.

4.1.4.3 WBP_Compass.

- 4.1.4.3.1 Відображення напрямків (N, E, S, W).
- 4.1.4.3.2 Динамічне оновлення залежно від напрямку погляду.

4.1.4.4 WBP_Minimap.

- 4.1.4.4.1 Відображення навколишнього середовища.
- 4.1.4.4.2 Динамічне обертання відносно напрямку камери.

4.1.4.5 W_BossHealthBar.

- 4.1.4.5.1 Окрема панель HP, що з'являється при початку бою з босом.

- 4.1.4.5.2 Назва боса, індикатор рівня.

4.1.4.6 WB_QuestGiverDialogue.

- 4.1.4.6.1 Виведення реплік персонажа.

4.1.4.6.2 WB_QuestGiver.

- 4.1.4.6.2.1 Отримання квесту.

4.1.4.6.3 WB_QuestObjective.

- 4.1.4.6.3.1 Відображення активних квестів.

- 4.1.4.6.3.2 Статус виконання.

4.1.4.7 WBP_Inventory.

- 4.1.4.7.1 Відображення інвентарю гравця.

- 4.1.4.7.2 Можливість переглядати, сортувати, використовувати або екіпірувати предмети.

- 4.1.4.7.3 Підтримка drag-and-drop функціональності.

4.1.4.7.4 Відображення опису предметів та їх характеристик.

4.1.4.8 WBP_Interact.

4.1.4.8.1 Відображення повідомлення при можливості взаємодії з об'єктом (наприклад, «Натисніть [E], щоб взаємодіти»).

4.1.4.8.2 Динамічне оновлення залежно від типу об'єкта.

4.1.4.9 WB_MainMenu.

4.1.4.9.1 Головне меню гри з доступом до нової гри, продовження, арени, налаштувань та виходу.

4.1.4.10 WB>LoadingScreen - Завантажувальний екран між переходами сцен.

4.1.4.11 WB_PauseMenu.

4.1.4.11.1 Меню паузи з можливістю продовжити гру, повернення до головного меню, зміни налаштувань.

4.1.4.11.2 Блокує ігрову взаємодію під час паузи.

4.1.5 Забезпечення функціонування взаємодії з інвентарем:

4.1.5.1 Підбір предметів.

4.1.5.1.1 Підбір через взаємодію.

4.1.5.1.2 Надання системою.

4.1.5.2 Огляд власних предметів.

4.1.5.3 Переміщення предметів у інвентарі.

4.1.5.4 Вибір зброї з інвентаря.

4.1.5.5 Вибір щита з інвентаря.

4.1.5.6 Застосування предметів: equip/use/eat.

4.1.5.7 Викидання непотрібних предметів.

4.1.5.8 Перегляд балансу ігрової валюти.

4.1.5.9 Перегляд здоров'я персонажа.

4.1.6 Забезпечення функціонування штучного інтелекту неігрових персонажів гри:

4.1.6.1 Базова поведінка NPC.

4.1.6.1.1 Патрюлювання по заданих маршрутах.

4.1.6.1.2 Реакція на гравця.

4.1.6.1.2.1 Виявлення.

4.1.6.1.2.2 Атака.

4.1.6.1.2.3 Втеча.

4.1.6.1.2.4 Реагування на шкоду.

4.1.6.1.3 Idle-анімації та природня поведінка(сидить, ходить, спілкується з іншими NPC).

4.1.6.1.4 Слідування за гравцем.

4.1.6.2 Ворожі NPC.

4.1.6.2.1 Виявлення гравця.

4.1.6.2.2 Перехід у бойовий стан.

4.1.6.2.3 Атакуюча поведінка(ближній/ дальній бій).

4.1.6.2.4 Ухилення, блокування.

4.1.6.2.5 Режим переслідування гравця.

4.1.6.2.6 Режим втечі.

4.1.6.2.7 Використання укриттів.

4.1.6.3 Дружні NPC.

4.1.6.3.1 Надання квестів.

4.1.6.3.2 Діалоги.

4.1.6.3.3 Використання NPC(наприклад: вихованець).

4.1.6.4 Навігація та орієнтація в просторі.

4.1.6.4.1 Обхід перешкод.

4.1.6.4.2 Вибір оптимального маршруту.

4.1.6.4.3 Використання NavMesh для руху.

4.1.6.5 Станова машина (Behavior Tree).

4.1.6.5.1 Використання Behavior Tree для логіки прийняття рішень.

4.1.6.5.2 Blackboard для зберігання стану NPC.

4.1.6.6 Стан штучного інтелекту.

4.1.6.6.1 Заморожений.

4.1.6.6.2 Бойовий.

4.1.6.6.3 Слідчий.

4.1.6.6.4 Пасивний або патрулюючий.

4.1.7 Забезпечення взаємодії з колізією об'єктів:

4.1.7.1 Загальні налаштування.

4.1.7.1.1 Визначення типів об'єктів для системи колізій.

4.1.7.1.1.1 Персонажі.

4.1.7.1.1.2 Статичні/динамічні об'єкти.

4.1.7.1.1.3 NPC.

4.1.7.1.1.4 Фізичні об'єкти.

4.1.7.1.1.5 Тригери, невидимі бар'єри.

4.1.7.1.2 Призначення колізійних профілів (Collision Presets)

відповідно до ролі об'єкта.

4.1.7.1.2.1 Pawn, WorldStatic, WorldDynamic, PhysicsBody, Trigger тощо.

4.1.7.2 Колізія персонажа.

4.1.7.2.1 Активне використання Capsule Component з відповідною сферою впливу.

4.1.7.2.2 Обмеження проходження крізь тверді об'єкти (наприклад, стіни, скелі).

4.1.7.2.3 Відключення колізії з об'єктами UI або прив'язаними елементами HUD.

4.1.7.2.4 Налаштування взаємодії з фізичними об'єктами: штовхання, торкання, блокування шляху.

- 4.1.7.2.5 Перешкоди для входу в закриті зони (двері, невидимі бар'єри) до певних умов.
- 4.1.7.3 Колізія бойових механік.
- 4.1.7.3.1 Виявлення влучань (Hit Detection) за допомогою Line Trace, Sphere Trace, або Box Trace.
- 4.1.7.3.2 Розрізnenня між фізичним ударом і візуальним ефектом.
- 4.1.7.3.3 Активація колізії зброї тільки під час атаки (наприклад, за допомогою Enable Collision у анімації).
- 4.1.7.4 Колізія для взаємодії (інтеракція).
- 4.1.7.4.1 Використання Sphere Overlap або Line Trace для визначення наявності об'єкта попереду персонажа.
- 4.1.7.4.2 Активація відповідного UI/підказки при входженні в зону колізії (наприклад, WBP_Interact).
- 4.1.7.5 Колізія для дверей, плит, тригерів.
- 4.1.7.5.1 Двері мають Box Collision для виявлення контакту з натискою плитою або ключовим об'єктом.
- 4.1.7.5.2 Плити активують подію лише за входження в колізію (OnComponentBeginOverlap).
- 4.1.7.5.3 Можливість обмеження повторного активаційного тригера – уникнення нескінченних подій.
- 4.1.7.5.4 Система перевірки умов: наявність квесту, ключа, подолання ворогів тощо.
- 4.1.7.6 Оптимізація та продуктивність.
- 4.1.7.6.1 Вимкнення колізії для декоративних або непотрібних об'єктів (наприклад, дрібні рослини).
- 4.1.7.6.2 Активація колізії лише за необхідності.
- 4.1.7.7 Основні вимоги:
- Забезпечити можливість збереження прогресу гравцем.
 - Забезпечити максимальну можливу швидкодію гри.

- Забезпечити інтерактивність ігрового світу.

4.1.7.8 Додаткові вимоги:

- Забезпечити адекватну поведінку штучного інтелекту.

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Користувач не може виконати недопустимі дії в інтерфейсі, такі як використання неіснуючих предметів, неприпустимі переміщення, та інше, що непередбачувано грою.

В разі збоїв у роботі гри (наприклад, зависання додатка), час на відновлення системи не повинен перевищувати 30 секунд для відновлення ігрового прогресу.

Гра повинна автоматично створювати контрольні точки в ключових моментах (наприклад, після завершення квесту, в бою, або після збереження). Якщо гравець не завершив гру, прогрес зберігається на найближчій контрольній точці.

Гра повинна забезпечувати загальноприйнятий нижній поріг, при якому гравець може грati без критичних затримок або смикань, а саме 30 FPS (кількість кадрів на секунду) при рекомендованих системних вимогах вказується як цільовий показник стабільної продуктивності. А також гра повинна мінімізувати споживання ресурсів, ігрові ресурси (текстури, моделі, звуки) повинні бути оптимізовані для економії пам'яті

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Для даної гри передбачено періодичне оновлення гри для виправлення помилок, оптимізації продуктивності, підтримка і оновлення контенту гри (нові локації, персонажі, предмети, квести).

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються

4.4 Вимоги до складу і параметрів технічних засобів

Мінімальна конфігурація технічних засобів:

- операційна система: Windows 10 (64-bit);
- процесор: Intel Core i3-8100 або AMD Ryzen 3 1200;
- оперативна пам'ять: 8 GB RAM;
- відеокарта: NVIDIA GeForce GTX 750 Ti або AMD Radeon R7 360;
- жорсткий диск: 10–15 GB вільного простору;
- DirectX: Версія 11;
- інше: Клавіатура та миша, доступ до інтернету не потребується;

Рекомендована конфігурація технічних засобів:

- операційна система: Windows 11 (64-bit);
- процесор: AMD Ryzen 7 3750H;
- оперативна пам'ять: 16 GB RAM;
- відеокарта: NVIDIA GeForce RTX 2060;
- жорсткий диск: SSD із мінімум 15 GB вільного простору;
- DirectX: Версія 12;
- інше: Клавіатура та миша, доступ до інтернету не потребується;
- роздільна здатність дисплею: 1920×1080 або вища (Full HD).

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства Windows 11.

4.5.1 Вимоги до вхідних даних

Вхідні дані для цієї гри не є критично важливими, оскільки більшість ігрового процесу зосереджена на внутрішній логіці та механіках, які не залежать від зовнішніх джерел даних.

4.5.2 Вимоги до вихідних даних

Вихідні дані гри чітко структуровані, що представляють з себе збереження поточного стану гри, включаючи всі дані, що дозволяють відновити гру на тому ж етапі. Це може включати позицію гравця, статус квестів, інвентар гравця, параметри персонажа: здоров'я, витривалість, валюту, досвід та рівень.

Приклад файлу збереження у форматі .sav:

```
{
  "PlayerID": "player_01",
  "Position": { "X": 120.5, "Y": 45.2, "Z": 10.0},
  "Health": 8.5,
  "Stamina": 100.0,
  "Level": 12,
  "Experience": 1050,
  "ExperienceToNextLevel": 1200
  "Inventory": [
    { "ItemID": "sword_01", "Quantity": 1 },
    { "ItemID": "banana_01", "Quantity": 3 } ]
}
```

Логи гри теж входять до вихідних файлів, логування помилки, виключення, повідомлення системи.

Вихідний файл: Лог файл для системи моніторингу, який включає важливі дані для аналізу: ErrorCode, Message, Time.

4.5.3 Вимоги до мови розробки

Розробку виконати з використанням мови програмування C++23 та інструментів візуального скриптування Blueprints, вбудованих в Unreal Engine 5.

4.5.4 Вимоги до середовища розробки

Розробку виконати на платформі Unreal Engine 5 із використанням інтегрованого середовища Unreal Editor, яке забезпечує повний цикл створення, налагодження та тестування ігрових проектів.

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді структурованого набору файлів проєкту Unreal Engine 5, що включає C++ класи, файли конфігурацій, ресурси Blueprints, а також необхідні матеріали, шейдери, текстури та інші ассети, згруповани відповідно до структури контенту Unreal Engine.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Згенерувати інсталяційну версію програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

- пояснювальна записка;
- технічне завдання;
- текст програми;
- програма та методика тестування;
- керівництво користувача.

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

- схема структурна варіантів використання;
- схема структурна класів програмного забезпечення;
- креслення вигляду екранних форм.

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	25.02	
2.	Розробка технічного завдання	15.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	19.04	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	30.04	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	05.05	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	10.05	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	14.05	Пояснювальна записка
8.	Розробка матеріалів графічної частини проекту	20.05	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	29.05	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до «Програми та методики тестування».

Пояснювальна записка до дипломного проекту

на тему: Гра у жанрі Role-Playing-Game(RPG) на Unreal Engine з
використанням ШІ

КП.ІП-1107.045440.02.81

Київ – 2025

ЗМІСТ

ВСТУП.....	5
1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ	6
1.1 Постановка завдання дипломного проєктування	6
1.2 Аналіз предметної області	6
1.3 Аналіз існуючих рішень.....	11
1.3.1 Аналіз відомих програмних продуктів	11
1.3.2 Аналіз відомих алгоритмічних та технічних рішень	20
1.4 Аналіз та моделювання бізнес-процесів	24
Висновки до розділу	28
2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	30
2.1 Варіанти використання програмного забезпечення	30
2.2 Розроблення функціональних вимог	38
2.3 Розроблення нефункціональних вимог	42
2.4 Аналіз системних вимог	43
2.5 Аналіз економічних показників програмного забезпечення	45
2.6 Постановка завдання на розробку програмного забезпечення.....	47
Висновки до розділу	47
3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	49
3.1 Архітектура програмного забезпечення	49
3.2 Архітектурні рішення та обґрунтування вибору засобів розробки	54
3.3 Конструювання програмного забезпечення	59
3.3.1 Опис структури вхідних/виходів даних	76
3.4 Аналіз безпеки даних	77
Висновки до розділу	77
4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	79
4.1 Аналіз якості ПЗ	79
4.2 Опис процесів тестування.....	81

4.3 Опис контрольного прикладу	88
Висновки до розділу	92
5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	93
5.1 Розгортання програмного забезпечення	93
5.2 Супровід програмного забезпечення	94
Висновки до розділу	95
ВИСНОВКИ	96
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	98
ДОДАТКИ	100

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	– Integrated Development Environment – інтегроване середовище розробки.
API	– Application programming interface, прикладний програмний Інтерфейс.
SDK	– Software development kit.
IT	– Інформаційні технології.
ОС	– Операційна система
AI	– Artificial intelligence, штучний інтелект
Engine	– Програмний рушій, центральна програмна частина
NPC	– Некерований гравцем персонаж.

ВСТУП

Актуальність теми розробки програмного забезпечення на тему «Гра у жанрі RPG на UE5 з використанням ШІ» обумовлена активним розвитком ігрової індустрії та підвищеним інтересом користувачів до ігор з високим рівнем інтерактивності. Одним із головних напрямів цього розвитку є інтеграція технологій штучного інтелекту, що дозволяє значно покращити якість геймплею.

На сьогодні можна спостерігати ефективне і вдале використання ШІ у створенні відеоігор. для розвитку нелінійного ігрового наративу та індивідуалізованого ігрового досвіду. Усі провідні ігрові компанії, зокрема Bethesda Softworks[1], CD Projekt RED[2] та інші, вже впроваджують ШІ у свої проекти.

У рамках даного ДП буде розроблено програмне забезпечення «RPG-гра з використанням ШІ» [RPG-AI-001], яке реалізує інтерактивне рольове середовище з інтелектуальною поведінкою персонажів. Особлива увага буде приділена розробці архітектури ШІ, процедурній генерації ігрового контенту та забезпеченню багатокористувацької взаємодії.

Метою розробки є підвищення якості ігрового досвіду користувачів шляхом реалізації штучного інтелекту для неігрових персонажів

Розробка призначена для забезпечення функціонування ігрових механік, що дозволяють реалізацію оригінального штучного інтелекту гри у жанрі RPG і використовується у галузі розваг для широкого кола користувачів.

Потенційними користувачами є гравці, гейм/левел дизайнери, новачки у сфері ШІ та розробники відеоігор.

1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка завдання дипломного проєктування

Дипломне проєктування передбачає виконання наступних завдань:

- аналіз предметної області та опис її ключових бізнес-процесів, визначення загального завдання розробки у рамках ДП;
- аналіз існуючих рішень (як програмних продуктів, так і алгоритмічних чи технічних рішень) обраного завдання розробки у рамках ДП;
- розроблення функціональних, нефункціональних та системних вимог до програмного забезпечення;
- аналіз економічних показників програмного забезпечення ДП;
- постановка завдання на розробку програмного забезпечення ДП;
- розроблення архітектури програмного забезпечення;
- розроблення архітектурних рішень та обґрунтування вибору засобів розробки програмного забезпечення;
- конструювання та розроблення програмного забезпечення;
- аналіз якості та тестування програмного забезпечення;
- розгортання та супровід програмного забезпечення;
- створення супроводжувальної документації до розробленого програмного забезпечення.

1.2 Аналіз предметної області

Відеоігри – це цифрові інтерактивні продукти, які поєднують у собі програмування, графіку, звук, сюжетну побудову та елементи геймдизайну для створення неповторного користувальського досвіду. Сучасна ігрова індустрія посідає провідні позиції серед галузей інформаційних технологій завдяки стрімкому розвитку. У процесі розробки ігор залучаються спеціалісти різних сфер – від кодерів і дизайнерів до сценаристів і звукових продюсерів, які працюють над іграми різних жанрів і для різних платформ.

Створення відеоігор[3] – це технічно складний процес, який може виконувати як одна особа, так і велика команда в межах студії чи великої компанії. Розробка буває як незалежною – із невеликим бюджетом, так і фінансованою великими інвесторами. Через складність роботи формується чіткий розподіл обов'язків, де кожна роль є ключовою для створення повноцінного ігрового проекту.

Для початку розглянемо роль проектного менеджера, котрий відповідає за декілька основних напрямків, це: планування, контроль та комунікація, тобто роль командного менеджера про взаємодію між командою, про допомогу з планування роботи, яку потрібно робити, розписати хто, в якій послідовності, за що відповідає, а також правильно прокомунікувати із замовником про зміни, котрі відбуваються

Наступна роль – це розробник, рушій проекту, він пише код, тобто створює нові функції продукту, а також виправляє помилки, баги. Розробників можна поділяти на багато підвідів: за спеціалізацією, за мовою програмування, тощо. Розробники дуже багато взаємодіють з тестувальниками, котрі перевіряють його роботу і надають йому фібек.

Тестувальник, або QA-інженер перевіряє наскільки написана функція відповідає вимогам та тому, що замовляв клієнт. Також знаходить помилки в розробці функцій і передає їх розробникам, пишучи тест-кейси, баг-репорти.

Наступна роль – це бізнес аналітик, в задачі якого входять збір думок від клієнтів, створення технічних завдань, приймають вибор у технічних рішеннях певних функцій

Також є геймдизайнер, вигадує і формує ідею гри, її механіки, сюжетні рамки та загальний геймплей. Саме він створює фундамент, на якому будують решту гри.

Дизайнери рівнів, схожі до геймдизайнерів, створюють внутрішній світ гри – вони розміщують об'єкти, продумують логіку проходження та забезпечують цікаві й продумані ситуації, які ведуть гравця через гру.

Сценаристи, що пишуть сюжет, діалоги та описують ігровий світ. Вони створюють історії та характери, які захоплюють гравця.

Художники – створюють усе візуальне наповнення гри, що придумали сценаристи та геймдизайнери: персонажів, світи, анімації, ефекти та інтерфейси. Їхня робота визначає вигляд гри.

Звукова команда додає грі атмосферу за допомогою музики та звукових ефектів. Це звукорежисери, композитори та інженери, які створюють аудіо-супровід.

Актори виконують ролі в іграх: одні працюють у студіях захоплення руху, щоб персонажі виглядали природно, інші озвучують героїв, надаючи їм емоцій і характеру.

Піар-менеджери відповідають за популяризацію гри: вони працюють із медіа, ведуть соціальні мережі, організовують презентації та формують інтерес до проєкту ще до його виходу.

Усі вони об'єднані спільною метою – втілити у життя ідею гри певного жанру. Адже саме жанр визначає основні риси геймплею, стиль подачі, темп і настрій гри. Щоб зрозуміти, як формується команда та якими принципами вона керується, спершу варто дати чітке визначення обраному жанру.

Варто зазначити, що існує безліч різносторонніх жанрів відеоігор, і те, що їх розробка заключається з схожих етапів розробки, але трішки можуть відрізнятись відносно типу гри та підходу самої компанії. На початковому етапі розробки формується загальне бачення майбутньої гри. Створюється дизайн-документ – ключовий документ, у якому детально описується ігровий всесвіт, сюжетна канва, основні механіки та принципи геймплею. Паралельно розробляються перші графічні ескізи – концепт-арт, який складає основне завдання для 3D-художників та Game-дизайнерів.

На цьому ж етапі приймаються стратегічні рішення: хто є цільовою аудиторією проєкту, на яких платформах гра буде доступна та яким чином її буде розповсюджено – через цифрові магазини, підписки чи інші канали. Всі

ці чинники впливають на обсяг необхідних ресурсів, розмір команди та набір фахівців, яких слід залучити.

Центральним елементом концепції гри є її жанр[4]. Це класифікація, яка базується передусім на типових діях, що виконує гравець у процесі проходження. На відміну від жанру в літературі чи кіно, де основну роль відіграє сюжет або тема, в іграх жанр визначається геймплейними характеристиками: наприклад, дослідження світу, вирішення головоломок, стратегічне планування чи бойові дії. Натомість сettтінг – це художнє оформлення гри, її атмосфера, світ і нараторивне тло, які можуть змінюватися незалежно від жанру.

Далі наведено одні з найбільш популярних існуючі жанри відеоігор.

Таблиця 1.1 – Жанри комп’ютерних ігор та їх опис

Жанр	Опис
Платформні ігри (Platform games)	У цих іграх основна механіка полягає в стрибках, переміщенні по різних рівнях і униканні перешкод. Одними з найбільш популярних представників жанру є Celeste, Hollow Knight та Rayman Legends.
Стрілялки (Shooter games)	В основному жанр фокусується на застосуванні вогнепальної зброї, де гравець бореться з ворогами, стріляючи в них або в цілі. Серед класичних ігор можна виділити CSGO, яка вплинула на багато сучасних шутерів.
Ігри-поєдинки (Fighting games)	Ігри, де персонажі змагаються один з одним у поєдинках, використовуючи різні бойові техніки та здібності. Класичними прикладами є Street Fighter V і Tekken 7.
Командні спорт-ігри (Sports-games)	Цей жанр включає ігри, що імітують реальні види спорту, такі як теніс або хокей. Вони можуть бути як для одного гравця, так і для кількох. Серед найбільш відомих можна згадати Rocket League та Skate 3.

Ігри з реалістичним відтворенням досвіду	Ігри, що прагнуть максимально точно відтворити реальне життя або певний процес, та можуть охоплювати безліч тем, від симуляторів містобудування до ігор, що імітують роботу фермера. <i>The Sims 4</i> та <i>Cities: Skylines</i> – яскраві приклади цього жанру.
Рольові ігри або RPG	Дозволяють гравцям створювати персонажів і занурюватися в сюжетні лінії, виконуючи різноманітні місії, взаємодіючи з іншими персонажами і розвиваючи навички. До популярних представників належать <i>The Witcher 3: Wild Hunt</i> , <i>Persona 5</i> та <i>Dark Souls</i> .
Стратегії (Strategy)	В цих іграх акцент робиться на управлінні ресурсами, плануванні та прийнятті стратегічних рішень, а не на безпосередніх бойових діях. <i>StarCraft II</i> та <i>Civilization VI</i> – це чудові приклади таких ігор.
Ігри з динамічним геймплеєм (Dynamic games)	Ці ігри орієнтовані на досягнення високих результатів, збори предметів або виконання різноманітних завдань. Серед відомих прикладів – <i>Pac-Man</i> та <i>Donkey Kong</i> , де гравець постійно прагне покращити свій результат на кожному рівні.

Попри динамічний розвиток ігрової індустрії, розробка відеоігор залишається надзвичайно складною сферою, що стикається з низкою проблем як технічного, так і організаційного характеру. Зокрема, високий поріг входу для інді-розробників: розробка навіть нескладної гри вимагає значного набору знань: від програмування й дизайну рівнів до роботи з 3D-моделюванням, анімацією, UX/UI-дизайном, звуком, сценарієм тощо. Повноцінна 3D-графіка потребує володіння такими інструментами, як *Unreal Engine*, *Unity*, *Blender*, *Substance Painter*, що створює суттєві труднощі для новачків або невеликих команд. Для створення гри часто необхідно комбінувати багато різних інструментів, які не завжди добре інтегруються між собою. Наприклад,

створення моделі в Blender, її текстурування в Substance Painter, анімація в Mixamo і перенесення всього до Unreal Engine – це багатоетапний процес, що потребує ручної адаптації та тестування.

У результаті всебічного дослідження найпопулярніших ігрових жанрів, їхніх особливостей, викликів та візуальних аспектів, було ухвалено рішення реалізувати проект у жанрі RPG з використанням тривимірної графіки. Такий вибір зумовлює необхідність володіння не лише навичками програмування, а й умінням працювати з візуальними ефектами, 3D-моделями персонажів, анімаціями та іншими компонентами. У підсумку це дозволить створити складний та повноцінний ігровий продукт, який відповідає високим стандартам якості, особливо з огляду на те, що проект реалізується одним програмістом.

1.3 Аналіз існуючих рішень

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації гри у жанрі Role-Playing-Game(RPG) на Unreal Engine з використанням ШІ. Далі будуть розглянуті готові програмні рішення, допоміжні програмні засоби та засоби розробки.

1.3.1 Аналіз відомих програмних продуктів

У даному підрозділі буде проведено огляд існуючих на ринку 3D RPG-ігор, зокрема буде зазначено їхніх розробників, використані ігрові рушії, орієнтовні терміни розробки та технологічний стек.

Aralon: Sword and Shadow[5] – це однокористувацька RPG-gra з відкритим світом, розроблена Crescent Moon Games. Гра вийшла спочатку для мобільних платформ, що вже задає певну специфіку її геймдизайну: обмеженість в обсягах контенту, спрощена графіка, та відносно невисока складність AI через апаратні обмеження мобільних пристроїв.



Рисунок 1.1 – Зображення головної сторінки гри Aralon: Sword and Shadow.

Гравець може досліджувати великий світ, виконувати квести, взаємодіяти з NPC, покращувати навички свого персонажа, використовувати інвентар та брати участь у боях. Світ гри розділений на кілька зон, між якими відбувається завантаження. Графічно гра зроблена досить лаконічно, але з урахуванням можливостей мобільних пристройів 2010-х років, вона була технічно якісною для свого часу.



Рисунок 1.2 – Зображення геймплею гри Aralon: Sword and Shadow.

Гра побудована як класична RPG з можливістю створення персонажа: вибір раси: людина, ельф, троль, вибір класу: воїн, розбійник, маг, рейнджер, прокачування характеристик і вмінь

Бої проходять у реальному часі з використанням зброї, заклять і здібностей. Також є крафтінг, риболовля, алхімія та інші побічні активності.

Основна сюжетна лінія обертається навколо героя, що дізнається про своє походження та повинен врятувати королівство від стародавньої загрози. На гравця чекають зради, змови, старі пророчества та геройчні рішення.

Хоч Aralon: Sword and Shadow є вдалим прикладом RPG з відкритим світом, її функціональні можливості обмежені платформою та епохою. Натомість дипломний проект орієнтується на більш сучасну реалізацію з глибшим AI, покращеною системою діалогів та вищим рівнем занурення у геймплей, що дозволяє значно підняти планку якості навіть у межах індивідуального розробника.

Наступним аналогом є Bound by Flame[6] – це рольова гра з елементами екшена, розроблена французькою студією Spiders та видана Focus Home Interactive у 2014 році. Гра вийшла на платформах Windows, PlayStation 3, PlayStation 4 та Xbox 360. Наразі її можна придбати в цифровому форматі через сервіс Steam[7]. Гра позиціонується як проект з акцентом на вибір гравця, бойову систему в реальному часі та моральні дилеми, пов'язані з внутрішнім конфліктом героя – людини, вселеної демонічною сутністю.



Рисунок 1.3 – Головна сторінка гри Aralon: Sword and Shadow.

Bound by Flame вирізняється рядом ключових особливостей, що формують глибину ігрового процесу. Однією з них є розгалужена система діалогів, у якій гравець має змогу обирати репліки, що безпосередньо впливають на розвиток сюжету, відносини з компаньонами та доступні подальші дії.

Бойова система реалізована в реальному часі та передбачає перемикання між кількома стилями ведення бою, що дозволяє гравцеві адаптуватися до різних типів ворогів і ситуацій. Окрім того, передбачене прокачування персонажа за кількома гілками навичок, що дає можливість сформувати унікальний стиль гри.

Важливою складовою гри є система супутників – кожен компаньйон має власний характер, погляди та може підтримати або вступити в конфлікт із головним героєм залежно від його рішень, аж до повного припинення співпраці. Особливу роль відіграє моральний вибір: гравцеві доведеться вирішувати, чи дозволити демонічній сутності, яка оселилася в головному герої, поступово його поглинати задля здобуття сили, або ж зберігати людяність, обмежуючи свій потенціал.

Візуальна складова гри створена за допомогою власного ігрового рушія студії Spiders – Silk Engine, що дозволило реалізувати оригінальний графічний стиль і підтримку різних платформ на час релізу.



Рисунок 1.4 – Зображення геймплею гри Bound by Flame.

Risen[8] – це відеогра в жанрі Action RPG, створена німецькою студією Piranha Bytes та випущена видавництвом Deep Silver. Ця гра стала першою в серії Risen і була випущена у жовтні 2009 року в Європі для Microsoft Windows та Xbox 360. Відзначена своєю глибокою сюжетною лінією та відкритим світом, Risen привернула увагу гравців, які цінують складні RPG з великою кількістю вибору та інтерактивних елементів.

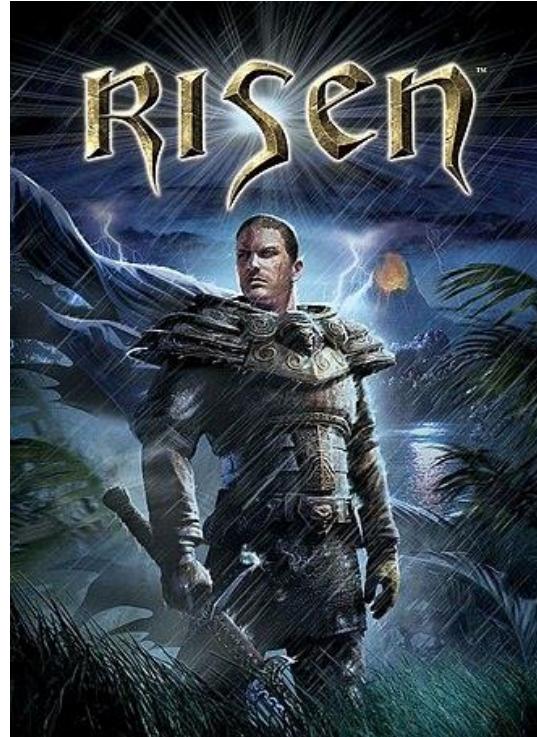


Рисунок 1.5 – Заставка гри Risen

Risen – це класична рольова гра з відкритим світом, що вирізняється глибокою системою розвитку персонажа. Гравець має змогу розвивати свого героя, інвестуючи очки навчання у різноманітні вміння: бойові навички, магію, алхімію, ковальство та інші. Такий підхід дозволяє створити унікального персонажа, адаптованого до індивідуального стилю гри. Однією з важливих особливостей гри є можливість приєднання до різних фракцій. Вибір фракції напряму впливає на сюжетну лінію, доступні квести та загальну траєкторію розвитку персонажа, що створює ефект занурення та варіативність проходження.



Рисунок 1.6 – Зображення геймплею гри Risen.

Бойова система реалізована в реальному часі та дозволяє використовувати різні типи зброї й магічних умінь. При цьому гравець може блокувати, ухилятися від атак і комбінувати удари, що надає динаміки й тактичної глибини бойовим зіткненням. Окрема увага приділена системі магії, яка базується на застосуванні магічних кристалів, рун і сувоїв. Це відкриває широкий спектр заклинань, як бойового, так і підтримувального чи захисного характеру.

Графіка в грі реалізована за допомогою власного рушія Genome Engine, який забезпечує деталізацію середовища, стабільну роботу з відкритим світом та атмосферну візуальну складову. Завдяки цьому гравець отримує повноцінний досвід занурення у фентезійний світ, насичений подіями, вибором і наслідками.

Що стосується штучного інтелекту (AI), то в грі «Risen» AI ворогів і NPC працює на основі скриптових шаблонів поведінки, що досить обмежує їхні можливості в адаптації до змін в ігровому середовищі. Водночас, дипломний проект передбачає реалізацію AI через поведінкові дерева з елементами навчання, що дозволяє ворогам реагувати на зовнішні стимули, змінювати тактику в залежності від ситуації, взаємодіяти з оточенням та навіть використовувати командну роботу. Цей підхід значно покращує

реалістичність ігрового процесу, що особливо важливо для RPG з відкритим світом.

Для порівняння цього проекту з аналогами можна скористатися таблицею 1.3

Таблиця 1.2 – Порівняння з аналогами

Функціонал	RPG-AI-001	Aralon: Sword and Shadow	Bound by Flame	Risen	Пояснення
Розробник	Один розробник	Crescent Moon Games	Spiders	Piranha Bytes	У ДП проєкт виконано одноосібно
Ігровий Рушій	Unreal Engine 5	Unity	PhyreEngine	Genome Engine	У ДП використано сучасний UE5 з Nanite, Lumen
Бюджет	0 \$ (нульовий)	~100,000 \$	~2 млн \$	від ~5 млн \$	ДП реалізовано повністю без фінансування
Кількість розробників	1	~5	~20	~25	Навіть малі студії мають команду, у ДП все створено самостійно
Тривалість розробки	6–8 місяців	~1.5 роки	~2 роки	~3 роки	У ДП – ефективна розробка за

					короткий термін
--	--	--	--	--	-----------------

Порівняльна таблиця функціоналу та особливостей, яка представлена в таблиці 1.3, буде використовуватися для порівняння механіки поведінки ворогів у контексті розробки ДП, оскільки дозволяє наочно оцінити відмінності та подібності між різними аспектами ігрового процесу.

Таблиця 1.3 – Порівняння функціоналу та особливостей механіки поведінки AI.

Функціонал поведінки AI	RPG-AI-001	Aralon: Sword and Shadow	Bound by Flame	Risen	Пояснення
Патрулювання території	+ (на шляху або випадково)	-	+	+ (по зоні)	У ДП є динамічні та прості маршрути патрулювання
Реакція на шум / зону агр	+	-	+ (фіксована зона)	+ (обмежена зона)	У ДП – зона агр змінюється залежно від рівня ворога
Зміна поведінки при низькому НР	+ (втеча або захист)	-	-	-	Реалізовано через дерево рішень
Командна взаємодія ворогів	+	+	-	-	Є поділ на команди

Уникнення атак / ухилення	+	-	-	+	У ДП вороги можуть ухилятись на складному рівні
Використання різних тактик	+	+	+	+(деякі типи)	У Bound by Flame тактики залежать від типу
Застосування вмінь / здібностей	+	-	+(через скрипт)	+(деякі типи)	У ДП – кастування залежить від ситуації

1.3.2 Аналіз відомих алгоритмічних та технічних рішень

У цьому розділі розглянуто основні алгоритмічні та технічні рішення, які можуть бути використані при створенні інтелектуальної поведінки персонажів у рамках дипломного проєкту, орієнтованого на жанр 3D RPG. Особливу увагу приділено можливостям Unreal Engine 5 (UE5) – рушія, обраного для реалізації даного проєкту.

Одним з найбільш поширених підходів до реалізації AI в іграх є використання дерев поведінки. Вони забезпечують гнучкий і модульний спосіб моделювання поведінки агентів. У UE5 система Behavior Tree вбудована та тісно інтегрується з іншими механізмами рушія, як вузлова структура, що дозволяє легко змінювати логіку, повторне використання поведінки через шаблони та поєднання з BlackBoard для передачі даних між підсистемами.

Цей підхід є ідеальним для реалізації поведінки ворогів у RPG: від патрулювання та переслідування до складних бойових маневрів.

Blackboard виступає як «пам'ять» NPC, де зберігається поточна інформація: цілі, розташування ворога, стан NPC тощо. Система дозволяє

централізовано змінювати поведінку, що робить її дуже ефективною в складних сценаріях.

AI Perception System[9] - система дозволяє NPC відчувати навколошній світ за допомогою сенсорів зору (Sight) для розпізнавання ворогів в межах поля зору, слуху (Hearing) для реагування на звуки (наприклад, кроки гравця) та сенсор шкоди (Damage): реакція на отримання ушкоджень.

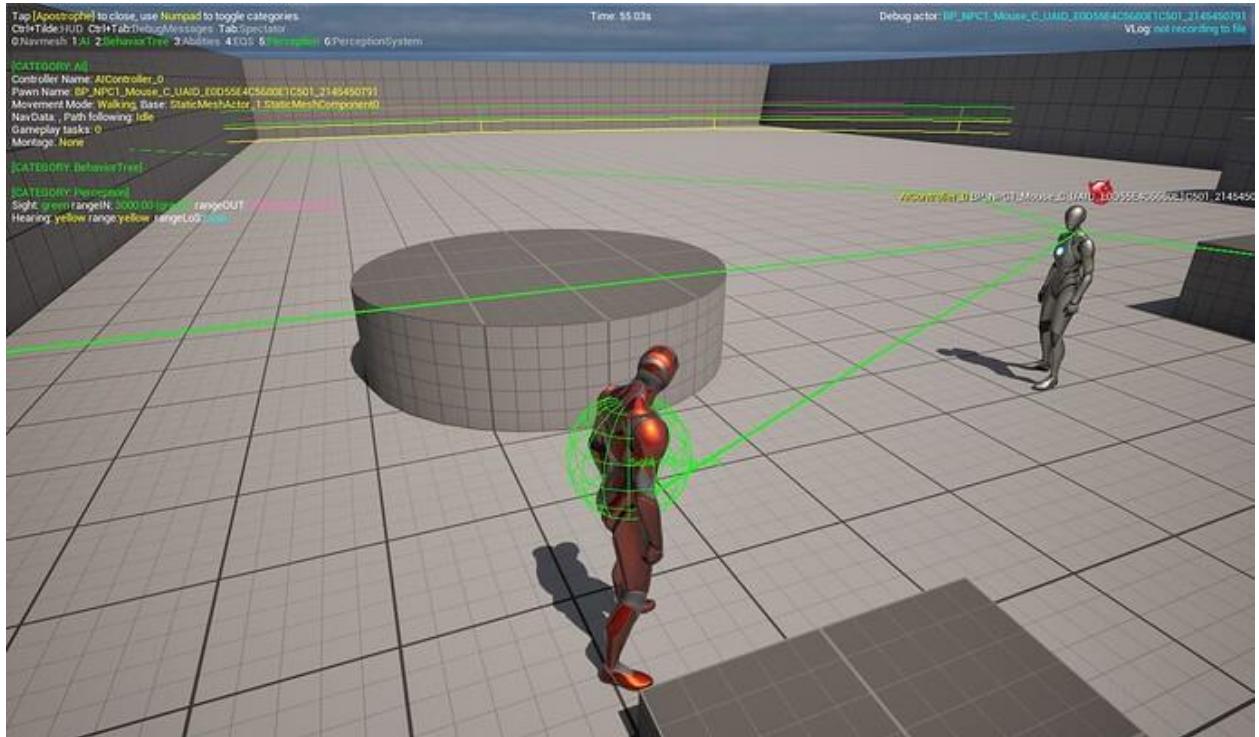


Рисунок 1.7 – Система сприйняття AI (AI Perception System).

Її гнучке налаштування дозволяє створити живий світ, в якому персонажі реагують на різноманітні стимули.

Unreal Engine 5 надає потужні інструменти для створення інтелектуальної поведінки NPC, що дозволяє розробникам втілювати складні й адаптивні сценарії у грі. Завдяки Behavior Tree Editor[10], можна легко створювати дерева рішень для управління логікою поведінки персонажів. Важливим компонентом є AI Controller, який дозволяє NPC виконувати інтелектуальні дії в залежності від ситуації.

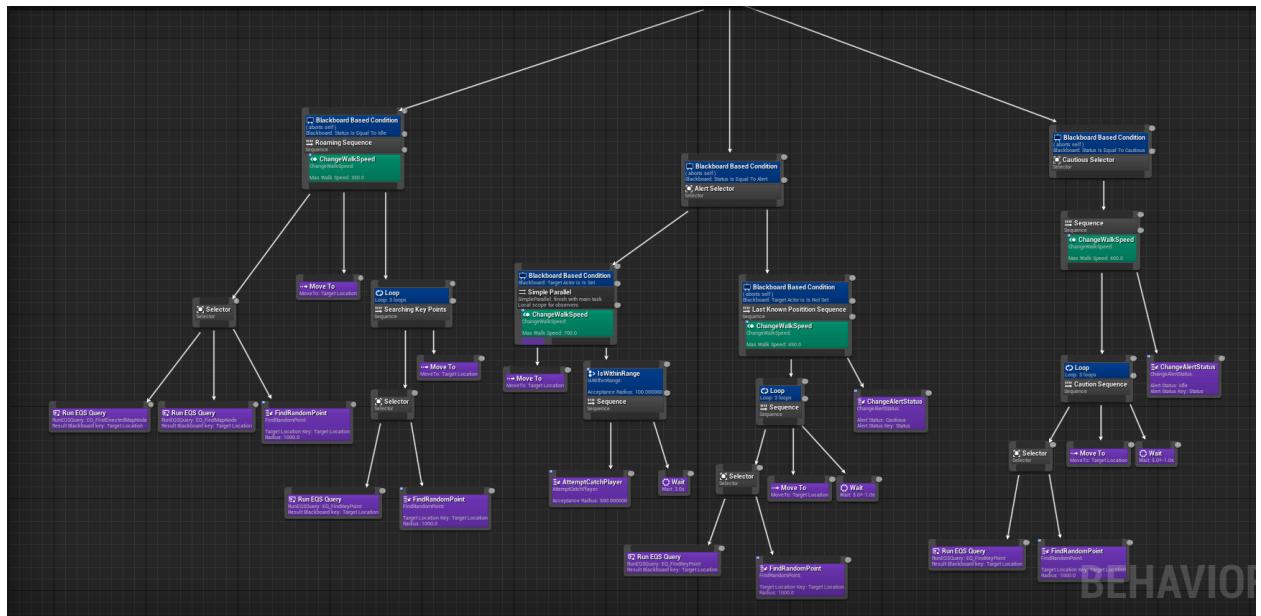


Рисунок 1.8 – Редактор дерев поведінки (Behavior Tree Editor) для налаштування AI.

Система NavMesh[11] відповідає за навігацію персонажів у просторі, допомагаючи їм уникати перешкод, переслідувати гравця або тікати. Це дає можливість NPC адаптувати свою поведінку навіть у динамічному середовищі гри.

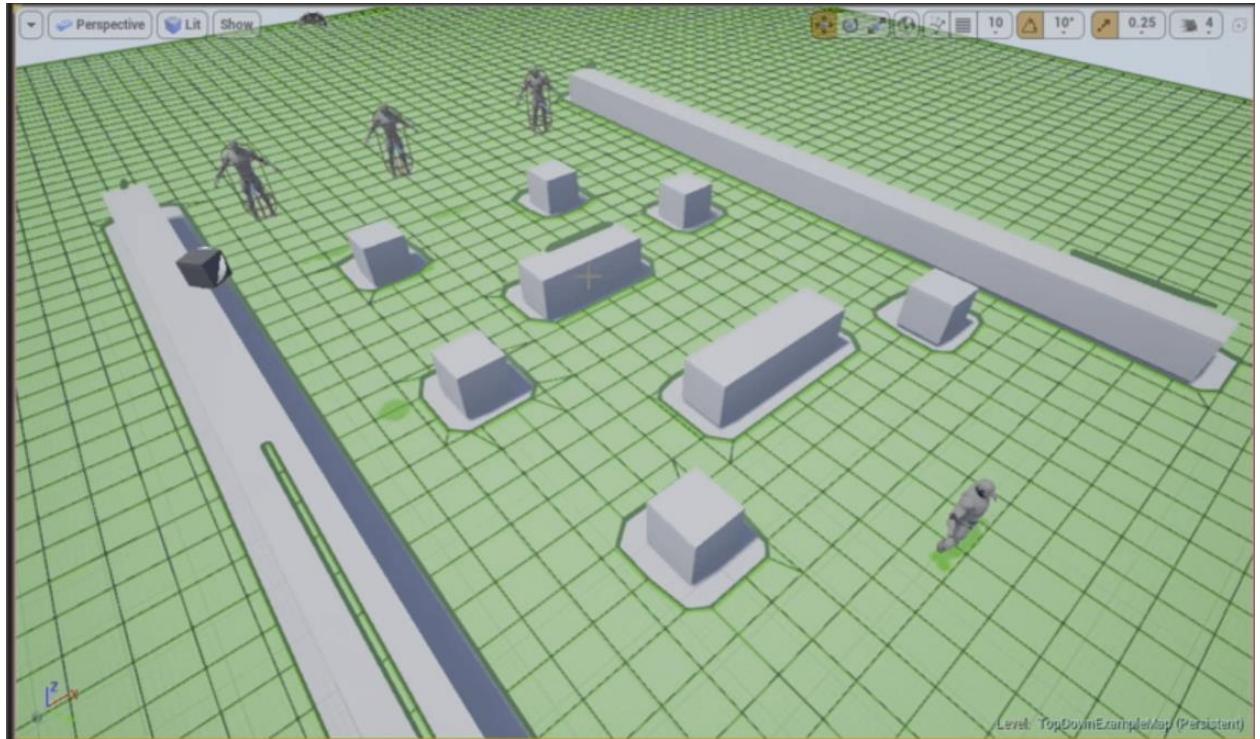


Рисунок 1.9 – Навігаційна мережа (NavMesh) для забезпечення руху AI в ігровому світі.

Крім того, для інтеграції поведінки з анімацією використовуються Animation Blueprints та State Machines[12]. Це дозволяє NPC не лише виконувати логічні дії, а й відповідно рухатися та реагувати на змінювані обставини, що робить їх поведінку більш реалістичною.

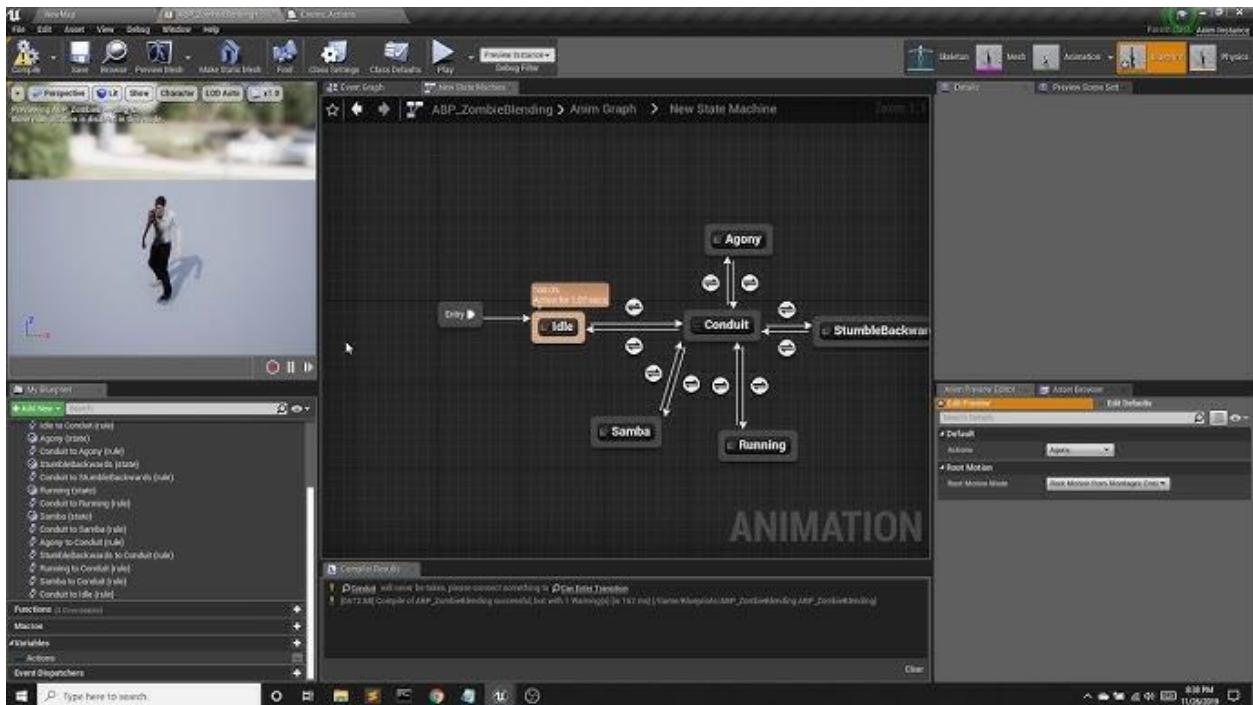


Рисунок 1.10 – Анімаційні блакитні плани (Animation Blueprints) та машини станів (State Machines) для управління анімацією персонажа в грі.

У таблиці 1.4 здійснено порівняння основних підходів та систем, що використовуються для розробки механіки штучного інтелекту (AI) у проєкті. У таблиці зазначено переваги, недоліки та доцільність використання кожної з представлених технологій, що допомагає обрати оптимальне рішення для реалізації різних аспектів AI.

Таблиця 1.4 – Порівняльний аналіз рішень щодо підходів у розробці механіки гри.

Підхід/система	Недоліки	Переваги	Доцільність використання
Behavior Tree	Гнучкість, модульність, інтеграція з UE5	Може стати складним при великій кількості станів	Основна система для AI у проєкті

Perception System	Реалістична реакція NPC на світ	Потребує налаштування для кожного NPC	Активно використовується в проєкті
NavMesh	Орієнтація просторі, автоматичне оновлення	Обмеження складних тривимірних середовища	Використовується для пошуку шляху

Для реалізації AI у дипломному проєкті було обрано Behavior Tree з Blackboard, як основна архітектура для реалізації логіки NPC, AI Perception System для імітації зору та слуху, NavMesh для пересування та обрання маршрутів та Animation Blueprint для зв'язку логіки з візуальними діями.

1.4 Аналіз та моделювання бізнес-процесів

У рамках даного ДП здійснено аналіз процесів, пов'язаних із функціонуванням RPG-гри з використанням ШІ. Із загальної множини процесів були обрані ключові, що безпосередньо впливають на якість ігрового процесу та забезпечують основні функціональні можливості програмного забезпечення.

Для моделювання процесу запуску гри використовується BPMN модель, що зображена на рисунку 1.14

Процес запуску гри:

- Користувач запускає застосунок.
- Початковий процес – відкриття застосунку, де система відображає екран з трьома кнопками: Continue, New Game та Exit
- Continue: Якщо користувач вибирає цю кнопку, система перевіряє, чи є збережена гра.
- New Game: Якщо користувач вибирає нову гру, застосунок ініціює процес запуску нової сцени гри.

- Exit: Якщо користувач вибирає цю кнопку, процес закінчується, застосунок закривається.
- Якщо є збережене ігрове середовище, система запускає збережену сцену.
- Якщо збереження відсутнє, система запускає нову сцену гри.
- Після запуску збереженої або нової сцени користувач потрапляє в ігровий світ і починається активний ігровий процес.

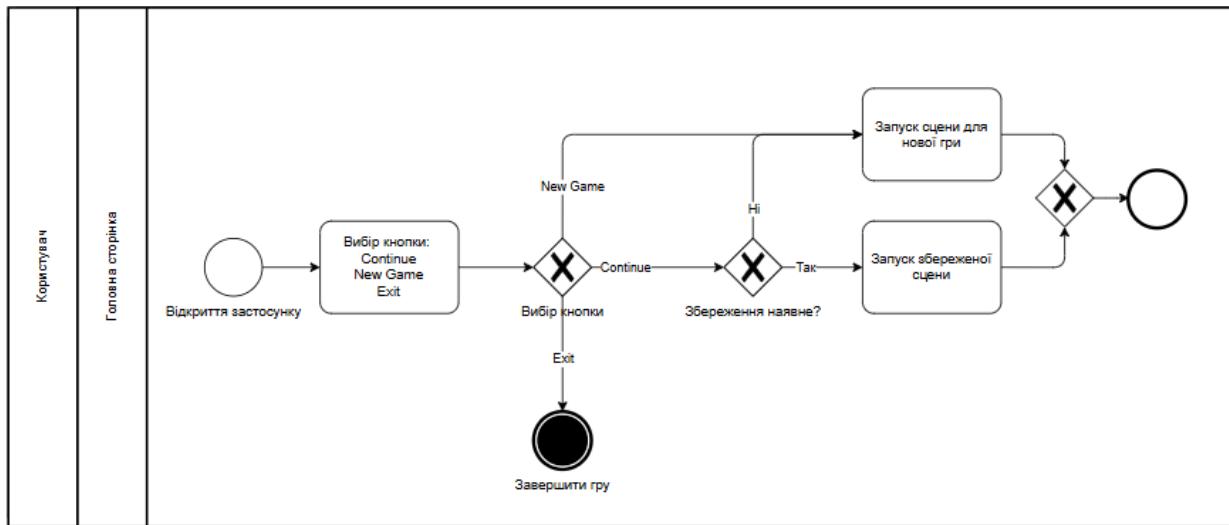


Рисунок 1.11 – BPMN модель процесу запуску гри.

Для моделювання процесу вибору режиму бою та реалізації бойової взаємодії використовується BPMN модель, що зображена на рисунку 1.15

Процес ініціації та проведення бою

Процес розпочинається з того, що користувач перебуває в ігровому світі та ініціює бойову взаємодію, бажаючи атакувати супротивника. Далі він обирає режим бою: близький бій або магічна атака.

- У режимі близького бою користувач приймає рішення щодо блокування:
- Якщо блок не активується, виконується атака, після чого система розраховує завдану шкоду.
- Якщо блок увімкнено, перевіряється, чи був блок ідеальним:

- Якщо так – відбувається париування.
- Якщо ні – виконується розрахунок ефективності блоку.
- У режимі магічного бою користувач здійснює прицілювання, після чого система проводить розрахунок влучання та шкоди.
- Завершальним етапом у будь-якому з варіантів є досягнення цілі та завершення дії бою.

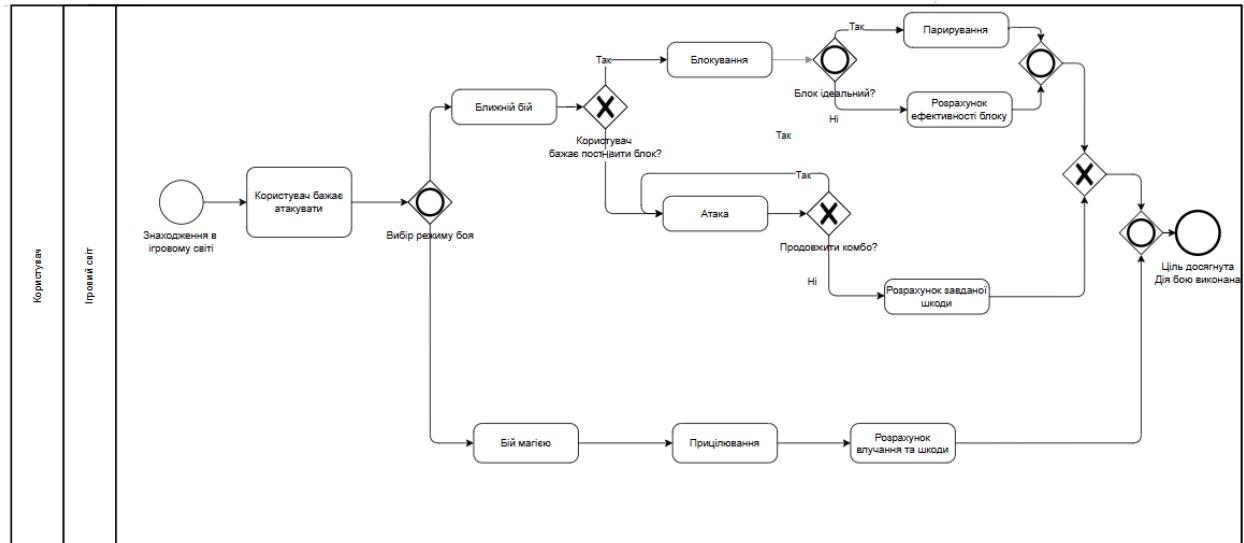


Рисунок 1.12 – BPBMN-модель вибору режиму бою та реалізації бойової взаємодії.

Для моделювання процесу процесу взаємодії користувача з об'єктами та персонажами в ігровому світі використовується BPBMN модель, що зображена на рисунку 1.16

Процес взаємодії користувача з об'єктами та персонажами в ігровому світі

Процес починається з того, що гравець перебуває в ігровому світі та бажає взаємодіяти з елементами середовища. Першим кроком є вибір об'єкта або персонажа для взаємодії, після чого система розгалужується залежно від типу цілі:

- Якщо це ворог, користувач приймає рішення, чи бажає атакувати. Якщо так – ініціюється бій. Якщо ні – активується втеча.

- Якщо це інший NPC (неігровий персонаж):

Перевіряється, чи гравець перебуває на достатній відстані: Якщо ні – гравець підходить ближче. Якщо так – розпочинається діалог.

- Якщо це предмет взаємодії, система перевіряє, чи має користувач доступ до предмета. Якщо так – відбувається отримання нагороди.
- Якщо це предмет інвентарю, виконується підняття предмета.
- Після завершення взаємодії користувач продовжує рух у світі.

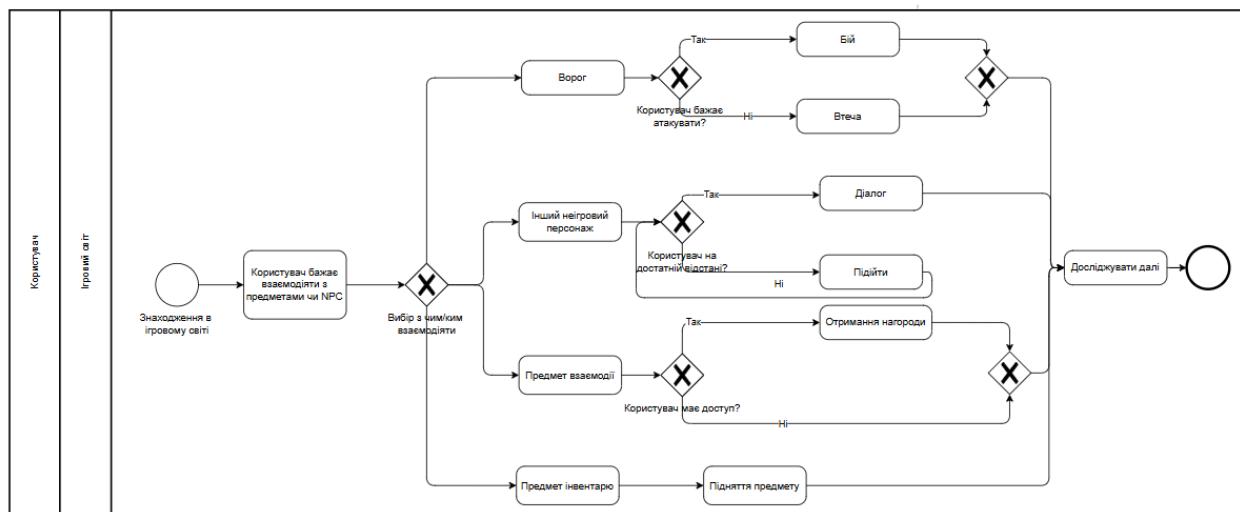


Рисунок 1.13 – BPMN модель процесу взаємодії користувача з об’єктами та персонажами в ігровому світі

Для моделювання ситуації, коли користувач вирішує вийти з ігрового світу до основного меню, використовується BPMN-модель, що зображена на рисунку 1.17

Процес збереження гри при виході в головне меню

Процес починається з того, що гравець перебуває в ігровому світі та ініціює вихід у основне меню, система пропонує користувачу зберегти прогрес

- Якщо гравець обирає «так», відбувається збереження даних, після чого здійснюється вихід до основного меню. це предмет інвентарю, виконується підняття предмета.
- Якщо гравець відмовляється зберігати, система виводить його до головного меню без збереження прогресу.

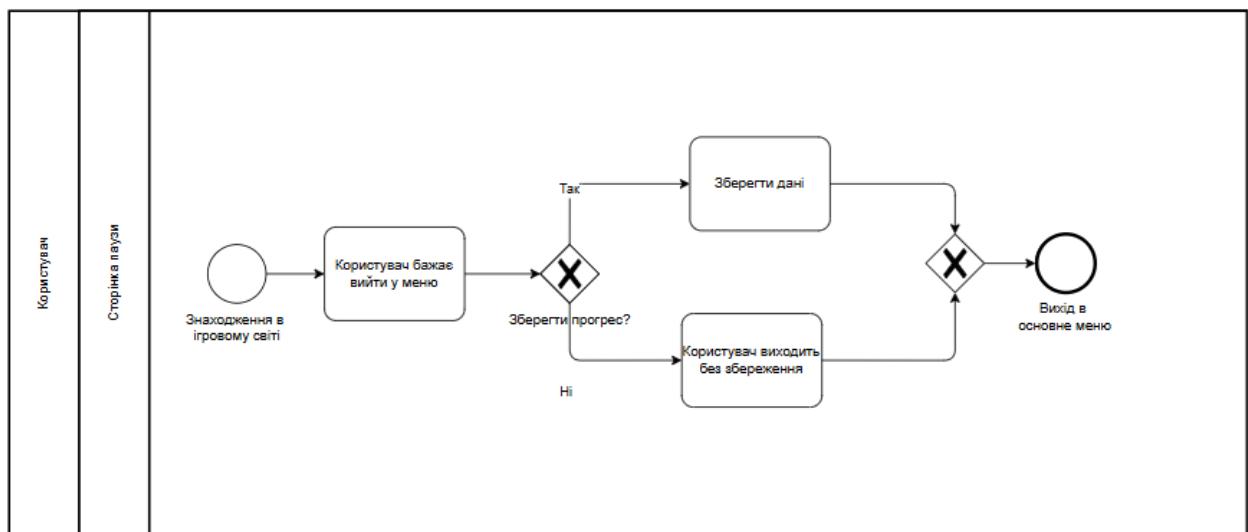


Рисунок 1.14 – BPMN модель процесу збереження гри при виході в головне меню

Висновки до розділу

У цьому розділі було здійснено комплексну підготовку до дипломного проектування, яка включає аналіз предметної області, формулювання цілей і завдань, а також огляд і оцінку наявних програмних та алгоритмічних рішень.

Окреслено основні завдання дипломного проекту: від аналізу предметної області до розгортання ПЗ і підготовки документації. Проведено аналіз діяльності фахівців та особливостей жанрів відеогор, що стало підґрунтям для вибору жанру RPG з 3D-графікою. Проаналізовано існуючі програмні продукти (RPG-ігри), їх рушії, бюджети, функціональність, а також здійснено порівняльний аналіз механік AI. Розглянуто технічні підходи: Behavior Tree, Perception System, NavMesh, Animation Blueprint, обґрунтовано

їх вибір для реалізації AI у проекті. Змодельовано бізнес-процеси гри за допомогою BPMN-діаграм, що дозволило формалізувати логіку ключових сценаріїв взаємодії користувача з грою.

У результаті сформовано концепцію програмного забезпечення для RPG-гри з інтегрованим штучним інтелектом.

2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Варіанти використання програмного забезпечення

У цьому підрозділі представлено діаграму варіантів використання (Use Case Diagram) для проекту RPG-гри. Вона ілюструє основні взаємодії користувача з ігровою системою. На рисунку 2.1 відображено діаграму варіантів використання з ключовими акторами та основними функціями.

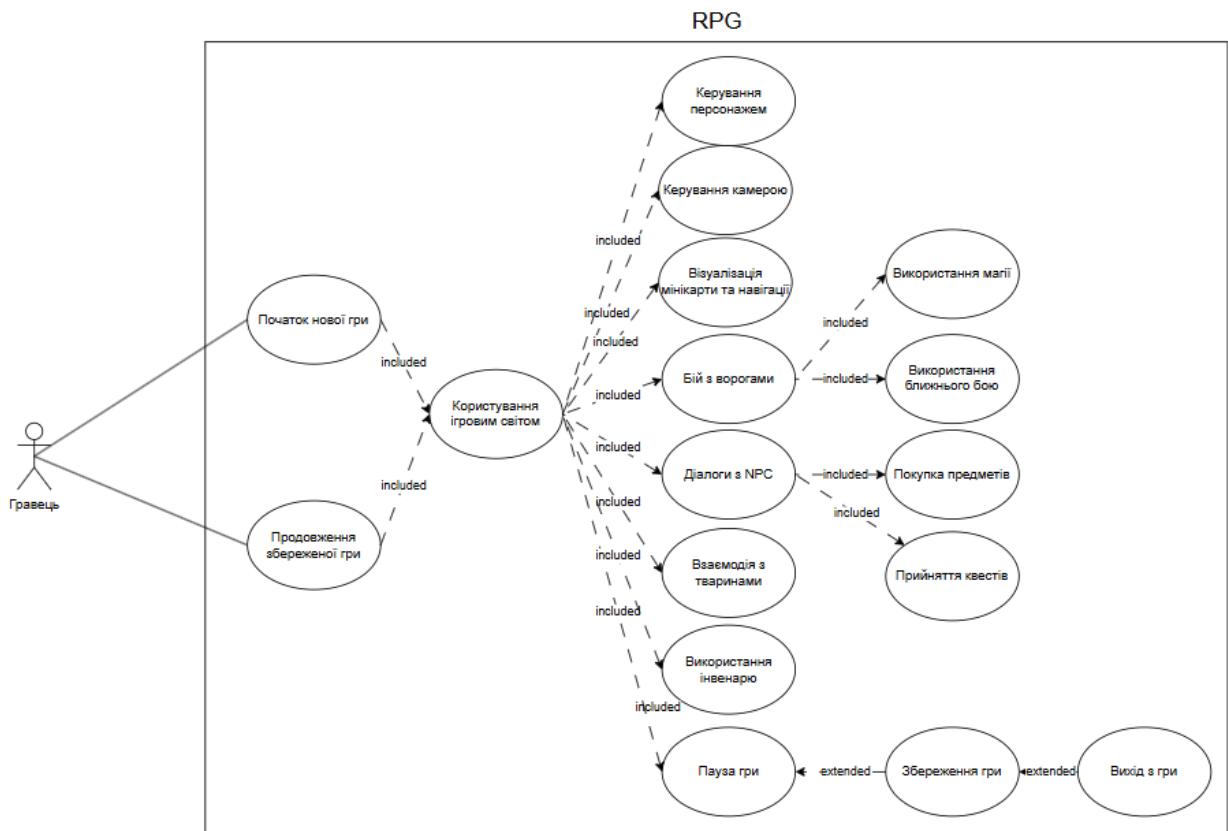


Рисунок 2.1 – Діаграма варіантів використання

В таблицях 2.1-2.15 наведено опис варіантів використання програмного забезпечення.

Таблиця 2.1 – Варіант використання UC-01

Use case name	Початок нової гри
Use case ID	UC-01
Goals	Ініціалізувати нову сесію гри з початковими налаштуваннями.

Actors	Гравець
Trigger	Вибір пункту «Нова гра» в головному меню.
Pre-conditions	Гра запущена, гравець у головному меню.
Flow of Events	<ol style="list-style-type: none"> 1. Гравець обирає «Нова гра». 2. Система завантажує стартову локацію. 3. Гравцю надається можливість керування персонажем
Extension	Гравець може налаштовувати параметри.
Post-Condition	Гравець перебуває у грі з новим персонажем і прогресом з нуля.

Таблиця 2.2 – Варіант використання UC-02

Use case name	Продовження збереженої гри
Use case ID	UC-02
Goals	Відновити прогрес з останнього збереження.
Actors	Гравець
Trigger	Гравець натискає «Продовжити гру» у меню.
Pre-conditions	Існує хоча б одне збереження гри.
Flow of Events	<ol style="list-style-type: none"> 1. Система перевіряє наявність збережень. 2. Завантажується останнє збереження. 3. Гравець знову керує персонажем у відповідному місці.
Extension	Якщо збереження пошкоджене – виводиться повідомлення про помилку.
Post-Condition	Гравець повертається у гру з усім попереднім прогресом.

Таблиця 2.3 – Варіант використання UC-03

Use case name	Вхід до ігрового світу
Use case ID	UC-03
Goals	Ініціалізувати вхід до ігрового середовища.
Actors	Гравець
Trigger	Завантаження гри.

Pre-conditions	Гра завантажена, доступне збереження або стартова точка.
Flow of Events	1. Система створює або завантажує світ. 2. Гравець отримує контроль над персонажем.
Extension	Світ може бути частково завантаженим
Post-Condition	Ігровий процес розпочато.

Таблиця 2.4 – Варіант використання UC-04

Use case name	Керування персонажем
Use case ID	UC-04
Goals	Керування діями персонажа.
Actors	Гравець
Trigger	Натискання клавіш або комбінації клавіш.
Pre-conditions	Персонаж активний у грі.
Flow of Events	1. Гравець натискає клавіші керування. 2. Персонаж виконує відповідну дію (рух, стрибок, взаємодія).
Extension	Залежно від стану персонажа – деякі дії можуть бути заблоковані.
Post-Condition	Дія виконана, стан світу оновлено.

Таблиця 2.5 – Варіант використання UC-05

Use case name	Керування камерою
Use case ID	UC-05
Goals	Змінювати ракурс огляду.
Actors	Гравець
Trigger	Рух миші
Pre-conditions	Персонаж активний у грі.
Flow of Events	1. Гравець змінює положення камери. 2. Відповідно змінюються зображення на екрані.

Extension	У деяких режимах камера може бути фіксованою.
Post-Condition	Камера встановлена у нове положення.

Таблиця 2.6 – Варіант використання UC-06

Use case name	Відображення мінікарти та навігації
Use case ID	UC-06
Goals	Отримати інформацію про місце розташування та маршрут.
Actors	Гравець
Trigger	Гравець бажає отримати навігаційну інформацію
Pre-conditions	Персонаж у грі та має доступ до карти
Flow of Events	<ul style="list-style-type: none"> 1. Система відображає мінікарту. 2. Гравець бачить своє місцезнаходження та точки інтересу.
Extension	На мапі можуть з'являтись нові маркери або вороги.
Post-Condition	Гравець має уявлення про навігацію.

Таблиця 2.7 – Варіант використання UC-07

Use case name	Бій з ворогами
Use case ID	UC-07
Goals	Перемогти противника в бою.
Actors	Гравець, Противник
Trigger	Вхід у бойову зону або ініціація бою.
Pre-conditions	Гравець має активну зброю/магію, противник виявлений.
Flow of Events	<ul style="list-style-type: none"> 1. Гравець вступає у бій. 2. Використовується близький бій або магія. 3. Система розраховує результат ударів. 4. Оновлюється стан противника.
Extension	Як противник, так і гравець може ухилятись або блокувати удари.

Post-Condition	Супротивник переможений або бій триває
----------------	--

Таблиця 2.8 – Варіант використання UC-08

Use case name	Використання магії
Use case ID	UC-08
Goals	Завдати шкоди супротивнику за допомогою заклинань.
Actors	Гравець
Trigger	Вибір режиму магії під час бою.
Pre-conditions	Противник у зоні досяжності
Flow of Events	<ol style="list-style-type: none"> 1. Гравець обирає магічну атаку. 2. При влучанні система завдає шкоди противнику.
Extension	Магія може промахнутись або бути заблокованою.
Post-Condition	Супротивник зазнає шкоди або блокує атаку

Таблиця 2.9 – Варіант використання UC-09

Use case name	Використання ближнього бою
Use case ID	UC-09
Goals	Завдати шкоди супротивнику у ближньому бою.
Actors	Гравець
Trigger	Вибір режиму ближнього бою
Pre-conditions	Противник у зоні досяжності, зброя доступна
Flow of Events	<ol style="list-style-type: none"> 1. Гравець наносить удари. 2. Можливе виконання комбо. 3. При влучанні система завдає шкоди противнику.
Extension	Частина ударів може бути відхиlena або заблокованою.
Post-Condition	Супротивник зазнає шкоди або блокує атаку

Таблиця 2.10 – Варіант використання UC-010

Use case name	Використання ближнього бою
Use case ID	UC-10
Goals	Завдати шкоди супротивнику у ближньому бою.
Actors	Гравець
Trigger	Вибір режиму ближнього бою
Pre-conditions	Противник у зоні досяжності, зброя доступна
Flow of Events	<ul style="list-style-type: none"> 1. Гравець наносить удари. 2. Можливе виконання комбо. 3. При влучанні система завдає шкоди противнику.
Extension	Частина ударів може бути відхиlena або заблокована.
Post-Condition	Супротивник зазнає шкоди або блокує атаку

Таблиця 2.11 – Варіант використання UC-011

Use case name	Діалог з NPC
Use case ID	UC-11
Goals	Провести розмову з неігровим персонажем.
Actors	Гравець
Trigger	Гравець наближається до NPC та активує взаємодію.
Pre-conditions	NPC активний і доступний для діалогу.
Flow of Events	<ul style="list-style-type: none"> 1. Відкривається інтерфейс діалогу. 2. Гравець продовжує діалог. 3. NPC відповідає, змінюється ставлення або відкриваються нові опції.
Extension	Можливість отримати квест або відкрити магазин.
Post-Condition	Діалог завершено, можлива нова дія (торгівля, завдання тощо).

Таблиця 2.12 – Варіант використання UC-012

Use case name	Використання інвентарю
Use case ID	UC-12
Goals	Користуватись предметами в інвентарі.
Actors	Гравець
Trigger	Відкриття інвентаря через меню.
Pre-conditions	Персонаж має доступ до інвентаря.
Flow of Events	<ol style="list-style-type: none"> 1. Відкривається вікно інвентаря. 2. Гравець переглядає предмети. 3. Обирає дію: використати, викинути, перемістити, екіпірувати.
Extension	Якщо предмет має умову (наприклад, рівень), вона перевіряється.
Post-Condition	Стан інвентаря змінено відповідно до дії.

Таблиця 2.13 – Варіант використання UC-013

Use case name	Збереження стану гри
Use case ID	UC-13
Goals	Зберегти поточний прогрес гри.
Actors	Гравець
Trigger	Вибір пункту «Зберегти гру» в меню паузи або на точці збереження.
Pre-conditions	Дозволено збереження на даному етапі.
Flow of Events	<ol style="list-style-type: none"> 1. Гравець відкриває меню паузи. 2. Гравець натискає «Зберегти гру». 3. Система записує поточні дані.
Extension	Якщо збереження неможливе – виводиться повідомлення.
Post-Condition	Прогрес успішно збережено.

Таблиця 2.14 – Варіант використання UC-014

Use case name	Пауза гри
Use case ID	UC-14
Goals	Тимчасово призупинити гру.
Actors	Гравець
Trigger	Натискання клавіші паузи.
Pre-conditions	Ігрова сесія активна.
Flow of Events	<ol style="list-style-type: none"> 1. Система призупиняє ігровий процес. 2. Відображається меню паузи. 3. Гравець може обрати дію: вийти в меню, зберегення, відновити.
Extension	-
Post-Condition	Гравець повертається до гри або виконує іншу дію.

Таблиця 2.15 – Варіант використання UC-015

Use case name	Вихід з гри
Use case ID	UC-15
Goals	Завершити сесію гри та вийти у головне меню або на робочий стіл.
Actors	Гравець
Trigger	Вибір пункту «Вийти» у головному меню.
Pre-conditions	Гра запущена.
Flow of Events	<ol style="list-style-type: none"> 1. Гравець відкриває меню. 2. Гравець обирає «Вийти». 3. Відбувається вихід з гри.
Extension	Можлива втрата прогресу, якщо не зберегтись
Post-Condition	Гравець завершив гру.

2.2 Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. В таблиці 2.15 наведено загальну модель вимог, а в таблиці 2.16 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити на рисунку 2.3.

Таблиця 2.15 – Загальна модель вимог

№	Назва	ID Вимоги	Пріоритети	Ризики
1	Інтерфейс взаємодії з користувачем	FR-1	Високий	Середній
2	Початок нової гри	FR-2	Високий	Середній
3	Продовження збереженої гри	FR-4	Високий	Середній
4	Переміщення персонажа по світу	FR-5	Високий	Середній
4.1	Управління камерою	FR-6	Високий	Середній
4.2	Використання бойових навичок	FR-7	Високий	Високий
4.3	Взаємодія з оточуючими об'єктами	FR-8	Високий	Середній
5	Підтримка карти та навігації	FR-9	Високий	Середній
5.1	Відображення об'єктів на карті	FR-10	Середній	Середній
5.2	Відображення локації поточного персонажа на карті	FR-11	Середній	Середній
5.3	Відображення противників	FR-12	Середній	Середній
6	Перегляд та взаємодія з предметами	FR-13	Високий	Середній
6.1	Здобуття та використання предметів	FR-14	Високий	Середній
6.2	Переміщення предметів у інвентарі	FR-15	Середній	Низький

6.3	Показ характеристик та опису предметів	FR-16	Середній	Низький
7	Відображення характеристик персонажа	FR-17	Високий	Низький
7.1	Показ рівня розвитку персонажа	FR-18	Високий	Низький
7.2	Перегляд ігрової валюти	FR-19	Високий	Низький
7.3	Покращення характеристик	FR-20	Високий	Середній
8	Здобуття предметів	FR-21	Високий	Низький
9	Спілкування з іншими персонажами (NPC)	FR-22	Високий	Середній
9.1	Початок діалогу	FR-23	Середній	Середній
9.2	Вибір у діалозі	FR-24	Середній	Середній
9.3	Наслідки вибору у діалозі	FR-25	Середній	Середній
10	Перегляд поточних характеристик персонажа	FR-26	Високий	Низький
11	Поведінка персонажів з штучним інтелектом	FR-27	Високий	Високий
12	Бій з ворогами	FR-28	Високий	Високий
12.1	Використання магії у бою	FR-29	Високий	Середній
12.2	Близький бій з ворогами	FR-30	Високий	Середній
12.3	Використання комбо	FR-31	Середній	Низький
13	Взаємодія з тваринами	FR-32	Середній	Низький
14	Пауза гри	FR-33	Високий	Низький
15	Збереження прогресу гри	FR-34	Високий	Високий
16	Вихід з гри	FR-35	Високий	Низький

Таблиця 2.16 – Перелік функціональних вимог

Назва	Опис
FR-1	Початок нової гри Користувач може розпочати нову гру, вибравши відповідну опцію з головного меню.
FR-2	Продовження збереженої гри. Користувач може завантажити раніше збережений прогрес.
FR-3	Збереження прогресу гри. Гравець може зберегти поточний стан гри вручну або автоматично
FR-4	Вихід з гри. Користувач може завершити ігрову сесію через меню.
FR-5	Інтерфейс взаємодії з користувачем. Гравець взаємодіє з грою через HUD, інвентар, карту та меню
FR-6	Переміщення персонажа по світу. Гравець керує переміщенням персонажа в межах ігрового середовища.
FR-7	Управління камерою. Користувач може змінювати кут огляду та масштаб за допомогою камери.
FR-8	Бій з ворогами. Персонаж вступає в бій з ворожими NPC, використовуючи зброю та навички.
FR-9	Використання магії у бою. Персонаж застосовує заклинання в бою проти ворогів.
FR-10	Близькій бій з ворогами. Персонаж атакує ворогів у близькому бою.
FR-11	Використання комбо. Гравець може виконувати серії атак для нанесення підвищеної шкоди.
FR-12	Взаємодія з оточуючими об'єктами.

	Гравець може активувати, збирати або використовувати об'єкти довкілля.
FR-13	Підтримка карти та навігації. Користувач може переглядати карту для орієнтації у світі.
FR-14	Відображення об'єктів на карті. Карта показує певні об'єкти (гори, рівнини тощо).
FR-15	Відображення локації персонажа на карті. На карті позначено поточне місце розташування гравця.
FR-16	Перегляд та взаємодія з предметами. Користувач переглядає предмети в інвентарі та використовує їх.
FR-17	Використання предметів. Гравець може використовувати предмети та ресурси.
FR-18	Переміщення предметів у інвентарі. Гравець змінює розташування об'єктів у власному інвентарі.
FR-19	Показ характеристик предметів. При виборі предмета гравець бачить його опис та ефекти.
FR-20	Відображення характеристик персонажа. Користувач переглядає параметри персонажа (сила, витривалість тощо).
FR-21	Показ рівня розвитку персонажа. Відображається поточний рівень та досвід до наступного.
FR-22	Перегляд ігрової валюти. Гравець бачить кількість наявної валюти у HUD або інвентарі.
FR-23	Покращення характеристик. Гравець розподіляє очки розвитку для посилення персонажа.
FR-24	Здобуття предметів. Предмети можуть бути знайдені, здобуті після бою або куплені.
FR-25	Спілкування з NPC. Гравець може почати діалог з неігровими персонажами.
FR-26	Початок діалогу.

	Гравець взаємодіє з NPC, що запускає діалогове вікно.
FR-27	Вибір у діалозі. Гравець обирає дії під час діалогу.
FR-28	Наслідки вибору в діалозі. Вибір у діалозі впливає на квести або ставлення NPC.
FR-29	Поведінка NPC з AI. Штучний інтелект NPC визначає їхню реакцію та дії в різних ситуаціях.
FR-30	Взаємодія з тваринами. Гравець може взаємодіяти з тваринами.
FR-31	Пауза гри. Гравець може зупинити гру в будь-який момент.

	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	FR-8	FR-9	FR-10	FR-11	FR-12	FR-13	FR-14	FR-15	FR-16	FR-17	FR-18	FR-19	FR-20	FR-21	FR-22	FR-23	FR-24	FR-25	FR-26	FR-27	FR-28	FR-29	FR-30	FR-31			
UC-1	+																				+	+												
UC-2		+																																
UC-3		+																																
UC-4																																+		
UC-5																																		
UC-6																																		
UC-7																																	+	
UC-8																																		
UC-9																																		
UC-10																																		
UC-11																																		
UC-12																																		
UC-13																																		
UC-14																																		
UC-15																																		+

Рисунок 2.2 – Матриця трасування вимог

2.3 Розроблення нефункціональних вимог

Нефункціональні вимоги визначають ключові характеристики якості, стабільності та продуктивності RPG гри, які суттєво впливають на користувальський досвід. Для проекту RPG жанру виділені найбільш пріоритетні аспекти продуктивності (Performance).

Гра повинна забезпечувати стабільну частоту кадрів на цільових платформах. Не менше 30 кадрів на секунду, що відповідає стандартам ігрової індустрії.

Час завантаження основних рівнів не повинен перевищувати 10 секунд. Стабільність роботи системи є критичною.

У найнавантаженіших сценах споживання оперативної пам'яті не повинно перевищувати 8 ГБ.

2.4 Аналіз системних вимог

У цьому аналізі розглядаються сучасні ігри, які здобули популярність завдяки своїм механікам, графіці та вимогам до системи. Усі наведені системні вимоги та інша інформація взяті з платформи Steam.

The Witcher 3: Wild Hunt[13] – це одна з найбільш відомих RPG-ігор, що пропонує відкритий світ з глибокою сюжетною лінією, складними персонажами і механіками, випущена в 2015, має такі мінімальні системні вимоги:

SYSTEM REQUIREMENTS	
MINIMUM:	RECOMMENDED:
OS *: 64-bit Windows 7, 64-bit Windows 8 (8.1)	OS: 64-bit Windows 10/11
Processor: Intel CPU Core i5-2500K 3.3GHz / AMD A10-5800K APU (3.8GHz)	Processor: Intel Core i5-7400 / Ryzen 5 1600
Memory: 6 GB RAM	Memory: 8 GB RAM
Graphics: Nvidia GPU GeForce GTX 660 / AMD GPU Radeon HD 7870	Graphics: Nvidia GTX 1070 / Radeon RX 480
DirectX: Version 11	DirectX: Version 12
Storage: 50 GB available space	Storage: 50 GB available space

Рисунок 2.3 – Системні вимоги для The Witcher 3: Wild Hunt.

Cyberpunk 2077[14] – це науково-фантастична RPG з відкритим світом, яка захоплює своїм візуальним стилем та безмежними можливостями для кастомізації персонажа. Станом на 2020 рік має такі системні вимоги:

SYSTEM REQUIREMENTS

MINIMUM:

Requires a 64-bit processor and operating system

OS: 64-bit Windows 10

Processor: Core i7-6700 or Ryzen 5 1600

Memory: 12 GB RAM

Graphics: GeForce GTX 1060 6GB or Radeon RX 580 8GB or Arc A380

DirectX: Version 12

Storage: 70 GB available space

Additional Notes: SSD required. Attention: In this

RECOMMENDED:

Requires a 64-bit processor and operating system

OS: 64-bit Windows 10

Processor: Core i7-12700 or Ryzen 7 7800X3D

Memory: 16 GB RAM

Graphics: GeForce RTX 2060 SUPER or Radeon RX 5700 XT or Arc A770

DirectX: Version 12

Storage: 70 GB available space

Additional Notes: SSD required.

Рисунок 2.4 – Системні вимоги для Cyberpunk 2077

Elden Ring (2022 рік)[15] – це вражаюча RPG від FromSoftware, що поєднує важкі бої, великий відкритий світ та складну історію. Гравець досліджує загадковий світ і бореться з монстрами в складних боях.

SYSTEM REQUIREMENTS

MINIMUM:

Requires a 64-bit processor and operating system

OS: Windows 10

Processor: INTEL CORE I5-8400 or AMD RYZEN 3 3300X

Memory: 12 GB RAM

Graphics: NVIDIA GEFORCE GTX 1060 3 GB or AMD RADEON RX 580 4 GB

DirectX: Version 12

Storage: 60 GB available space

Sound Card: Windows Compatible Audio Device

Additional Notes:

RECOMMENDED:

Requires a 64-bit processor and operating system

OS: Windows 10/11

Processor: INTEL CORE I7-8700K or AMD RYZEN 5 3600X

Memory: 16 GB RAM

Graphics: NVIDIA GEFORCE GTX 1070 8 GB or AMD RADEON RX VEGA 56 8 GB

DirectX: Version 12

Storage: 60 GB available space

Sound Card: Windows Compatible Audio Device

Additional Notes:

Рисунок 2.5 – Системні вимоги для Elden Ring

Можна зробити висновки, що сучасні ігри (як Cyberpunk 2077 та Elden Ring) вимагають потужних систем, зокрема для забезпечення плавної роботи на високих налаштуваннях графіки. Ігри з відкритим світом і складною графікою потребують сучасних графічних карт і багатоядерних процесорів, що здатні підтримувати високу якість візуальних ефектів і стабільну частоту кадрів.

Нижче подано обґрунтовані системні вимоги для RPG гри з поділом на мінімальні й рекомендовані.

Мінімальна конфігурація технічних засобів:

- Операційна система: Windows 10 (64-bit)
- Процесор: Intel Core i3-8100 або AMD Ryzen 3 1200
- Оперативна пам'ять: 8 GB RAM
- Відеокарта: NVIDIA GeForce GTX 750 Ti або AMD Radeon R7 360
- Жорсткий диск: щонайменше 10–15 GB на пристрой
- DirectX: Версія 11
- Інше: Клавіатура та миша, доступ до інтернету не потребується

Рекомендована конфігурація технічних засобів:

- Операційна система: Windows 11 (64-bit)
- Процесор: AMD Ryzen 7 3750H
- Оперативна пам'ять: 16 GB RAM
- Відеокарта: NVIDIA GeForce RTX 2060
- Жорсткий диск: SSD із щонайменше 15 GB на пристрой
- DirectX: Версія 12
- Інше: Клавіатура та миша, доступ до інтернету не потребується

2.5 Аналіз економічних показників програмного забезпечення

Економічний аналіз програмного забезпечення є важливим етапом оцінки доцільності його розробки, а також визначення потенційної комерційної ефективності проєкту. У межах цього підрозділу обрано Planning Poker, оскільки цей метод дозволяє ефективно оцінити завдання, досягти узгодженості в розумінні складності задач, а також швидко адаптувати оцінки при зміні вимог(див. табл. 2.17).

Таблиця 2.17 – Приклад оцінки завдань методом Planning Poker.

Завдання	Оцінка(SP)
Забезпечення функціонування головного меню гри.	5

Керування камерою.	3
Пересування гравця.	5
Можливості персонажа.	8
Бойова система персонажа.	13
Система характеристик персонажа.	8
Навігація та міні-карта.	8
Забезпечення функціонування ігрового інтерфейсу(компоненти HUD).	13
Забезпечення функціонування взаємодії з інвентарем.	13
Квестова система.	13
Забезпечення функціонування штучного інтелекту неігрових персонажів гри.	21
Забезпечення взаємодії з колізією об'єктів.	13
Всього:	123

У рамках реалізації проєкту було оцінено 123 Story Points основних завдань. Проте з огляду на індивідуальне виконання, самостійне вивчення технологій та складність творчих етапів, фактичний обсяг трудозатрат склав приблизно 480 SP, що відповідає 6 місяцям стабільної роботи по 20 годин на тиждень. Така різниця підкреслює важливість врахування додаткових факторів, що виходять за межі технічної оцінки.

Зробимо суб'єктивні припущення про економічні розрахунки, що 1 SP дорівнює 1 людино-години, а 1 людино-година дорівнює 500 грн. Тоді час, необхідний на виконання $480 : 8 \text{ год/день} = 60 \text{ робочих днів}$. Вартість реалізації проєкту однією людиною $480 * 500 = 240\,000 \text{ грн}$. Якщо над розробкою RPG-гри працює одна людина, то на повне виконання запланованого Product Backlog знадобиться приблизно 60 повних робочих днів, а загальна оцінена вартість роботи складе 240 000 грн. Такий аналіз дозволяє точно планувати терміни розробки навіть у випадку індивідуального підходу, що часто трапляється на етапах прототипування або інді-розробки.

2.6 Постановка завдання на розробку програмного забезпечення

Метою дипломного проєкту є забезпечення високої якості ігрового процесу шляхом створення прототипу гри у жанрі RPG із реалізацією реалістичної поведінки персонажів, інтерактивного середовища та інтуїтивно зрозумілого інтерфейсу. У межах розробки планується впровадити основні механіки RPG-гри, включаючи систему управління персонажем, бойову систему, інвентар, діалогову взаємодію з NPC та прості квести.

Завданнями, що підлягають вирішенню в процесі реалізації проєкту, є

- Проектування структури гри та її архітектури малі букви.
- Реалізація системи керування персонажем і його взаємодії з

оточенням.

- Створення бойової системи, інвентаря та діалогової системи.
- Розробка користувальського інтерфейсу.
- Реалізація основних функцій притаманих жанру.
- Впровадження системи AI-механізмів у загальну логіку гри.
- Оптимізація продуктивності на етапах розробки.

У результаті буде створене навчальне програмне забезпечення, що демонструє базову функціональність RPG-гри та може бути основою для подальшого розвитку або вдосконалення.

Висновки до розділу

У даному розділі було сформовано та проаналізовано вимоги до програмного забезпечення, що розробляється – індивідуального проєкту гри жанру RPG. Розділ охоплює як функціональні та нефункціональні аспекти, так і техніко-економічне обґрунтування доцільності реалізації гри.

Було побудовано діаграму варіантів використання (Use Case Diagram), яка візуалізує основні взаємодії користувача з ігровою системою. Вона відображає ключові функціональні можливості, зокрема: запуск гри, взаємодію з інвентарем, бойову систему, переміщення, збереження прогресу

тощо. Для кожного варіанту використання було надано детальний опис у вигляді таблиць, що дозволяє краще зрозуміти логіку поведінки системи.

Далі представлено функціональні вимоги до гри. Вони описують очікувану поведінку системи: від керування персонажем і бойових механік до управління інвентарем та діалогової системи. Також розглянуто нефункціональні вимоги, які визначають обмеження та характеристики якості ПЗ: продуктивність, зручність інтерфейсу, кросплатформеність, вимоги до часу відгуку, безпеки даних та стабільності.

Наступним кроком здійснено аналіз системних вимог. Визначено мінімальні апаратні ресурси для запуску гри (CPU, GPU, RAM), а також програмне середовище (операційна система, необхідні бібліотеки та рушій – наприклад, Unity). Далі сформульовано постановку завдання на розробку ПЗ. Визначено цілі, задачі та обсяг функціональності, що реалізовуватиметься у процесі створення ігрового застосунку.

У цілому, розділ 2 закладає теоретичну, технічну та економічну основу для подальшої реалізації гри. Отримані вимоги та аналіз дозволяють впевнено перейти до етапу проектування та програмування.

За результатами розділу сформовано технічне завдання на розробку програмного забезпечення. Окремий документ за шаблоном «ДП_шаблон_ч1_ТЗ»

3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

Для розробки RPG гри на рушії Unreal Engine 5 (UE5) із використанням штучного інтелекту (AI) було обрано монолітну архітектуру програмного забезпечення. Монолітна архітектура передбачає побудову системи як єдиного цільного додатка, у якому всі компоненти взаємодіють у межах одного процесу. Тобто, уся логіка гри, обробка даних, інтерфейс користувача та модулі штучного інтелекту розміщаються в єдиному програмному блоці, компоненти гри не запускаються окремо, а працюють як єдина система.

Саме така архітектура дозволить витрачати менше часу на налаштування взаємодії між модулями, крім того сам ігровий рушій Unreal надає потужні інструменти, які легко поєднувати в єдиному проєкті із зручним використанням систем поведінки, як Behavior Tree, Blackboard та інші AI-інструменти, які легко інтегруються в монолітну архітектуру.

З недоліків можна виділити обмежену масштабованість, тобто зі збільшенням складності гри будуть виникати труднощі з налагодженням коду, тому що будуть дуже багато різних компонентів, однак монолітна архітектура є оптимальним вибором, оскільки вона дозволяє дозволяє швидко створити функціональний прототип та сконцентруватися на геймплейних та AI-елементах, не витрачаючи зайвий час на мікросервісну або модульну організацію.

Монолітна архітектура, що зображена на рисунку 3.1 є прикладом для проектів, які не вимагають великої гнучкості чи масштабування, як в нашому випадку(Рис. 3.1).

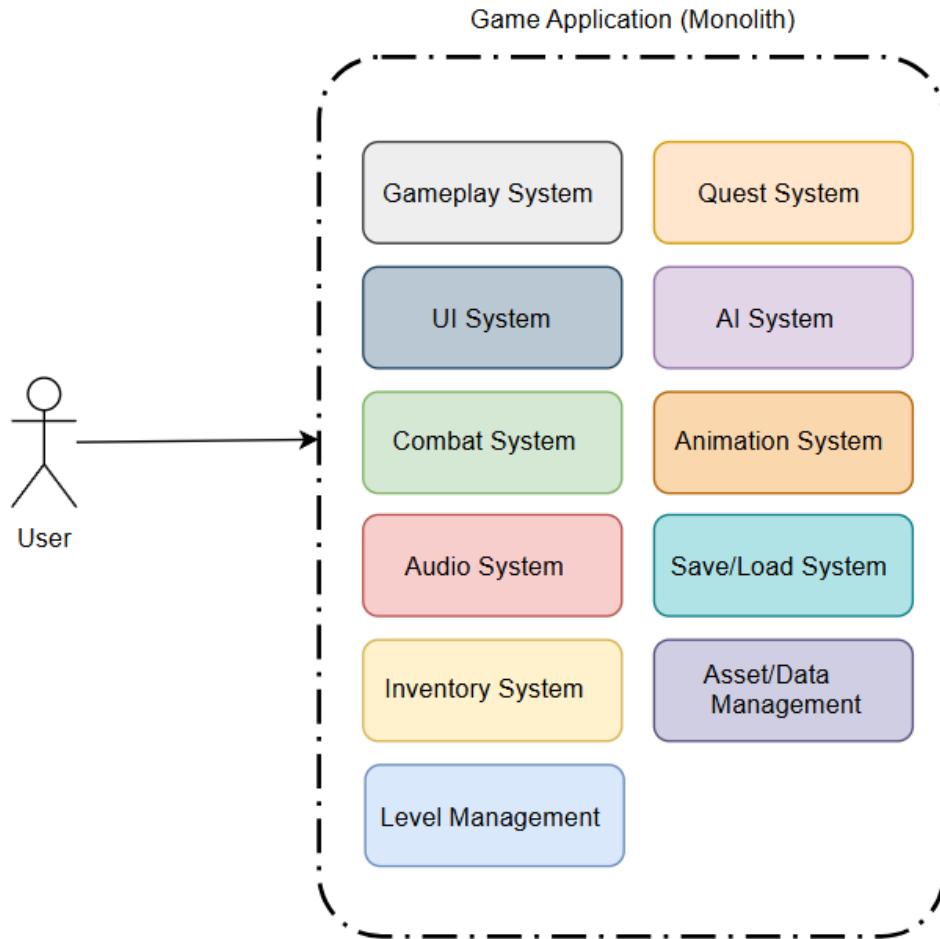


Рисунок 3.1 – Монолітна архітектура гри

Щоб краще деталізувати компоненти архітектури, виконано візуальне представлення архітектури у вигляді діаграми C4 Model, яка дозволяє описати структуру системи на різних рівнях абстракції. Використані архітектурні діаграми для представлення високорівневої архітектури, рівні 1 та 2 моделі C4 Model для представлення архітектури(Рис. 3.2-3.3).



[System Context] Diagram for RPG game

This diagram shows the RPG game as a single system and its interaction with the primary user, the Player.

Рисунок 3.2 – Рівень 1: Контекстна діаграма (System Context Diagram)

RPG-гра є основною системою, яку запускає гравець. Вся логіка виконання, включно з AI-модулями, геймплейною логікою та візуалізацією, відбувається в межах одного цілісного додатка. У навчальному контексті гра функціонує автономно, тобто без зовнішніх API або сторонніх інтеграцій.

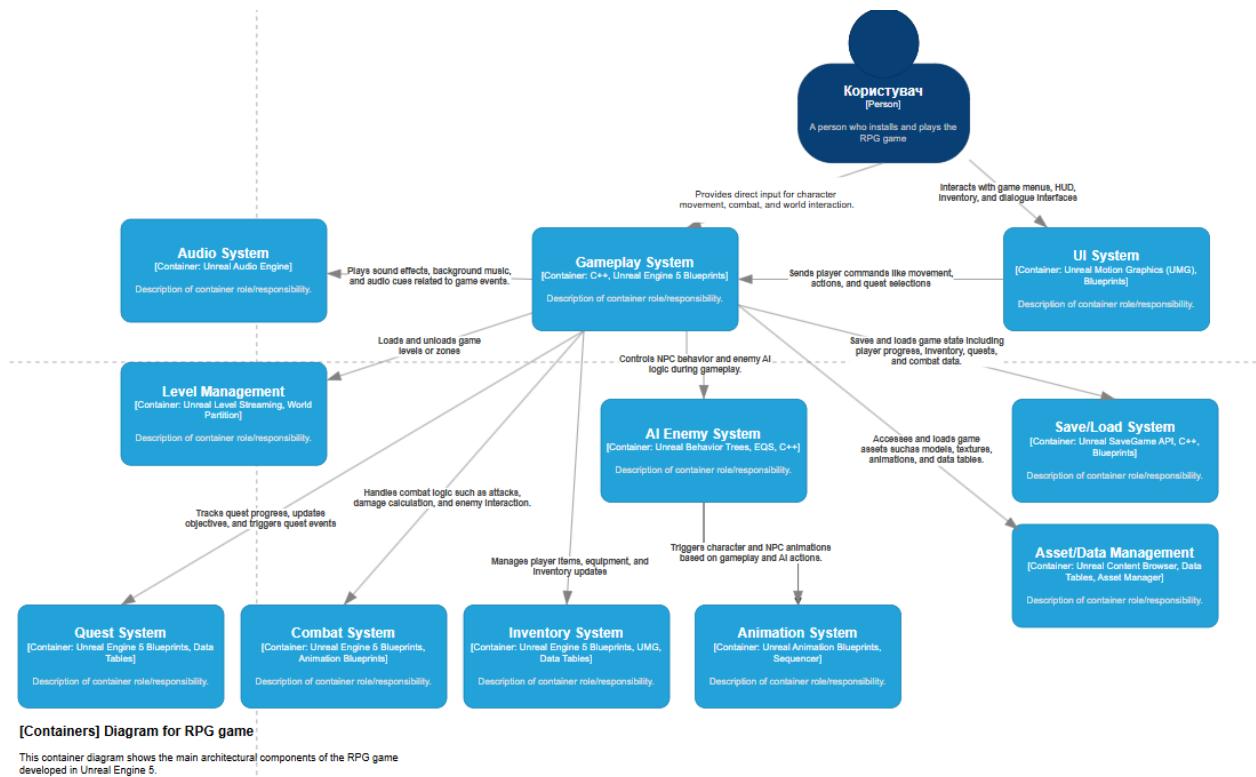


Рисунок 3.3 – Рівень 2: Діаграма контейнерів (Container Diagram).

Цей рівень показує основні «контейнери» всередині системи, які реалізують ключову функціональність:

UI System (Система інтерфейсу користувача), відповідає за відображення всіх елементів інтерфейсу: меню, HUD, інвентар, вікна діалогів, квести, компас, мінікарта тощо. Реалізується через UMG (Unreal Motion Graphics) у Blueprint'ах, іноді з частковим використанням C++ для складної логіки.

Gameplay System (Геймплейна система), основна логіка гри – обробка вводу гравця, поведінка персонажа, керування станами гри, подіями та викликами інших систем (наприклад, бою чи квестів). Реалізується через Blueprints та C++.

Inventory System (Система інвентарю) зберігає дані про речі, екіпірування, взаємодію з предметами, інвентарний інтерфейс, перетягування тощо. Використовуються Blueprints для інтерфейсу та взаємодії, а C++ – для структури предметів і збереження.

Combat System (Бойова система) відповідає за атаки, нанесення шкоди, бойові здібності, статуси та бойові стани. Реалізується за допомогою Blueprint-ів (для візуальної логіки) та C++ (для розрахунків та ефективності).

Quest System (Система квестів) керує квестами: цілі, етапи, умови виконання, нагороди, діалоги. Реалізована за допомогою Blueprint-ів, таблиць даних (DataTables) та, при потребі, C++ структур.

AI System (Система штучного інтелекту) керує поведінкою NPC - патрулювання, атаки, втеча, діалоги. Реалізується через Behavior Tree, Blackboard, AI Controllers та Blueprint-ах та C++ для розширення.

Save/Load System (Система збереження/завантаження) зберігає прогрес гравця, інвентар, статистику, стан світу та квестів. Реалізується через Unreal SaveGame класи, Blueprint або C++.

Audio System (Аудіо-система) відповідає за звукові ефекти, музику, озвучення, звуки середовища. Реалізується через Sound Cues, Sound Waves, а також тригери в Blueprint-ах і компоненти звуку.

Animation System (Анімаційна система) керує анімаціями персонажів: рух, атаки, взаємодія, емоції. Реалізується через Animation Blueprints, стейт-машини, монтажі. C++ може використовуватись для складної логіки або процедурних анімацій.

Level Management (Управління рівнями) відповідає за завантаження та перезавантаження рівнів, стрімінг підрівнів, переміщення між зонами. Реалізовано через Level Streaming та UE5, Blueprint.

Asset/Data Management (Управління ресурсами та даними) керує ігровими активами: текстурами, моделями, звуками, таблицями даних (DataTables), налаштуваннями. Реалізовано через вбудовані системи Unreal Engine: Content Browser, Data Assets, DataTables, Blueprint Config.

Усі контейнери фізично об'єднані в єдину програму (моноліт), що відповідає архітектурі, типовій для UE5. Такий підхід дає змогу зосередитися на функціоналі, не витрачаючи ресурси на складну взаємодію між сервісами.

3.2 Архітектурні рішення та обґрунтування вибору засобів розробки

Зрозуміло, що одним з найважливіших кроків у розробці стояв вибір ігрового рушія. Ігровий рушій – це ключовий елемент програмного забезпечення, який забезпечує технічну основу для створення відеогор. Він об'єднує різноманітні системи, такі як графіка, фізика, звук, анімація, штучний інтелект, мережеві функції та інші компоненти, що дозволяють розробникам зосередитися на геймплейній частині гри. Завдяки уніфікації та стандартизації внутрішньої структури проекту, рушій значно полегшує розробку, тестування та масштабування ігрових продуктів.

Таким чином, розглянемо та проаналізуємо кілька найвідоміших ігрових рушіїв останніх років, звернувши увагу на їхні ключові сильні та слабкі сторони. Це дозволить краще зрозуміти, в яких ситуаціях доцільно використовувати той чи інший рушій, залежно від цілей проекту та технічних вимог.

Unreal Engine 5[16] – один із найпотужніших і найпопулярніших ігрових рушіїв сучасності, який широко використовується для створення ігор класу AAA. Цей рушій побудований на мові програмування C++, тож розробники, які володіють цією мовою, можуть гнучко налаштовувати проект під свої потреби. Водночас, для початківців передбачено використання візуального скриптування через систему Blueprints, що дозволяє створювати ігрову логіку без глибокого знання коду.

Unreal Engine підтримує розгортання ігор на різноманітні платформи, зокрема Xbox One, PlayStation 4, iOS, macOS та Windows. Рушій надає повний набір інструментів для розробки – від 3D-моделювання до створення складних ландшафтів і візуальних ефектів, що робить його привабливим як для новачків, так і для досвідчених професіоналів.

Проте через велику кількість можливостей Unreal Engine може здатися складним для опанування. Вивчення цього рушія вимагає часу та зусиль, але, з огляду на його популярність у професійному середовищі, володіння ним є важливою перевагою для кожного, хто прагне розвиватися в ігровій індустрії.



UNREAL ENGINE

Рисунок 3.4 – Логотип ігрового рушія Unreal Engine

Unity[17] – це один із найпопулярніших та найзручніших у використанні ігрових рушіїв, який часто обирають як початківці, так і професіонали. Він дозволяє створювати як 2D, так і 3D ігри для практично будь-якої платформи – від мобільних пристройів до ПК, Xbox та інших консолей. Завдяки широким можливостям налаштування, Unity можна застосовувати як для розробки простих мобільних ігор, так і для складніших проектів, включно з іграми рівня AAA.

Однією з великих переваг Unity є підтримка інтеграції з програмами для 3D-моделювання, такими як Blender, Maya, Cinema 4D та інші. Це значно полегшує імпорт графіки й моделей у проект. Unity також надає великий набір інструментів – наприклад, систему візуального редагування анімацій, сучасні засоби обробки частинок, а також інтуїтивно зрозуміле середовище для розробки.

Unity має гарний попит та велику кількість доступних ресурсів: документація, відеоуроки, форуми та готові рішення в Asset Store. Попри це, для створення дійсно якісних ігор усе ж знадобляться час, зусилля та певний бюджет. Проте, з правильним підходом Unity здатен забезпечити повноцінний функціонал для реалізації майже будь-якої ідеї.



Рисунок 3.5 – Логотип ігрового рушія Unity

Godot Engine[18] – це сучасний кросплатформенний ігровий рушій, який дозволяє розробляти 2D та 3D ігри з єдиного, зручного інтерфейсу. Він надає вбудований набір усіх необхідних інструментів для створення ігор, що дозволяє розробникам зосередитися на творчому процесі, не витрачаючи час на розробку базової функціональності з нуля. Готові проєкти можна експортувати на різноманітні платформи – Windows, macOS, Linux, Android, iOS, а також у вебформаті (HTML5) буквально в один клік.

Основною мовою програмування в Godot є GDScript – спеціально створена мова, яка схожа на Python і легко засвоюється навіть новачками. Хоча деякі розробники спершу скептично ставляться до використання окремої мови, більшість швидко адаптуються до її логіки. Для тих, хто віддає перевагу іншим мовам, рушій також підтримує C# і C++, а в майбутньому очікується повноцінна реалізація VisualScript – візуального середовища для розробки без написання коду, схожого на Blueprint в Unreal Engine.

Godot приваблює свою відкритістю (open-source), легкістю, невеликими системними вимогами та активною спільнотою. Це ідеальний вибір для інді-розробників, експериментальних проєктів та освітніх цілей, особливо якщо важлива кросплатформеність і простота у використанні.



Рисунок 3.6 – Логотип ігрового рушія Godot

Нижче, на таблиці 3.1 наведено основні переваги та недоліки розглянутих популярних ігрових рушіїв.

Таблиця 3.1 – Ігрові двигуни, недоліки та переваги

Назва	Переваги	Недоліки
UE5	<ol style="list-style-type: none"> 1. Потужна технічна підтримка та регулярні оновлення 2. Постійне розширення функціоналу та поява нових інструментів 3. Великий набір можливостей для 3D-моделювання, анімації та візуальних ефектів 	<ol style="list-style-type: none"> 1. Потрібно ретельно оптимізувати проєкти, щоб уникнути просідань FPS
Unity	<ol style="list-style-type: none"> 1. Гнучка та доступна ліцензійна політика 2. Простий в опануванні та використанні 3. Велика та активна спільнота 4. Величезна бібліотека асsetів, плагінів та готових рішень 	<ol style="list-style-type: none"> 6. Програє конкурентам за продуктивністю в деяких аспектах 7. Обмежений функціонал для високорівневої 3D-графіки 8. Великі розміри фінальних збірок гри

	5. Популярний серед інді-розробників та комерційних студій	
GODOT	1. Компактний, не займає багато місця 2. Повністю безкоштовний з відкритим кодом 3. Гнучка система анімації майже кожного елементу 4. Наявність якісної безкоштовної документації та навчальних матеріалів	1. Обмежені можливості для складної 3D-графіки 2. Відносно невелика спільнота 3. Недостатньо інструментів для реалізації великих комерційних проектів

Порівнявши найпопулярніші сучасні ігрові рушії, а також проаналізувавши їхні сильні й слабкі сторони, я зупинив свій вибір на Unreal Engine. Для реалізації моого проекту цей рушій виявився найбільш придатним, оскільки дає змогу максимально розкрити потенціал майбутньої гри як у технічному, так і в візуальному аспектах.

Однією з головних причин, чому я обрав саме Unreal, є через візуальне програмування, що значно спрощує процес створення прототипів ігрової механіки без ручного кодування. Завдяки інтуїтивно зрозумілому інтерфейсу користувач може створювати логіку гри шляхом з'єднання візуальних вузлів, що значно спрощує й прискорює розробку, особливо на початкових етапах.

У підсумку, Unreal Engine 5 – це потужний інструмент для створення тривимірних проектів з високим рівнем деталізації та гнучким підходом до розробки. Саме тому я вважаю його найкращим вибором для реалізації свого ігрового задуму.

3.3 Конструювання програмного забезпечення

Основна частина розробки гри буде реалізована за допомогою системи візуального програмування, відомої як Blueprints, яка є однією з ключових технологій в Unreal Engine. Щоб краще зрозуміти її можливості та переваги, варто розглянути цю систему більш детально.

Blueprints[19] – це потужний інструмент візуального сценарного програмування, що дозволяє створювати логіку ігрових механік без необхідності писати код мовою C++. Вона побудована на принципі з'єднання вузлів (node-based interface), кожен з яких виконує певну функцію або дію. Розробники мають змогу будувати цілі системи взаємодії, обробки подій, анімації чи логіки штучного інтелекту, використовуючи лише інтуїтивний графічний інтерфейс редактора. Цей підхід значно спрощує процес створення ігрового функціоналу, особливо для невеликих команд або розробників-початківців, які не мають великого досвіду програмування. Крім того, Blueprints дозволяє швидко прототипувати ідеї, миттєво тестувати зміни в редакторі та візуалізувати складні процеси, які в традиційному коді були б важкими для відстеження.

На практиці Blueprints використовується навіть у професійних комерційних проектах: наприклад, студії часто комбінують Blueprints з C++, створюючи основну архітектуру на коді, а дрібні механіки та логіку – на Blueprints

Одним із перших етапів у процесі створення гри стало формування ігрового середовища, а саме – початкового ландшафту, який згодом слугуватиме основою для розміщення об'єктів, архітектури та маршруту пересування персонажа. Робота з ландшафтом здійснювалася безпосередньо в Unreal Engine 5 за допомогою вбудованих інструментів, доступних у режимі редагування під назвою Landscape Mode (див. рис. 3.4).

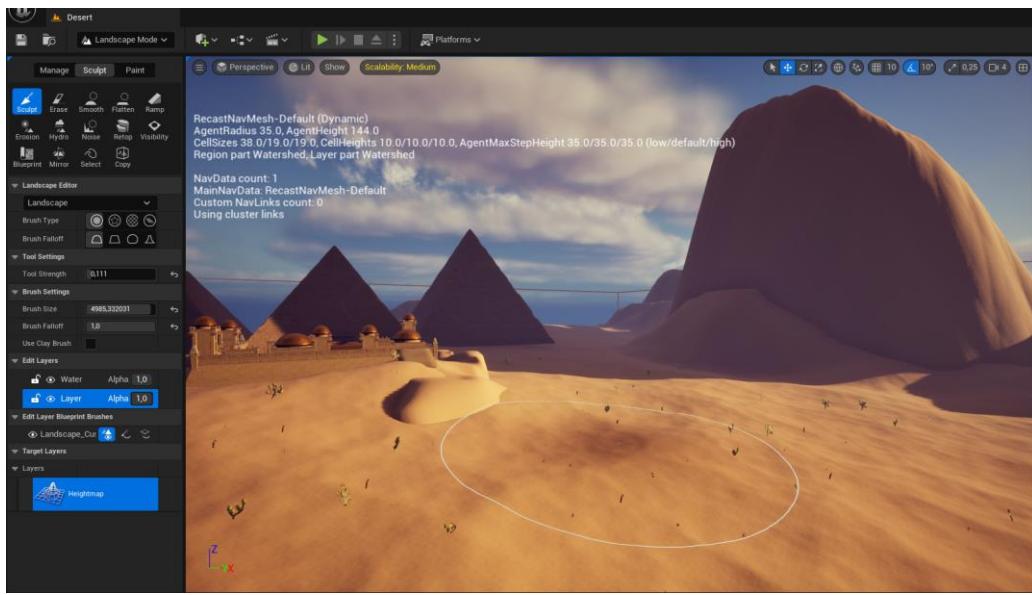


Рисунок 3.7 – Інтерфейс інструмента Landscape Mode в Unreal Engine.

На початковому етапі за допомогою Landscape Mode було створено базову площину, що послугувала «полотном» для подальшого формування рельєфу. Після налаштування параметрів сітки, розміру й роздільної здатності, було здійснено первинну генерацію території. Подальше редагування рельєфу здійснювалося за допомогою інструментів, таких як Sculpt, Flatten, Smooth та Erosion. Особливу увагу було приділено навігації: ландшафт був спроектований так, щоб забезпечити гравцеві зручні шляхи переміщення, уникнути надто крутих схилів або мертвих зон, які можуть спричинити фрустрацію під час гри. Логічне розбиття простору на умовні зони – наприклад, стартову область, зону конфлікту, приховані маршрути тощо – дозволило закласти основу для майбутнього геймдизайну.

Крім топографії, важливим етапом стала розробка матеріалів для візуального оформлення поверхні. Було створено складений матеріал, який включає кілька шарів текстур (за допомогою Layer Blend) – зокрема траву, кам'янистий ґрунт, суху землю тощо. Це надало ландшафту реалістичного вигляду й дозволило автоматично або вручну фарбувати різні частини карти відповідно до їхнього призначення.

Після створення базового ландшафту наступним логічним етапом стало наповнення сцени рослинністю, що відіграє ключову роль у формуванні атмосфери відкритого ігрового світу. Для цього було використано

спеціалізований інструмент Unreal Engine – Foliage Mode, який дозволяє зручно та швидко розміщувати численні об'єкти навколошнього середовища (див. рис. 3.8).

До сцени були додані різноманітні елементи довкілля: дерева, чагарники, трава та окремі декоративні рослини. Більшість моделей було імпортовано з бібліотеки Starter Content, а також із високоякісної колекції Megascans, що дозволяє використовувати фотorealістичні 3D-ресурси. Під час розміщення було налаштовано параметри щільності, масштабування, варіації висоти та випадковість орієнтації об'єктів, що надало природності розміщенню зелених зон

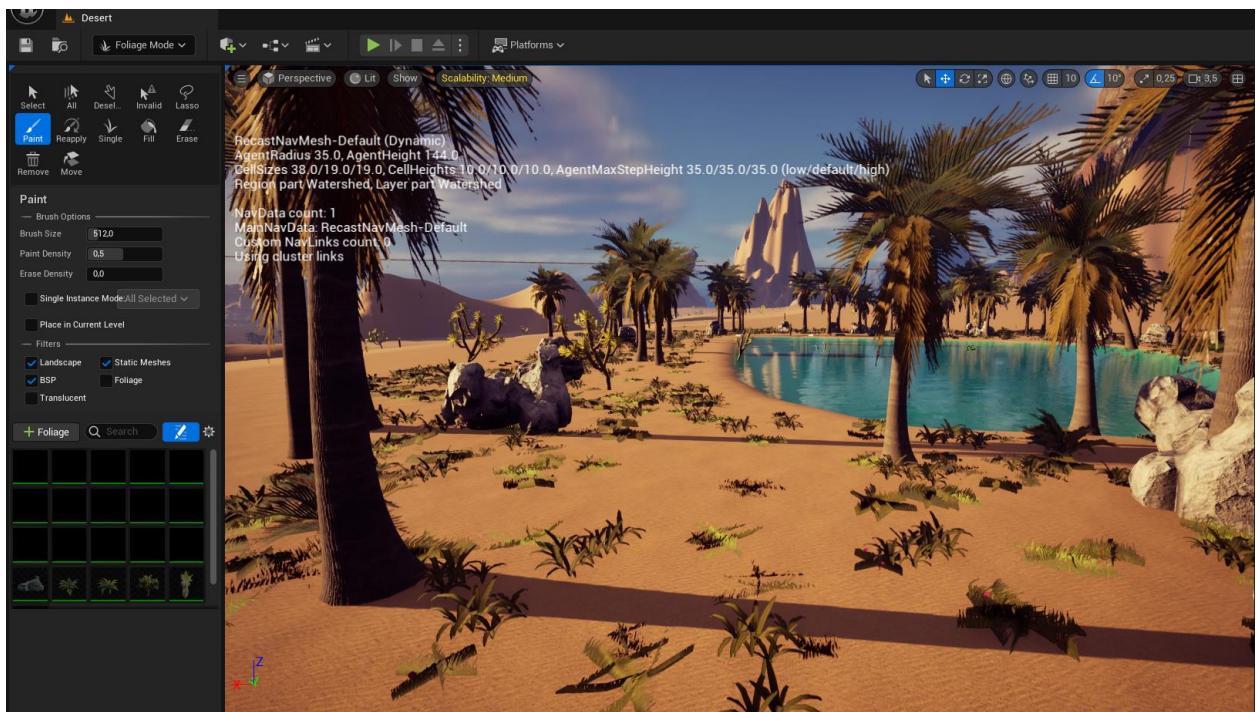


Рисунок 3.8 – Генерація рослинності з налаштуванням випадковості.

Однак, додавання великої кількості дрібних об'єктів значно підвищує навантаження на систему, особливо під час рендерингу великих відкритих територій. Тому особливу увагу було приділено оптимізації. Усі моделі рослинності було попередньо оброблено з урахуванням кількості полігонів. Крім того, для оптимального управління пам'яттю та рендерингом у великих просторах було задіяно систему World Partition – інноваційну функцію Unreal Engine 5, яка автоматично розділяє ігровий світ на сектори та динамічно завантажує лише ті ділянки карти, які перебувають у полі зору гравця.

Після завершення формування ландшафту наступним етапом стало проєктування ігрового оточення, яке має створити атмосферу світу, у якому гравець буде взаємодіяти з елементами навколишнього середовища. Це середовище не лише відіграє декоративну роль, а й безпосередньо впливає на сприйняття, навігацію та загальну геймплейну динаміку. Конструкція ігрового оточення реалізується за допомогою кількох ключових інструментів і систем Unreal Engine 5 а саме: система візуального сценарного програмування, що дозволила створювати логіку взаємодії між об'єктами, контент із Marketplace – Unreal Engine має вбудовану платформу Marketplace, де розробники можуть знайти та використовувати тисячі готових ресурсів: 3D-моделі, текстури, анімації, VFX-ефекти та навіть готові сцени. Частину такого контенту було інтегровано в проект для розширення візуального різноманіття.

Одним із центральних елементів візуального оформлення в Unreal Engine є матеріали. Вони складаються з одного або кількох текстур, які поєднуються за допомогою спеціального редактора матеріалів. Саме матеріал визначає, як виглядатиме об'єкт у грі: його колір, ступінь прозорості, наявність блиску, рельєфність поверхні (через Normal map) та інші параметри (див. рис. 3.9).

Так, наприклад, один і той самий 3D-меш (наприклад, камінь або стіна) може виглядати абсолютно по-різному залежно від того, який матеріал до нього застосовано. Більш того, ті самі матеріали можуть використовуватись не лише на статичних об'єктах, а й для UI-елементів, ефектів частинок, води, скла та інших складних візуальних ефектів.

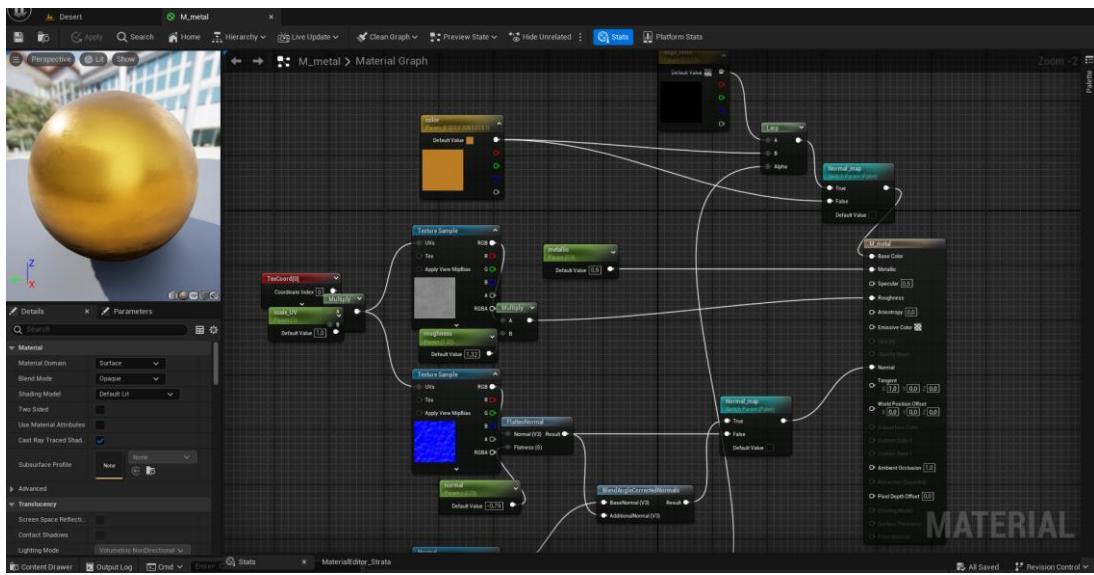


Рисунок 3.9 – Приклад створення матеріалу.

Наступним етапом стало поетапне наповнення сцени різноманітними об'єктами та елементами декору, які формують цілісне ігрове середовище та сприяють зануренню гравця в атмосферу гри. До таких об'єктів належать архітектурні структури, кам'яні утворення, дерев'яні конструкції, реквізит середовища (props), а також допоміжні елементи, що надають сцені деталізації та реалізму.

Зображення прикладів доданих об'єктів до ігрового середовища наведені на рисунку 3.10.

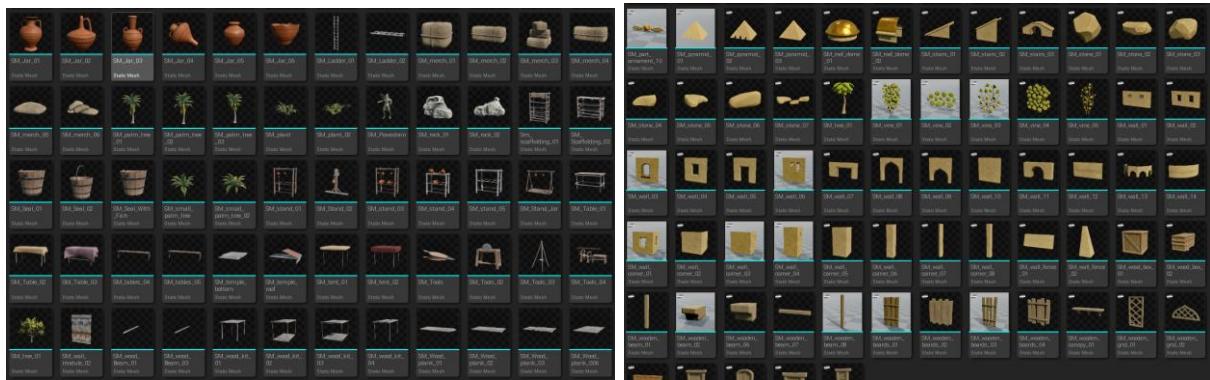


Рисунок 3.10 – Використані об'єкти.

Після завершення формування базового ігрового світу виникла потреба у створенні інтерфейсу користувача (UI), важливої складової будь-якого ігрового проекту, яка забезпечує інтуїтивну взаємодію гравця з ігровими механіками, станом персонажа, інвентарем та іншими елементами.

У Unreal Engine реалізація користувацького інтерфейсу здійснюється за допомогою Unreal Motion Graphics (UMG) - потужної візуальної системи, вбудованої безпосередньо в рушій. UMG дозволяє створювати інтерактивні графічні інтерфейси без необхідності написання великого обсягу коду. Система заснована на Widget Blueprint - спеціальних схемах, які поєднують елементи дизайну з логікою їхньої поведінки.

Для створення графічного інтерфейсу достатньо відкрити Widget Blueprint, за допомогою візуального редактора розмістити необхідні компоненти на полотні, налаштувати їх стиль і поведінку, а також додати відповідну логіку – наприклад, обробник події натискання на кнопку виходу в головне меню чи оновлення тексту при зміні кількості очок досвіду.

У межах розробки даної RPG-гри було створено низку ключових UI-компонентів, реалізованих за допомогою системи UMG.

WB_HUD, основний інтерфейс, що постійно відображається під час гри. Він містить основну інформацію про стан гравця. Відображення поточного рівня, а також прогресу до наступного рівня через індикатор досвіду. Також показує кількість золота або інших ресурсів, які є в розпорядженні гравця. Та відповідає за динамічне спливаюче повідомлення про події (здобуття предмета, досягнення тощо)

WBP_HealthBar, індикатор здоров'я гравця. Віповідає за графічне зображення HP, що оновлюється в реальному часі. Має стилістичне відображення у вигляді сердець.

WBP_Compass – функціональний компас для орієнтації гравця у просторі. Відображає напрямки (N, E, S, W) позначення головних сторін світу. Динамічно оновлюється відповідно до напрямку камери в залежності від кута огляду гравця.

WBP_Minimap, мінікарта, що допомагає гравцю орієнтуватися у середовищі. Відображає навколошнє середовище, показує ландшафт, об'єкти, NPC. Як і компас, обертається залежно від напрямку камери, динамічна адаптація під поточний погляд.

W_BossHealthBar – це спеціальна панель для боїв із босами. Поява під час бою з босом, відображається лише за потреби. Назва боса, індикатор рівня дає гравцеві уявлення про потужність ворога.

WB_QuestGiverDialogue, інтерфейс діалогу з NPC, які дають квести. Діалоги відображаються у текстовій формі з відповідями. Має можливість прийняти завдання з описом. Основна функція - відображення списку поточних завдань, їх статус виконання та відстеження прогресу виконанняожної мети.

WBP_Inventory, система інвентарю яка включає: відображення інвентарю гравця, сортування, використання, екіпірування предметів, підтримка Drag-and-drop, а також відображення описів та характеристик у вигляді додаткової інформація при наведенні.

WBP_Interact, віджет-підказка для взаємодії з об'єктами. Повідомлення типу “Натисніть [E], щоб взаємодіяти” спливає, коли гравець підходить до об'єкта. Динамічно змінюється залежно від типу об'єкта (двері, предмет, NPC).

WB_MainMenu, віджет відповідальний за головне меню гри. Доступ до: нової гри, продовження, арени, налаштувань та виходу, це центральна точка входу в гру.

WB_PauseMenu, меню паузи. Блоکує взаємодію з грою, що забезпечує повну паузу. Має функції продовження гри, вихід у головне меню, доступ до налаштувань.

WB>LoadingScreen, допоміжний віджет для екрану завантаження. Показується під час переходів між рівнями, містить анімацію, підказки або поради.

Для прикладу, щоб створити віджет, який відображає поточний стан здоров'я персонажа, необхідно розробити спеціальний Blueprint-клас у Unreal Engine. Цей клас слугує своєрідним контейнером для логіки UI, що відповідає за візуалізацію та динамічне оновлення індикатора здоров'я. У межах цього Blueprint-класу визначаються ключові змінні, які управляють показниками здоров'я: MaxHP – максимальна кількість очок здоров'я, яку

персонаж може мати в ігровому світі. CurrentHP – актуальний рівень здоров'я персонажа, який змінюється в процесі гри через отримані ушкодження або лікування. Ці дві змінні взаємопов'язані і формують основу для обчислення відсоткового заповнення шкали здоров'я. Саме це відсоткове значення використовується у функціях оновлення прогрес-бару, що забезпечує коректне візуальне відображення стану гравця.

Віджет містить набір спеціалізованих функцій, які слід викликати щоразу, коли змінюється здоров'я персонажа. Ці функції беруть на вхід нові значення CurrentHP і перераховують відсоток заповнення шкали, оновлюючи графічний компонент прогрес-бару. Таким чином, індикатор завжди відповідає реальному стану гри без затримок і помилок.

Для наочності, на рисунках 3.11–3.12 наведено схему реалізації ключових функцій, що оновлюють стан здоров'я в Blueprint (логіка обробки змінних, оновлення віджета) та фінальний вигляд віджета у контексті ігрової сцени – з детальним відображенням прогрес-бара та його стилістичним оформленням.

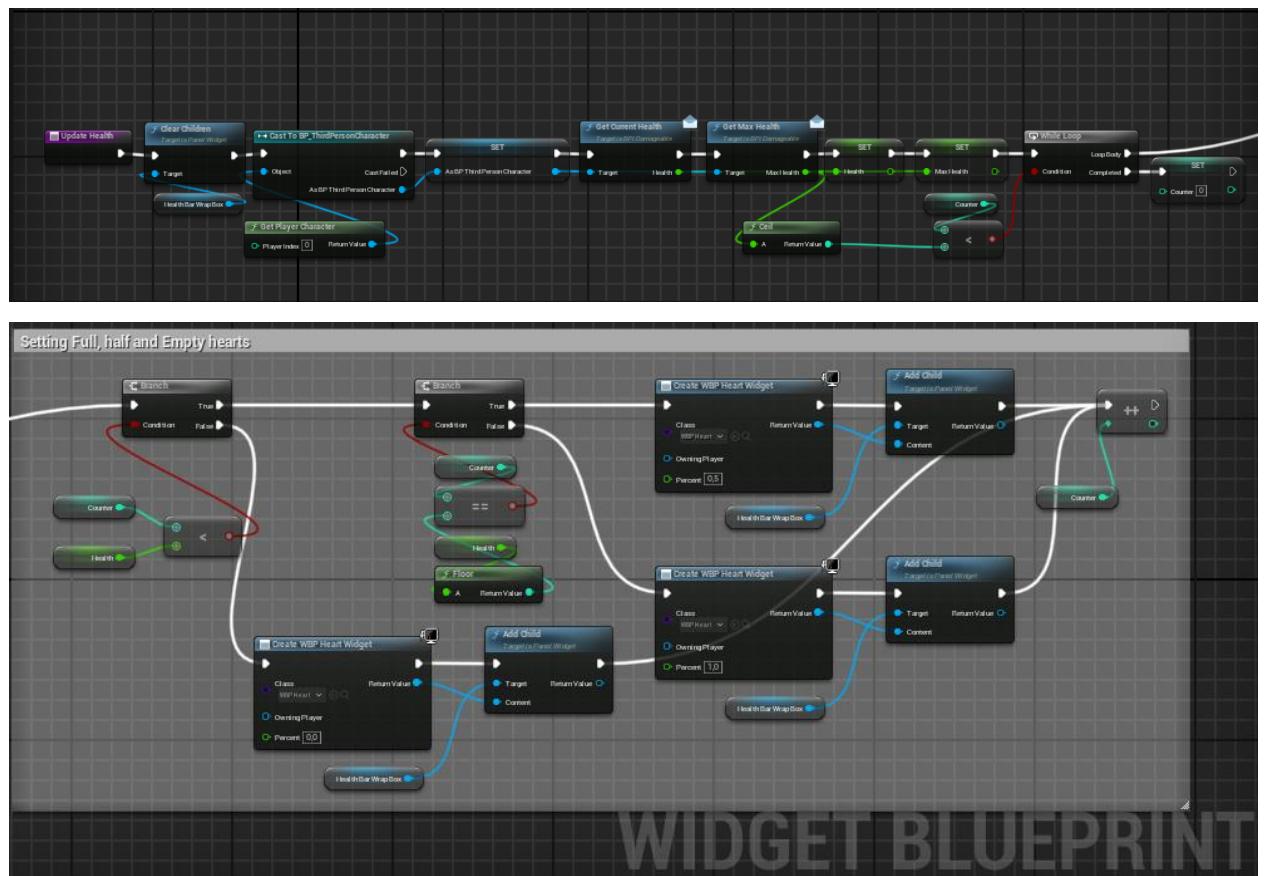


Рисунок 3.11 – Логіка функції віджету здоров'я.



Рисунок 3.12 – Поточний вигляд віджету здоров'я.

Отже, цей віджет виконує роль ключового візуального інструменту, що інформує гравця про поточний стан здоров'я персонажа безпосередньо під час ігрового процесу.

Перед тим, як перейти до реалізації ігрової логіки, необхідно створити повноцінного 3D-персонажа, який стане центральним героєм гри. У Unreal Engine цей процес поєднує в собі кілька ключових аспектів: візуальну складову, систему анімації та інтерактивну логіку, що дозволяє персонажу взаємодіяти з ігровим середовищем.

Першим етапом є створення або імпорт тривимірної моделі персонажа. Для цього зазвичай використовуються професійні графічні редактори, такі як 3Ds Max, Blender, Cinema 4D або Autodesk Maya. Модель може бути створена повністю вручну, що дає максимальний контроль над деталями та стилем, або ж можна скористатися готовими шаблонами і базовими моделями, які потім доопрацьовуються під конкретні потреби проєкту.

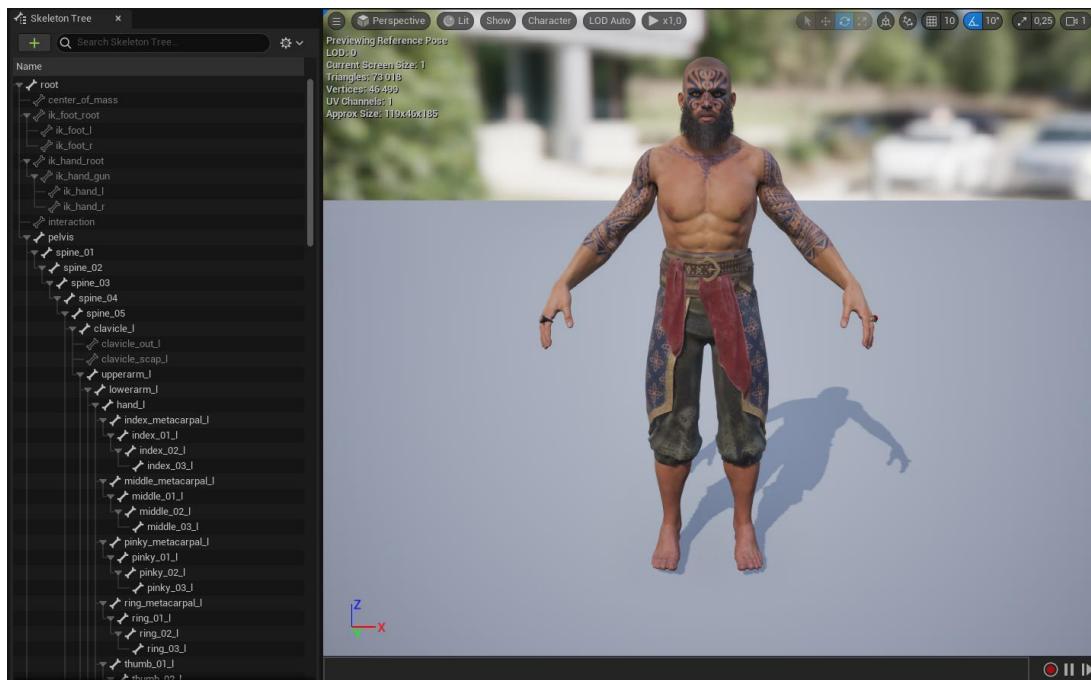


Рисунок 3.13 – Модель головного персонажа.

Такий підхід забезпечує міцну основу для подальшої роботи з персонажем, зокрема, для налаштування скелетної анімації, імплементації системи управління рухами і взаємодією в ігровому світі.

Після завершення моделювання геометрії персонажу на неї накладаються текстири, які формують його зовнішній вигляд – це можуть бути шкіра, одяг, броня та інші деталі.

Наступним кроком є інтеграція скелетної сітки (rigging) до 3D-моделі. Це дає змогу анімувати окремі частини тіла персонажа, забезпечуючи плавність та природність рухів. Як правило, для цього використовують стандартний скелет Epic Skeleton, який оптимізований для Unreal Engine, або розробляють власний кастомний шаблон, адаптований під унікальні потреби конкретного героя. Цей етап є фундаментальним для подальшої роботи з анімацією та ігровою взаємодією персонажа.



Рисунок 3.14 – Скелет для головного персонажа.

Щоб надати персонажу живість і природність, створюється повний набір анімацій, який охоплює різні стани та дії: ходьбу, біг, стрибки, атаки, смерть та інші рухи, характерні для ігрового процесу. Ці анімації можуть бути створені вручну аніматорами або отримані через імпорт із зовнішніх бібліотек, таких як Mixamo – популярний онлайн-сервіс із готовими анімаціями, або

Unreal Marketplace, де можна знайти професійно зроблені анімаційні пакети. Такий підхід значно спрощує і прискорює процес оживлення персонажа, дозволяючи фокусуватися на унікальних особливостях геймплею.

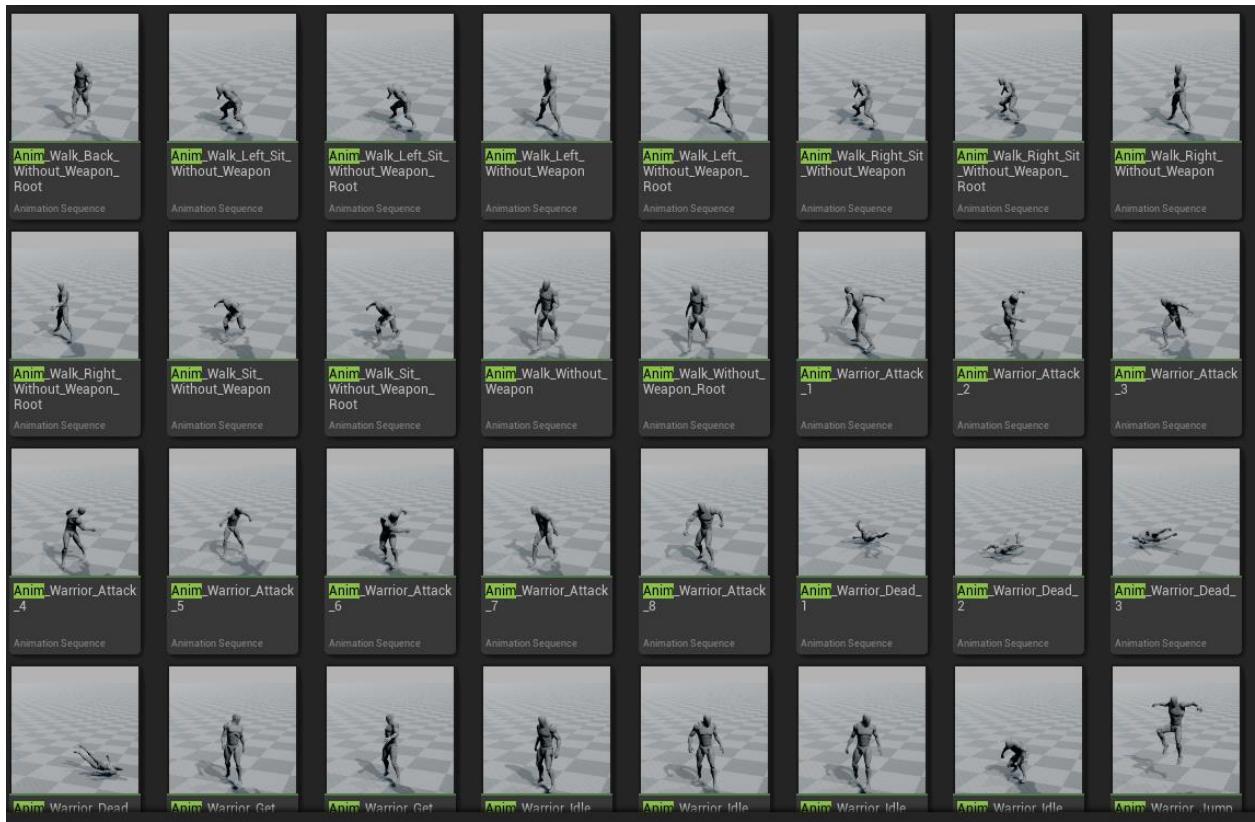


Рисунок 3.15 – Невеликий набір анімацій для демонстрації.

Усі анімації інтегруються в спеціальну систему – Animation Blueprint (ABP), яка відповідає за динамічне управління та плавне переключення між різними станами руху персонажа. Ця система враховує різноманітні параметри, як-от швидкість руху, стан у повітрі, виконання атаки чи інші дії, щоб коректно відтворювати потрібну анімацію в кожен момент часу.

Animation Blueprint тісно пов’язується з основним класом персонажа – BP_Player, що є ключовим blueprint-класом, який управляє поведінкою героя в грі. Завдяки такій архітектурі, анімації стають не просто набором кліпів, а гнучкою системою, що адаптується під поточний стан ігрового процесу, створюючи природний та живий образ персонажа.

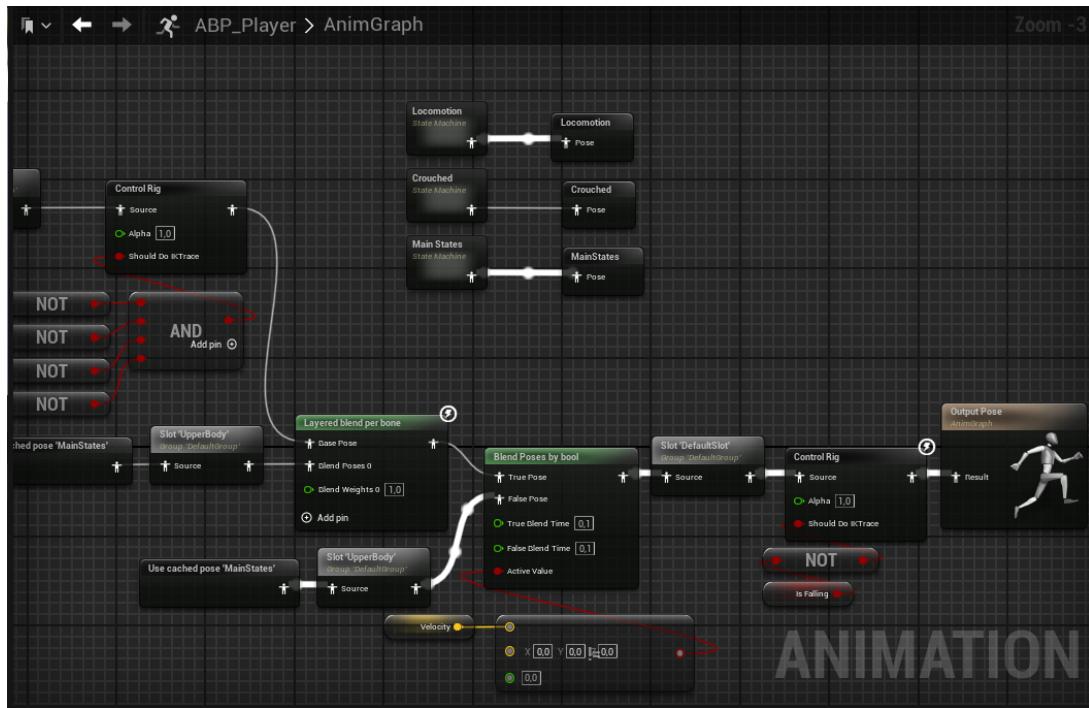


Рисунок 3.16 – Основна логіка Animation Blueprint.

У процесі розробки проекту було прийнято рішення використовувати напівавтоматизований підхід до створення персонажа. Як базу було взято готову 3D-модель, яку адаптували під специфіку гри: налаштували текстури для досягнення бажаного вигляду, підкоригували скелетну структуру для коректної анімації, а потім імпортували цей комплект до Unreal Engine 5.

Паралельно було розроблено унікальні анімації, які інтегрували через Animation Blueprint, що дозволило гнучко керувати переходами між різними станами персонажа. Такий підхід поєднав переваги готових рішень із можливістю індивідуального налаштування, скоротивши час розробки і зберігши при цьому унікальність героя в грі.

Наступним кроком, буде створення загальної логіки для персонажа. Для прикладу на рисунках 3.17-18 зображені логіку конкретних особливостей:

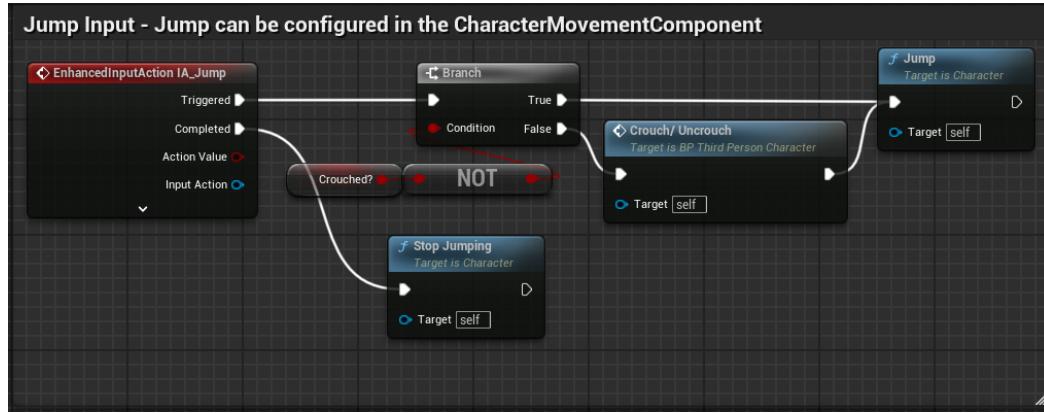


Рисунок 3.17 – Логіка лише одного прижка для персонажа.

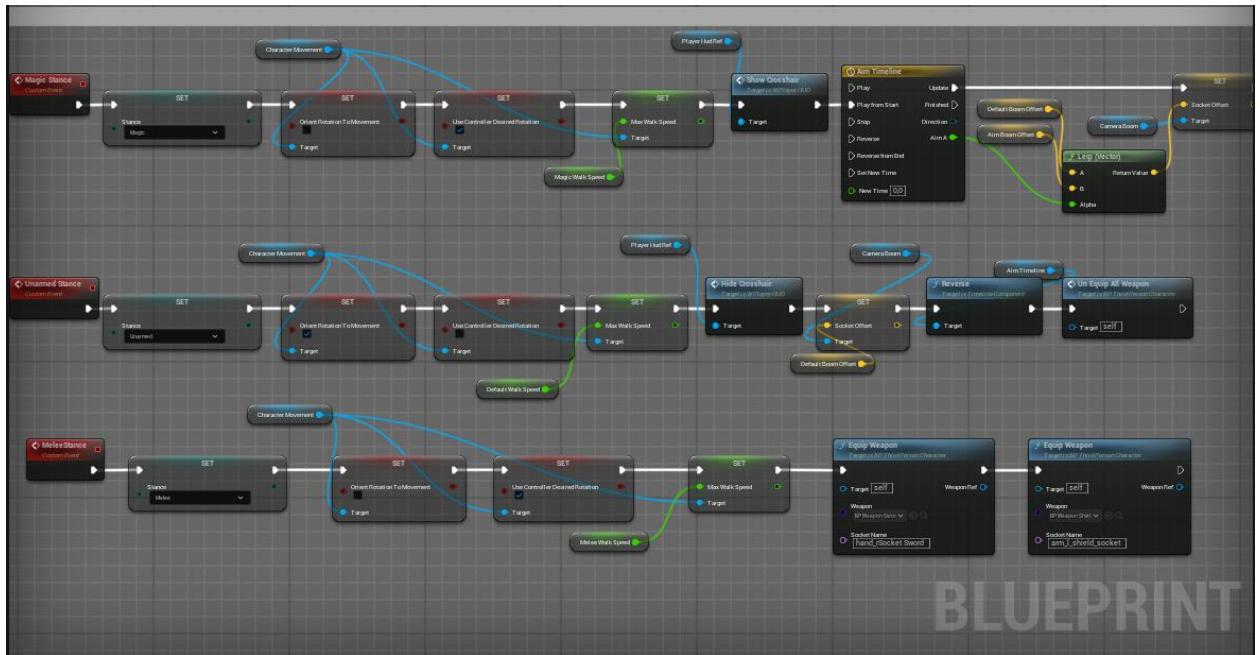


Рисунок 3.18 – Логіка вибору стану для атаки у бою для персонажа.

Описувати кожну функцію окремо немає практичного сенсу, оскільки структура Blueprint логіки персонажа є досить масштабною й охоплює широкий спектр функціональності – від переміщення й анімації до обробки бою, отримання шкоди, смерті та взаємодії з оточенням.

На рисунку 3.19 представлено лише невелику частину загальної структури Blueprint логіки гравця, яка демонструє приклад реалізації конкретних особливостей поведінки персонажа. Решта логіки структуровано за аналогічним принципом із використанням коментарів, функцій та подій, що забезпечують зрозумілу модульну побудову.

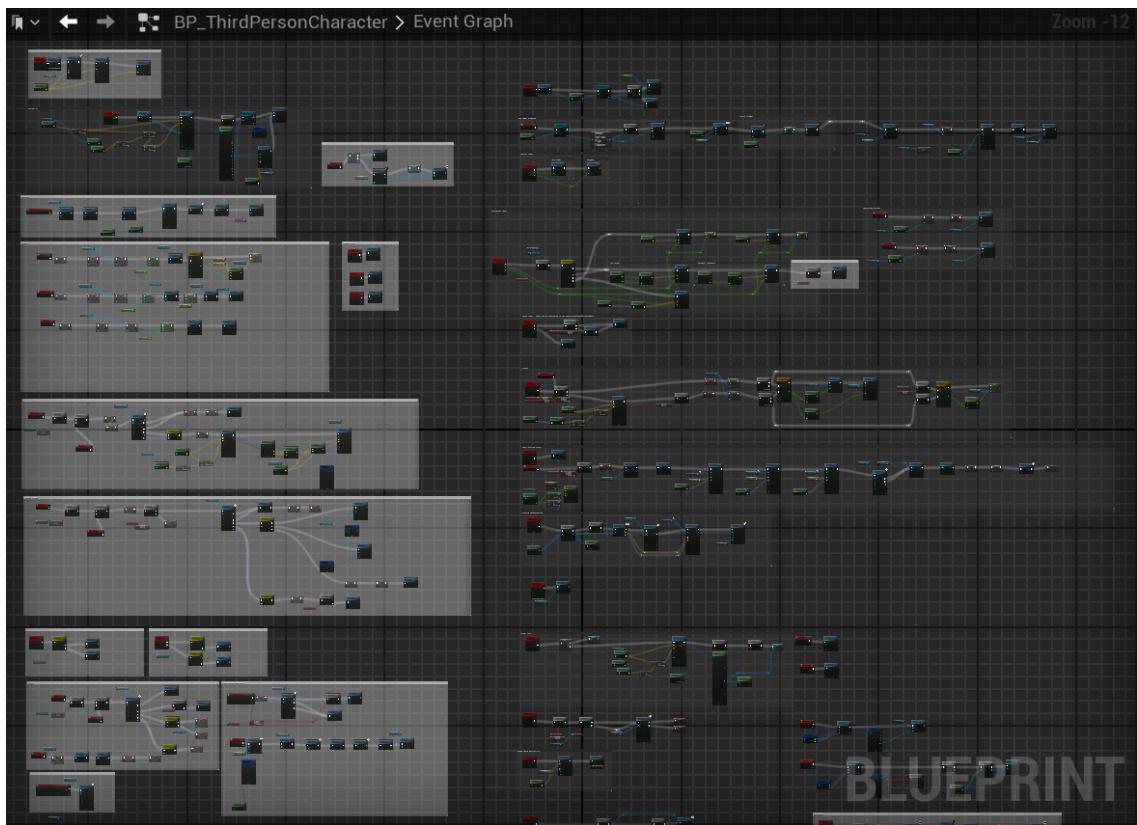


Рисунок 3.19 – Частина загальної структури Blueprint логіки гравця.

У рамках розробки RPG-гри було реалізовано систему штучного інтелекту (ШІ) для ворожих персонажів, яка забезпечує їхню інтерактивну поведінку в ігрому світі.

Система ШІ реалізована за допомогою Behavior Tree (BT) та Blackboard (BB) - стандартних інструментів Unreal Engine, що дозволяють будувати гнучку та масштабовану поведінку NPC.

Blackboard зберігає ключові змінні стану, як-от ціль гравця (TargetActor), поточна точка патрулювання (PatrolPoint) тощо.

Behavior Tree визначає логіку прийняття рішень в залежності від значень цих змінних.

Для кожного типу ворожих NPC було розроблено окреме дерево поведінки, яке враховує особливості бою й поведінкову складність. Наприклад, простий супротивник використовує базову логіку переслідування та прямої атаки гравця. Така структура показана на рисунку 3.32.

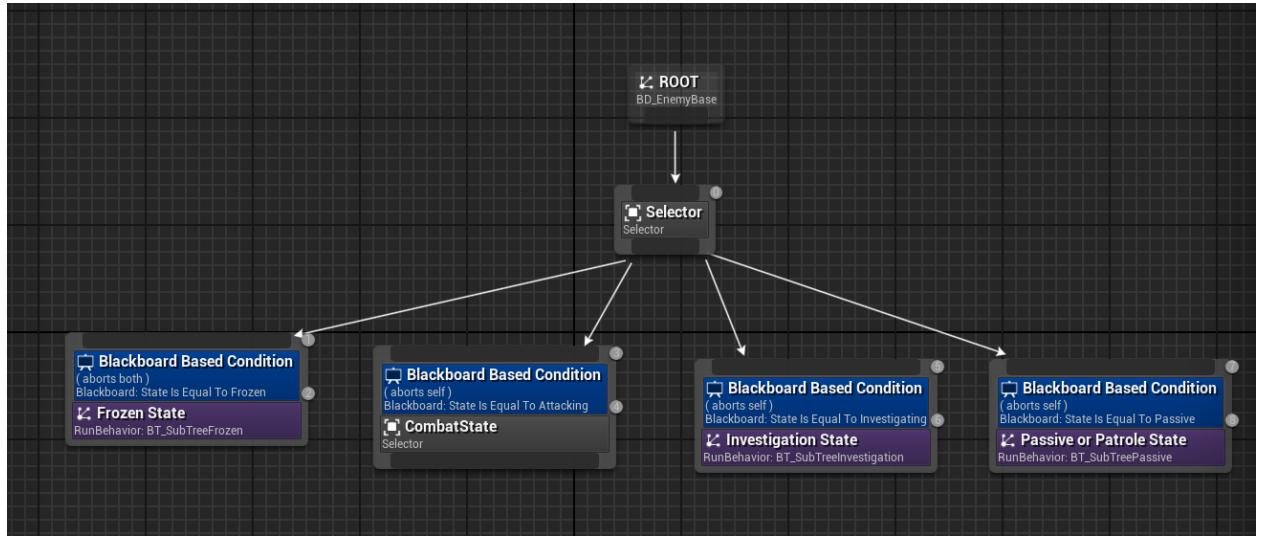


Рисунок 3.20 – Behavior Tree для Base Enemy.

Натомість просунуті вороги мають складніші дерева, що включають: патрулювання в зоні спавну, адаптивну зміну атак залежно від дистанції, динамічне ухилення, умовне відступання при низькому НР, вибір між близьким та дальнім боєм. Подібна розгалужена логіка демонструється на рисунку 3.33, який відображає лише частину можливостей складного Behavior Tree з великою кількістю гілок і декораторів.

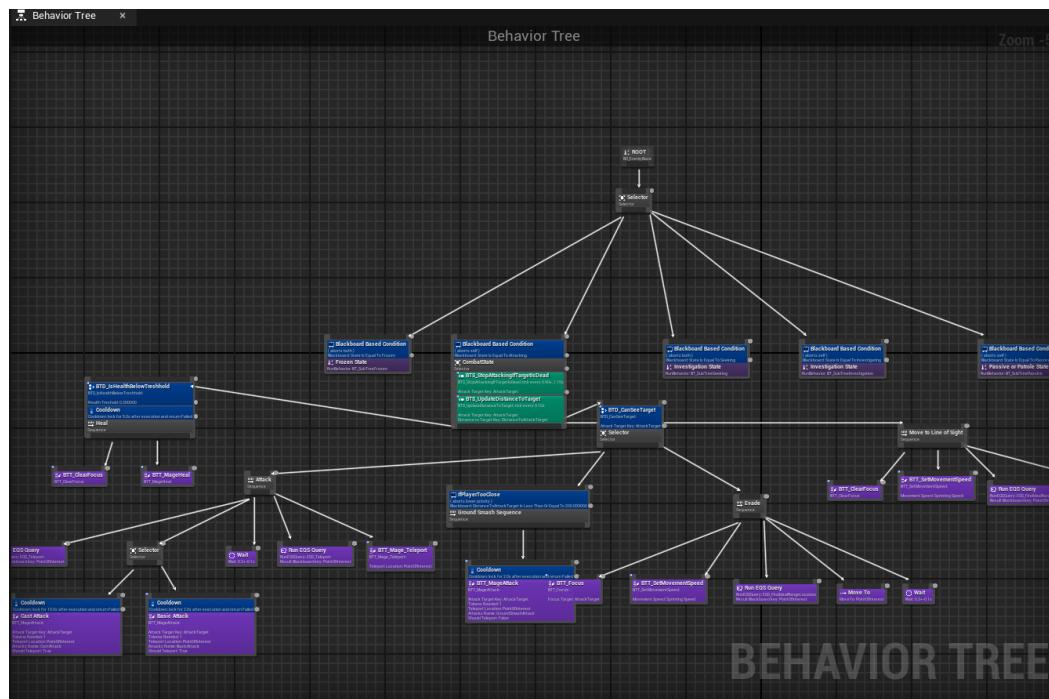


Рисунок 3.21 – Behavior Tree для Range Enemy.

Боси ж мають найскладніші Behavior Tree, що включають кілька фаз бою, спеціальні унікальні здібності, паузи перед виконанням атак та розвинену логіку взаємодії з оточенням.

Behavior Tree для ворогів складаються з численних гілок, що відповідають за різні аспекти поведінки NPC. Для розширення базових можливостей системи були розроблені власні завдання (Behavior Tree Tasks, BTT) і декоратори (Behavior Tree Decorators, BTD), які дозволяють реалізувати унікальні дії та умови для прийняття рішень ворогами. Ці кастомні елементи допомагають гнучко керувати поведінкою, забезпечуючи більш природну та адаптивну реакцію противників у грі.

Для ілюстрації, нижче наведено кілька прикладів кастомних завдань і декораторів, створених для розширення поведінки ворогів у грі:

- BTT_FindRandomPatrol – завдання, що визначає випадкову точку патрулювання.
- BTT_ChasePlayer – завдання, що змінює швидкість руху NPC та спрямовує його до гравця.
- BTT_PerformAttack – запускає анімацію атаки та викликає функцію завдання шкоди.
- BTD_IsPlayerVisible – перевіряє, чи бачить NPC гравця за допомогою системи сприйняття.
- BTD_IsInAttackRange – перевіряє, чи знаходиться гравець в межах досяжності атаки.

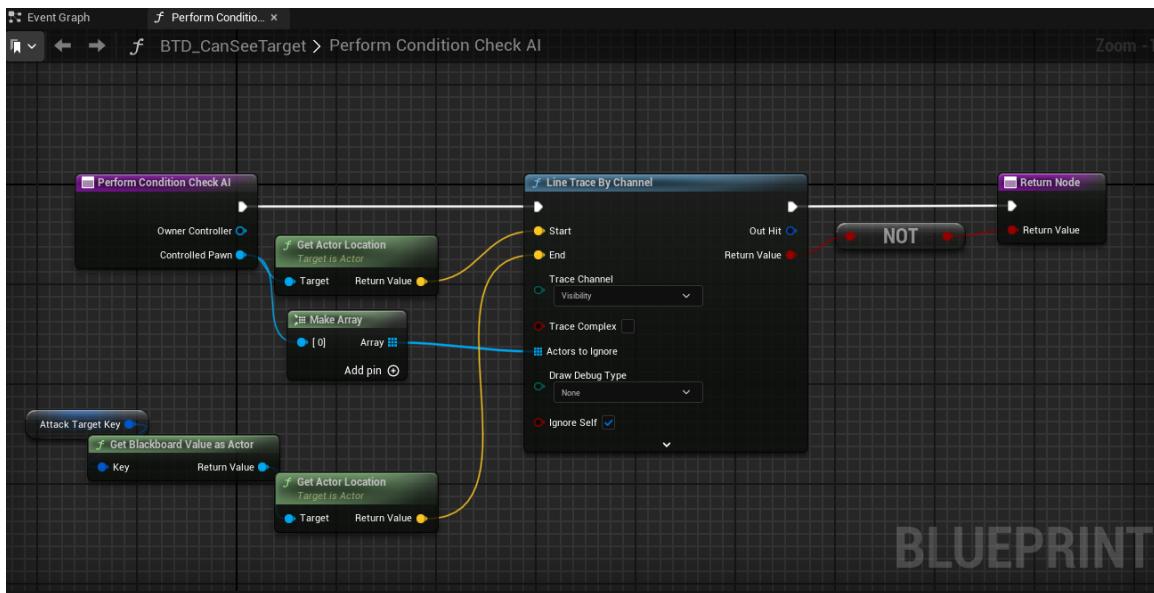


Рисунок 3.21 – Приклад логіки декоратора BTD_CanSeeTarget.

Важливою складовою RPG є бойова система BPC_AttackSystem, яка тісно інтегрована з поведінкою NPC. Ця система відповідає за вибір типу атаки, контроль кулдауну між атаками, синхронізацію анімацій ударів із нанесенням шкоди, а також реалізує інші важливі механізми бою.

Кожен ворог (NPC) має власний AI Controller, який під час запуску гри активує відповідний Behavior Tree та керує логікою штучного інтелекту. Контролер також використовує додаткові компоненти, зокрема BPC_AttackSystem та AIPerception, для виявлення гравця і реакції на події середовища.

Паралельно AI Controller оновлює необхідні змінні в Blackboard, базуючись на отриманих даних про сприйняття та поточний стан NPC, що забезпечує гнучку та адаптивну поведінку ворогів у бою.

Звукове оформлення - одна з ключових складових RPG, що безпосередньо впливає на атмосферу ігрового світу. Під час розробки було реалізовано як фонову музику, так і 3D-звуки, що динамічно взаємодіють із гравцем у режимі реального часу.

Дляожної ігрової області створено або підібрано відповідні музичні треки, які змінюються залежно від розташування гравця. Це реалізовано за допомогою SoundCue та Trigger Volume – спеціальних зон, що активують або деактивують певні аудіодоріжки. Для плавних переходів між музичними треками використовується ефект Fade In/Out. У грі застосовується тривимірне позиціювання звуків (3D Sound Positioning), що забезпечує зміну звучання в залежності від відстані до джерела, його напрямку та наявності перешкод між гравцем і джерелом звуку. Завдяки цьому досягається реалістичний ефект присутності.

Основні технології, використані для звукового оформлення: Sound Attenuation Settings – налаштовують згасання звуку на відстані, Spatialization – визначає напрямок, з якого надходить звук, Occlusion – забезпечує приглушення звуку, якщо між джерелом і гравцем є перешкоди, SoundCue –

дозволяє створювати складні аудіо логіки, включно з комбінаціями звуків та випадковим вибором варіантів відтворення.

Реалізовано окремі аудіоелементи для кроків персонажа на різних поверхнях, звуків ударів і зброї, магічних ефектів, взаємодії з інвентарем та UI, діалогів і реакцій NPC, звуків навколишнього середовища.

Всі звукові ефекти підключені через Blueprint-події або анімовані нотифікатори (AnimNotifies), що забезпечує точну синхронізацію аудіо з візуальними подіями.

3.3.1 Опис структури вхідних/виходних даних

Вхідні дані для цієї гри не є критично важливими, оскільки більшість ігрового процесу зосереджена на внутрішній логіці та механіках, які не залежать від зовнішніх джерел даних.

Вихідні дані гри чітко структуровані, що представляють з себе збереження поточного стану гри, включаючи всі дані, що дозволяють відновити гру на тому ж етапі. Це може включати позицію гравця, статус квестів, інвентар гравця, параметри персонажа: здоров'я, витривалість, валюту, досвід та рівень.

Приклад файлу збереження у форматі .sav:

```
{
    "PlayerID": "player_01",
    "Position": { "X": 120.5, "Y": 45.2, "Z": 10.0 },
    "Health": 8.5,
    "Stamina": 100.0,
    "Level": 12,
    "Experience": 1050,
    "ExperienceToNextLevel": 1200
    "Inventory": [
        { "ItemID": "sword_01", "Quantity": 1 },
        { "ItemID": "banana_01", "Quantity": 3 } ]}
```

}

Логи гри теж входять до вихідних файлів, логування помилки, виключення, повідомлення системи.

Вихідний файл: Лог файл для системи моніторингу, який включає важливі дані для аналізу: ErrorCode, Message, Time.

Тексти програмного коду наведені в окремому документі «Текст програми».

3.4 Аналіз безпеки даних

Це програмне забезпечення працює автономно, без підтримки одночасного доступу кількох користувачів, не здійснює запити до зовнішніх ресурсів та не має підключення до Інтернету. Воно не збирає та не зберігає жодних приватних або персональних даних. З огляду на ці характеристики, впровадження додаткових заходів інформаційної безпеки або проведення глибокого аналізу потенційних загроз не є необхідним.

Висновки до розділу

У цьому розділі було детально описано основні етапи розробки інтерфейсу користувача та ігрової логіки для RPG-проекту на базі Unreal Engine. Розпочато з огляду системи Unreal Motion Graphics (UMG), що використовується для створення та налаштування графічних елементів інтерфейсу, таких як індикатори здоров'я, рівень досвіду, компас, мінікарта, інвентар, діалоги, квести, меню паузи тощо. Було наведено перелік основних віджетів (Widget Blueprints), їх функціональні особливості та способи інтеграції з ігровими даними.

Далі описано процес підготовки 3D-моделі головного персонажа: створення або імпорт моделі, накладання текстур, скелетна анімація та налаштування анімацій через Animation Blueprint. Пояснено основи логіки персонажа, що реалізується у Blueprint-класах, а також механізми взаємодії анімацій з ігровою логікою.

У розділі також розглянуто систему штучного інтелекту для ворогів, реалізовану через Behavior Tree та Blackboard. Пояснено структуру різних рівнів складності поведінки NPC - від простих ворогів з базовою атакою до босів із багатофазними боями і унікальними здібностями. Вказано на розробку власних завдань та декораторів для Behavior Tree, що розширяють стандартний функціонал.

Крім того, висвітлено бойову систему (BPC_AttackSystem), яка забезпечує синхронізацію анімацій атаки та нанесення шкоди, керування кулдаунами та взаємодію з AI Controller ворогів.

Загалом, у розділі представлено комплексний опис основних технічних рішень для формування ігрового світу, інтерфейсу, анімації, логіки персонажів та аудіо, що забезпечує цілісність ігрового досвіду.

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз якості ПЗ

Для оцінки були використані обґрунтовані метрики, що відповідають сучасним практикам геймдизайну, розробки і тестування ПЗ. Основна увага приділена нефункціональним вимогам, серед яких: продуктивність, масштабованість, надійність, а також якість реалізації AI-системи.

У рамках оцінки якості програмного забезпечення було використано набір ключових нефункціональних метрик, результати яких подано нижче. Аналіз проводився у середовищі розробки Unreal Engine 5 із використанням внутрішніх інструментів моніторингу, а також сторонніх програм для визначення продуктивності.

Frame Rate (FPS). Під час тестування у режимі Shipping було зафіксовано середню стабільну частоту кадрів 60 FPS (рис. 4.1). Навіть у сценах з великою кількістю NPC та ефектів FPS не опускався нижче 30.



Рисунок 4.1 – Frame Rate (FPS).

Loading Time, час завантаження основних рівнів становив від 4 до 8 секунд, що повністю відповідає вимогам (не більше 10 с). На рис. 4.2 показано підрахунок зазначеного таймером часу.

No	Level	Loading time(s)
1	Arena	7.8
2	MainMenu	3.4
3	Desert	7.8
4	ThirdPersonMap	6.4
5	Arena	5.7
6	MainMenu	4.8
7	Desert	8
8	ThirdPersonMap	6.6
9	Arena	6.4
10	MainMenu	3.7
11	Desert	7.5
12	ThirdPersonMap	6.7
13	Arena	5.9
14	MainMenu	3.8
15	Desert	7.6
16	ThirdPersonMap	6.1
AverageTime		6.1375

Рисунок 4.2 – Loading Time.

Stability (Надійність). Програма була протестована упродовж 12 годин геймплею без жодного критичного збою чи падіння. Логи UE5 підтверджують відсутність непередбачуваних винятків та стабільну роботу системи пам'яті.

Memory Usage, моніторинг використання оперативної пам'яті показав, що у найнавантаженіших сценах споживання не перевищувало 4 ГБ, що менше допустимого ліміту у 8 ГБ (рис. 4.3). Ефективне розвантаження неактивних ресурсів забезпечується за рахунок використання Streaming Levels.

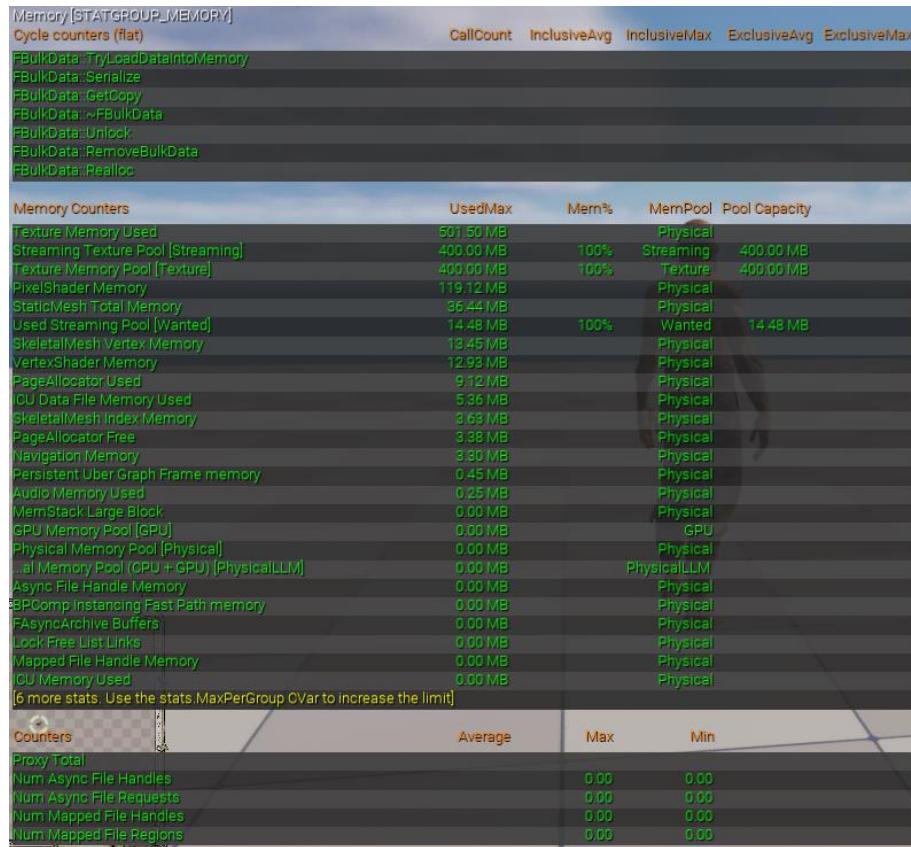


Рисунок 4.3 – Memory Usage.

Таким чином, усі встановлені метрики якості були дотримані, що свідчить про високий рівень оптимізації, стабільності та продуктивності розробленого ПЗ.

4.2 Опис процесів тестування

Тестування виконується згідно документу «Програма та методика тестування».

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 4.1 – 4.12.

Таблиця 4.1 – Тест 1 Перевірка базового руху персонажа

Початковий стан системи	Персонаж знаходиться на рівній поверхні без перешкод.
Вхідні дані	-

Опис проведення тесту	Гравець затискає клавішу W - персонаж починає рухатись вперед. Потім натискає A, щоб перевірити рух ліворуч, S - назад, D - праворуч. Користувач змінює напрям під час руху та тестує переміщення з натиснутими двома кнопками (наприклад, W + D). Спостерігає за відгуком анімації та колізією з оточенням.
Очікуваний результат	Анімація переміщення відтворюється, персонаж рухається згідно з напрямком вводу.
Фактичний результат	Персонаж рухається плавно, керування працює стабільно.

Таблиця 4.2 – Тест 2 Базова атака в близькому бою

Початковий стан системи	Персонаж стоїть перед ворогом на близькій відстані.
Вхідні дані	-
Опис проведення тесту	Гравець підходить до ворога та натискає кнопку атаки. Спостерігає, чи з'являється анімація удару. Потім перевіряє, чи змінюється стан ворога (наприклад, зменшується НР, з'являються ефекти).
Очікуваний результат	Анімація атаки, ворог отримує шкоду, можлива візуалізація ефекту
Фактичний результат	Ворог отримує шкоду, зменшується НР.

Таблиця 4.3 – Тест 3 Взаємодія з NPC

Початковий стан системи	Персонаж знаходиться біля NPC.
Вхідні дані	-

Опис проведення тесту	Гравець підходить до NPC, на екрані з'являється підказка “Натисніть E для взаємодії”. Натискає E - відкривається діалогове вікно. Гравець читає репліки, перемикає їх натисканням клавіші або кнопки на екрані.
Очікуваний результат	Відкривається діалогове вікно з репліками NPC.
Фактичний результат	Діалог відкрився, тексти відображаються.

Таблиця 4.4 – Тест 4 Отримання квесту

Початковий стан системи	Активна розмова з NPC, який видає квест.
Вхідні дані	-
Опис проведення тесту	Гравець підходить до NPC, розпочинає діалог, читає опис квесту та натискає “Прийняти”. Потім перевіряє, чи додано новий квест.
Очікуваний результат	Квест з'являється у віджеті «Active Quests».
Фактичний результат	Квест збережено та відображається в UI.

Таблиця 4.5 – Тест 5 Збереження гри

Початковий стан системи	Персонаж має прогрес - здобутий лут, виконані дії.
Вхідні дані	-
Опис проведення тесту	Гравець відкриває меню паузи, обирає “Зберегти гру”. Потім виходить в головне меню, натискає “Завантажити”.
Очікуваний результат	Прогрес збережено, гра завантажується з того ж місця.

Фактичний результат	
---------------------	--

Таблиця 4.6 – Тест 6 Використання магічної атаки

Початковий стан системи	Персонаж має стан використання магії та ворог попереду.
Вхідні дані	-
Опис проведення тесту	Гравець натискає клавішу, щоб активувати магію. Спостерігає за ефектами та шкодою.
Очікуваний результат	Магія активується, ворог пошкоджений.
Фактичний результат	Збігається з очікуваним

Таблиця 4.6 – Тест 6 Поведінка AI Enemy при критичному НР

Початковий стан системи	Обрати для перевірки Range Enemy, що патрулює.
Вхідні дані	-
Опис проведення тесту	Гравець провокує бій, зменшуючи НР ворогу до критичного.
Очікуваний результат	Ворог тікає за перешкоду, зцілює себе та повертається до бою чи патруля.
Фактичний результат	Ворог утік, зцілив себе, гравець не у полі зору, повернувся до патруля.

Таблиця 4.7 – Тест 7 Підтримка союзника у бою

Початковий стан системи	Гравець має союзника. Поблизу вороги.
-------------------------	---------------------------------------

Вхідні дані	-
Опис проведення тесту	Гравець починає бій. Упевнеться, що союзник бачить ворогу, Спостерігати за діями союзника.
Очікуваний результат	Союзник вступає в бій. Система реагує коректно на зміну НЗ ворога та союзника.
Фактичний результат	Союзник допомагає у бою.

Таблиця 4.8 – Тест 8 Перевірка застосування параметрів

Початковий стан системи	Гра запущена, гравець перебуває в головному меню.
Вхідні дані	-
Опис проведення тесту	Гравець відкриває меню налаштувань, змінює графічну якість або роздільну здатність. Потім повертається в гру або перезапускає її, щоб перевірити, чи збережено зміни.
Очікуваний результат	Зміни застосовано правильно, нові налаштування залишаються після перезапуску, інтерфейс адаптується під нову роздільну здатність.
Фактичний результат	Налаштування збережені, графіка змінена, після перезапуску всі зміни залишились.

Таблиця 4.9 – Тест 9 Перевірка механіки смерті персонажа та завершення гри

Початковий стан системи	Персонаж має низький рівень здоров'я, поруч активний ворог.
Вхідні дані	-
Опис проведення тесту	Гравець навмисно потрапляє під атаки ворога або під час бою ігнорує захист, щоб перевірити втрату НР. Коли

	шкала здоров'я досягає нуля, гравець спостерігає, чи спрацьовує сценарій поразки.
Очікуваний результат	НР падає до 0, гра завершується, з'являється екран поразки або відповідний віджет.
Фактичний результат	Після зниження НР до 0 з'являється екран затухання. Гравець не може більше керувати персонажем.

Таблиця 4.9 – Тест 9 Перевірка механіки смерті персонажа та завершення гри

Початковий стан системи	Персонаж має низький рівень здоров'я, поруч активний ворог.
Вхідні дані	-
Опис проведення тесту	Гравець навмисно потрапляє під атаки ворога або під час бою ігнорує захист, щоб перевірити втрату НР. Коли шкала здоров'я досягає нуля, гравець спостерігає, чи спрацьовує сценарій поразки.
Очікуваний результат	НР падає до 0, гра завершується, з'являється екран поразки або відповідний віджет.
Фактичний результат	Після зниження НР до 0 з'являється екран затухання. Гравець не може більше керувати персонажем.

Таблиця 4.10 – Тест 10 Перевірка навігації меню гри

Початковий стан системи	Гра запущена, гравець в головному меню.
Вхідні дані	-
Опис проведення тесту	Гравець послідовно переходить між пунктами меню: «Нова гра», «Налаштування», «Завантажити гру», «Вийти з гри». Для кожного пункту клікає мишкою. У меню налаштувань змінює кілька параметрів, повертається назад.

Очікуваний результат	Всі пункти меню відкриваються коректно, налаштування застосовуються і зберігаються, при виході з меню повертається в головне меню.
Фактичний результат	Всі пункти меню працюють без помилок, зміни в налаштуваннях застосовуються і зберігаються, навігація плавна.

Таблиця 4.11 – Тест 11 Перевірка взаємодії з об’єктами світу

Початковий стан системи	Гравець у відкритому світі біля декількох об’єктів
Вхідні дані	-
Опис проведення тесту	Гравець підходить до кожного об’єкта, натискає кнопку взаємодії. Перевіряє: чи отримується предмет, відкриваються чи закриваються двері, вмикається чи вимикається світло.
Очікуваний результат	Взаємодія з об’єктами проходить коректно: предмети отримано, двері реагують, лампи змінюють стан.
Фактичний результат	Взаємодія працює належним чином, без зависань чи помилок.

Таблиця 4.12 – Тест 12 Використання предметів інвентаря

Початковий стан системи	В інвентарі є кілька предметів: меч, щит, їстівні предмети.
Вхідні дані	-
Опис проведення тесту	Гравець відкриває інвентар, вибирає з’їсти їстівний предмет. Потім обирає меч та щит і екіпірує їх.
Очікуваний результат	Їстівні предмети відновлюють відповідні ресурси, меч та щит екіповані та готові до бою.

Фактичний результат	Усі предмети працюють як очікується, ефекти від їх використання видно одразу.
---------------------	---

4.3 Опис контрольного прикладу

Контрольний приклад демонструє ключовий функціонал штучного інтелекту у грі - механіку бою з ворогом із застосуванням штучного інтелекту, а також взаємодію з інвентарем та системою квестів. Приклад містить опис основних сценаріїв, можливих розгалужень та особливостей поведінки гравця і NPC.

Melee Enemy(рис.4.4) – це базовий супротивник ближнього бою. Ворог патрулює між заданими точками, використовуючи систему NavMesh для переміщення.



Рисунок 4.4 – Поведінка Melee Enemy з базовим деревом рішень та використанням NavMesh/NavLink

Під час патрулювання він може виявити гравця або в полі зору, або почувши звук. Якщо гравець помічений, ворог переходить у бойовий режим: починає переслідування, обходячи перешкоди за допомогою NavMesh і NavLink. Коли підходить достатньо близько, виконує атачу ближнього бою з відповідною анімацією та врахуванням таймінгів ударів.

Range Enemy – це ворог, що спеціалізується на дальньому бою(рис.4.5).



Рисунок 4.5 – Поведінка Range Enemy (дальній бій).

Він діє обережно та стратегічно. У звичайному режимі патрулює заданий маршрут і змінює напрямок, якщо стикається з перешкодами. Як тільки помічає гравця, починає стріляти з безпечної дистанції, намагаючись не підпустити до себе ворога.

Коли здоров'я падає нижче певного рівня, ворог вмикає режим втечі – починає відступати, уникуючи бою й використовуючи навігацію для обходу перешкод. Якщо втекти не вдається, Range Enemy може або спробувати знову атакувати, або, якщо гравець зупинив переслідування, повертається до патрулювання.

Mage Enemy(рис4.6) – це складніший і більш тактичний ворог. Його поведінка залежить від ситуації: він може вирішити атакувати, захищатися або відступити. Mage не просто стріляє з місця - він аналізує поле бою.



Рисунок 4.6 – Поведінка Mage Enemy (AI-мага).

Ворог використовує телепортацію, щоб раптово змінити позицію наприклад, уникнути атаки або зайняти вигідну точку для наступного закляття. У бою застосовує різні магічні атаки: від прямих влучань по гравцю до АОЕ-ударів по зоні. Якщо його здоров'я падає, може зцілити себе.

Mage Enemy не бореться бездумно. Якщо бачить, що сили не на його боці, може відступити або тимчасово зникнути з поля бою. Така поведінка створює відчуття, що гравець має справу з розумним супротивником, який не просто б'є навмання, а діє стратегічно.

Цей ворог використовує складне дерево поведінки та кастомну логіку, що робить кожен бій з ним непередбачуваним.

Щоб уникнути хаосу в бою, реалізована система токенів, яка обмежує кількість ворогів, що можуть одночасно атакувати гравця (рис 4.7). Перед тим як атакувати, кожен NPC перевіряє, чи є вільний токен, своєрідний «дозвіл» на атаку. Якщо токенів немає, ворог не кидається в бій, а чекає своєї черги або рухається по периметру, ніби шукаючи момент для удару.

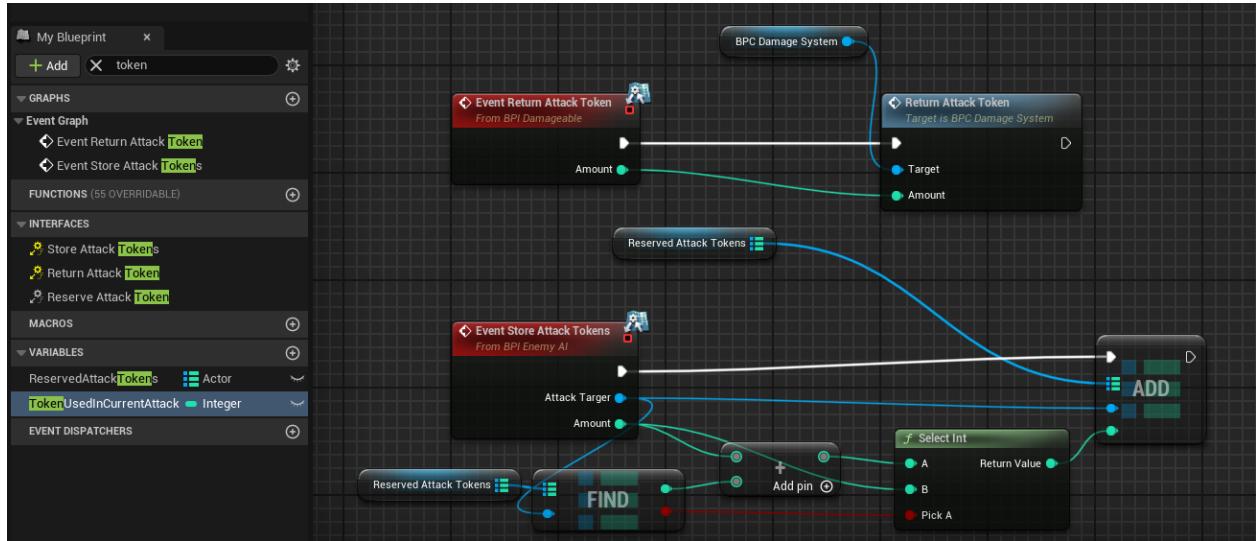


Рисунок 4.7 – Система токенів для обмеження NPC у бою.

Коли один із супротивників завершує атаку або помирає, його токен звільняється, і його може взяти інший NPC. Такий підхід дозволяє зберігати контроль над боєм: вороги не навалюються гуртом, а діють злагоджено, створюючи ілюзію тактичної поведінки.

Використання предметів інвентаря для тактичної переваги(рис. 4.8)

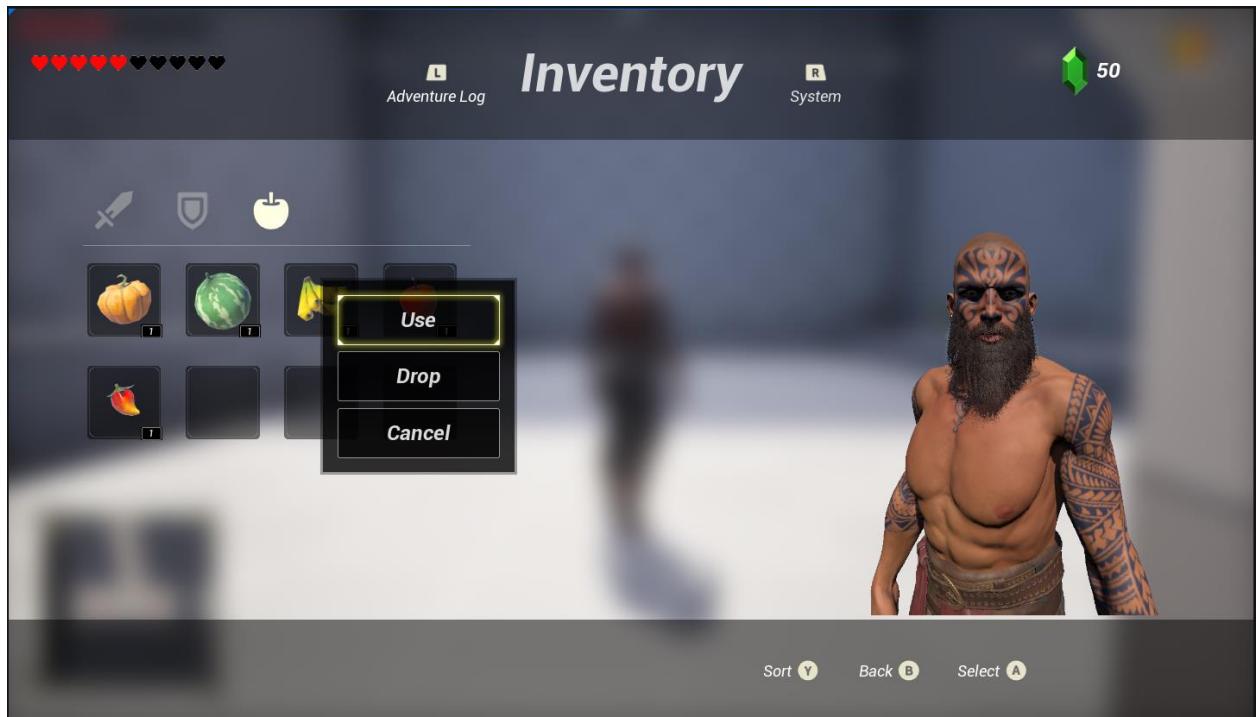


Рисунок 4.8 – Використання предметів інвентаря.

Інвентар це не просто місце для зберігання предметів, а повноцінна частина бойової системи. У будь-який момент гравець може використати

предмет, змінити зброю. Це дає тактичну перевагу прямо в бою: можна швидко зцілитися, або нанести шкоду ворогам.

Усі ці дії пов'язані з інтерфейсом: вони анімовані, мають затримку застосування та можуть бути обмежені умовами, наприклад, кулдауном або кількістю використань. Це додає реалізму та глибини.

Таким чином, інвентар, бойова система та UI працюють разом, створюючи живий, гнучкий і захопливий геймплей, де гравець постійно приймає рішення й адаптується до ситуації на полі бою.

Висновки до розділу

У цьому розділі проведено аналіз якості програмного забезпечення, що включає оцінку за ключовими нефункціональними метриками, такими як стабільність, продуктивність, використання пам'яті, швидкість реакції AI та час завантаження.

Для перевірки відповідності цим метрикам було виконано ручне тестування, описане у тестових сценаріях. Всі ключові компоненти, зокрема навігація меню, рух персонажа, взаємодія з об'єктами та використання інвентаря, працюють відповідно до очікувань, без збоїв або помилок.

Контрольний приклад ілюструє реалізацію складного AI в рамках бойової системи. Продемонстровано поведінку ворогів, всі вони мають індивідуальні патерни дій, реакції на дії гравця, механіку ухилення, патрулювання та зміну тактики в залежності від ситуації.

Таким чином, усі аспекти якості, тестування та реалізація III поєднані, забезпечуючи стабільну та цікаву ігрову взаємодію.

5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розгортання програмного забезпечення

Основним інструментом розгортання є система пакування (Packaging System) Unreal Engine 5, яка дозволяє компілювати проект у виконувані файли для різних платформ: Windows, macOS, консолі, мобільні пристрої тощо. Розгортання гри складається з кількох основних кроків.

Перевірка цілісності та готовності ресурсів та встановлення необхідних параметрів збірки, вибір цільової платформи, конфігурація збірки: Development, Shipping зображені на рисунку 5.1.

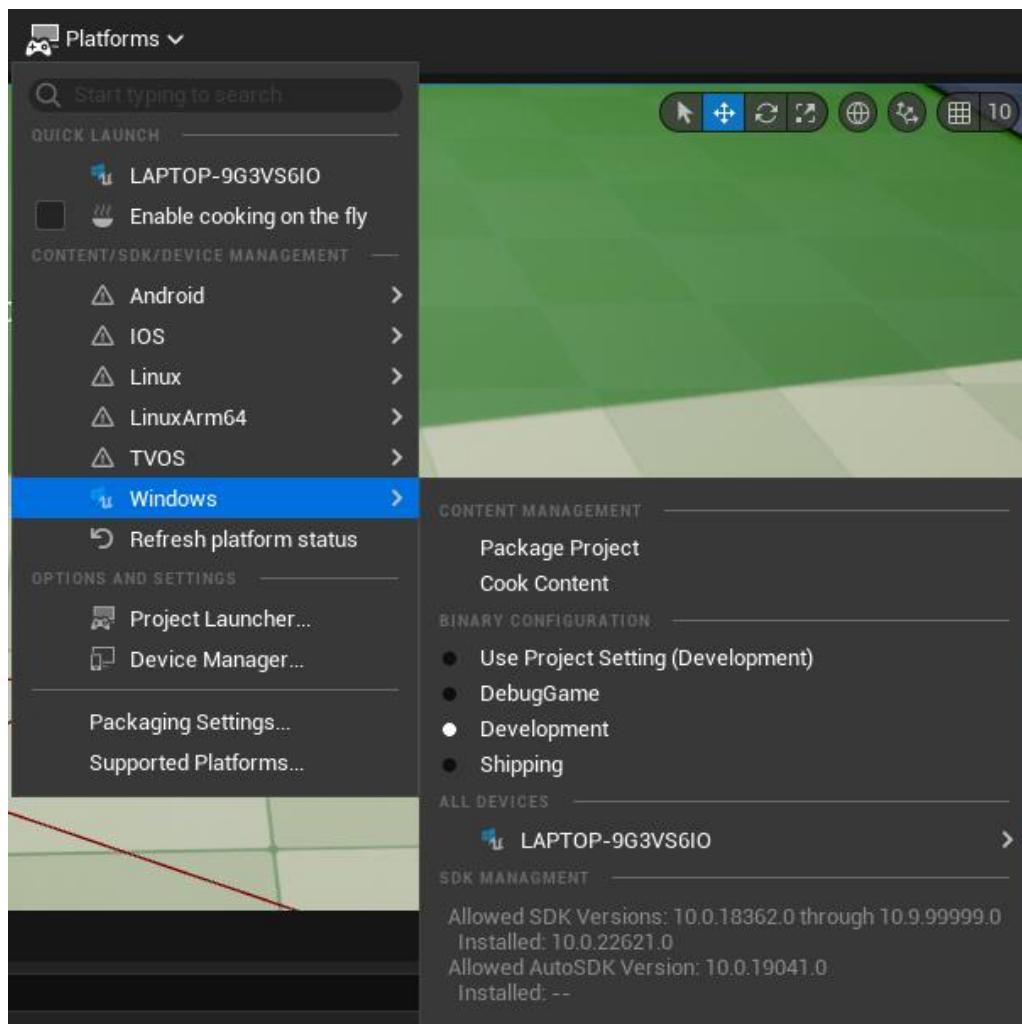


Рисунок 5.1 – Необхідні параметри та конфігурація збірки в UE5.

Упаковка проекту(рис 5.2). В редакторі Unreal Engine обирається команда Package Project. Вказується платформа (наприклад, Windows 64-bit). Визначається каталог для збереження результату збірки.

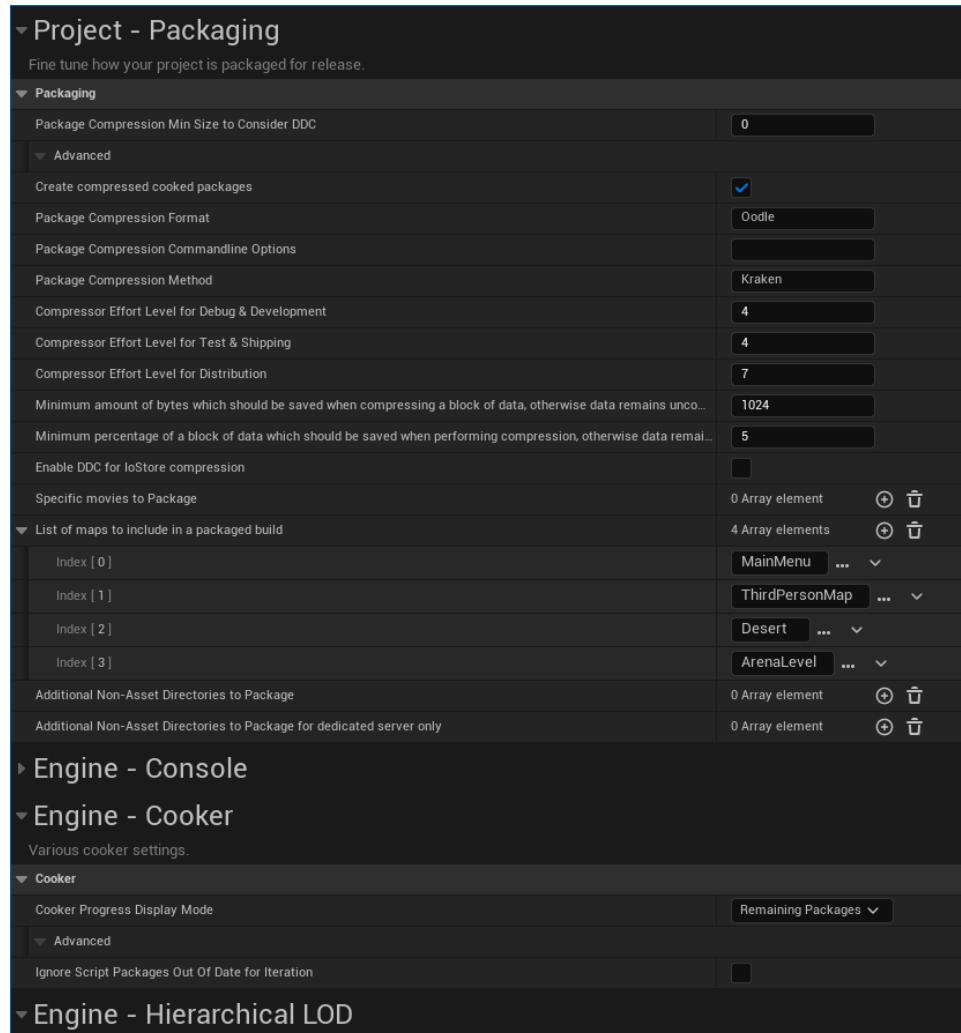


Рисунок 5.2 – Пакування проекту в UE5.

Тестування збірки та її розповсюдження. перевірка коректності запуску, продуктивності, основного функціоналу. Копіювання пакованого проекту на дистрибутивні платформи (Steam, Epic Games Store, локальні сервери). Налаштування оновлень і підтримки.

5.2 Супровід програмного забезпечення

Інструкція користувача наведена в окремому документі «ІП-11_Головня_КК_2025».

Супровід RPG гри, розробленої на Unreal Engine 5, включає виправлення багів та оновлення контенту за допомогою стандартних інструментів UE5.

Після внесення змін у проект необхідно знову виконати білд та розгортання, описану в розділі 5.1. Готовий збірний пакет гри потім завантажується вручну на платформу дистрибуції. При цьому всі налаштування користувача та збереження прогресу залишаються без змін, забезпечуючи безперервний ігровий досвід.

Висновки до розділу

У цьому розділі розглянуто процес розгортання та супроводу програмного забезпечення, створеного на базі Unreal Engine 5. Основну увагу приділено механізму упаковки проєкту у виконувані файли для цільових платформ.

Розгортання передбачає кілька етапів: перевірку готовності ресурсів, встановлення параметрів збірки (Development, Shipping), вибір платформи та запуск процесу упаковки. У результаті формується стабільна збірка, яку потім тестиють на предмет працездатності та продуктивності. Після успішного тестування, проєкт публікується.

У розділі також описано процедуру супроводу гри. Внесення змін у код чи контент вимагає повторної компіляції та повторного розгортання. Ці оновлення не зачіпають персональні налаштування користувачів або їхній прогрес у грі, що забезпечує безперервність ігрового процесу. Вся робота з оновленнями виконується з використанням стандартних інструментів UE5.

Таким чином, реалізовано повноцінний цикл життєвого супроводу ПЗ: від створення збірки до підтримки та оновлення після релізу.

ВИСНОВКИ

Метою розробки було підвищення якості ігрового досвіду користувачів шляхом реалізації штучного інтелекту для неігрових персонажів.

У процесі роботи поставлена мета цілком досягнута. Реалізовано весь обсяг запланованих підсистем, передбачених функціональними вимогами, що підтверджує відповідність результатів очікуванням.

Забезпечено функціонування головного меню гри. Було реалізовано повноцінне головне меню, яке дозволяє користувачеві запускати нову гру, продовжувати збережену, переглядати налаштування, а також виходити з програми.

Забезпечено функціонування ігрових механік. Розроблено базову логіку ігрового процесу, включаючи управління рухом гравця, стрибки, атаки, облік здоров'я та смерті персонажів. Також реалізовано механіку нанесення шкоди, відновлення здоров'я та використання предметів. Ігрова логіка побудована таким чином, щоб легко розширювати її новими функціональностями в майбутньому.

Забезпечено функціонування ігрового інтерфейсу (компоненти HUD). Створено інтерфейс користувача, який включає всі необхідні елементи HUD: індикатори здоров'я, досвіду, рівня, компас, мінікарту, іконки активних ефектів та повідомлень. Інтерфейс динамічно оновлюється в процесі гри відповідно до дій гравця та змін у стані персонажа.

Забезпечено функціонування взаємодії з інвентарем. Було реалізовано систему інвентарю, яка дозволяє гравцеві підбирати, переглядати, використовувати та екіпірувати предмети. Також забезпечено зручну візуалізацію слотів інвентарю, можливість перетягування предметів, взаємодію між предметами та інтерфейсом екіпіровки.

Забезпечено функціонування штучного інтелекту неігрових персонажів гри. Розроблено систему AI для NPC, яка забезпечує опціональні ситуації для гри, як: патрулювання території, виявлення гравця, переслідування та участь

у бою. Реалізовані також прості діалогові механіки та поведінка NPC поза боєм. Штучний інтелект працює стабільно в рамках заданих сценаріїв.

Забезпечено взаємодію з колізією об'єктів. Упроваджено систему колізій, яка обробляє зіткнення персонажа з навколишнім середовищем, ігровими об'єктами та іншими персонажами. Це дозволяє забезпечити фізично коректну взаємодію з ігровим світом, уникнути проходження крізь стіни, а також реалізувати активацію подій при вході в зони тригерів.

Незважаючи на те, що проект знаходиться на початковій стадії реалізації, він є повноцінною основою для подальшого розвитку. Його архітектура дозволяє ефективно масштабувати та доповнювати функціональність відповідно до сучасних вимог у сфері ігрової індустрії.

Отже, продовження розробки є доцільним і може привести до створення повноцінного ігрового продукту, що відповідає актуальним очікуванням користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1) Bethesda Softworks Official site. URL:
<https://bethesda.net/en/dashboard> (дата звернення: 15.03.2025)
- 2) CD Projekt RED Official site. URL:
<https://www.cdprojektred.com/en/about-us> (дата звернення: 15.03.2025)
- 3) Main Components of Game Development. URL:
<https://www.argentics.io/main-component-of-game-development-stages> (дата звернення: 21.03.2025)
- 4) Game Genres Overview. QATestLab. URL:
<https://training.qatestlab.com/blog/technical-articles/games-genres> (дата звернення: 22.03.2025)
- 5) Crescent Moon Games. Aralon: Sword and Shadow. URL:
<https://www.crescentmoongames.com/> (дата звернення: 24.04.2025)
- 6) Bound by Flame on Steam. URL:
https://store.steampowered.com/app/243930/Bound_By_Flame/ (дата звернення: 24.04.2025)
- 7) Steam Platform. URL: <https://store.steampowered.com/about/> (дата звернення: 24.04.2025)
- 8) Risen on Steam. URL:
<https://store.steampowered.com/app/40300/Risen/> (дата звернення: 24.04.2025)
- 9) AI Perception System in Unreal Engine. URL:
<https://dev.epicgames.com/documentation/en-us/unreal-engine/ai-perception-in-unreal-engine> (дата звернення: 03.05.2025)
- 10) Behavior Tree Editor in Unreal Engine. URL:
<https://dev.epicgames.com/documentation/en-us/unreal-engine/behavior-trees-in-unreal-engine> (дата звернення: 04.05.2025)
- 11) NavMesh API in Unreal Engine. URL:
<https://dev.epicgames.com/documentation/en-us/unreal-engine/API/Runtime/NavigationSystem/NavMesh> (дата звернення: 05.05.2025)

- 12) Animation Blueprints and State Machines in Unreal Engine. URL: <https://dev.epicgames.com/documentation/en-us/unreal-engine/animation-blueprints-in-unreal-engine> (дата звернення: 05.05.2025)
- 13) The Witcher 3: Wild Hunt on Steam. URL: https://store.steampowered.com/app/292030/The_Witcher_3_Wild_Hunt/ (дата звернення: 17.03.2025)
- 14) Cyberpunk 2077 Official site. URL: <https://www.cyberpunk.net/us/en/> (дата звернення: 21.03.2025)
- 15) Elden Ring on Steam. URL: https://store.steampowered.com/app/1245620/ELDEN_RING/ (дата звернення: 22.03.2025)
- 16) Unreal Engine 5 Official site. URL: <https://www.unrealengine.com/en-US> (дата звернення: 15.05.2025)
- 17) Unity Engine Official site. URL: <https://unity.com/> (дата звернення: 15.05.2025)
- 18) Godot Engine Official site. URL: <https://godotengine.org/> (дата звернення: 16.05.2025)
- 19) Blueprints in Unreal Engine. URL: <https://dev.epicgames.com/documentation/en-us/unreal-engine/blueprints-visual-scripting-in-unreal-engine> (дата звернення: 22.05.2025)

ДОДАТКИ

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРИКОВ

“___” _____ 2025 р.

**ГРА У ЖАНРІ ROLE-PLAYING-GAME(RPG) НА UNREAL ENGINE 3
ВИКОРИСТАННЯМ ШІ**

Текст програми

КП.ІП-1107.045440.03.12

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Максим ГОЛОВЧЕНКО

Нормоконтроль:

_____ Максим ГОЛОВЧЕНКО

Виконавець:

_____ Олександр ГОЛОВНЯ

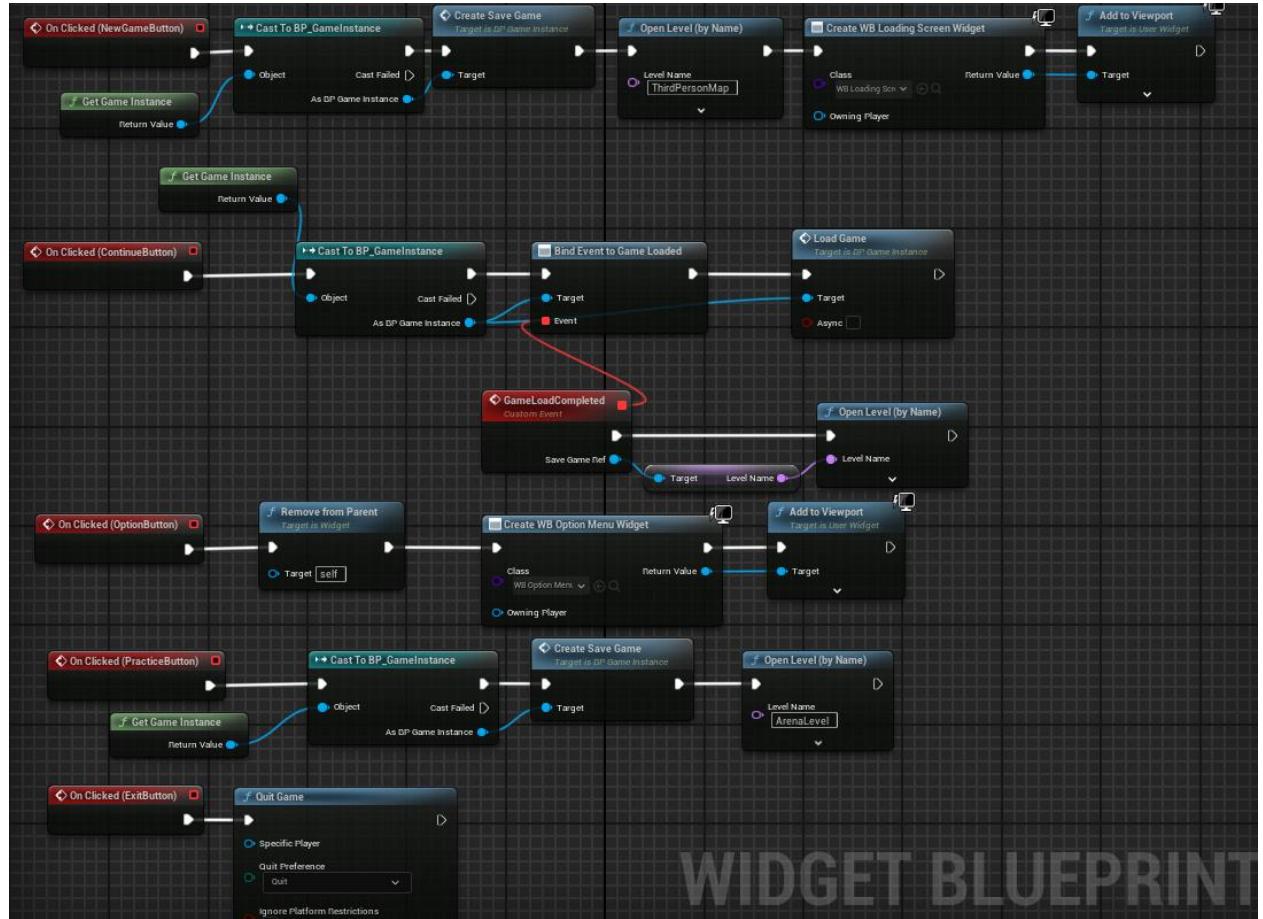
Київ – 2025

Посилання на репозиторій з повним програмним кодом

https://github.com/YeaLowww/YeaLow_RPG/

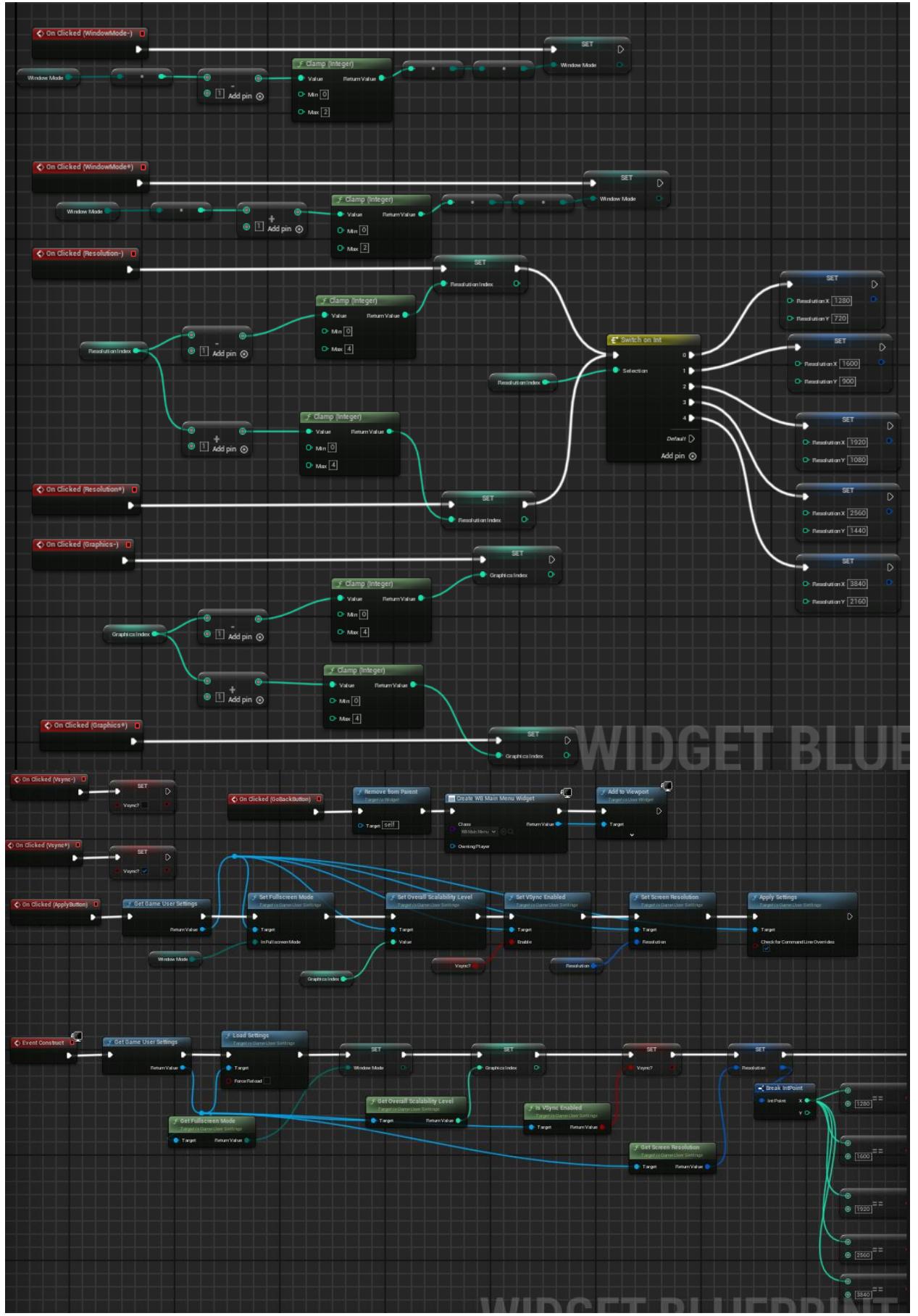
Реалізація функціональної задачі забезпечення функціонування головного меню гри:

Файл WB_MainMenu

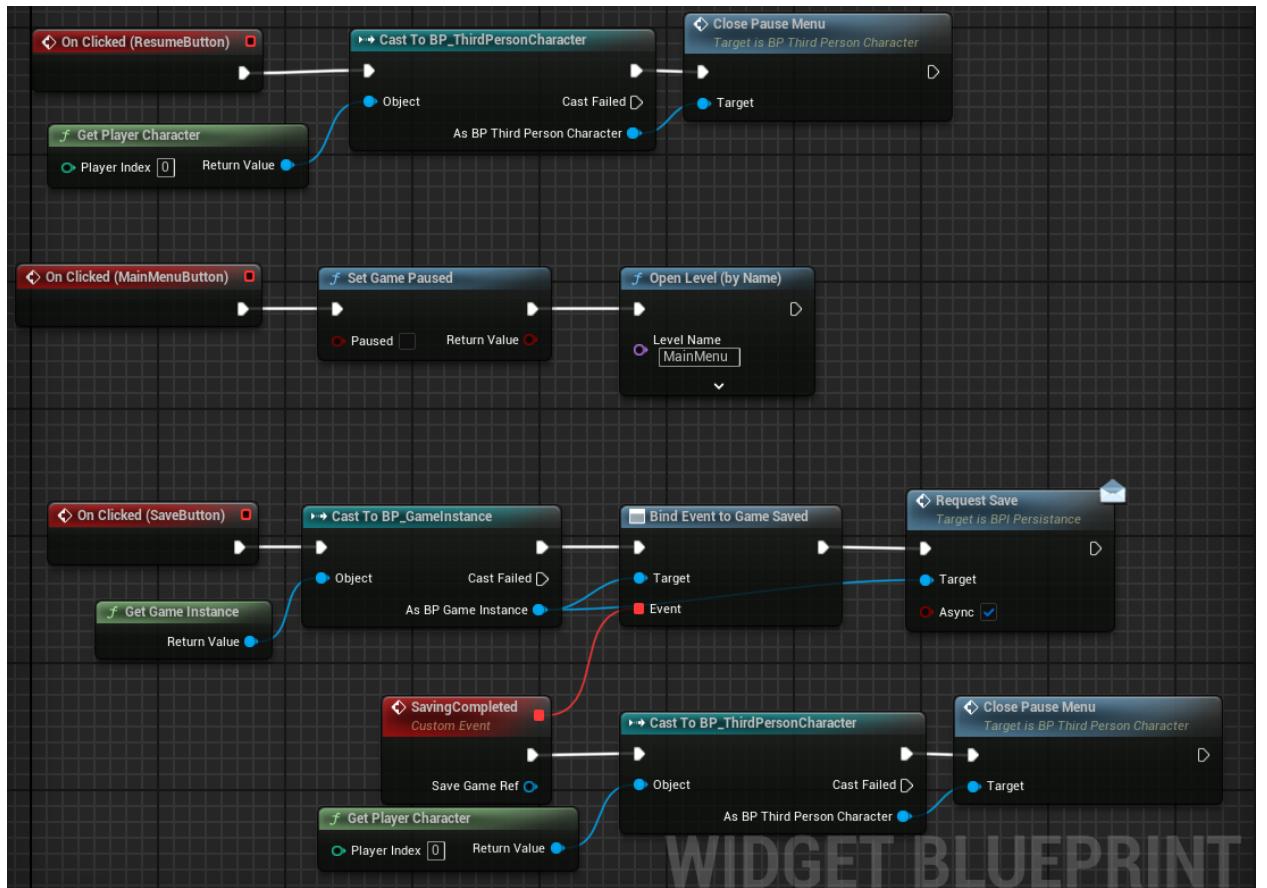


Файл WB_OptionMenu

WIDGET BLUEPRINT

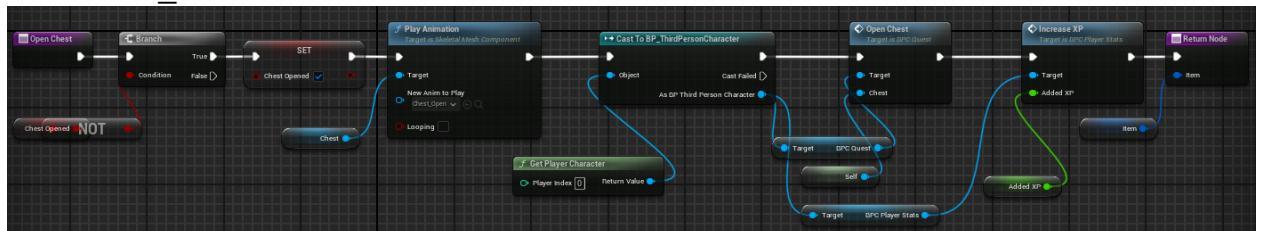


Файл WB_PauseMenu

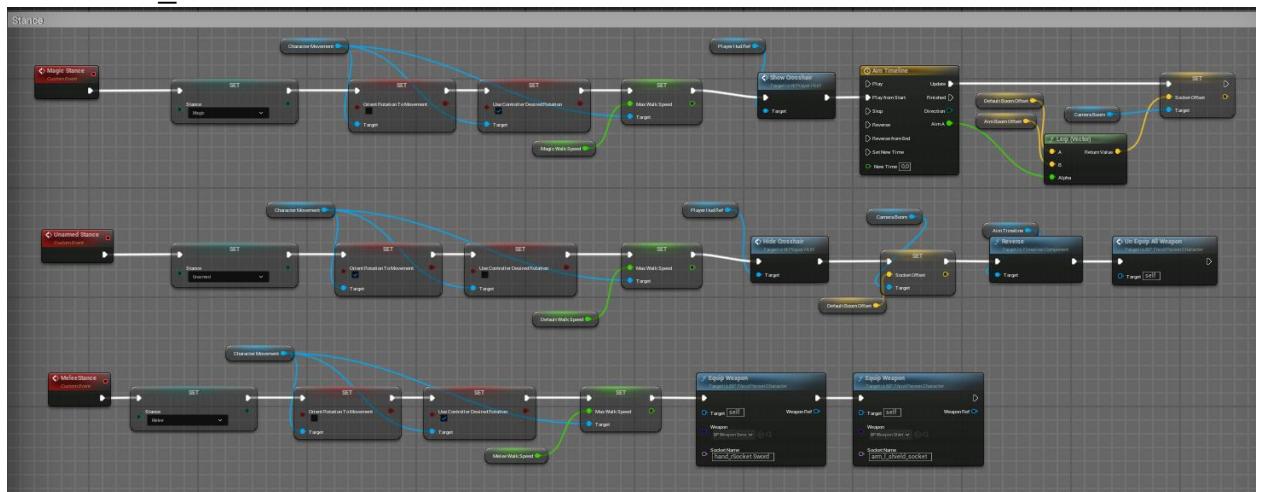


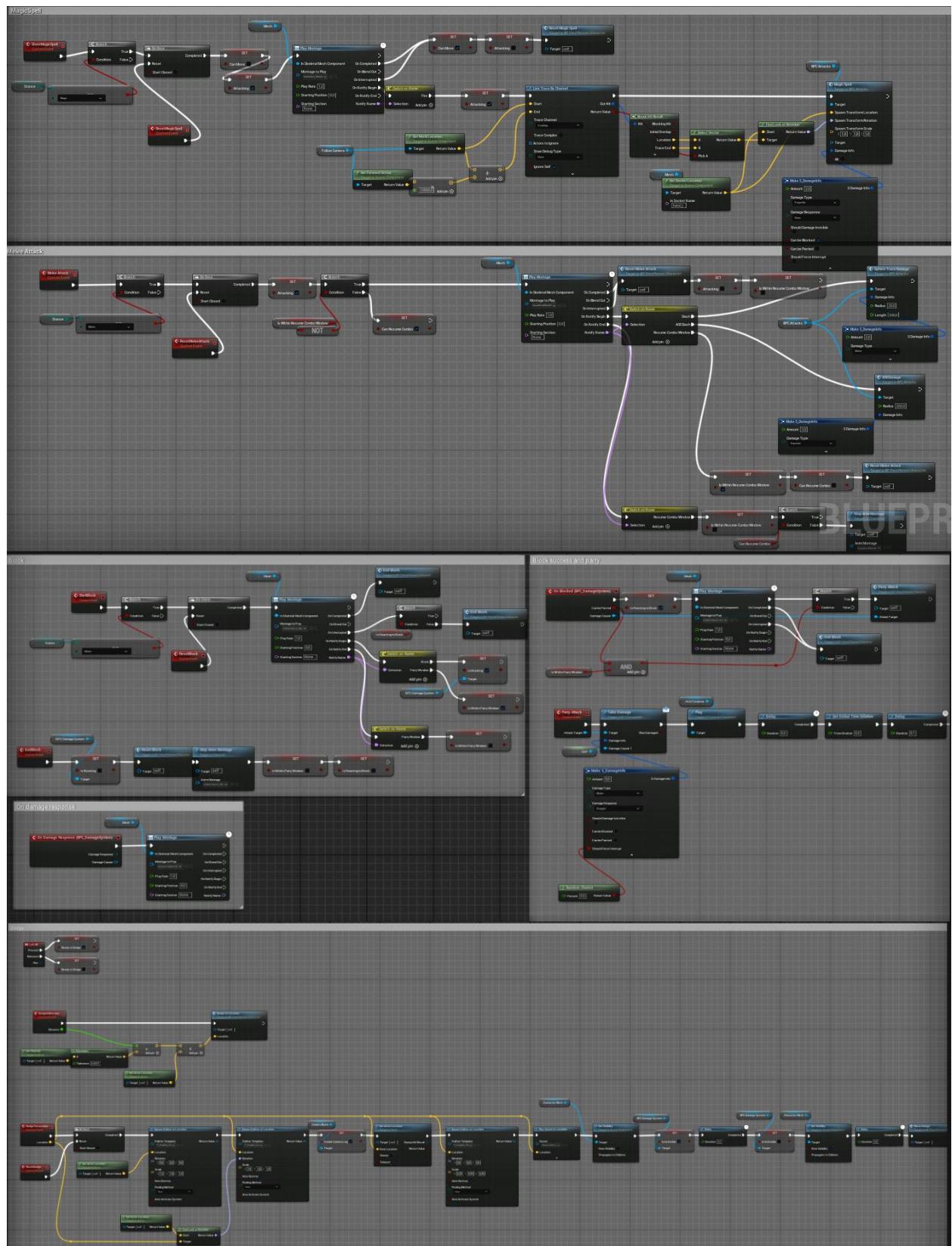
Реалізація функціональної задачі забезпечення функціонування ігрових механік:

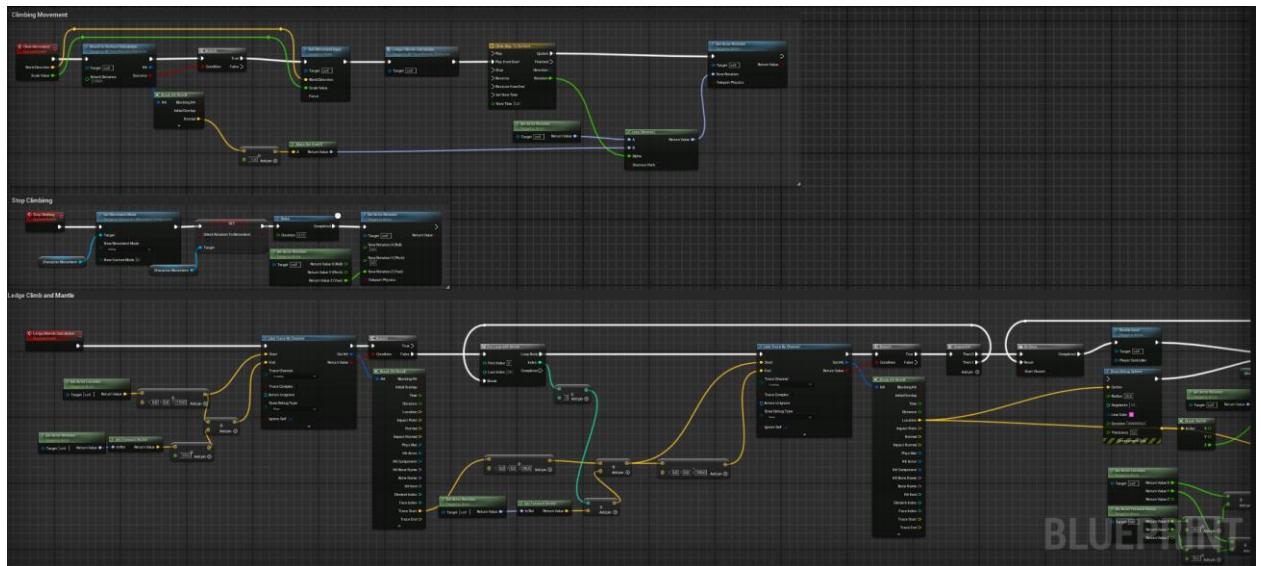
Файл PB_Chest



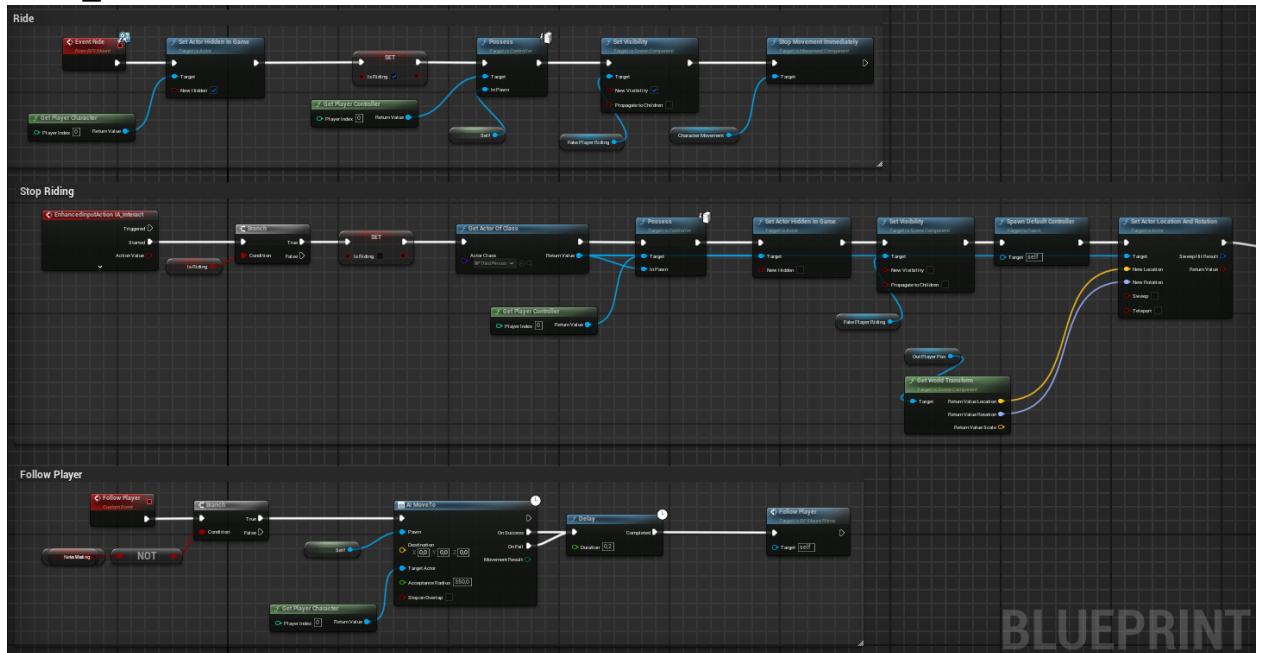
Файл PB_ThirdPersonCharacter



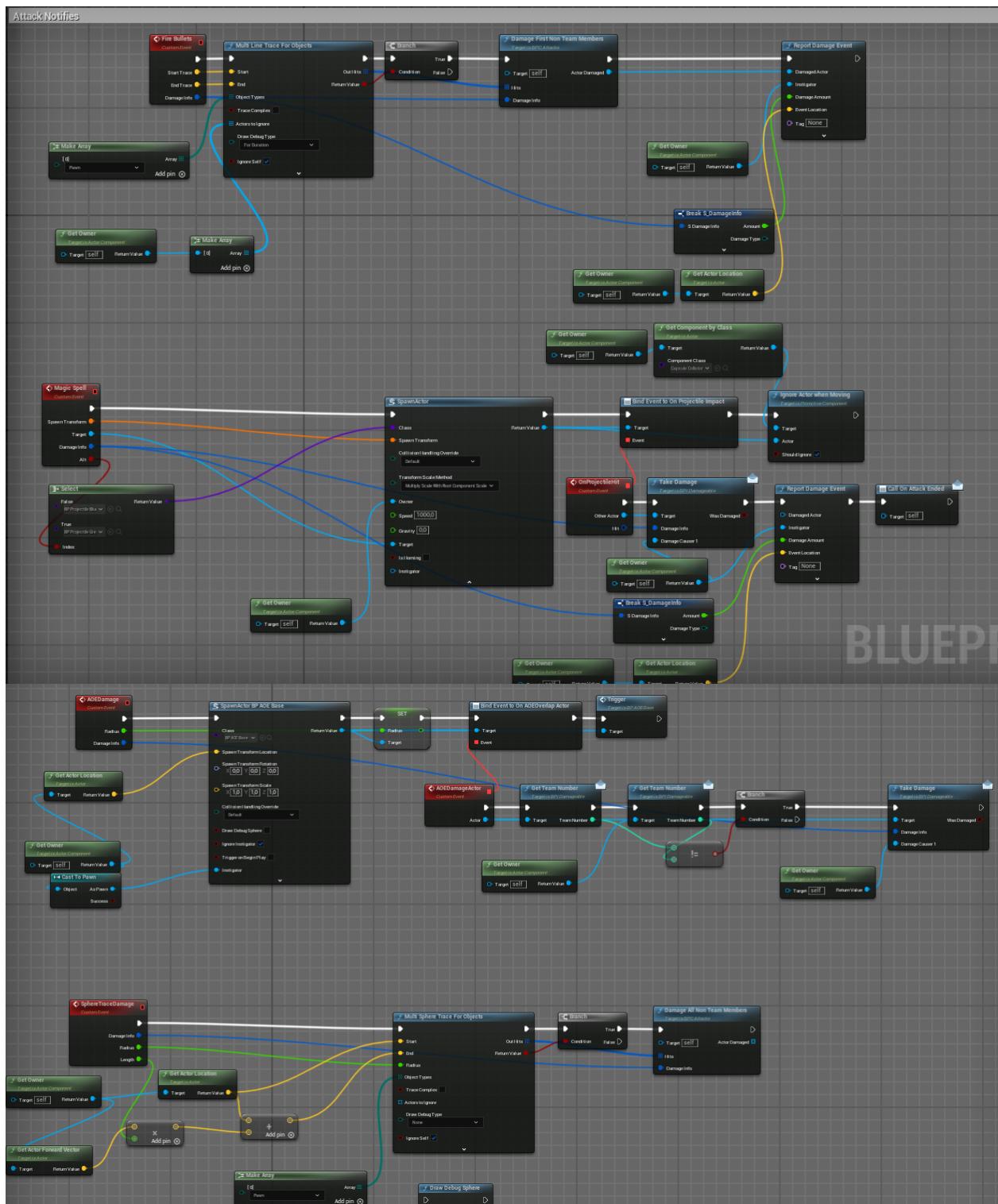


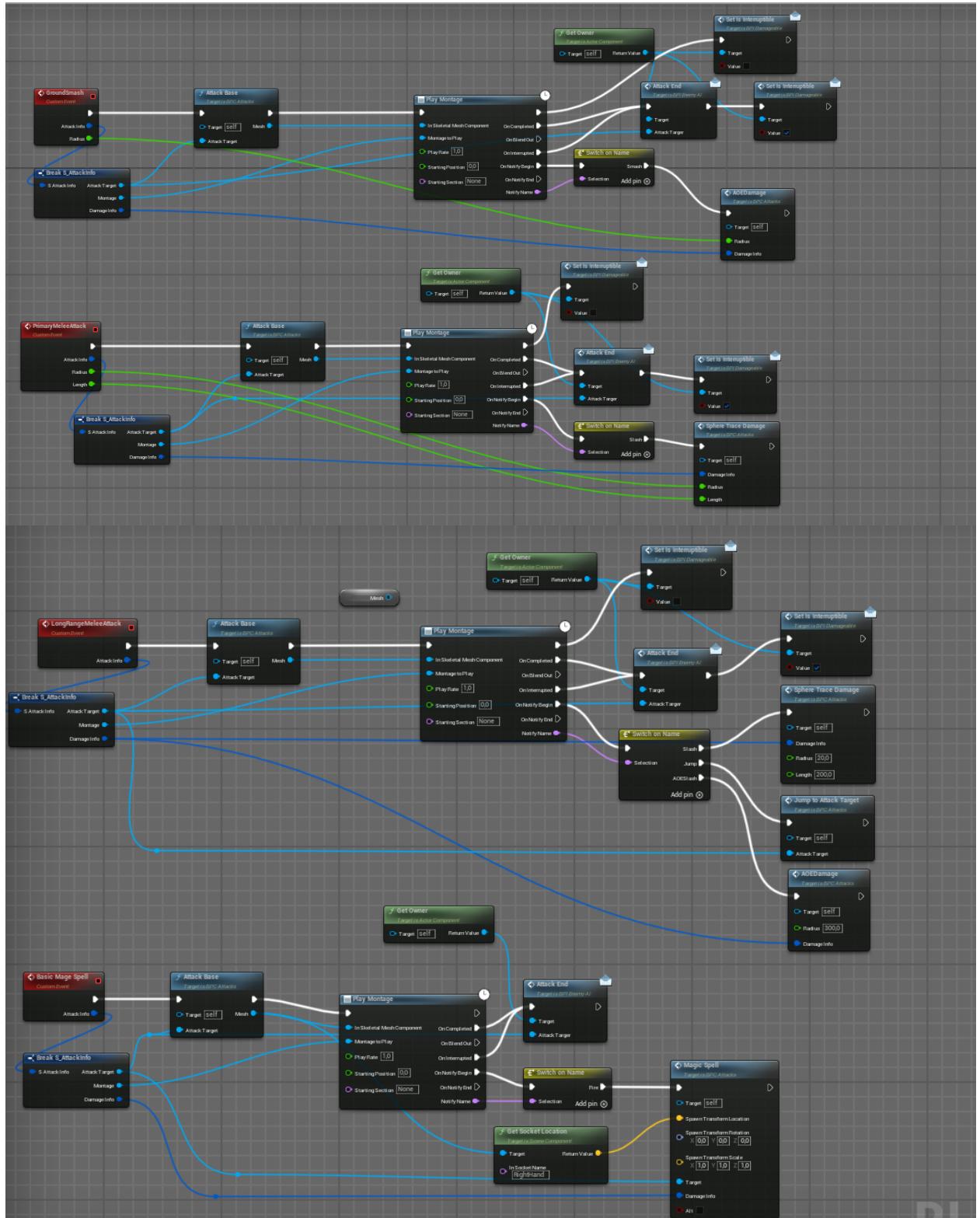


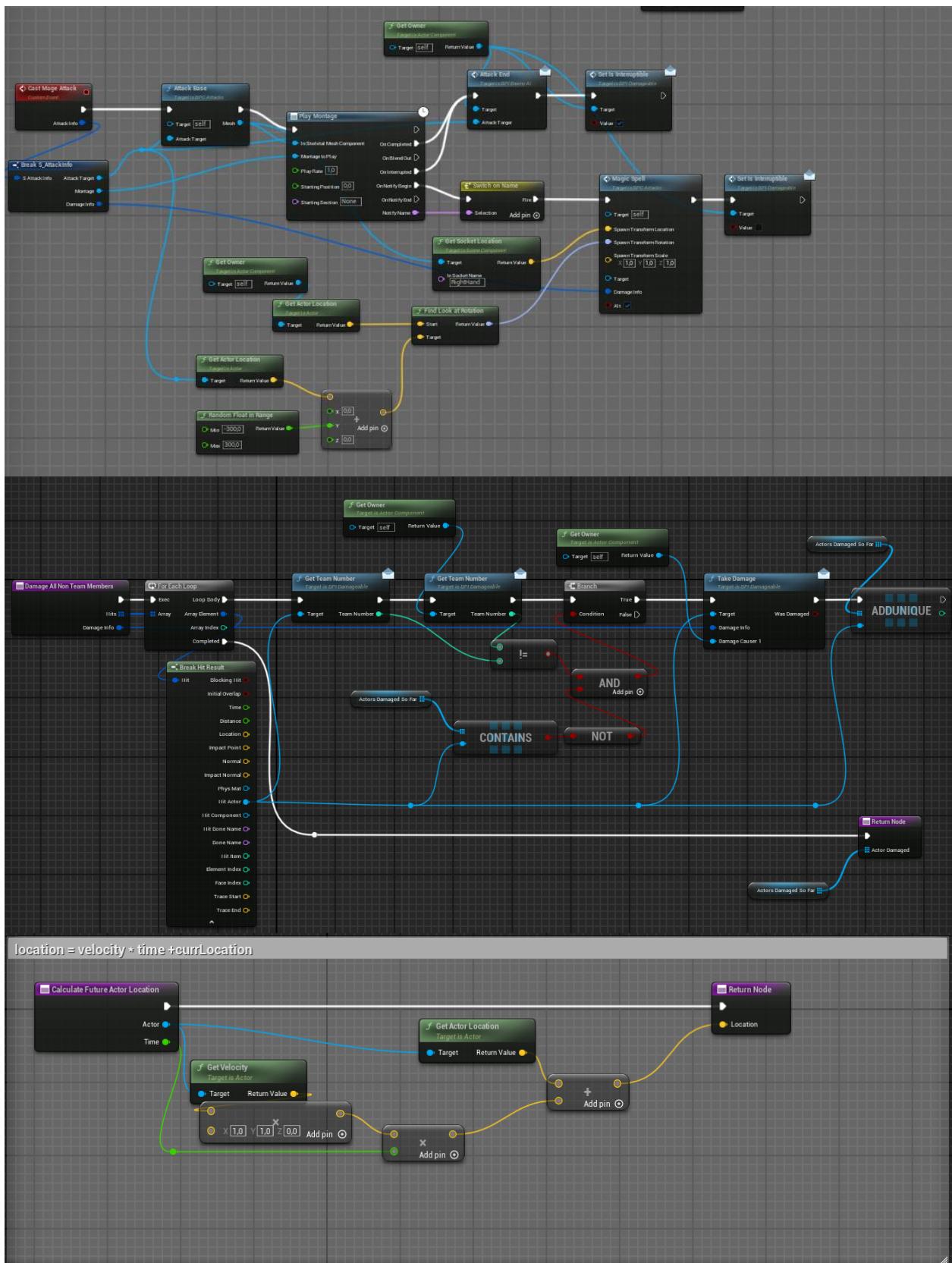
PBC_MountRhino



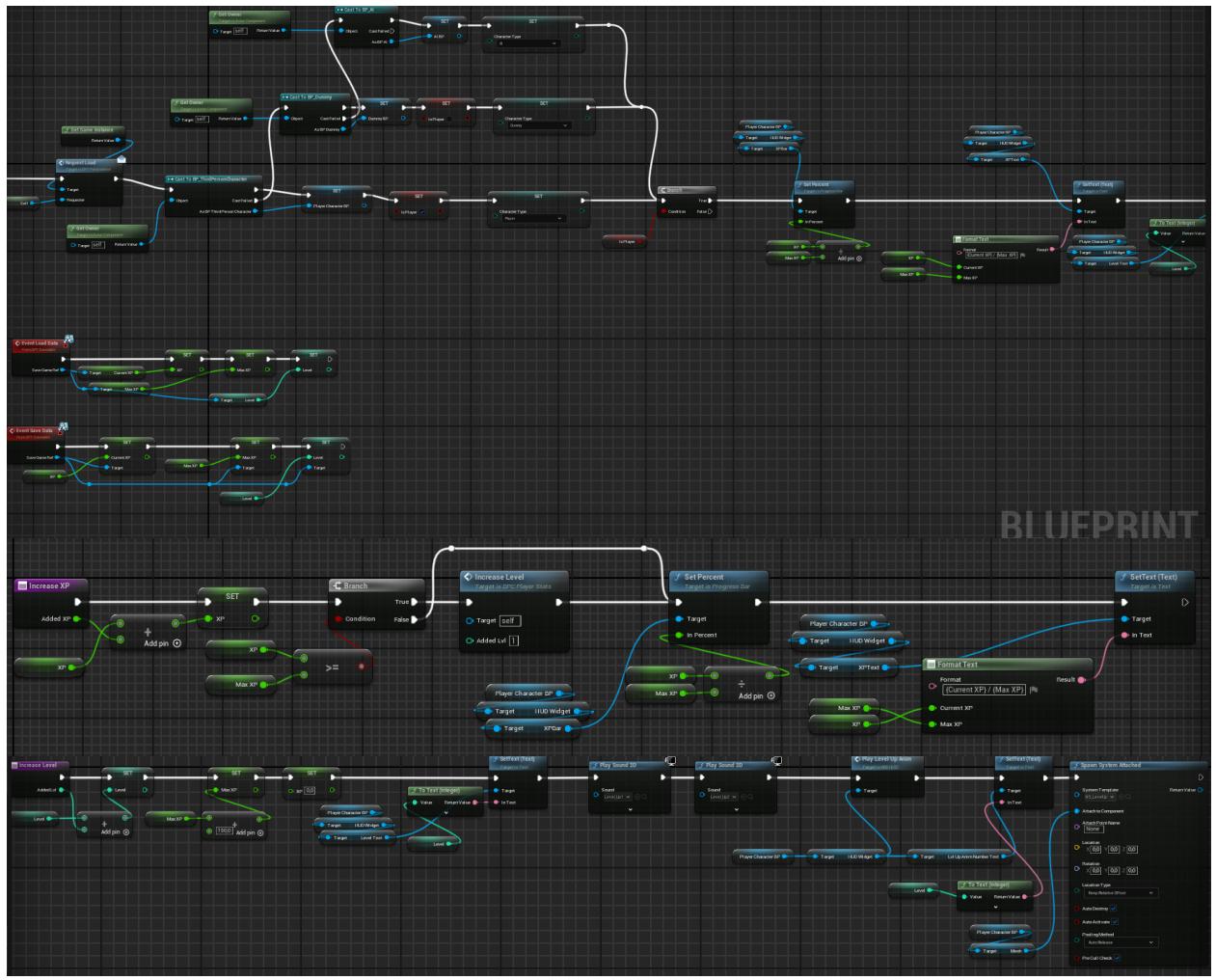
Файл PBC_Attacks





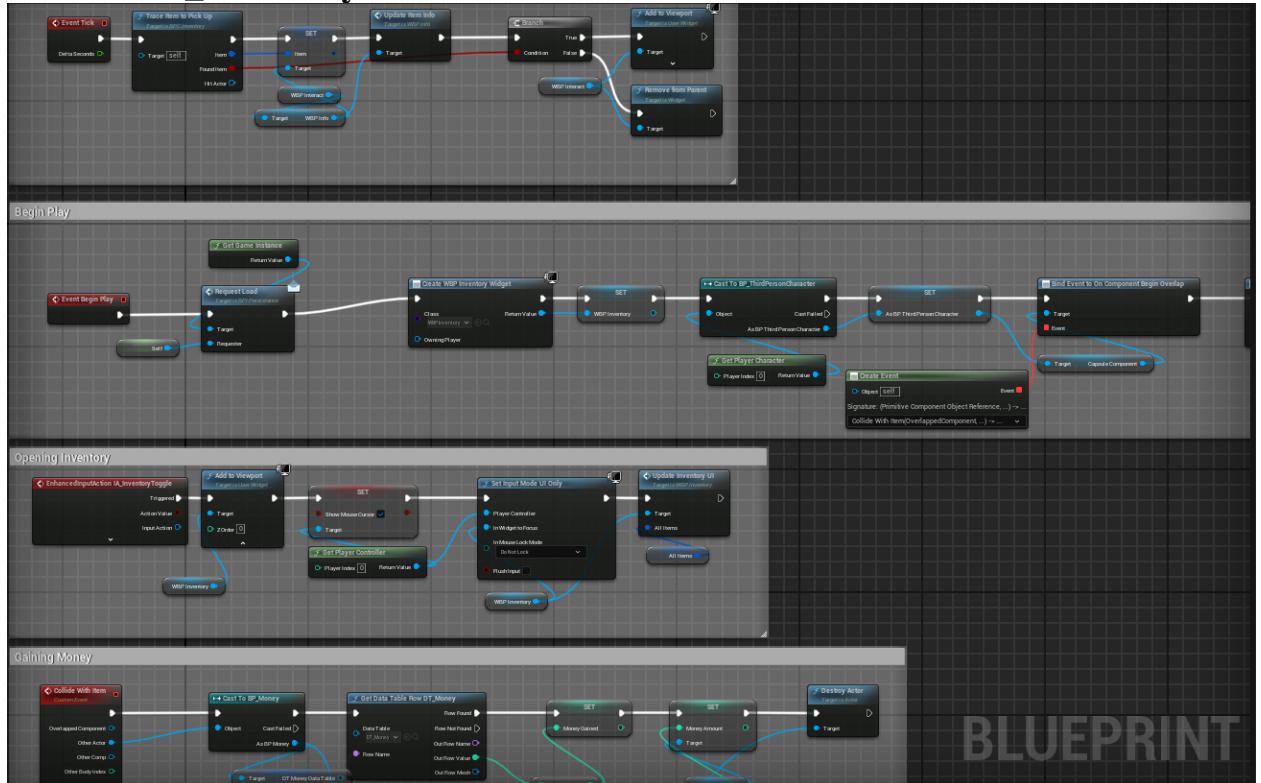


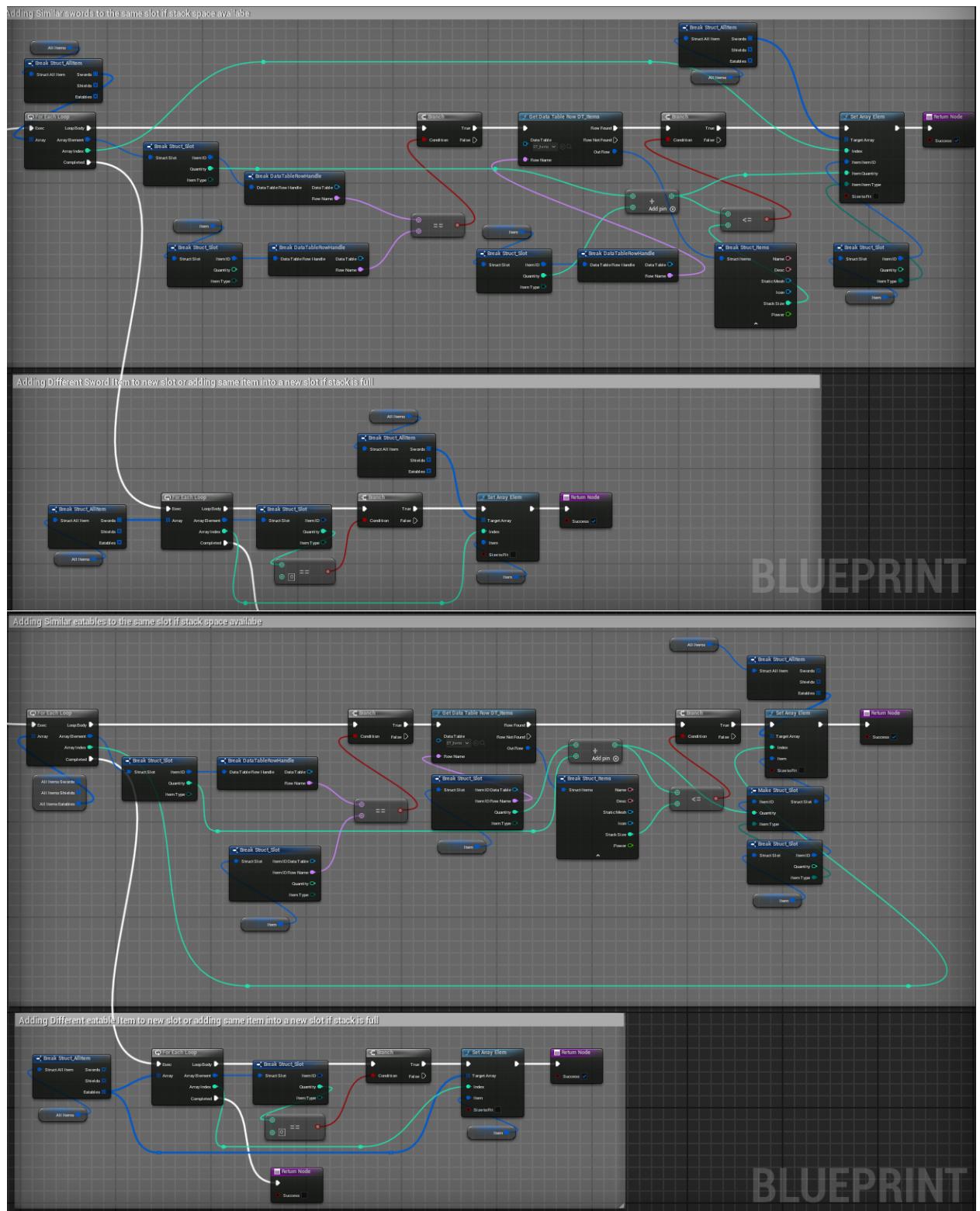
Файл PBC_PlayerStats



Реалізація функціональної задачі забезпечення функціонування взаємодії з інвентарем:

Файл PBC_Inventory

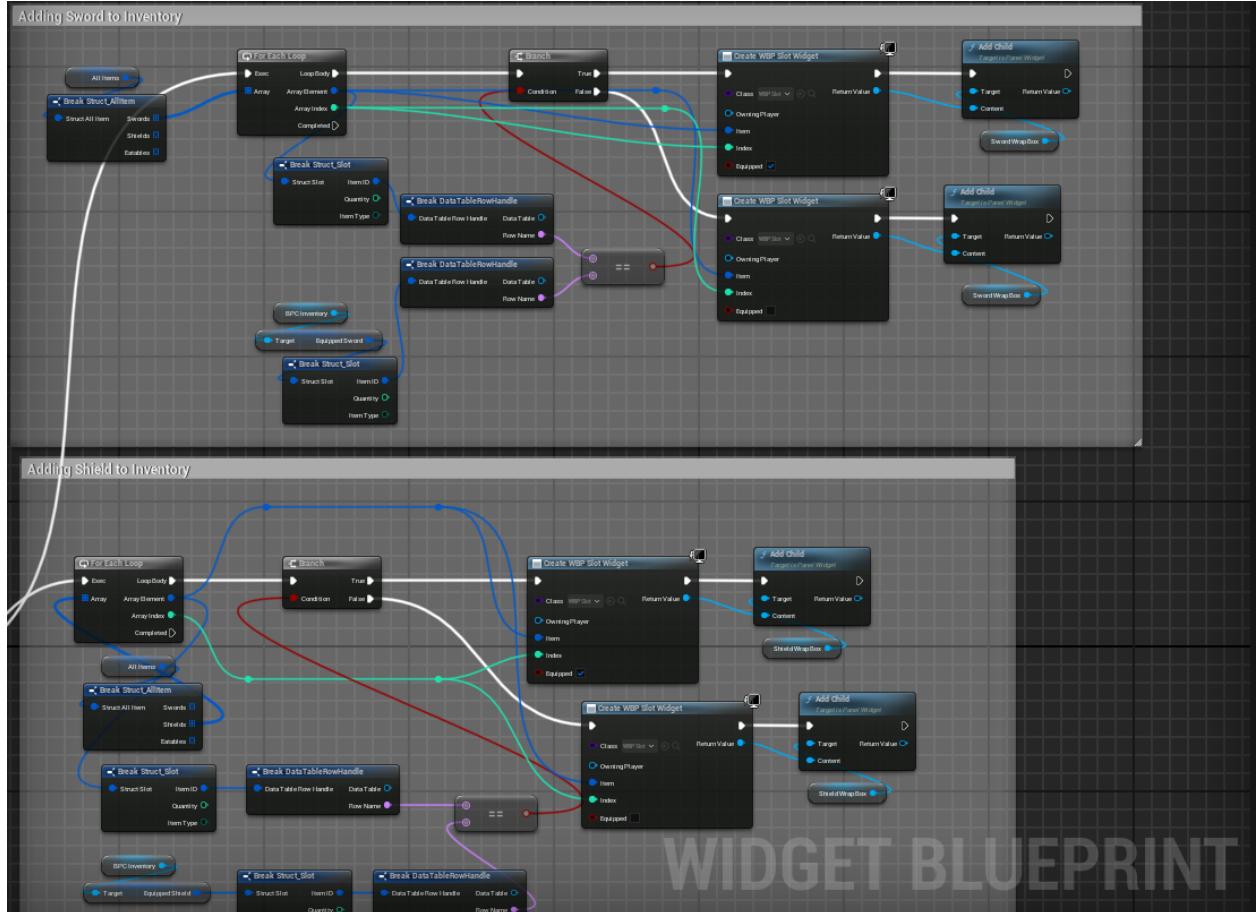




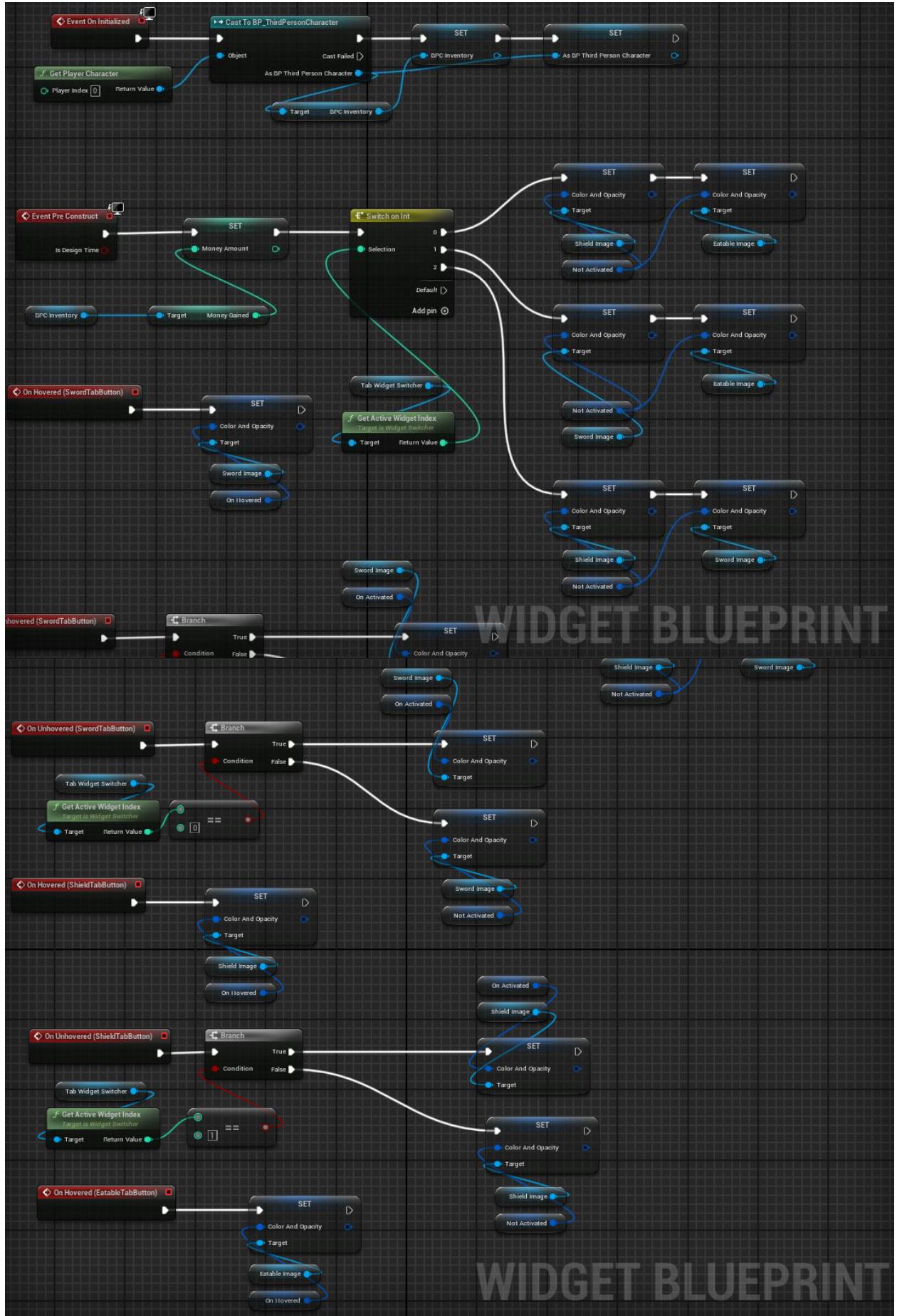
Файл DT_Items

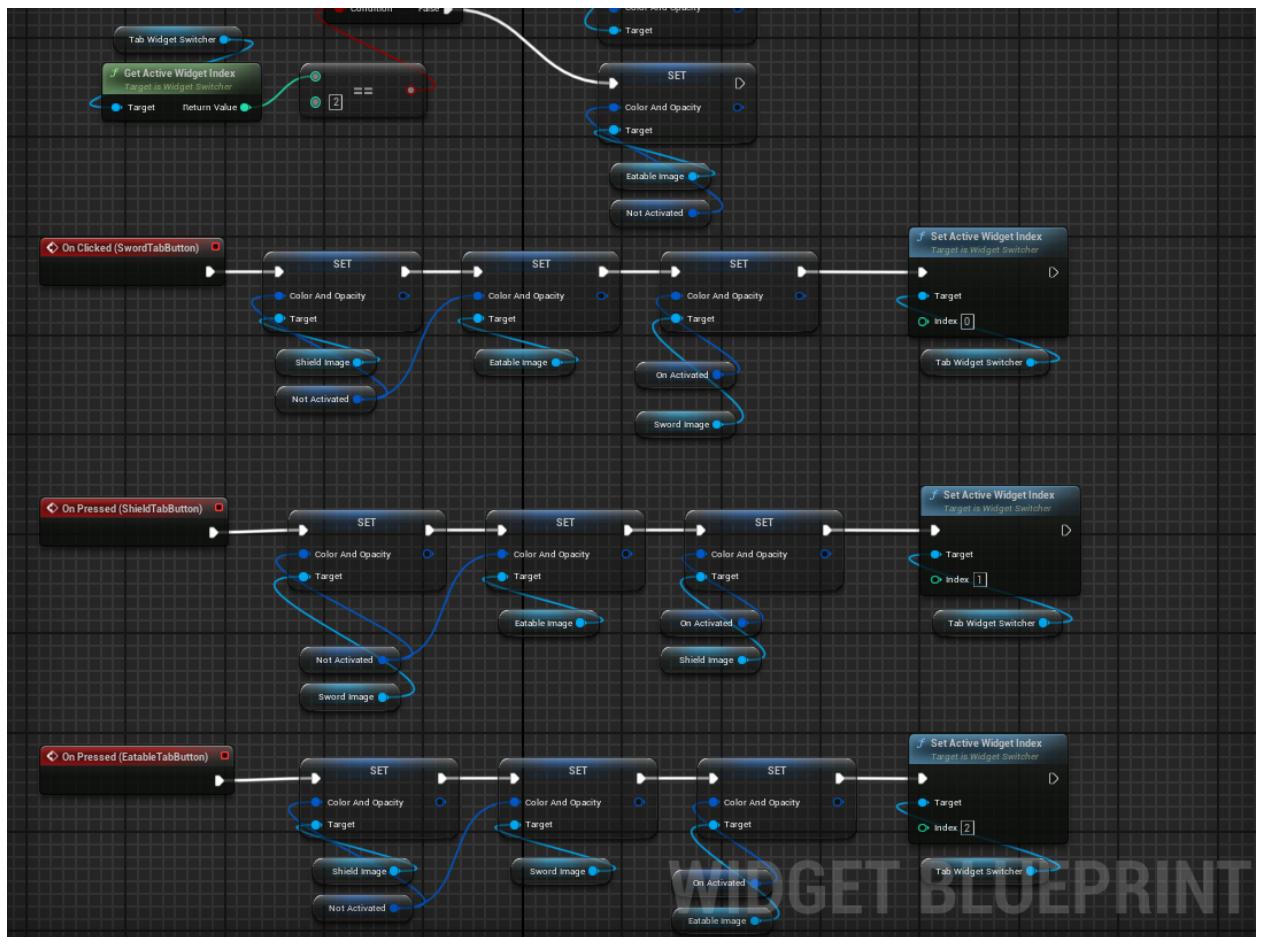
Data Table			Icon	Stack Size	Power
Row Name	Name	Desc	Static Mesh		
1 Blizzard_Sword	Blizzard Sword	A magical rod that can cast extreme cold in a wide range. These days it's the weapon of /Script/Engine.StaticMesh'/Game/Inventory/Assets/Swords/Blizzard_Rod	/Script/Engine.Texture2D'/Game/Inventory/Assets/Swords/Blizzard_Rod	1	35.000000
2 Knight_Sword	Knight Sword	Knights of Hyrule once carried this sword. These days it's the weapon of /Script/Engine.StaticMesh'/Game/Inventory/Assets/Swords/Knight_SBro_1	/Script/Engine.Texture2D'/Game/Inventory/Assets/Swords/Knight_SBro_1	50.000000	
3 Spiked_Sword	Spiked Sword	A reinforce Bokoblin club made to maximize damage. It's sharpened to /Script/Engine.StaticMesh'/Game/Inventory/Assets/Swords/Spiked_Bok	/Script/Engine.Texture2D'/Game/Inventory/Assets/Swords/Spiked_Bok	1	45.000000
4 DayBreaker_Shield	DayBreaker Shield	This shield was cherished by the Gerudo Champion Urbosa. The gold use /Script/Engine.StaticMesh'/Game/Inventory/Assets/Shields/Daybreaker/I	/Script/Engine.Texture2D'/Game/Inventory/Assets/Shields/Daybreaker/I	12	23.000000
5 RoyalGuard_Shield	RoyalGuard Shield	This shield was forged using ancient Sheikah technology. It boasts exter /Script/Engine.StaticMesh'/Game/Inventory/Assets/Shields/Royal_Guard	/Script/Engine.Texture2D'/Game/Inventory/Assets/Shields/Royal_Guard	12	33.000000
6 SteelLizal_Shield	SteelLizal Shield	This Lizal shield is adorned with several metal shells as a means of reinforce /Script/Engine.StaticMesh'/Game/Inventory/Assets/Shields/Steel_Lizal_Sl	/Script/Engine.Texture2D'/Game/Inventory/Assets/Shields/Steel_Lizal_Sl	12	40.000000
7 EA_Apple	Apple	A common fruit found on trees all around Hyrule. Eat it fresh, or cook it to /Script/Engine.StaticMesh'/Game/Inventory/Assets/Eatables/Fruit/CO_Aj	/Script/Engine.Texture2D'/Game/Inventory/Assets/Eatables/Fruit/CO_Aj	5	1.500000
8 EA_Pumpkin	Pumpkin	An extremely tough pumpkin raised in village fields. When cooked, that to /Script/Engine.StaticMesh'/Game/Inventory/Assets/Eatables/Fortified_P	/Script/Engine.Texture2D'/Game/Inventory/Assets/Eatables/Fortified_P	4	4.000000
9 EA_Hydromelon	Hydromelon	This resilient fruit can flourish even in the heat of the desert. The hydromel /Script/Engine.StaticMesh'/Game/Inventory/Assets/Eatables/Hydromelo	/Script/Engine.Texture2D'/Game/Inventory/Assets/Eatables/Hydromelo	4	3.000000
10 EA_Banana	Banana	This fruit grows mainly in tropical forests of the Faron region. When it's u /Script/Engine.StaticMesh'/Game/Inventory/Assets/Eatables/Mighty_Ban	/Script/Engine.Texture2D'/Game/Inventory/Assets/Eatables/Mighty_Ban	20	2.500000
11 EA_SpicyPepper	SpicyPepper	This pepper is exploding with spice. Cook it with to create dishes that will /Script/Engine.StaticMesh'/Game/Inventory/Assets/Eatables/Spicy_Pepp	/Script/Engine.Texture2D'/Game/Inventory/Assets/Eatables/Spicy_Pepp	5	1.000000

Файл WBP_Inventory

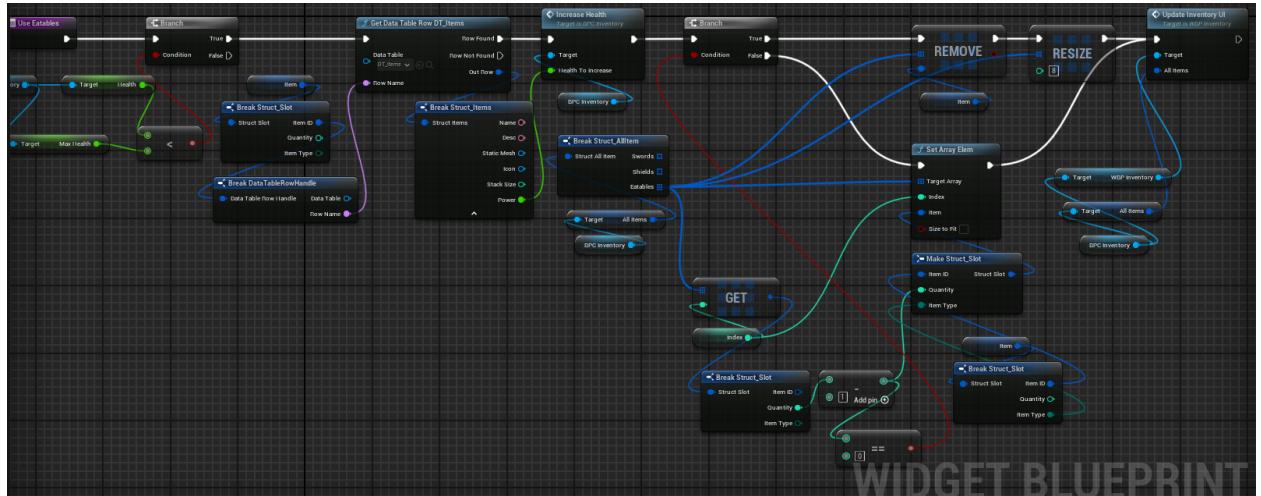


WIDGET BLUEPRINT



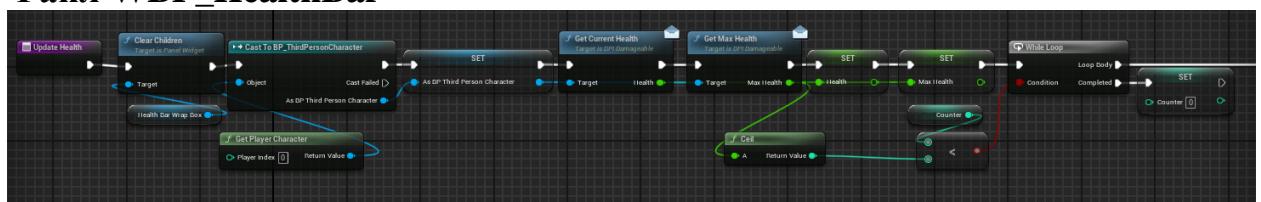


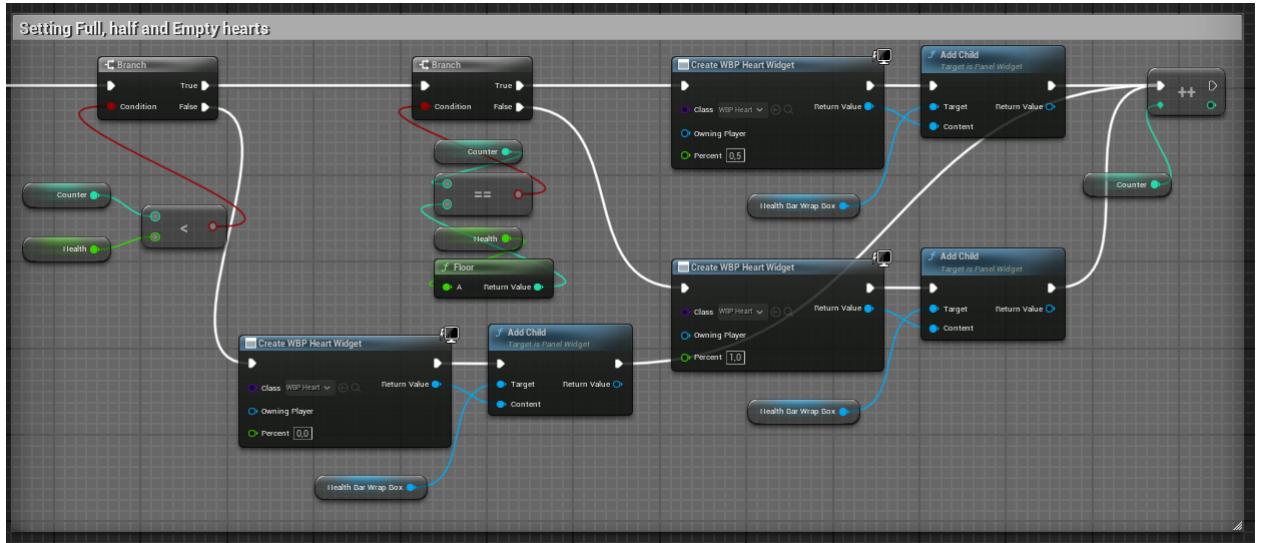
Файл WBP_Slot



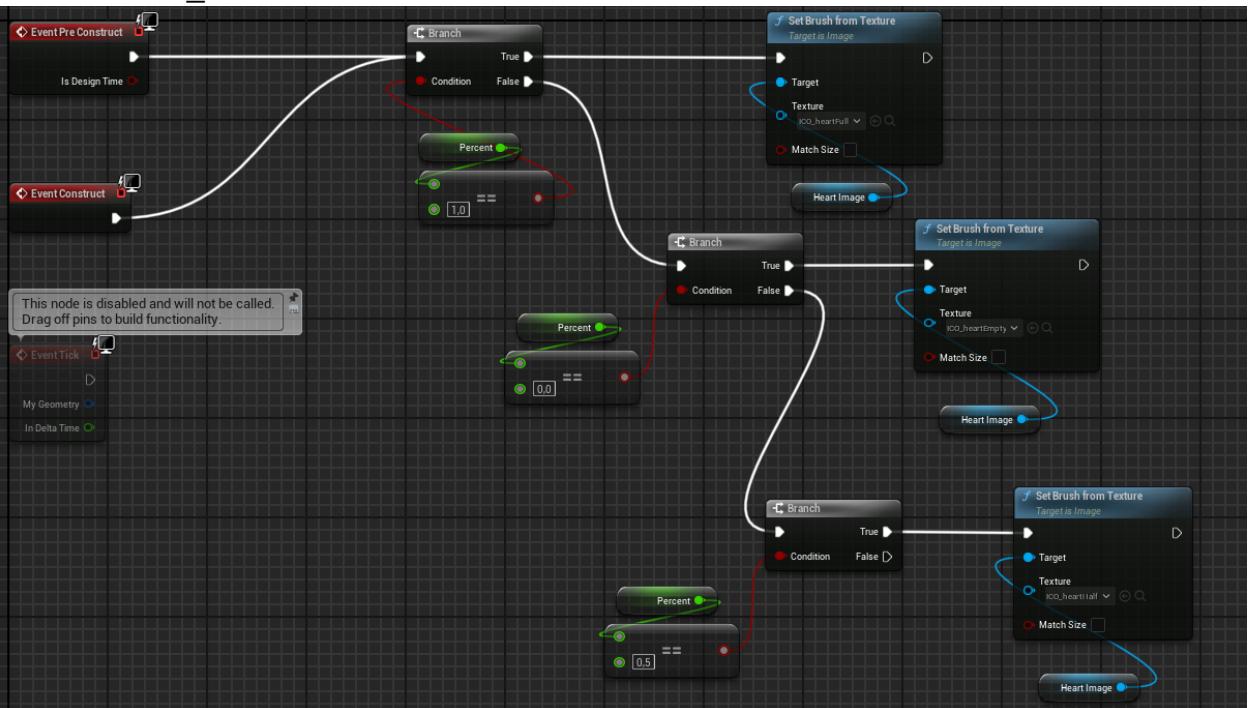
Реалізація функціональної задачі забезпечено функціонування ігрового інтерфейсу (компоненти HUD):

Файл WBP_HealthBar

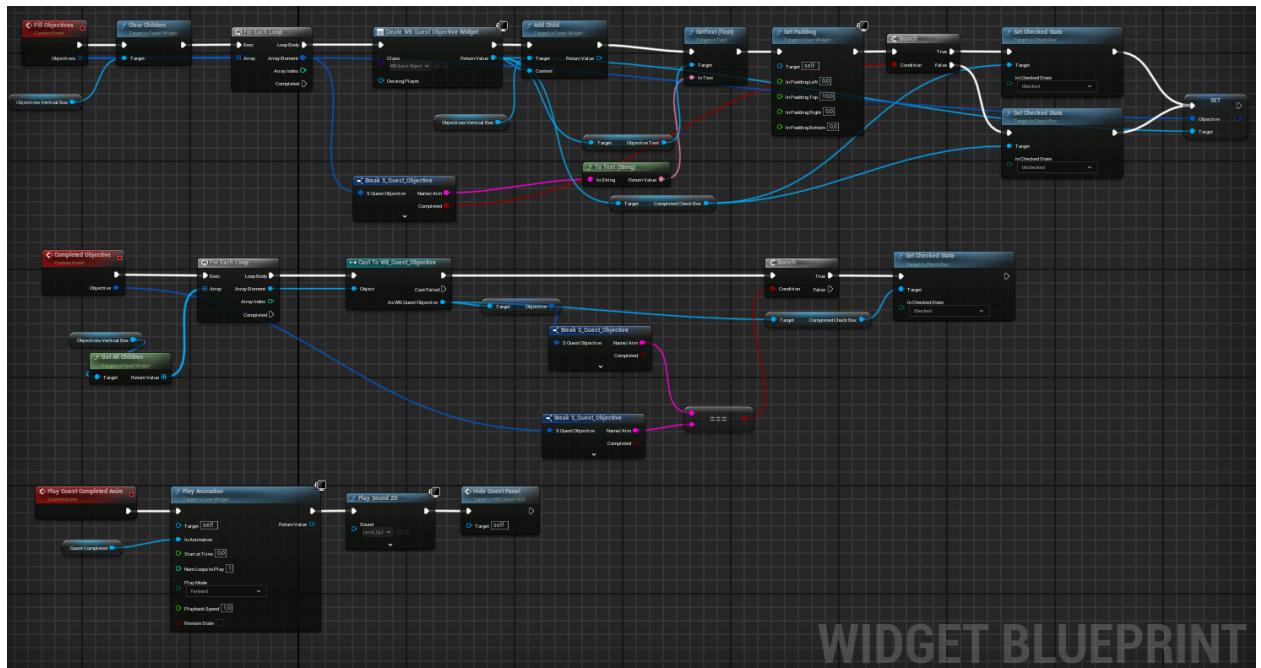




Файл WBP_Heart

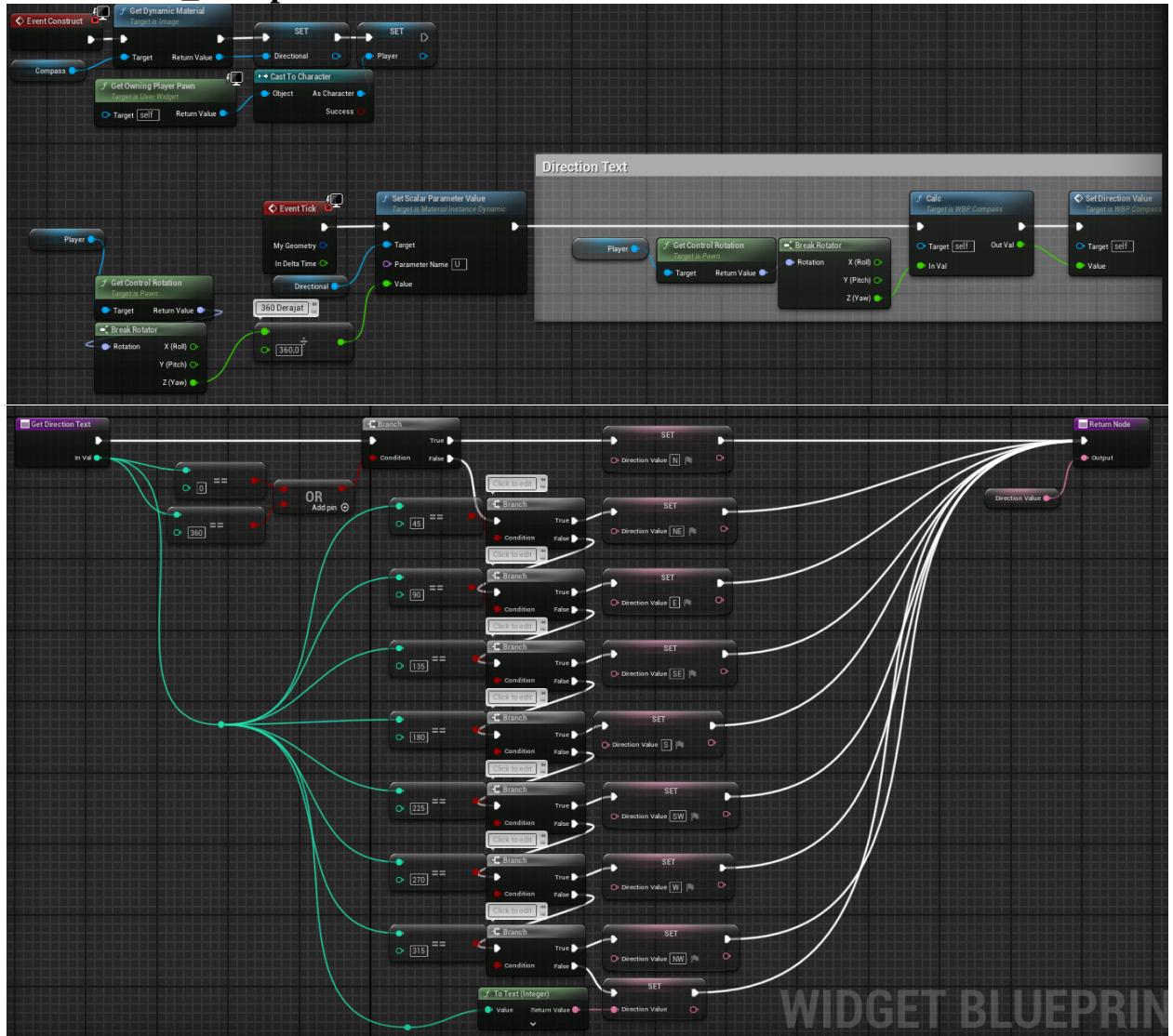


Файл WB_QuestHUD

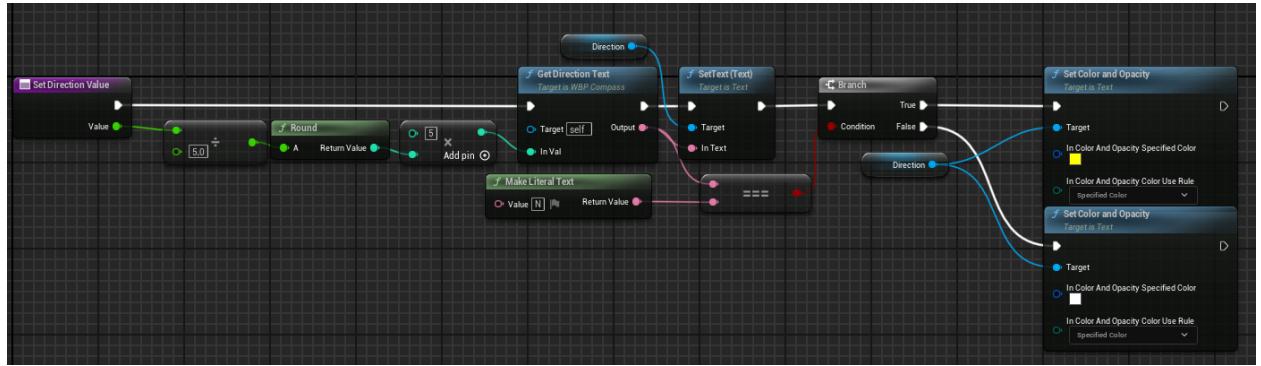


WIDGET BLUEPRINT

Файл WBP_Compass

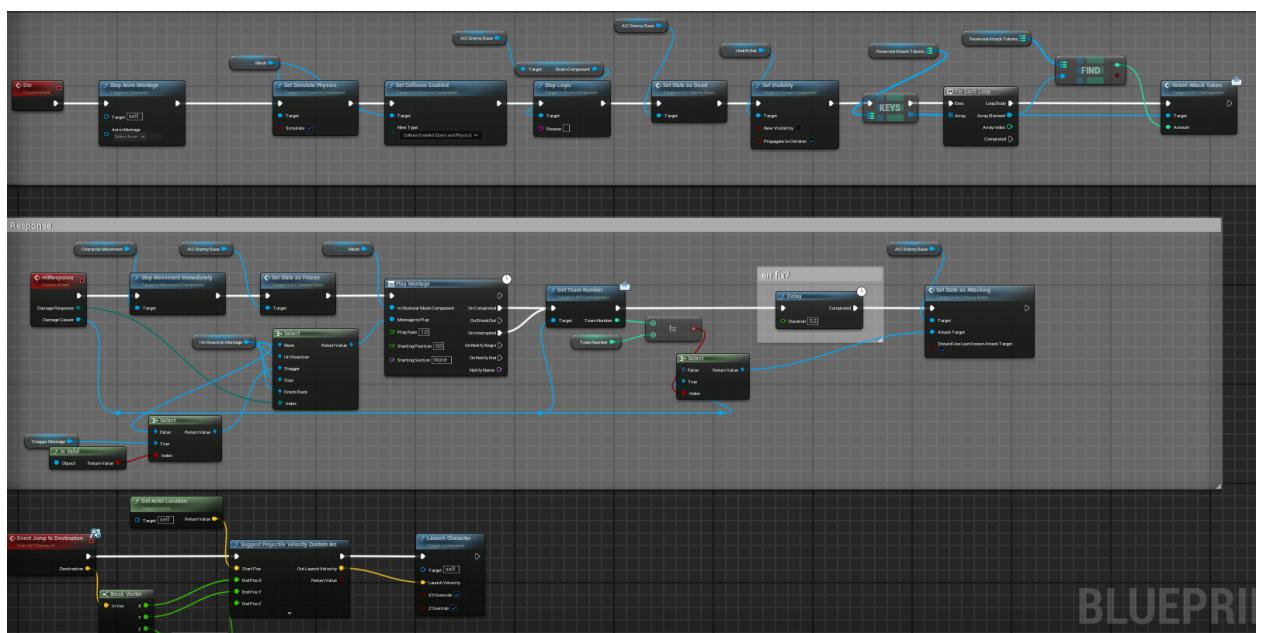
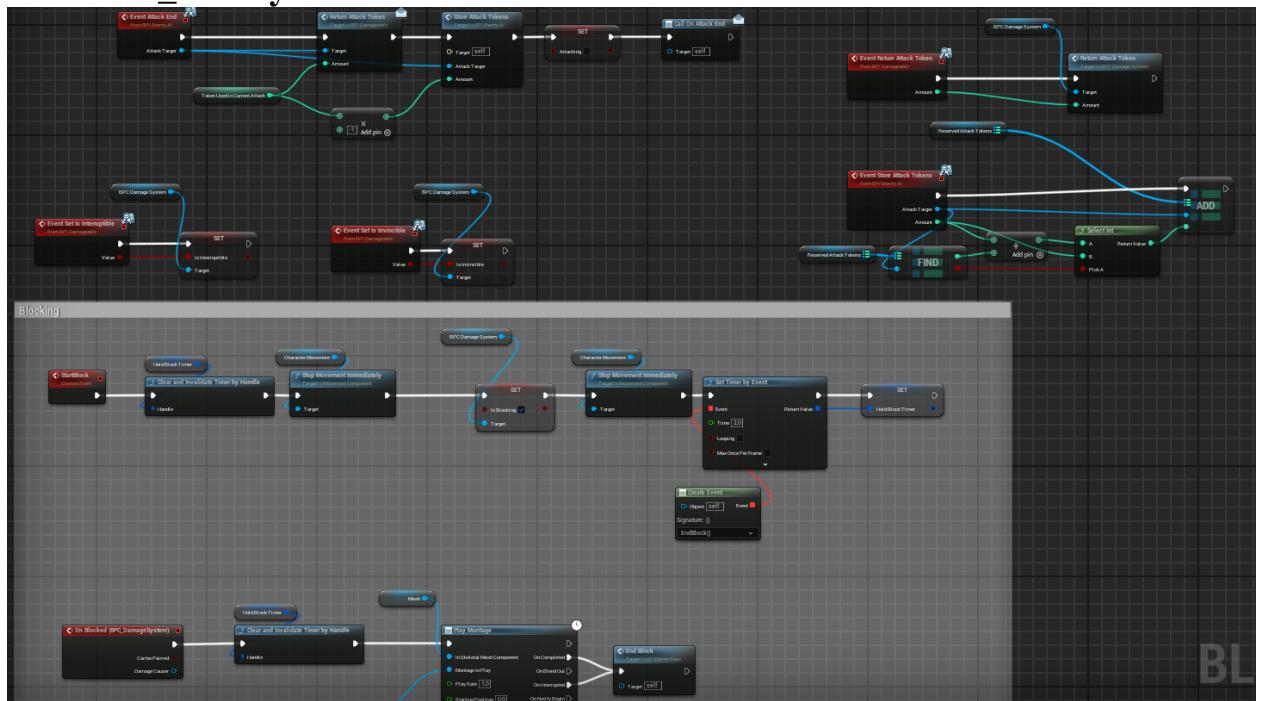


WIDGET BLUEPRINT

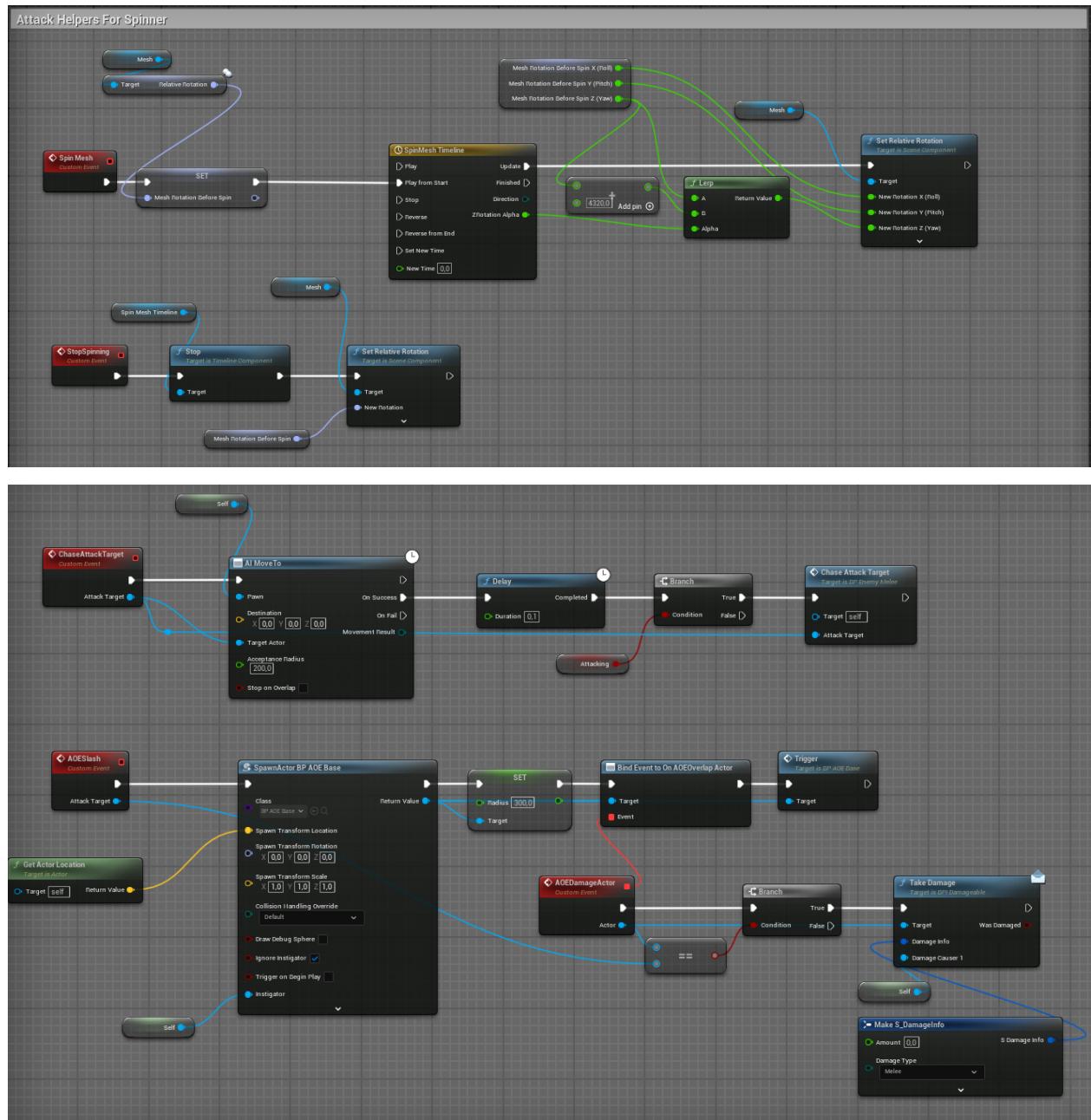


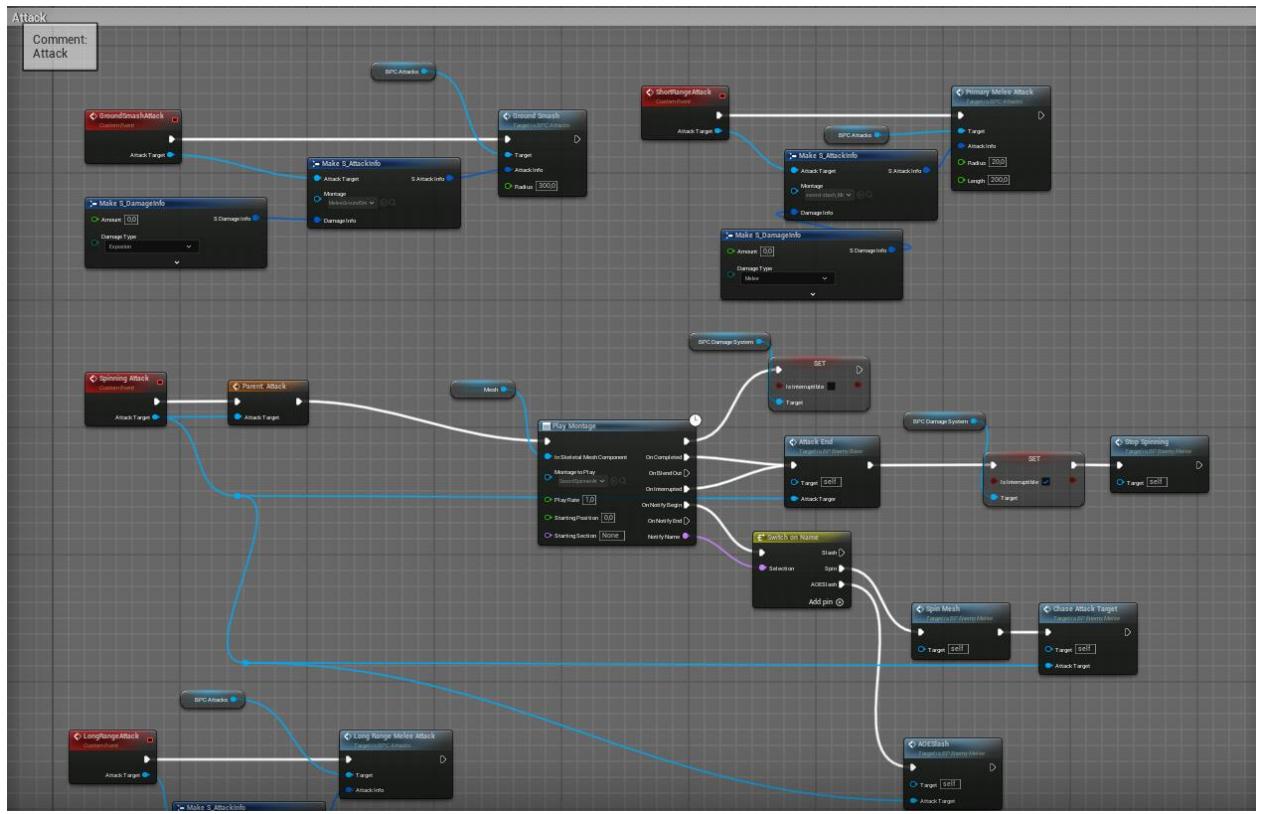
Реалізація функціональної задачі забезпечення функціонування штучного інтелекту неігрових персонажів гри:

Файл BP_EnemyBase

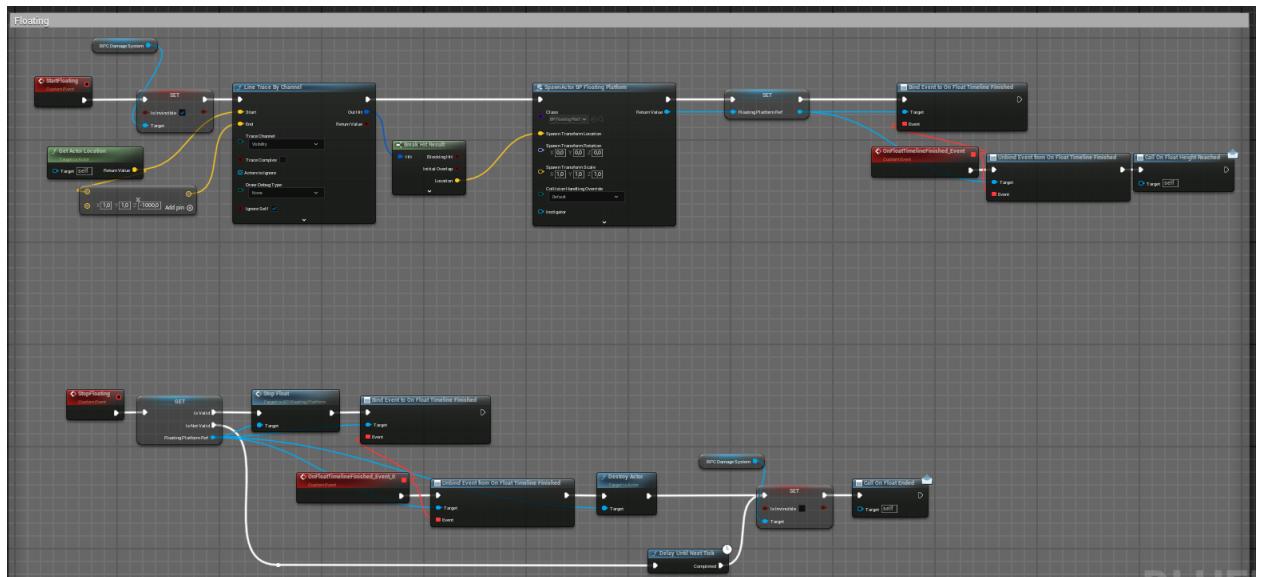
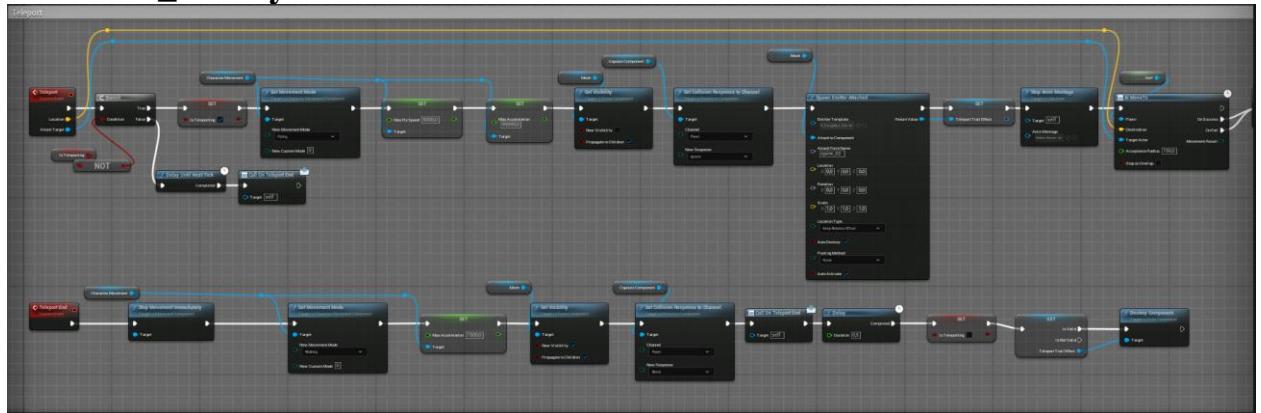


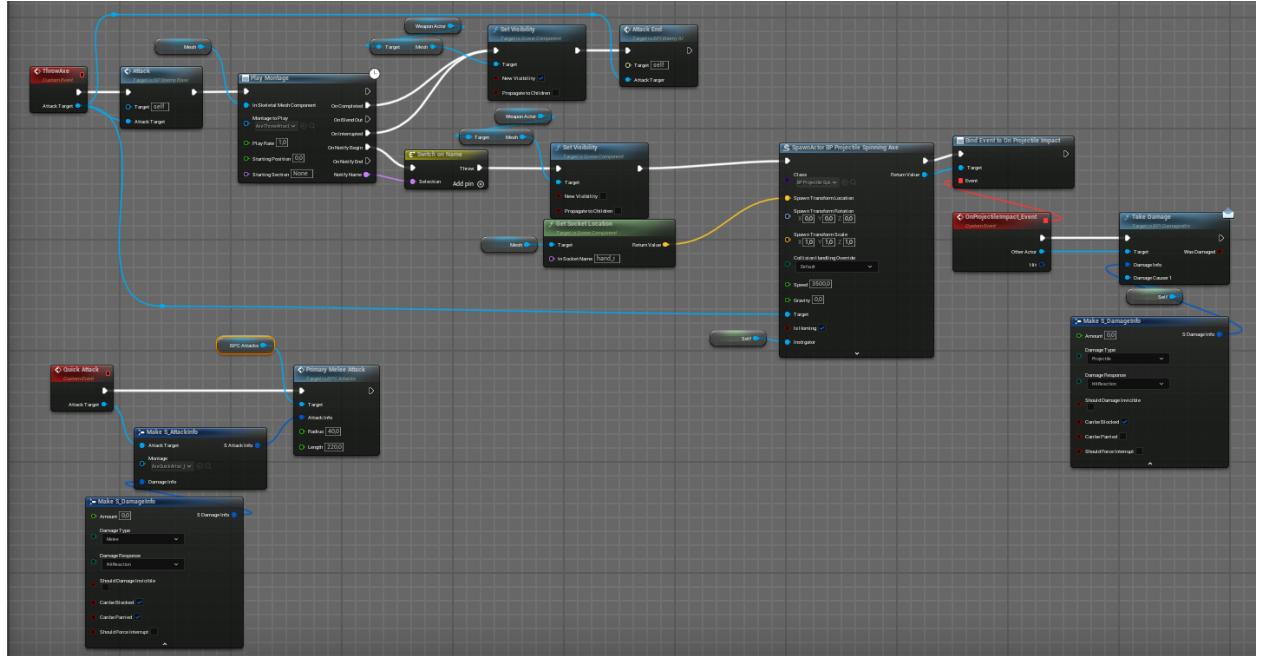
Файл BP_EnemyMelee



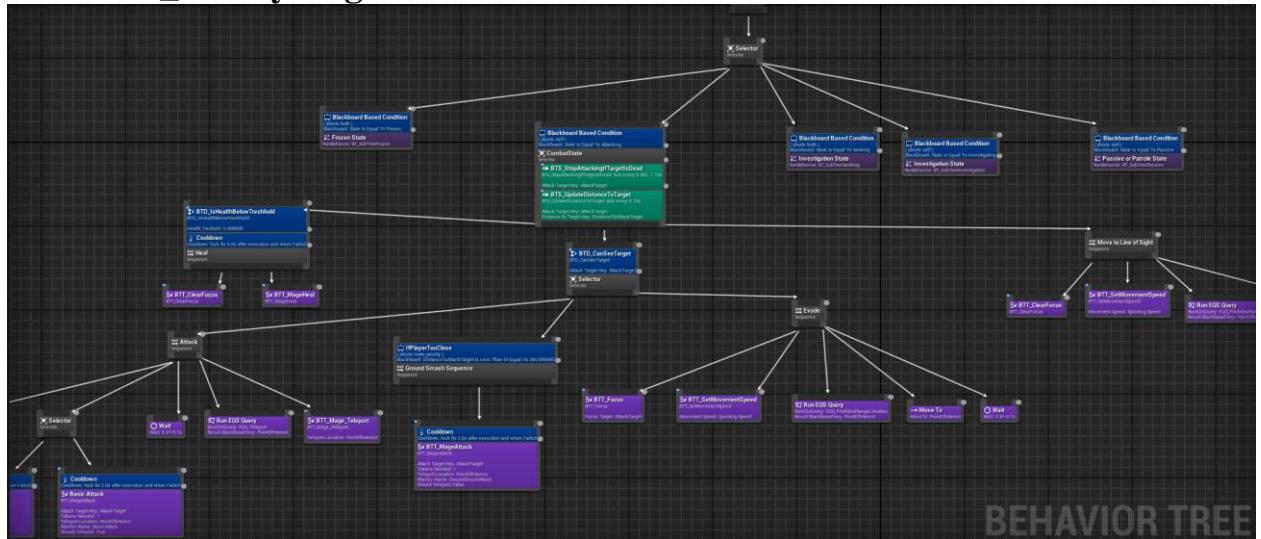


Файл BP_EnemyBoss

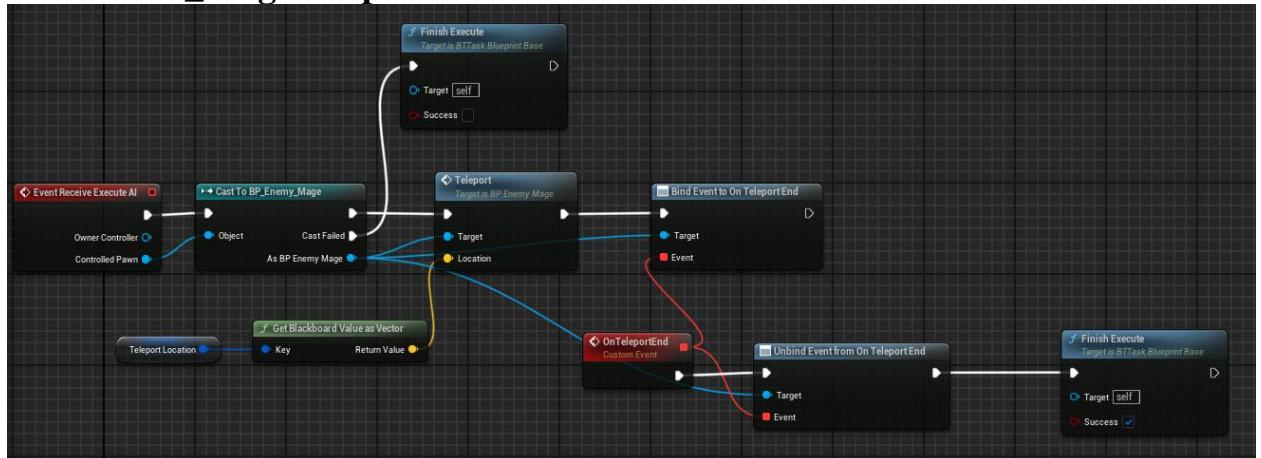




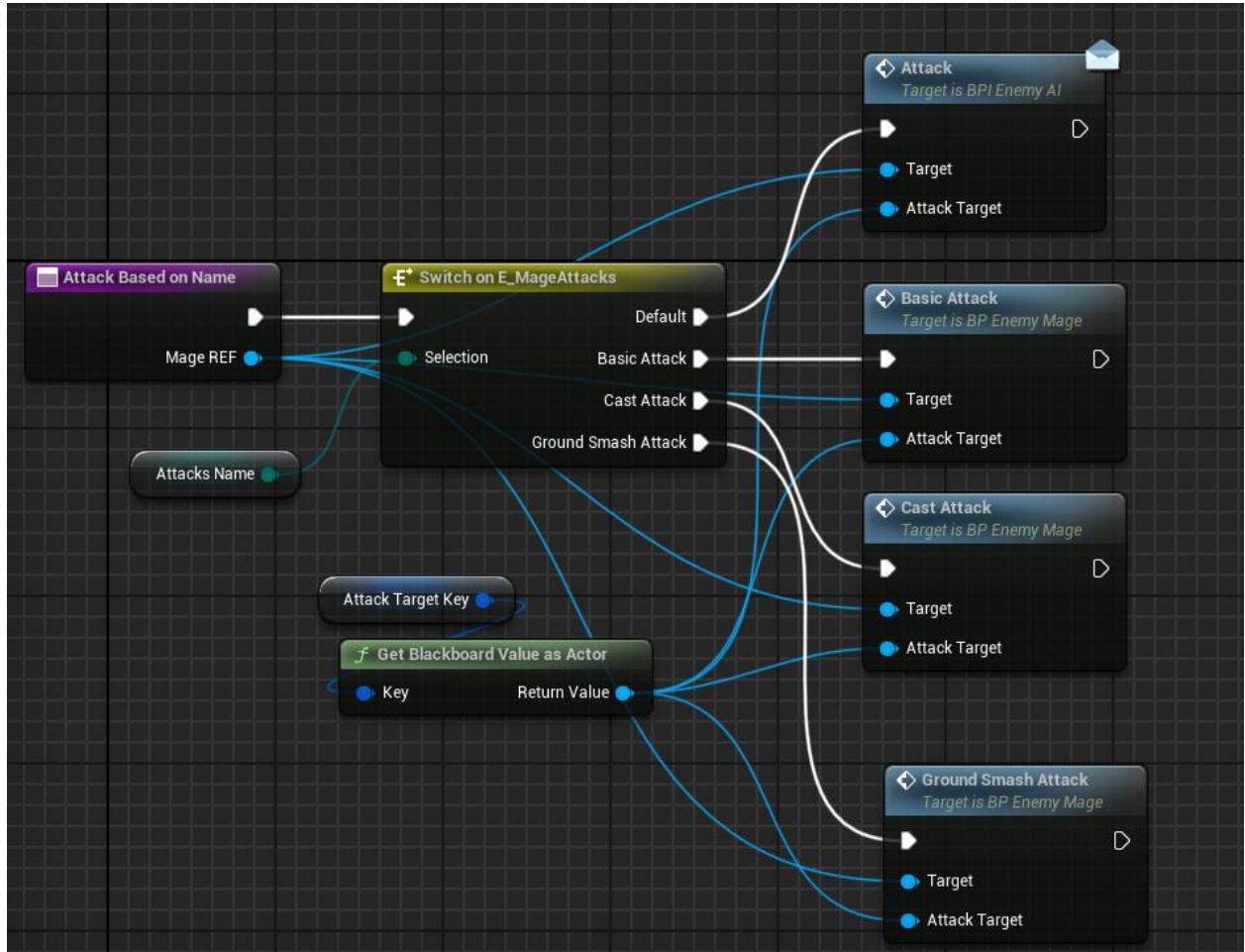
Файл BT_EnemyMage



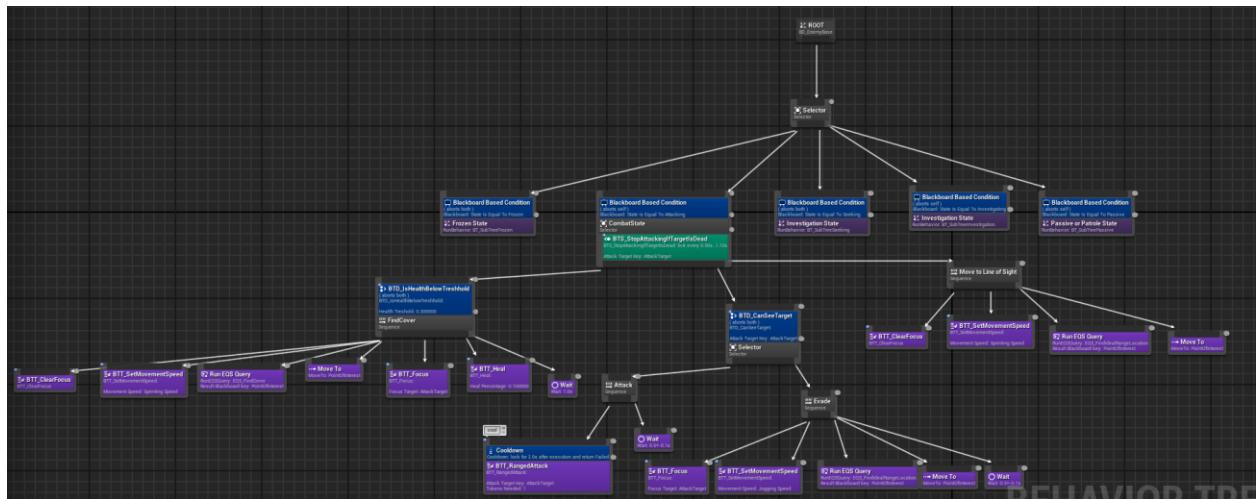
Файл BTT_MageTeleport



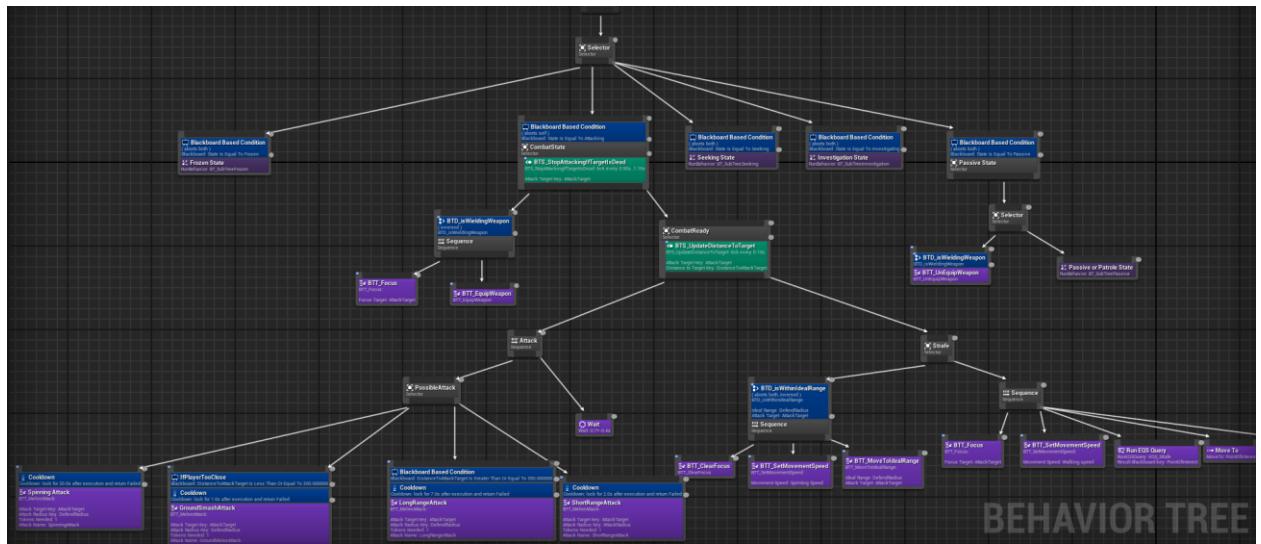
Файл BTT_MageAttack



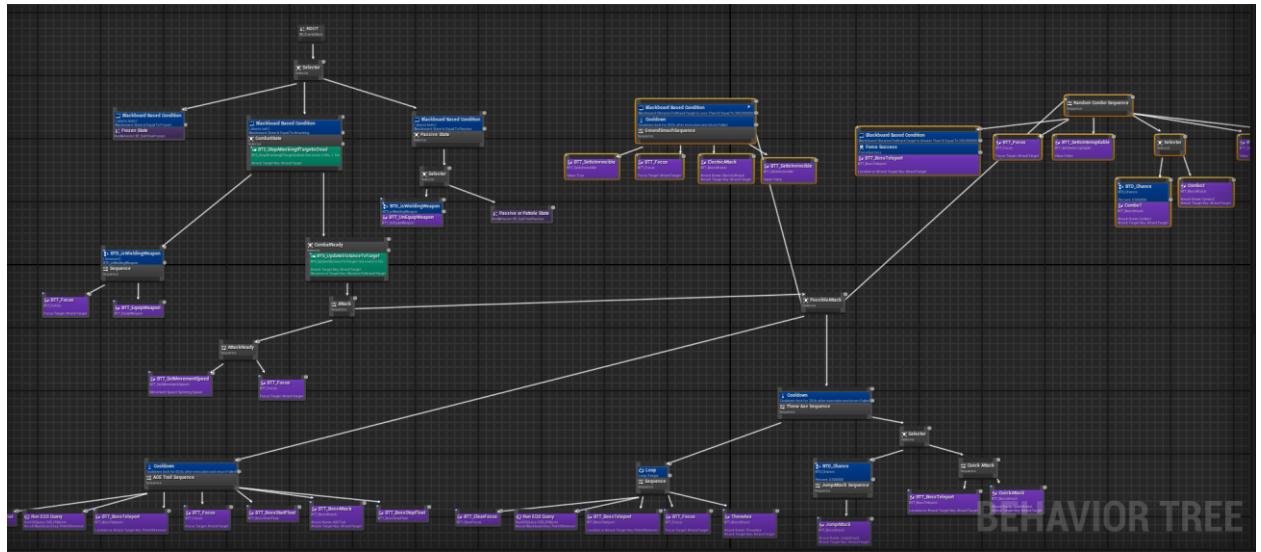
Файл BT_EnemyRange



Файл BT_EnemyMelee



Файл BT_EnemyBoss



Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРИКОВ
“___” _____ 2025 р.

**ГРА У ЖАНРІ ROLE-PLAYING-GAME(RPG) НА UNREAL ENGINE 3
ВИКОРИСТАННЯМ ШІ
Програма та методика тестування
КПІ.ІП-1107.045440.04.51**

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Максим ГОЛОВЧЕНКО

Нормоконтроль:

_____ Максим ГОЛОВЧЕНКО

Виконавець:

_____ Олександр ГОЛОВНЯ

Київ – 2025

ЗМІСТ

1	ОБ'ЄКТ ВИПРОБУВАНЬ.....	3
2	МЕТА ТЕСТУВАННЯ	4
3	МЕТОДИ ТЕСТУВАННЯ.....	5
4	ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ	6

1 ОБ'ЄКТ ВИПРОБУВАНЬ

Об'єктом випробування є мануальне тестування гри у жанрі RPG, розробленої на ігровому рушії Unreal Engine 5. Моею метою є перевірка ігрових механік, загального геймплею та візуальної частини гри. Під час тестування, акцентовано увагу на функціонування головного меню гри, безпосередньо ігрові механіки притаманні жанру, функціонування ігрового інтерфейсу, взаємодії з інвентарем, функціонування штучного інтелекту ворогів та взаємодії з колізією об'єктів.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

- перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;
- знаходження проблем, помилок і недоліків з метою їх усунення;
- перевірка зручності графічного інтерфейсу;

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

- функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;
- системне тестування – перевіряється усе програмне забезпечення в цілому;
- мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Під час проведення тестування будуть використовуватись допоміжні, вбудовані в Unreal Engine засоби, як-от: Output Log, Unreal Insights, Stat Commands(stat fps, stat unit, stat memory, stat ai, тощо), Gameplay Debugger, Crash Reporter.

Порядок проведення тестування буде наступним:

- мануальне тестування функціональності;
- тестування продуктивності та стабільності;
- тестування поведінки ШІ та внутрішніх систем;
- документування результатів.

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРИКОВ

“___” _____ 2025 р.

ГРА У ЖАНРІ ROLE-PLAYING-GAME(RPG) НА UNREAL ENGINE 3

ВИКОРИСТАННЯМ ШІ

Керівництво користувача

КП.ІП-1107.045440.05.34

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Максим ГОЛОВЧЕНКО

Нормоконтроль:

_____ Максим ГОЛОВЧЕНКО

Виконавець:

_____ Олександр ГОЛОВНЯ

Київ – 2025

ЗМІСТ

1	ПРИЗНАЧЕННЯ ПРОГРАМИ	3
2	ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ.....	4
2.1	Системні вимоги для коректної роботи	4
2.2	Завантаження застосунку	4
2.3	Перевірка коректної роботи	5
3	ВИКОНАННЯ ПРОГРАМИ.....	6

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

Гра у жанрі RPG – це програмне забезпечення, реалізоване з використанням ігрового рушія Unreal Engine 5, дозволяє гравцеві керувати ігровим персонажем та ігровими механіками гри.

Розробка призначена для забезпечення функціонування цих механік, що дозволяють реалізацію цікавого штучного інтелекту для використання у галузі розваг для широкого кола користувачів з метою підвищення якості ігрового досвіду.

Кінцева збірка гри створена за допомогою інструментів пакування Unreal Engine 5 під платформу Windows . Усі необхідні ресурси (моделі, текстури, звуки, анімації) включено у збірку.

Формат кінцевої збірки у вигляді інсталяційного каталогу із виконуваним файлом та супутніми файлами. Гра не потребує додаткового встановлення, достатньо розпакувати архів і запустити виконуваний файл.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Мінімальна конфігурація технічних засобів:

- операційна система: Windows 10 (64-bit);
- процесор: Intel Core i3-8100 або AMD Ryzen 3 1200;
- оперативна пам'ять: 8 GB RAM;
- відеокарта: NVIDIA GeForce GTX 750 Ti або AMD Radeon R7 360;
- жорсткий диск: 10–15 GB вільного простору;
- DirectX: Версія 11;
- інше: Клавіатура та миша, доступ до інтернету не потребується;

Рекомендована конфігурація технічних засобів:

- операційна система: Windows 11 (64-bit);
- процесор: AMD Ryzen 7 3750H;
- оперативна пам'ять: 16 GB RAM;
- відеокарта: NVIDIA GeForce RTX 2060;
- жорсткий диск: SSD із мінімум 15 GB вільного простору;
- DirectX: Версія 12;
- інше: Клавіатура та миша, доступ до інтернету не потребується;
- роздільна здатність дисплею: 1920×1080 або вища (Full HD).

2.2 Завантаження застосунку

Для отримання доступу до програмного забезпечення, включаючи фінальну збірку гри, виконуваний файл, інсталяційну версію та програмні файли, слід завантажити архів із офіційного репозиторію або локального сховища, розпакувати архів у бажану директорію та запустити виконувальний файл.

2.3 Перевірка коректної роботи

Щоб переконатися у правильності встановлення програмного забезпечення, слід запустити та переконатись, що гра відкривається без помилок або попереджень. Переконатись, що всі основні компоненти інтерфейсу завантажуються(головне меню, кнопки, початок гри).

3 ВИКОНАННЯ ПРОГРАМИ

Після запуску гри завантажується головне меню, де гравцю доступні опції розпочати нову гру, продовжити збережену гру (за наявності збереження), змінити налаштування гри або ж вихід із гри(рисунок 3.1).

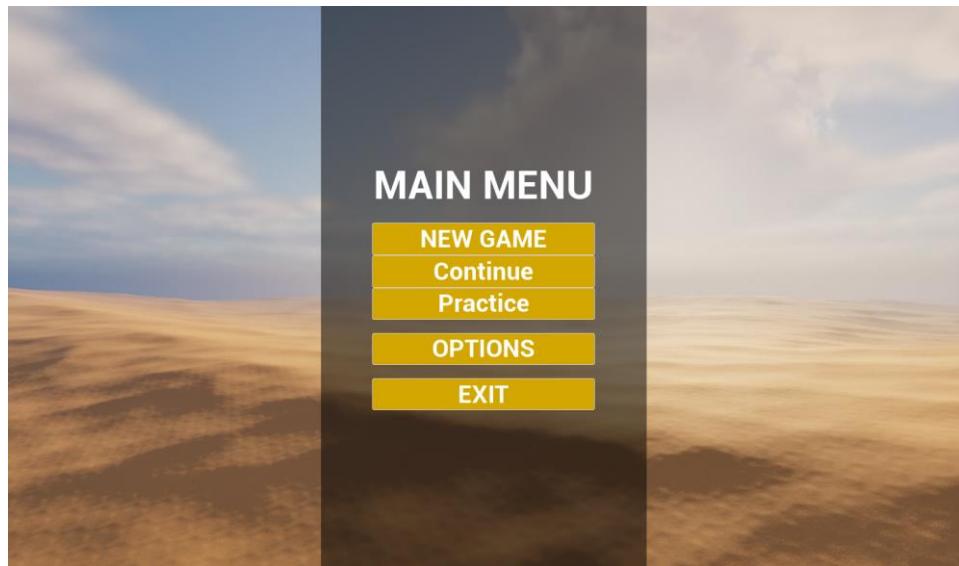


Рисунок 3.1 – Головне меню гри.

Після натискання на New Game розпочинається вступна сцена гри, після якої гравець отримує контроль над персонажем. Гравець вступає в ігровий світ, де на початку його зустріне базовий туторіал як керувати персонажем(рисунок 3.2).



Рисунок 3.2 – Ігровий світ.

Під час гри, доступні основні механіки, які забезпечують повноцінний ігровий процес та взаємодію гравця з ігровим світом.

Гравець може отримати завдання (квести) від неігрових персонажів (NPC) або через інтерактивні об'єкти в ігровому світі. Їх виконання надає гравцю предмети та збільшує характеристики. Гравець має можливість спілкуватися з NPC, що можуть бути торговцями, наставниками або супутниками(рисунок 3.3).

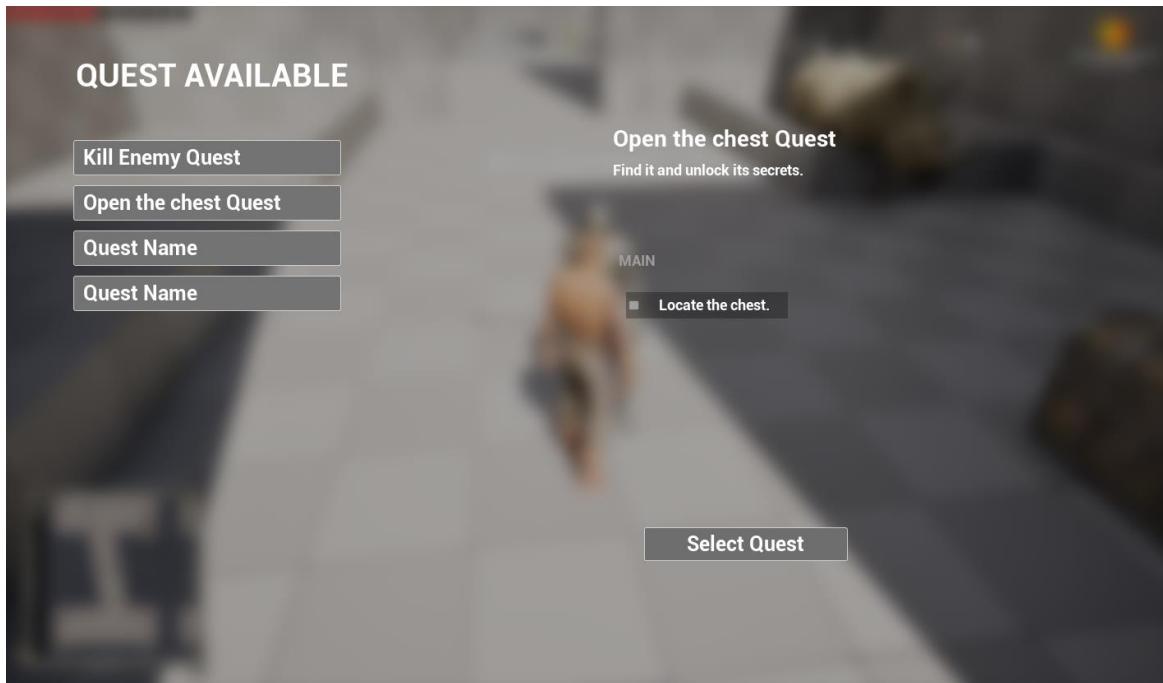


Рисунок 3.3 – Квестова та діалогова системи.

Під час дослідження світу гравець може знаходити та збирати різноманітні предмети: ресурси, спорядження, їжу та інші об'єкти, які зберігаються в інвентарі й можуть бути використані пізніше(рисунок 3.4).



Рисунок 3.4 – Взаємодія з об’єктами.

Інвентар дозволяє гравцеві переглядати, використовувати, екіпірувати або відкидати предмети. Інтерфейс інвентарю реалізований з урахуванням зручності користувача та підтримує категоризацію предметів: зброя, броня, ресурси(рисунок 3.5).

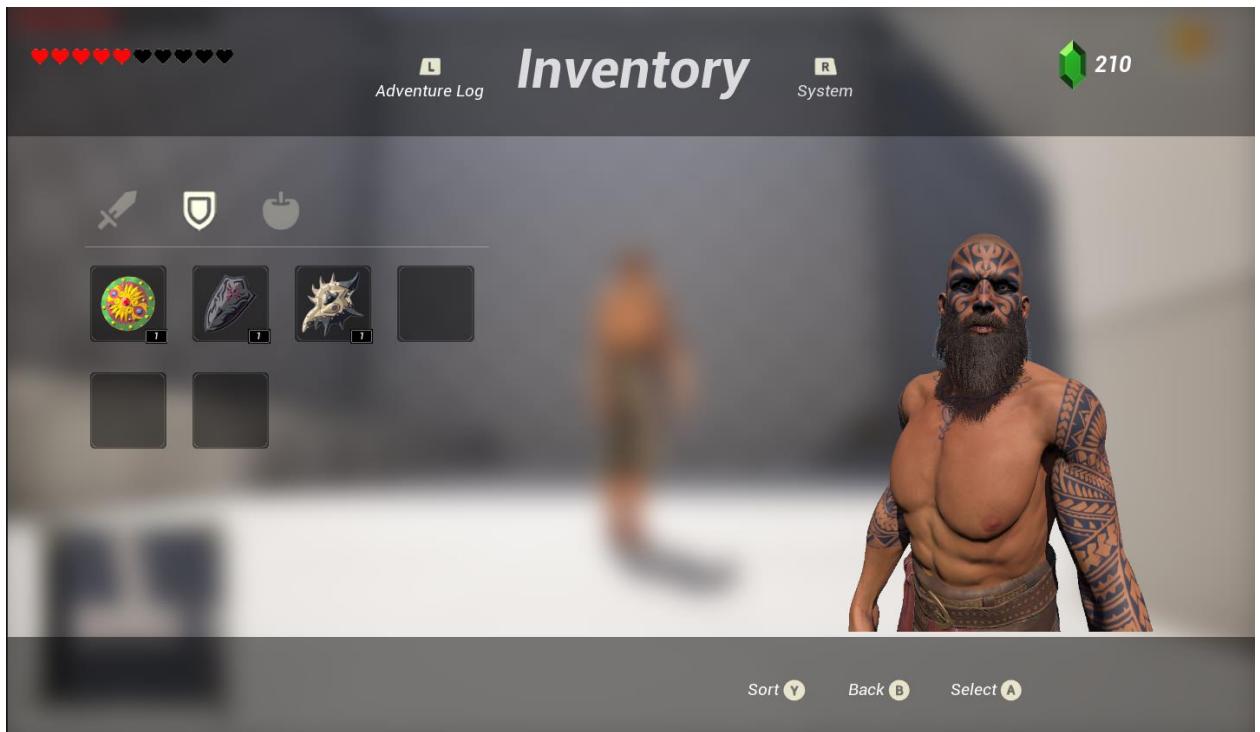


Рисунок 3.5 – Взаємодія з інвентарем.

Досліджуючи світ гравець може брати участь у бойових зіткненнях, використовуючи зброю ближнього або дальнього бою. Система бою враховує характеристики персонажа, тип зброї, швидкість атаки та ворожу поведінку (AI). Передбачено декілька видів ворожих NPC, такі як: Range enemy, Melee enemy, Mage enemy та Boss(рисунок 3.6).



Рисунок 3.6 – Типи ворожих NPC.

Під час гри, гравець може поставити гру на паузу, або ж вийти до головного меню обравши пункт Main Menu(рисунок 3.7). Для завершення гри обрати у головному меню пункт Exit, після чого програма завершиться свою роботу. Ігровий прогрес може зберігатися гравцем вручну за допомогою пункту Save Game в меню паузи.



Рисунок 3.7 – Меню паузи та виходу у головне меню.

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРИКОВ

“___” _____ 2025 р.

ГРА У ЖАНРІ ROLE-PLAYING-GAME(RPG) НА UNREAL ENGINE 3

ВИКОРИСТАННЯМ ШІ

Графічний матеріал

КП.ІП-1107.045440.06.99

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Максим ГОЛОВЧЕНКО

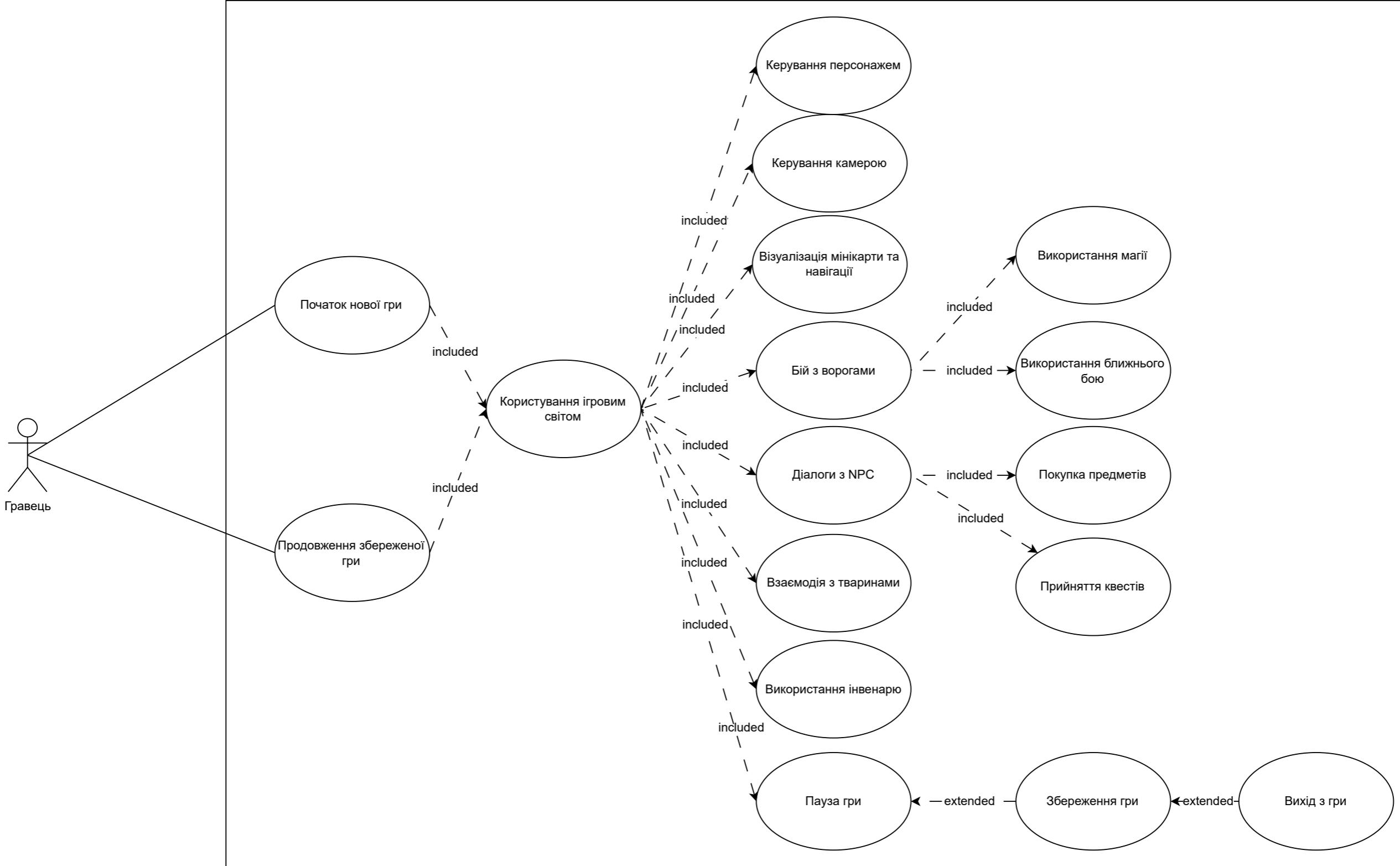
Нормоконтроль:

_____ Максим ГОЛОВЧЕНКО

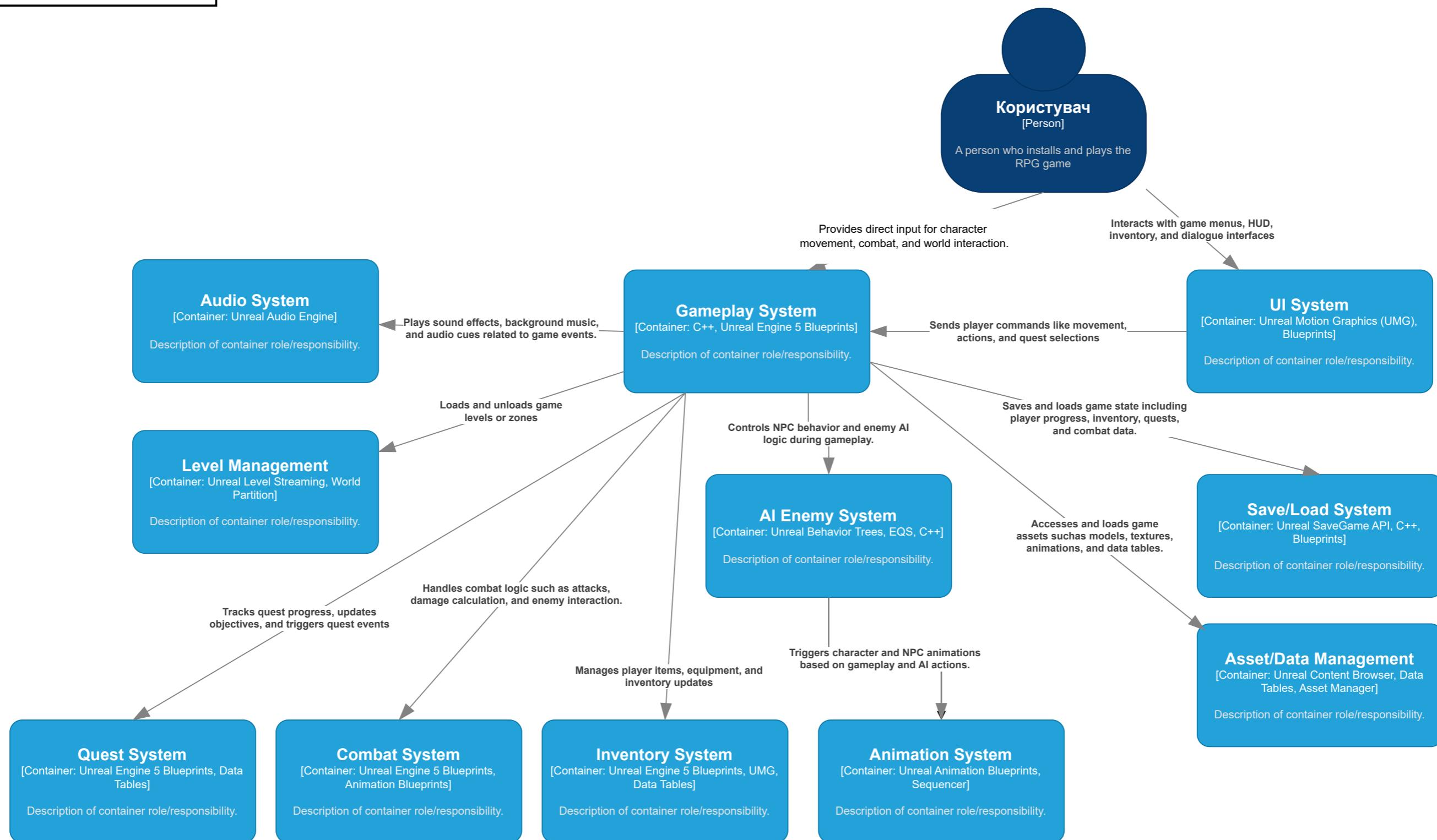
Виконавець:

_____ Олександр ГОЛОВНЯ

Київ – 2025



Літера	Маса	Масштаб
Діаграма варіантів використань		
Зм.	Арк.	№ документа
Розробив	Головня О.Р.	Підпис
Перевірив	Головченко М.М	Дата
Т. контр.		
Н. контр.	Головченко М.М	
Затвердив	Жаріков Е.В.	
Гра у жанрі Role-Playing-Game(RPG) на Unreal Engine з використанням ШІ		
КПІ ім. Ігоря Сікорського Кафедра ІПІ гр. ІП-11		

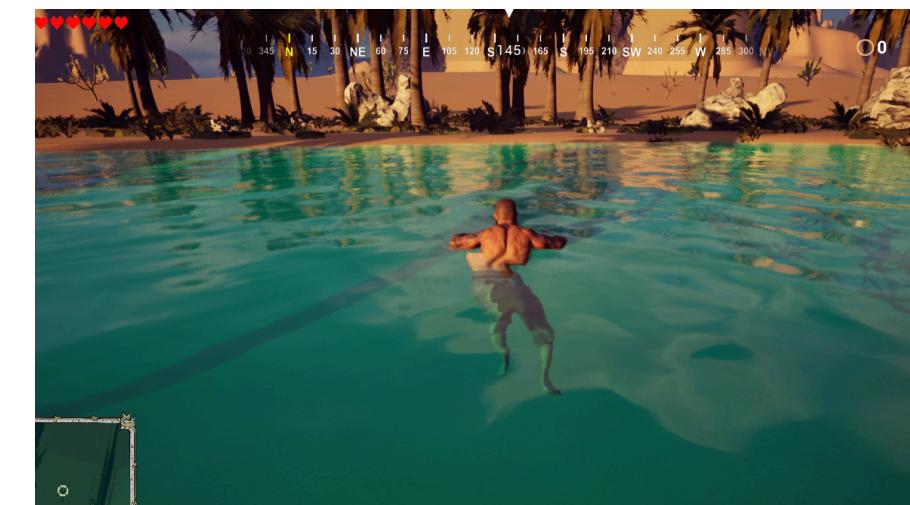
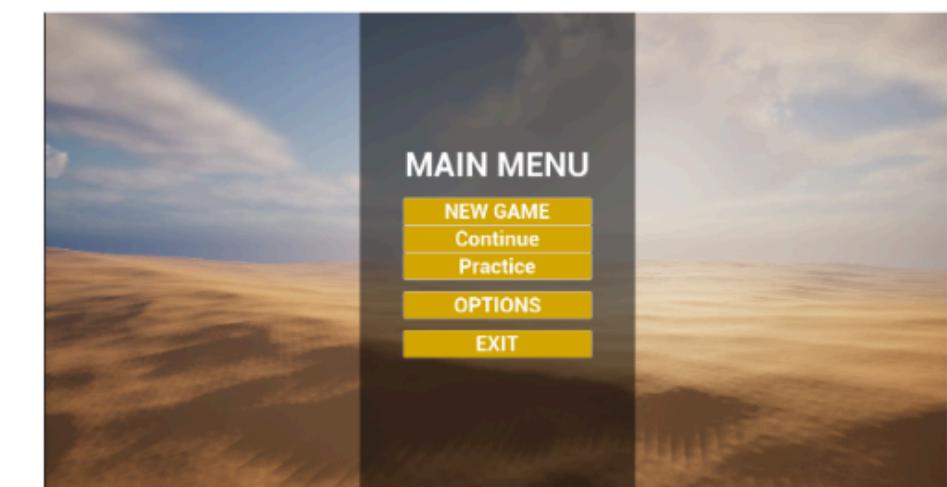
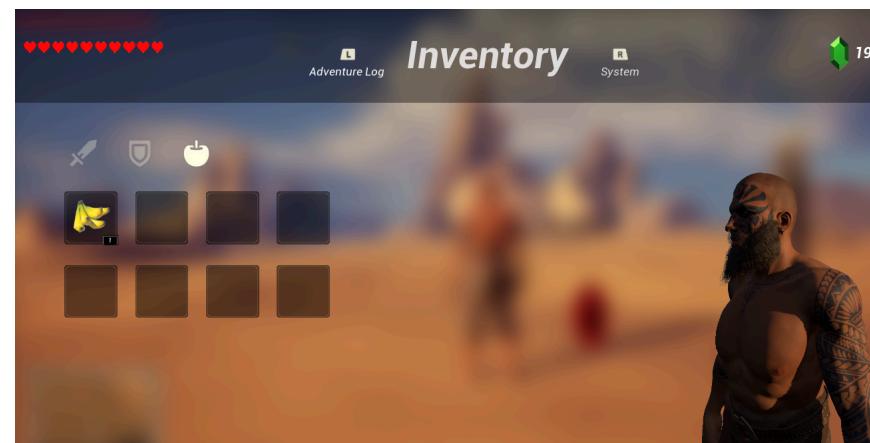


[Containers] Diagram for RPG game

This container diagram shows the main architectural components of the RPG game developed in Unreal Engine 5.

					КПІ.ІП-1107.045440.06.99.ССМ	
Зм.	Арк.	№ документа	Підпис	Дата	Діаграма контейнерів	
Розробив	Головня О.Р.					
Перевірив	Головченко М.М					
Т. контр.						
Н. контр.	Головченко М.М				Гра у жанрі Role-Playing-Game(RPG) на Unreal Engine з використанням ШІ	
Затвердив	Жариков Е.В.				КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-11	
					Літера	Маса
					Аркуш	Аркушів

KPI.IP-1107.045440.06.99.KE



Зм.	Арк.	№ документа	Підпис	Дата
Розробив		Головня О.Р.		
Перевірив		Головченко М.М.		
Т. контр.				
Н. контр.		Головченко М.М.		
Затвердив		Жаріков Е.В.		

KPI.IP-1107.045440.06.99.KE

Скріни програми

Гра у жанрі Role-Playing-Game(RPG) на Unreal Engine з використанням ШІ

Літера	Маса	Масштаб
Аркуш	Аркушів	

КПІ ім. Ігоря Сікорського
Кафедра ІПІ
гр. ІП-11