Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційна робота ступеня бакалавра
(бакалавра, спеціаліста, магістра)

Здобувача Сергіха Ярослава Віталійовича
(ПІБ)

академічної групи 126-21-1
(шифр)

спеціальності 126 «Інформаційні системи та технології»
(код і назва спеціальності)

за освітньо-професійною програмою «Інформаційні системи та технології»
(офіційна назва)

на тему «Розробка ігрового застосунку з використанням штучного інтелекту на ігровому рушії Unreal Engine 4»
(назва за наказом ректора)

Керівник	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	доц. Соколова Н. О.			
розділів:				

Рецензент	доц. Клименко А.В.			

Нормоконтролер	проф.Коротенко Г.М.			

Дніпро
2025

ЗАТВЕРДЖЕНО:
завідувач кафедри

інформаційних технологій
та комп'ютерної інженерії
(повна назва)

_____ Гнатушенко В.В
(підпис) (прізвище)

«___» _____ 2025 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня ___бакалавра___
(бакалавра, спеціаліста, магістра)

здобувачу Сергіх Я.В. **академічної групи** 126-21-1
(прізвище та ініціали) (шифр)

спеціальності 126 «Інформаційні системи та технології»

за освітньо-професійною програмою «Інформаційні системи та технології»

на тему «Розробка ігрового застосунку з використанням штучного інтелекту на ігровому рушії Unreal Engine 4»

затверджену наказом ректора НТУ «Дніпровська політехніка» від 05.05.2025 № 336-с

Розділ	Зміст	Термін виконання
Розділ 1	Дослідження предметної області та постановка задачі розробки ігрового додатку	03.02.2025 – 30.04.2025
Розділ 2	Проєктні рішення	03.02.2025 – 03.06.2025

Завдання видано _____ Соколова Н. О.
(підпис керівника) (прізвище, ініціали)

Дата видачі 03.02.2025

Дата подання до екзаменаційної комісії _____

Прийнято до виконання _____ _____
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 97 с., 82 рис., 12 табл., 7 додатків, 14 джерел. ІГРОВИЙ РУШІЙ, РОЛЬОВА ГРА, BLUEPRINTS, UNREAL ENGINE, JRPG, ШТУЧНИЙ ІНТЕЛЕКТ, ІНТЕРАКТИВНІСТЬ.

Об'єкт кваліфікаційної роботи: JRPG-гра з елементами штучного інтелекту.

Предмет кваліфікаційної роботи: методи реалізації поведінки персонажів на основі ШІ в Unreal Engine.

Мета роботи: створення ігрового застосунку жанру JRPG з інтеграцією ШІ для моделювання поведінки персонажів.

У вступі визначено актуальність розробки інтерактивних ігор із ШІ, сформульовано мету, завдання, об'єкт і предмет розробки.

У першому розділі розглянуто розвиток JRPG-жанру та ігрових рушіїв, проаналізовано архітектуру Unreal Engine і використання Blueprints для реалізації ігрової логіки.

У другому розділі наведена проектна складова проєкту: логіку NPC, бойову систему, ШІ-алгоритми та побудову інтерфейсу з урахуванням інтерактивності.

У висновках узагальнено результати, подано пропозиції щодо розвитку системи та розширення поведінкових сценаріїв.

Практичне значення: робота створює основу для ігор із гнучкою поведінкою персонажів, що підвищує занурення користувача. Рішення придатне для сюжетно-орієнтованих RPG-проєктів.

Розроблене програмне забезпечення може бути запроваджено у комерційні проєкти, орієнтовані на сюжетно-орієнтовані рольові ігри.

ABSTRACT

Explanatory note: 97 pages, 82 figures, 12 tables, 7 appendices, 14 sources.

GAME ENGINE, ROLE-PLAYING GAME, BLUEPRINTS, UNREAL ENGINE, JRPG, ARTIFICIAL INTELLIGENCE, INTERACTIVITY.

The object of the qualification work: a JRPG game with elements of artificial intelligence.

The subject of the work: the methods of implementing character behavior based on AI in Unreal Engine.

The goal of the work: to create a JRPG-style game application with integrated AI to simulate character behavior.

The introduction outlines the relevance of developing interactive games with AI, defines the goal, objectives, object, and subject of the work.

The first chapter examines the development of the JRPG genre and game engines, analyzes the architecture of Unreal Engine, and explores the use of Blueprints for game logic implementation.

The second chapter presents the project component: NPC behavior logic, combat system, AI algorithms, and the creation of a user interface with interactivity in mind.

The conclusion summarizes the results and provides suggestions for system development and behavioral scenario expansion.

Practical significance: the work lays the foundation for games with flexible character behavior, enhancing user immersion. The solution is suitable for story-driven RPG projects.

The developed software can be implemented in commercial projects focused on narrative-based role-playing games.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

2D – двовимірна графіка

3D – тривимірна графіка

AI – Artificial Intelligence

JRPG (Japanese Role-Playing Game) – японська рольова гра

BLUEPRINT (BP) – візуальна система програмування в Unreal Engine

NPC (Non-Playable Character) – неігровий персонаж

UNREAL ENGINE – ігровий рушій

WIDGET (WBP) – елемент інтерфейсу користувача в Unreal Engine

ВСТУП

Розробка відеоігор почалася з простих аркадних автоматів близько 1970-х років і стрімко розвивалася разом із технологічним прогресом комп'ютерної індустрії. Якщо спочатку ігри були доволі примітивними – як у графічному оформленні, так і в складності геймплею, – то вже до початку XXI століття вони еволюціонували у складні інтерактивні середовища з багаторівневими сюжетами, реалістичною фізикою, високоякісною візуалізацією та розвиненою системою штучного інтелекту. У більшості сучасних ігор штучний інтелект (ШІ) виконує надзвичайно важливу роль: він забезпечує природну поведінку неігрових персонажів (NPC), створює відчуття живого, динамічного світу та робить гру більш захопливою й варіативною.

Ігрові проєкти приваблюють суспільство передусім своїми унікальними сюжетами, інтерактивністю, механікою вибору та можливістю приміряти на себе різні ролі в ігровому всесвіті. Одним із таких популярних жанрів є JRPG (Japanese Role-Playing Game), що вирізняється глибокою оповіддю, покроковою боєвою системою, поступовим розвитком персонажів через систему рівнів, навичок і здібностей, а також яскравим і впізнаваним візуальним стилем. Жанр японської рольової гри вдало поєднує елементи дослідження світу, стратегічного планування боїв та емоційно забарвлених сюжетних ліній, що забезпечує високий рівень занурення і приваблює широку аудиторію гравців.

Швидке зростання індустрії відеоігор стало можливим завдяки поєднанню розвитку обчислювальних потужностей апаратного забезпечення та удосконалення методів програмування. Сучасні ігрові рушії, такі як Unreal Engine, надають розробникам потужний інструментарій для створення як фотореалістичної графіки, так і складних логічних систем – зокрема, штучного інтелекту, що включає поведінкові дерева, сенсорні модулі, аналіз навколишнього середовища та реакцію NPC на дії гравця в режимі реального часу.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ ІГРОВОГО ДОДАТКУ

1.1 Аналіз існуючих ігрових застосунків з використанням штучного інтелекту

Ідеологія жанру JRPG (Japanese Role-Playing Game) бере свій початок з 1980-х років, коли японські розробники почали творити власне бачення рольових ігор, натхненних західними проєктами на кшталт Ultima та Wizardry. Однією з перших помітних ігор у жанрі JRPG стала Dragon Quest, випущена у 1986 році компанією Enix (рис. 1.1). Її поєднання покрокових боїв, сюжетної глибини та характерного візуального стилю заклало підґрунтя для подальшого розвитку жанру. Пізніше, у 1987 році, з'явилася перша гра серії Final Fantasy від Square, яка з часом стала глобальним явищем та значно вплинула на популярність JRPG по всьому світі.

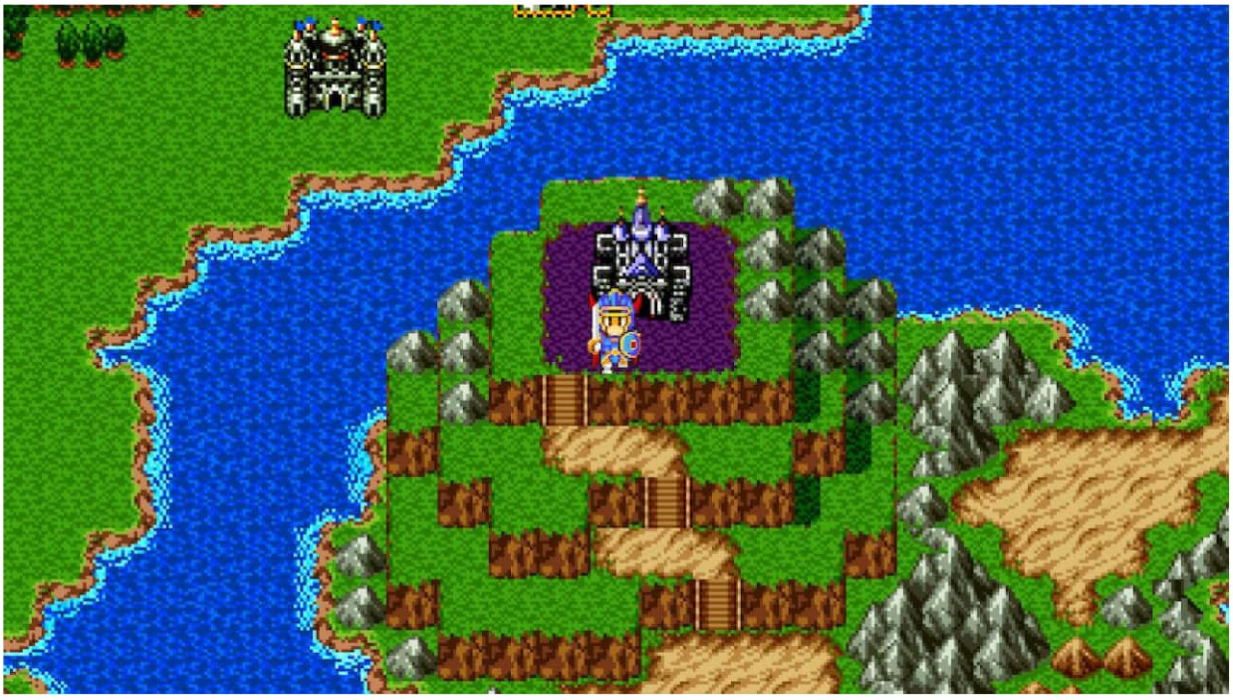


Рисунок 1.1 – Гра Dragon Quest, створена в 1986 році

На відміну від західних RPG, які часто акцентують увагу на відкритих світах та виборі гравця, JRPG здебільшого зосереджуються на заздалегідь прописаному сценарії, глибоких взаємозв'язках між персонажами, покроковій або гібридній бойовій системі та емоційній залученості гравця. Ці ігри часто досліджують теми дружби, боротьби зі злом, особистого зростання, і мають яскравий художній стиль.

У 2024 році жанр японських рольових ігор (JRPG) продемонстрував суттєве збільшення популярності на світовому ринку. Одним із ключових чинників цього зростання стала поява ремейків класичних ігор, які поєднують ностальгію з сучасними технологіями. Зокрема, гра Persona 3 Reload, випущена 2 лютого 2024 року, стала грою, що найшвидше продалася в історії компанії Atlus, досягнувши позначки в 1 мільйон проданих копій всього за тиждень . Цей успіх підкреслює відродження інтересу до жанру JRPG серед гравців усього світу [1].

Варто відзначити, що серія Persona, яка почалася з гри Revelations: Persona у 1996 році (рис. 1.2), поступово здобула міжнародне визнання. Особливої популярності вона набула після виходу Persona 5, що сприяло зростанню продажів наступних ігор серії. Загалом, станом на кінець 2023 року, серія Persona продала понад 17,7 мільйона копій по всьому світу [2].

Рисунок 1.2 – Гра Revelations: Persona, створена в 1996 році

Ці дані свідчать про стабільний інтерес до жанру JRPG, зокрема до ігор, що пропонують глибокий сюжет, стратегічний геймплей та емоційно насичені історії. Зростаюча популярність жанру також відображає тенденцію до відродження класичних ігор з оновленою графікою та механікою, що приваблює як нових гравців, так і давніх шанувальників жанру.

З розвитком технологій та ігрових рушіїв, особливо після 2010-х років, JRPG почали активно запроваджувати елементи штучного інтелекту. AI-системи стали відповідальними не лише за базову поведінку супротивників, а й за адаптивність бою, патрулювання, реакції на дії гравця, взаємодію з оточенням та навіть процедурну генерацію сценаріїв. Це підвищує занурення гравця у світ гри та дозволяє створювати гнучкіші й динамічніші геймплейні ситуації.

У сучасних рольових іграх жанру JRPG все більше уваги надається якості поведінки неігрових персонажів (NPC), зокрема ворожих, що мають безпосередній вплив на ігровий рівень складності, динаміку бою та занурення гравця у світ гри. Відомі серії ігор , такі як Shin Megami Tensei та Persona, слугують яскравим прикладом використання адаптивного штучного інтелекту, який реалізує як патрулювання простору, так і складні бойові рішення.

У грі Persona 5 ворожі NPC патрулюють коридори в палацах, орієнтуючись на зір – коли гравець потрапляє в поле зору ворога, той розпочинає переслідування за гравцем (рис. 1.3) [3]. При цьому реалізовано систему попередження: ворог має індикатор уваги, що збільшується при підозрілих діях гравця.

Рисунок 1.3 – Приклад патрулювання ворога у грі Persona 5 Royal

У Persona 4 та Persona 3 вороги поводяться менш складно: вони рухаються за певною траєкторією або випадково по ігровій місцевості, реагуючи на появу гравця лише в безпосередній близькості (рис. 1.4) [4]. Однак навіть така проста модель забезпечує базову динаміку, дозволяючи гравцеві обирати момент для нападу.

Рисунок 1.4 – Приклад патрулювання ворога у грі Persona 3 Reload

У Shin Megami Tensei V: Vengeance значно покращено сприйняття навколишнього середовища ворогами. Вони здатні реагувати на звуки, створені гравцем під час руху, стрибків або атак (рис. 1.5) [5]. Крім того, їхня поведінка адаптується до складності місцевості – вороги не просто патрулюють територію, а можуть змінювати маршрут, зважаючи на позицію гравця, або навіть організовувати переслідування з фланговими маневрами.

Рисунок 1.5 – Приклад реагування ворога на гравця у Shin Megami Tensei V: Vengeance

1.2 Аналіз засобів розробки ігрових додатків на платформі Windows

Розробка ігрових застосунків для Windows вимагає вибору відповідного ігрового рушія, що впливає на продуктивність, якість графіки, складність розробки логіки, підтримку штучного інтелекту, а також зручність інтеграції з інструментами UI/UX. Серед розробників ігор найчастіше застосовуються такі рушії, як Unreal Engine, Unity та Godot, кожний з яких має власні переваги та обмеження.

Unity є одним з найпоширеніших рушіїв завдяки своїй універсальності та підтримці великої кількості платформ. Його головними перевагами є проста у використанні, система компонування об’єктів, вбудована фізика та великий маркетплейс з готовими рішеннями. Unity підтримує мову C#, що робить поріг входу нижчим для новачків. Проте Unity поступається візуальній якості та гнучкості графічного рендерингу у порівнянні з рушіями вищого класу, такими як Unreal Engine (рис. 1.6) [6].

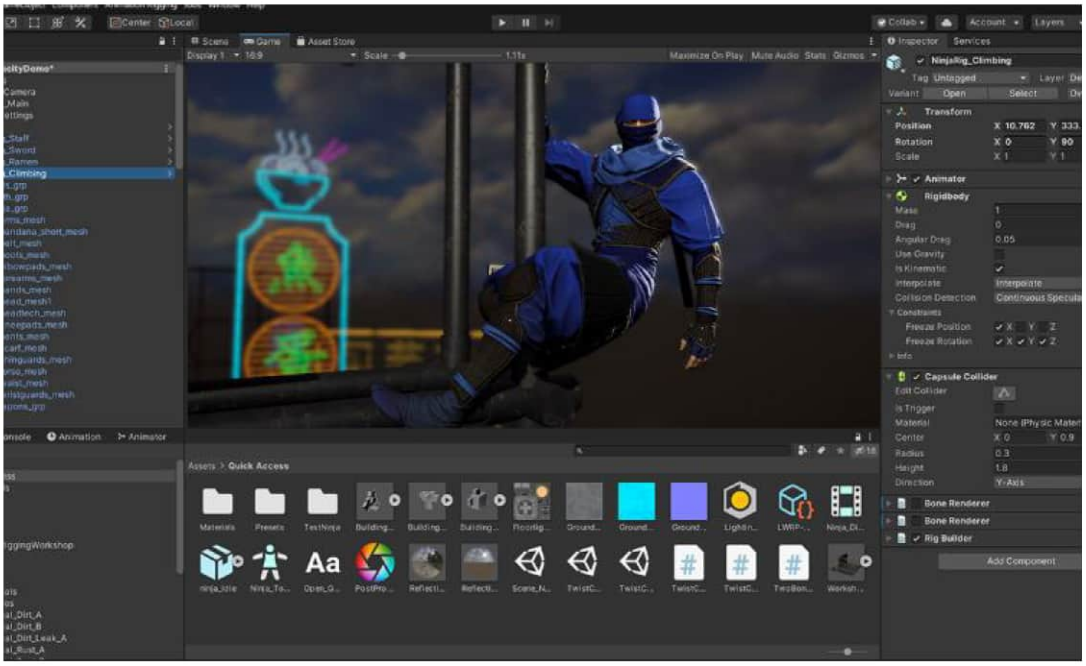


Рисунок 1.6 – Інтерфейс ігрового рушія Unity

Godot набуває все більшої популярності завдяки відкритому вихідному коду, легкості освоєння та активній спільноті. Цей рушій особливо зручний для 2D-ігор, де він переважає конкуренцію у швидкості розробки та оптимізації. Основна мова програмування в Godot – GDScript. Він полегшує роботу для тих, хто не знайомий з класичними об'єктно-орієнтованими мовами. Проте для створення складних 3D-сцен або сучасної графіки Godot поки що поступається Unreal Engine і Unity за функціональністю та візуальними можливостями (рис. 1.7) [7].

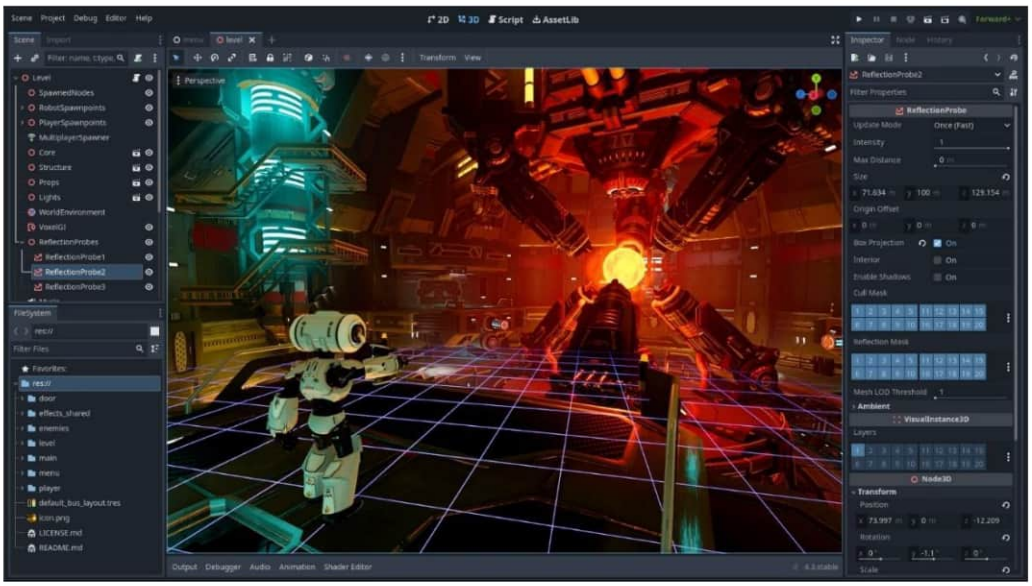


Рисунок 1.7 – Інтерфейс ігрового рушія Godot

Unreal Engine (UE) вирізняється з-поміж інших високим рівнем фотореалізму, підтримкою сучасних графічних технологій, таких як трасування променів у реальному часі, а також гнучкістю в реалізації логіки гри (рис. 1.8) [8]. Основними інструментами втілення логіки є мова C++ і візуальна система Blueprints, яка дозволяє створювати взаємодію без написання коду [9]. Unreal Engine також має просунуту систему реалізації штучного інтелекту через Behavior Tree, що дає змогу створювати адаптивну поведінку NPC [10].

Рисунок 1.8 – Інтерфейс ігрового рушія Unreal Engine

1.3 Огляд середовища розробки Unreal Engine

Беручи до уваги завдання реалізувати JRPG-гру з елементами штучного інтелекту, великою кількістю бойової логіки та детальним 3D-середовищем, Unreal Engine 4 був обраний як найбільш відповідний рушій для даного проекту.

Unreal Engine 4 (UE4) – це сучасний ігровий рушій, розроблений компанією Epic Games, який забезпечує цілісний цикл створення ігор: від розробки прототипів до реалізації складних 3D-проєктів із високим рівнем візуалізації. Завдяки своїй потужності та гнучкості UE4 широко застосовується як інді-розробниками, так і великими студіями для створення ігор у жанрах FPS, RPG, симуляторів та інших.

Першим кроком до роботи з Unreal Engine є встановлення рушія через Epic Games Launcher, який надає доступ до різних версій Unreal Engine, документації та маркетплейсу. Після встановлення рушія користувач може створити новий проект, обравши шаблон (наприклад, Blank, Third Person, Top Down) та бажаний тип програмування – Blueprints, C++ або комбіновану модель. (рис. 1.9)

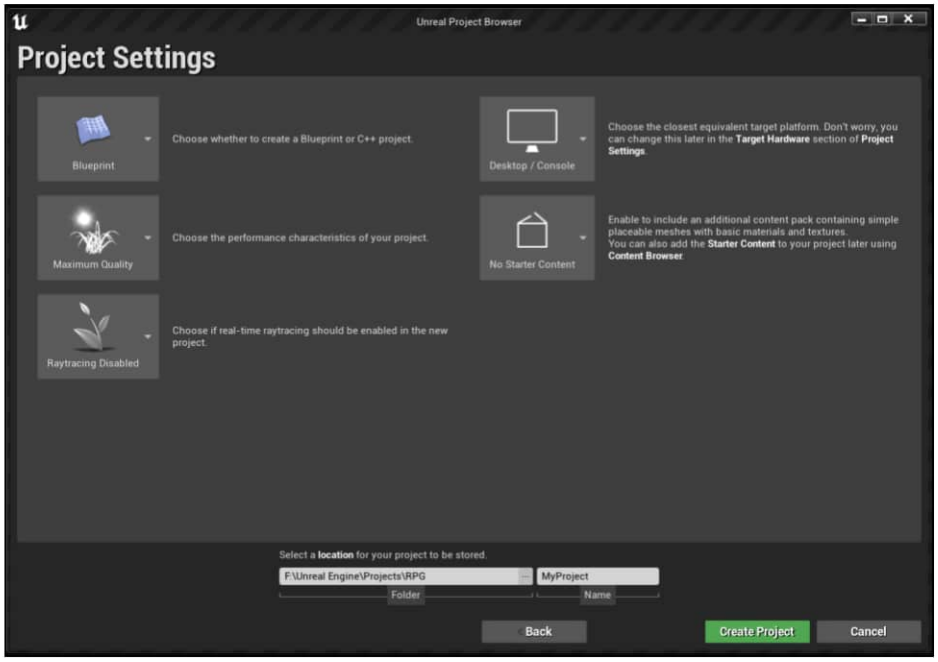


Рисунок 1.9 – Налаштування проєкту в Unreal Engine 4

При створенні проєкту користувач також обирає якість графіки (наприклад, Maximum Quality або Scalable 3D) та інші налаштування, що впливають на продуктивність і вигляд майбутньої гри. Завдяки цьому UE4 дозволяє одразу працювати у підготовленому середовищі, без потреби налаштовувати рушій з самого початку.

Головне вікно Unreal Engine 4 складається з кількох основних панелей, які формують основу робочого процесу розробника (рис. 1.10):

1) Viewport (вікно перегляду) – центральна область, де відображається і редагується 3D-сцена. Тут розробник може розміщувати об'єкти, налаштовувати освітлення, камери, а також тестувати логіку гри.
2) World Outliner – перелік усіх об'єктів, розміщених у поточній сцені. Кожен об’єкт можна обрати, згрупувати, чи швидко знайти.
3) Details Panel – панель властивостей обраного об’єкта. Дозволяє змінювати положення, розмір, матеріали, колізії та інші характеристики.
4) Content Browser – файловий менеджер рушія, що містить усі ресурси проєкту: 3D-моделі, текстури, звуки, анімації, Blueprint-и, карти тощо.
5) Toolbar – панель інструментів, з якої здійснюється запуск гри (Play), збірка проєкту (Build), компіляція Blueprints та інші важливі дії.

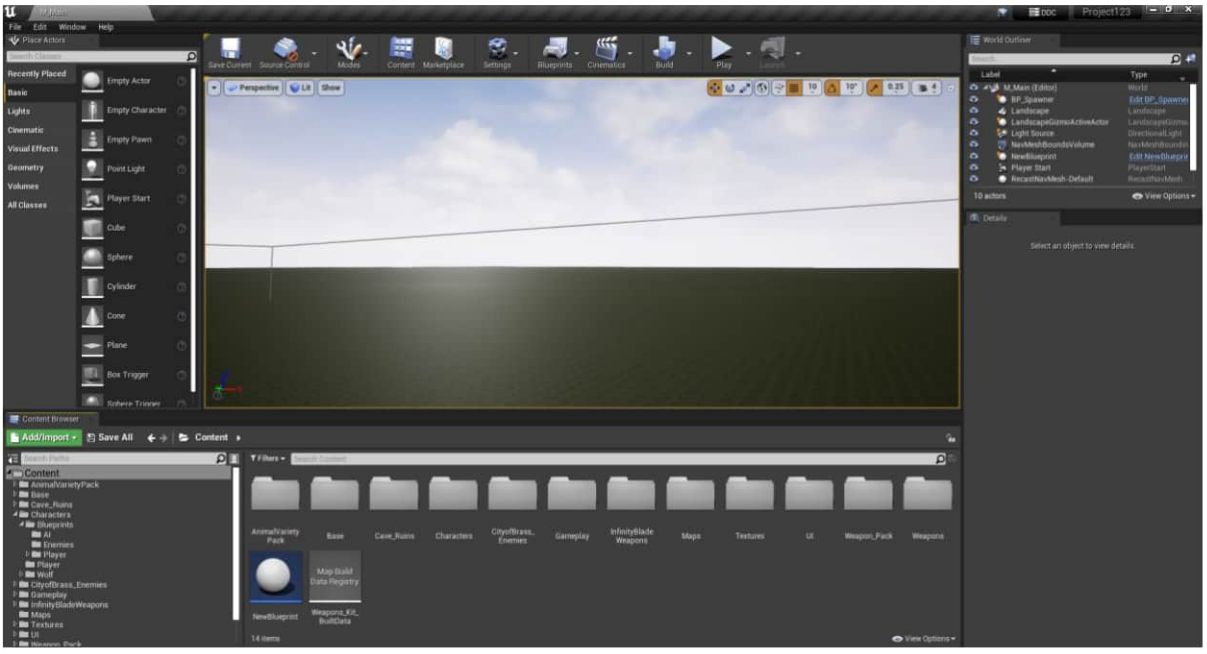


Рисунок 1.10 – Інтерфейс середовища Unreal Engine 4

Однією з ключових особливостей UE4 є система Blueprints – візуальний інструмент для розробки логіки гри без написання коду (рис. 1.11). Користувач створює об'єкти типу Event Graph, де логіка збудована з вузлів: події, умови, цикли, дії тощо. Blueprints ідеально підходять для швидкого прототипування, а також для команд без досвіду програмування на C++ [11].

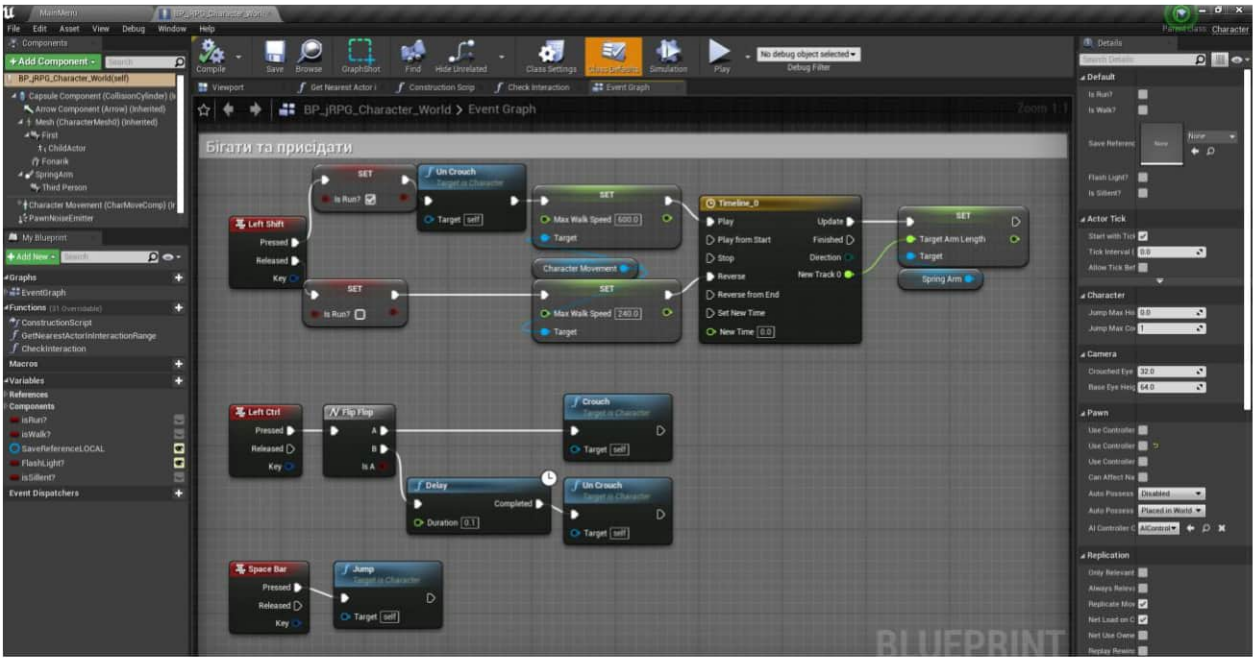


Рисунок 1.11 – Візуальне програмування за допомогою Blueprints в Unreal Engine 4

UE4 дозволяє створювати складні сцени завдяки Level Editor – модулю, що включає інструменти для розміщення геометрії, ландшафтів, освітлення та зон тригерів. Рівень (Level) є базовою одиницею ігрового світу. Окрім того, UE4 підтримує багаторівневу організацію сцен (наприклад, рівні для бойових арен, інтер'єрів, кат-сцен тощо).

Для реалізації ШІ рушій має вбудовану систему Behavior Tree та Blackboard (рис. 1.12), яка дозволяє створювати поведінкові дерева для ворожих персонажів. Це особливо актуально для ігор жанру JRPG, де супротивники повинні приймати рішення в залежності від ситуації.

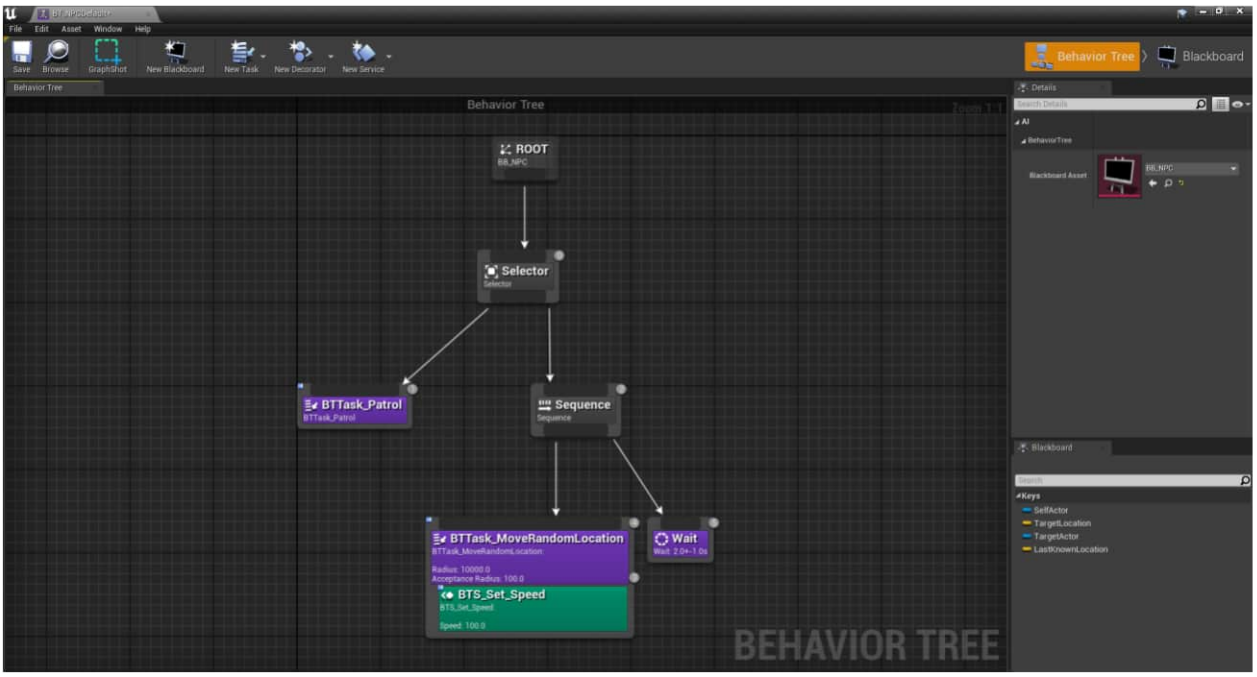


Рисунок 1.12 – Візуальне програмування за допомогою Behavior Tree в Unreal Engine 4

1.4 Висновки до першого розділу. Постановка задач розробки

Отже, аналіз наявних ігрових застосунків з використанням штучного інтелекту показав, що така технологія активно використовується в сучасній ігровій індустрії з метою збільшення зацікавлення гравців, адаптивності ігрового процесу та втілення складніших сценаріїв. Відомі приклади, такі як Shin Megami Tensei V: Vengeance, Persona 3, Persona 4 та Persona 5, демонструють рівень реалізації штучного інтелекту у вигляді поведінки супротивників, адаптації сценарію до дій гравця та динамічних систем ухвалення рішень.

У процесі аналізу також обґрунтовано вибір рушія Unreal Engine 4, який надає широкі можливості для розробки JRPG-застосунку із використанням поведінкових дерев, візуального програмування (Blueprints), потужної 3D-графіки та підтримки AI-компонентів. Проведений огляд рушія дозволив визначити його сильні боки та інтегрувати їх у процес планування архітектури гри.

Метою даної кваліфікаційної роботи є розробка ігрового застосунку в жанрі JRPG з елементами штучного інтелекту на основі рушія Unreal Engine 4. Головним завданням є втілення сценарію гри, де персонажі взаємодіють із навколишнім середовищем та ворогами з урахуванням динамічної поведінки, що забезпечується за допомогою AI-систем.

Для досягнення поставленої мети в кваліфікаційній роботі необхідно вирішити такі завдання:

1) здійснити аналіз існуючих ігрових проєктів, у яких реалізовано AI-системи, з метою виявлення актуальних підходів та технологій;
2) обґрунтувати вибір ігрового рушія, мов програмування та інструментів, які будуть використані під час реалізації проєкту;
3) розробити концепцію гри та визначити архітектуру майбутнього застосунку, зокрема структуру AI-модулів;
4) створити візуальне середовище гри, ігрових персонажів та сценарні події;
5) реалізувати ключові функціональні модулі, включаючи штучний інтелект для персонажів, бойову систему та систему збереження прогресу;
6) протестувати функціональність ігрового застосунку з метою перевірки коректності роботи AI-компонентів та забезпечення загальної стабільності гри;
7) оцінити ефективність реалізованого програмного рішення та його відповідність початковим вимогам.

РОЗДІЛ 2. ПРОЄКТНІ РІШЕННЯ

2.1 Концепція гри

2.1.1 Жанрова специфіка

Представлена гра відноситься до жанру японських рольових ігор (JRPG), поєднуючи тактичні битви та інтерактивне дослідження світу. JRPG традиційно асоціюється з глибоким сюжетом, розвитком персонажів, покроковими боями, а також візуальним стилем.

Головна риса, яка відрізняє JRPG від західних рольових ігор - це лінійна або напівлінійна структура сюжету та стратегічний акцент у битвах: гравець змушений ретельно планувати дії, управляти командою, вдало застосовувати покрокові здібності та предмети.

У даному проєкті додатково впроваджено елементи штучного інтелекту, розширюючи межі жанру: противники мають складну поведінкову логіку, неігрові персонажі реагують на дії гравця, а бойова система різноманітна тактичними шарами.

Окрему увагу зосереджено на створенні атмосфери звичайної школи (рис. 2.1) та потойбічного світу, що контрастує з реальністю, викликаючи відчуття тривоги, загадковості та небезпеки. Детально опрацьовані інтер'єри класів, коридорів та шкільних приміщень допомагають гравцеві поринути у знайоме, але водночас тривожне оточення. Безпосередня взаємодія звичного шкільного середовища та паранормальних елементів, як-от спотворені простори, дивні явища та загрозливі створіння, породжує притаманну жанру напругу, зацікавленість і психологічне напруження, що поступово посилюється з розвитком сюжету.

Рисунок 2.1 – Ігрова локація Map_School

2.1.2 Цільова аудиторія

Розроблена гра орієнтована на підлітків та молодь, віком від 15 до 25 років. Це ті, хто має інтерес до японської культури, аніме-естетики, а також рольових ігор з елементами дослідження і тактики. Цільова аудиторія вже знайома з іграми серій Persona, Shin Megami Tensei або подібні. Вона має схильність до сюжетно насичених пригод, що виконані у стилізованому візуальному оформленні.

Враховуючи те, що гра містить загадкову атмосферу школи, дослідження покинутих класів і потойбічного миру, вона буде привабливою для гравців, яким подобається містика, ізольовані простори, а також ігри з вираженою атмосферою й поступовим розкриттям сюжету. Особливий акцент на дослідження локацій та бої з ворогами спрямований на тих гравців, котрі цінують тактичну глибину та виклик, а не тільки візуальний стиль або ж сам сюжет.

2.1.3 Загальний задум гри

Гра розробляється в жанрі JRPG з ізометричним видом, що складається з двох ключових режимів: дослідження місцевості та покрокові битви. Дія

розгортається в школі, яка має паралельний вимір, спотвореним віддзеркаленням реальності. Гравець контролює головного героя старшокласника-екзорциста (рис. 2.2), котрий разом з друзями намагається звільнити школу від злих сил.

Рисунок 2.2 – Idle анімація головного героя

Мета гравця – поступово досліджувати школу, ухиляючись або вступаючи в бій з ворожими істотами, що охороняють коридори. У міру проходження локацій стає відомо, що деякі кімнати зачинені. І для їх відкриття, гравцю потрібно знайти всі ключі та просуватися до головного ворога.

Важливу функцію виконує атмосфера занедбаної школи (рис. 2.3), наповнена дивними шумами, понівеченими меблями, темними коридорами та класами, які породжують гнітюче та моторошне відчуття. Елементи довкілля – такі як мерехтливе освітлення, скрип дверей, відгук кроків – збільшують занурення у світ, де небезпека може чатувати за кожним поворотом. Головна мета гравця – здолати боса, який втілює джерело зла в конкретному крилі школи. Після його перемоги локація звільняється від впливу потойбічних сил, атмосфера стає світлішою та безпечною, а команда повертається у звичайний світ.

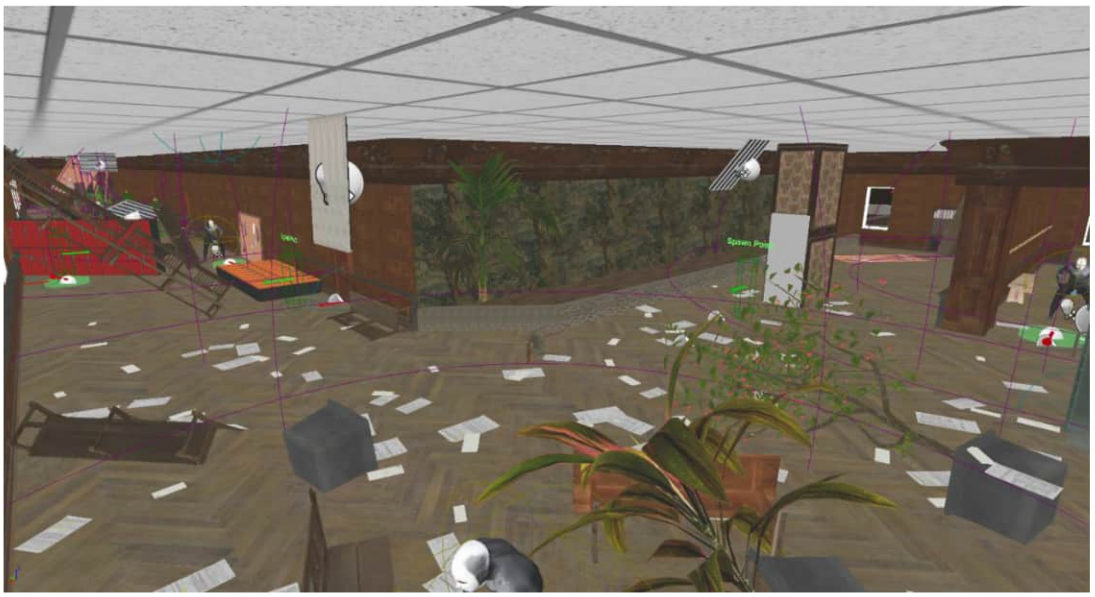


Рисунок 2.3 – Ігрова локація Map_SchoolCursed в Viewport

Діалоги з NPC мають ознайомлювальну та сюжетну функцію: вони роз'яснюють основні ігрові механіки, як-от взаємодія з предметами, орієнтація в локації, а також розкривають мотиви героїв. Після фінальної битви відбувається завершення історії та відображення титрів.

2.1.4 Особливості гри

Гра поєднує у собі традиційні риси JRPG з елементами простоти та ефективності в реалізації штучного інтелекту та інтерфейсу. Основні аспекти проєкту розкриваються у наступному:

1) Дослідження школи в потойбіччі: Гравець вільно подорожує по коридорах, класах та інших приміщеннях покинутої школи. Кожна кімната унікальна: має власну атмосферу, кольорову палітру та музику, які посилюють відчуття занурення в ігровий світ.
2) Покрокова бойова система: У битвах гравець керує групою героїв, застосовуючи фізичні атаки, спеціальні вміння а також предмети. Різні вороги демонструють власну тактику, навички та характеристики, що додає різноманітності в зіткненні.
3) Система розвитку персонажів: Кожен персонаж може бути екіпірований знайденими речами, які покращують його параметри. Ефективний підбір

ресурсів та розподілення команди безпосередньо впливають на ігровий процес.

4) Інтерактивні об'єкти та діалоги: У світі гри є скрині зі скарбами, двері з різними вимогами для відкриття, а також NPC, з якими можна взаємодіяти. Саме ці персонажі знайомлять гравця з ігровою механікою, надають підказки та розкривають основну концепцію гри.
5) AI-поведінка ворогів та NPC: Вороги патрулюють локації, дотримуючись визначених маршрутів, реагуючи на появу гравця в полі зору і розпочинаючи переслідування. В бою використовують базову тактику та атакують різні цілі. Дружні NPC, які ходять по звичайній школі та класах, додають живої атмосфери.
6) Музичний супровід: Кожна ігрова зона має власний музичний трек, що динамічно змінюється в залежності від ситуації (дослідження або бій). Саундтрек створює напружену атмосферу та підкреслює загальну атмосферу гри.
7) Фінальна частина та очищення школи: Після перемоги над фінальним босом у останній локації, школа повністю очищується від потойбічної присутності. Відтворюються заключні сцени та титри, що завершують гру.

Усі ці складові взаємодіють, створюючи інтегроване ігрове середовище. Забезпечує безперервний перехід між дослідженням локацій та битвами. Цей підхід дозволяє зберегти цілісність ігрового процесу, постійно підтримуючи інтригу та мотивацію гравця, та занурити його у неповторну атмосферу похмурого навчального закладу, де панують надприродні явища.

2.2. Архітектура проєкту

2.2.1 Структура проєкту в Unreal Engine 4

Архітектура ігрового застосунку створена згідно з принципами модульності та чіткого розділення функціональності. Проєкт виконано на

ігровому рушії Unreal Engine 4, де кожен функціональний блок поміщено в окрему директорію для зручності розробки, підтримки та подальшого масштабування. Усі внутрішні взаємодії між модулями здійснюються через механізми подій, біндінгу, інтерфейсів та таблиць даних.

До ключових модулів проєкту відносяться:

1. AI – містить логіку поведінки NPC, їх патрулювання, пересування та реакції на дії гравця.
2. Blueprints – головна директорія з візуальними сценаріями: актори, контролери, компоненти, інтерфейси, система збереження.
3. Datatables – таблиці з даними про персонажів, ворогів, діалоги, локації; забезпечують централізоване зберігання змінних.
4. Enumerations – переліки можливих значень, які використовуються для визначення типів предметів, станів, статусів та іншого.
5. Game – усі ресурси проєкту (текстури, моделі, звукові файли, анімації) та похідні BP-класи.
6. Structures – спеціальні контейнери, що визначають формат збереження даних у Datatables.
7. Widgets – всі елементи користувацького інтерфейсу: HUD, інвентар, меню налаштувань та інші.

На рис. 2.4 та рис. 2.5 показано загальну структуру проєкту, яка демонструє взаємодії між ключовими модулями.

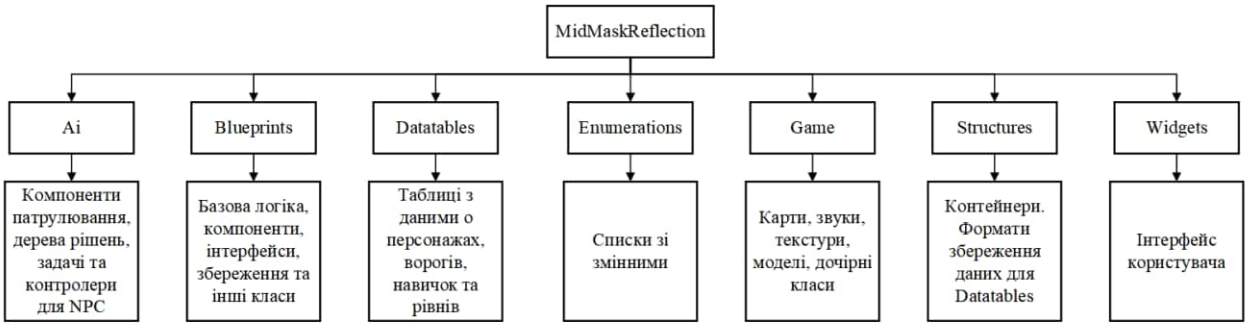


Рисунок 2.4 – Діаграма структури проєкту в Unreal Engine 4

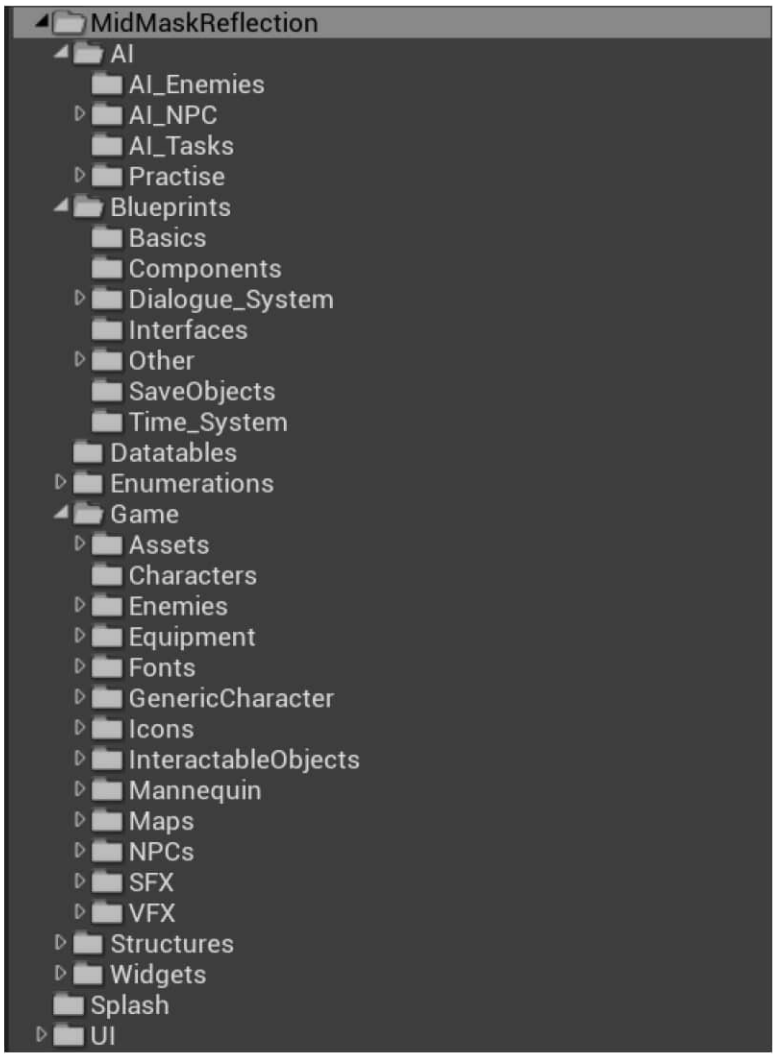


Рисунок 2.5 – Файлова структура проєкту в Unreal Engine 4

2.2.2 Структура готового проєкту

По завершенні створення в Unreal Engine 4 проєкт переходить до етапу пакування, під час якого формується готовий до запуску файл гри та супровідні каталоги з усіма потрібними даними. Це дає змогу запустити гру на операційній системі Windows без використання середовища розробки.

У запакованому проєкті міститься основний виконуваний файл MidMaskReflection.exe, відповідальний за запуск гри. Папка Engine включає всі системні бібліотеки рушія Unreal Engine, необхідні для функціонування графіки, звуку, фізики та інших технічних елементів. Папка MidMaskReflection містить внутрішні ресурси гри: локації, моделі, текстури, анімації, логіку,

скрипти, налаштування та інші складові, які забезпечують повноцінний ігровий процес (рис. 2.6).

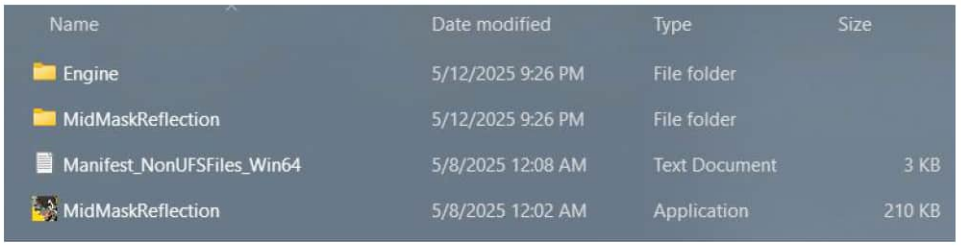


Рисунок 2.6 – Файлова структура готового продукту

2.2.3 Основні програмні модулі

Unreal Engine 4 базується на об'єктно-орієнтованій архітектурі, де ключові складові логіки створені через Blueprint-класи, що представляють собою візуальні сценарії для налагодження взаємодії між об'єктами без необхідності писати код. Завдяки цьому можна швидко прототипувати та реалізовувати ігрову логіку. Діаграму взаємодії класів зображено на рисунку 2.7.

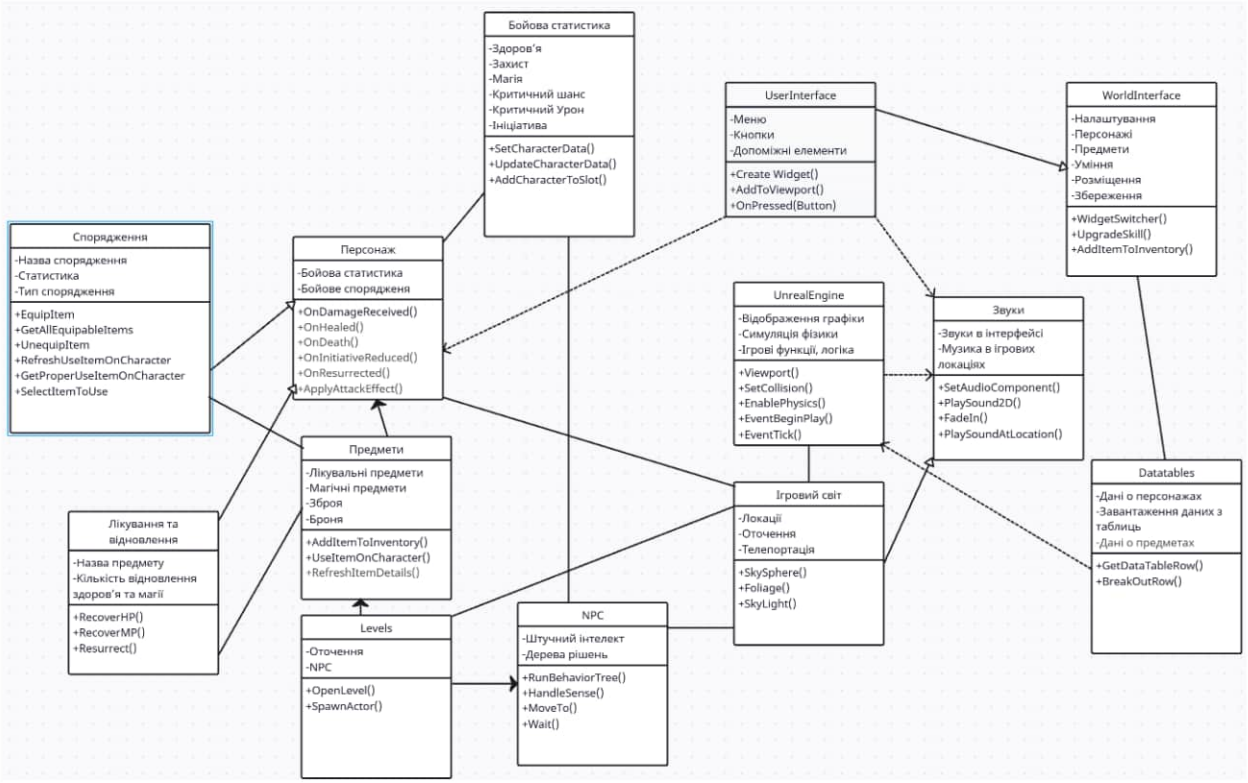


Рисунок 2.7 – Діаграма класів

Основні структурні елементи, які використовуються в Unreal Engine:

1. Actor (Актор) – фундаментальний об'єкт, що розташовується в сцені. Він може містити фізичний вигляд, реалізовувати логіку чи служити контейнером для компонентів.
2. Component (Компонент) – допоміжна частина, прикріплювана до Actor'a з метою розширити його можливості (наприклад, детектування зіткнень, показ моделі, відтворення звуку).
3. Controller (Контролер) – особливий клас, відповідальний за управління персонажем або NPC. Розрізняють PlayerController (для гравця) та AIController (для керування NPC).
4. Widget (Віджет) - елементи інтерфейсу (UI), що відповідають за вивід інформації, меню, діалогових вікок та інших елементів взаємодії з користувачем на екран комп’ютера.
5. Behavior Tree (Дерево поведінки) – система прийняття рішень для NPC, дозволяє визначити складну поведінку через набір умов, дій та їх пріоритетів.
6. Blueprint Class (Клас Blueprint) – шаблон класу з візуально представленою логікою, що може містити змінні, функції, події та інші сутності.

Ці складові взаємодіють згідно з їх призначенням у грі: логіка бою комунікує з інтерфейсом через віджети, дерева рішень NPC використовують контролери, які звертаються до самого дерева або компонента патрулювання. Такий підхід сприяє створенню гнучкої, модульної та легко масштабованої архітектури ігрового застосунку.

2.2.4 Взаємодія гравця між модулями

Перед початком взаємодії гравця з ігровим додатком запускаються внутрішні системи, що відповідають за обробку ігрових подій, реакцію на введення користувача та відстеження поточного стану ігрового середовища. Ці процеси працюють у режимі реального часу, забезпечуючи оперативну

реакцію на будь-які дії гравця. Кожна така дія супроводжується зворотним зв’язком, який може проявлятися у вигляді оновлення інтерфейсу, візуальних змін у середовищі або активації звукових ефектів. Подібна реакція допомагає підтримувати високу динаміку гри та створює ефект повної присутності. Узгодженість усіх систем у режимі реального часу забезпечує плавне, логічне й інтуїтивно зрозуміле керування, що позитивно впливає на загальне враження від гри. Взаємодію гравця з додатком відображено в таблиці 2.1.

Таблиця 2.1 – Взаємодія гравця з ігровим додатком

Дія гравця	**Реакція гри (логіка)**	**Відображення в інтерфейсі**
Переміщення по світу	Обробка введення, оновлення позиції персонажа	Камера слідує за персонажем, оновлюється анімація руху
Взаємодія з об’єктом	Перевірка умов, запуск скрипту події	Відкриття діалогового вікна, відображення анімації
Відкриття інвентаря	Виклик логіки інвентаря, зчитування поточних предметів	Відображення меню з речами, інтерфейс з кнопками дії
Початок бою (при зіткненні з ворогом)	Ініціалізація бою, перехід у бойову локацію	Завантаження бойового інтерфейсу
Вибір дії в бою	Надсилання команди до бойової системи, оновлення станів	Анімації атак, оновлення бойового інтерфейсу
Завантаження гри	Зчитування даних з файлу, оновлення стану світу та персонажа	Завантаження останньої сцени
Перехід у головне меню	Завершення поточної сесії, перехід до стартового екрану	Відображення головного меню

2.3. Ігрові персонажі

2.3.1 Головний герой

Головним персонажем, яким гравець керує у грі, є Нарукамі. Саме з його перспективи гравець проходить увесь сюжет, досліджує навколишній світ, взаємодіє з оточенням та розкриває таємниці гри. Нарукамі обладнаний ліхтариком, який слугує основним засобом освітлення темних коридорів і кімнат. Гравець може відкривати скрині, відчиняти двері, а також ініціювати діалоги з іншими персонажами, активуючи події або отримуючи корисну інформацію. Всі дії відбуваються в режимі дослідження, однак у разі зустрічі з ворогом гра миттєво перемикається в бойовий режим.

Керування головним героєм забезпечується класом BP_jRPG_Controller_World (рис. 2.8). Цей контролер відповідає за пересування гравця, взаємодію з ігровим інтерфейсом, управління інвентарем, а також обробку всіх взаємодій з персонажами за допомогою інтерактивної системи. Контролер містить в собі основні компоненти, які забезпечують логічні системи (табл. 2.2). Повна Blueprint логіка відображена в додатку А.

Таблиця 2.2 – Компоненти контролера BP_jRPG_Controller_World

Назва компоненту	**Функції**
InventoryManager	Відповідає за зчитування предметів у гравця, які потім показуються в інтерфейсі.
CharactersManager	Зчитує інформацію про персонажів у команді та відображає їх у меню.
SaveManager	Зберігає поточний прогрес гравця, локацію, відкриті об’єкти та переможених ворогів.
BattleManager	Ініціює битву при контакті з ворогом, зберігає координати поточної локації.
ActionsManager	Зчитує бойові навички персонажів та відображає доступні дії в інтерфейсі

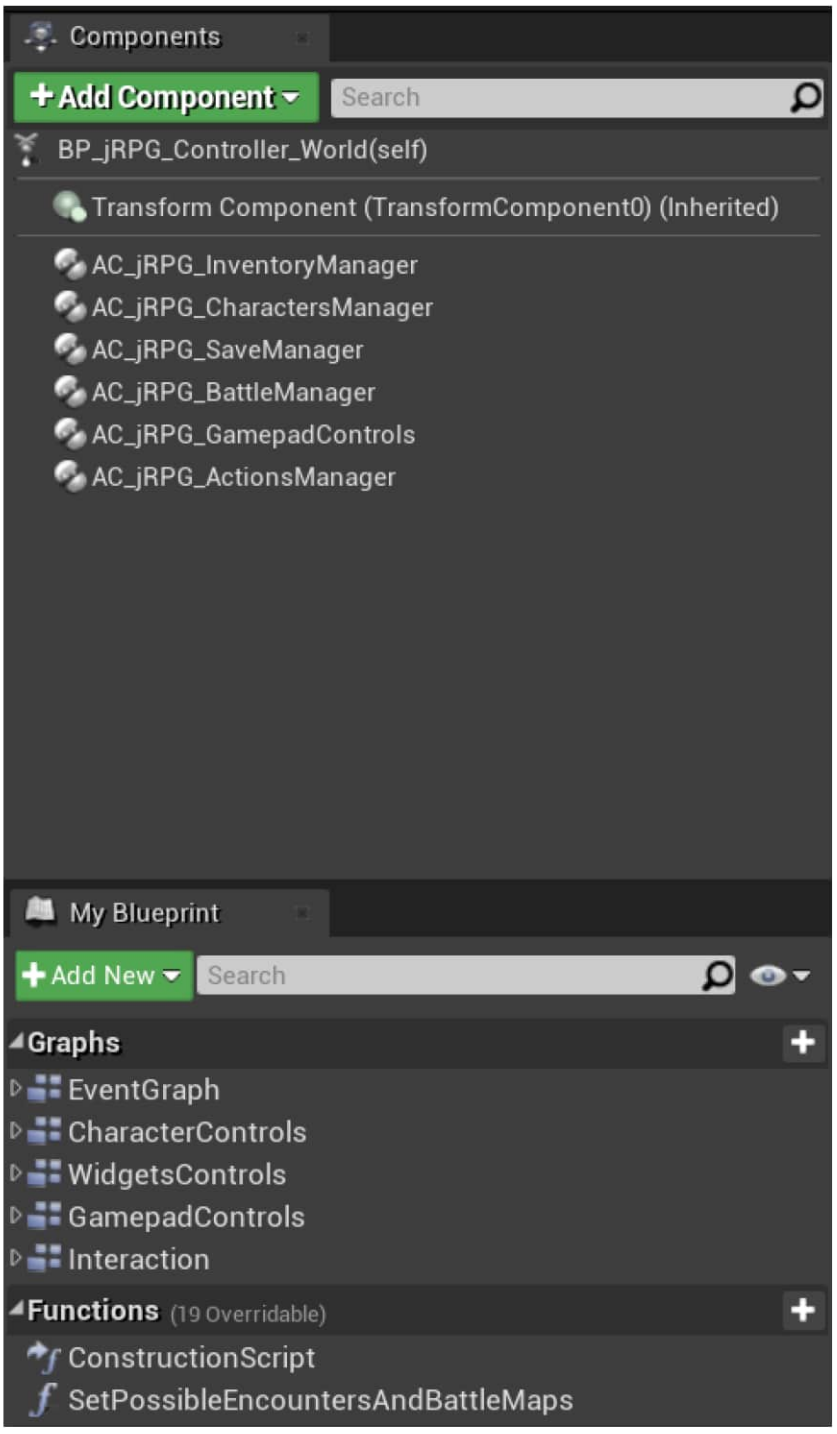


Рисунок 2.8 – Компоненти контролеру головним персонажем

Клас BP_jRPG_Character_World виступає фундаментальним класом персонажа, що бачить гравець у всесвіті гри (рис. 2.9). Він керує візуальним відображенням, рухами та базовою поведінкою в грі. Цей клас об'єднує компоненти, які визначають як внутрішні аспекти персонажа, так і його зовнішній вигляд (табл. 2.3). Повну Blueprint логіку та Controller надано в додатку А.

Таблиця 2.3 – Компоненти класу BP_jRPG_Character_World

Назва компоненту	**Тип**	**Призначення**
Capsule Component	Collision Component	Визначає фізичні границі персонажа для взаємодії зі світом
Arrow Component	Scene Component	Служить орієнтиром у просторі (не бере участі в логіці гри)
Mesh	Skeletal Mesh	Модель персонажа, яка анімується під час руху
First Person Camera	Camera Component	Камера з видом від першої особи
Third Person Camera	Camera Component	Камера з видом від третьої особи
Child Actor (Flashlight)	Child Actor Component	Клас Ліхтарика, який з'являється після підбору об'єкта
Character Movement	Movement Component	Компонент, який відповідає за налаштування швидкості, стрибків тощо
PawnNoiseEmitter	Audio Component	Створює шум, який можуть чути NPC із системою AI

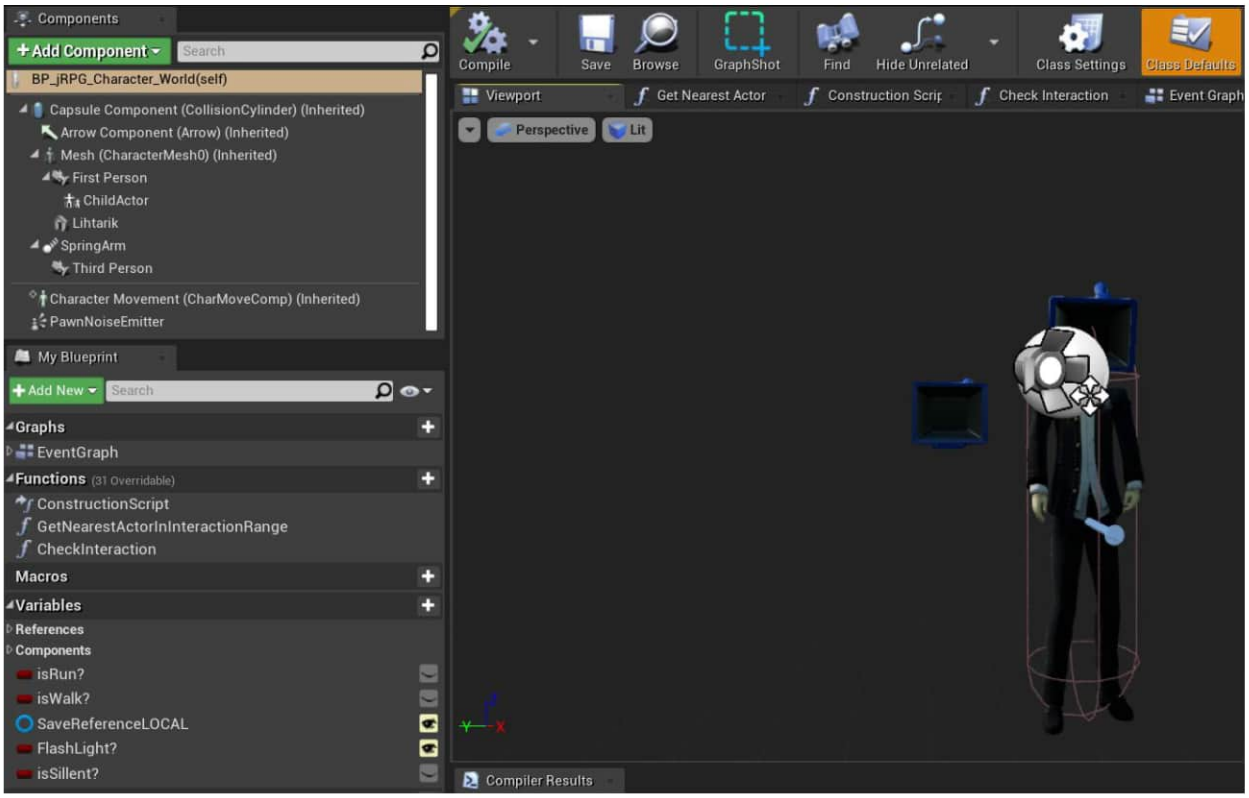


Рисунок 2.9 – Компоненти класу головного героя

2.3.2 Бойові персонажі

У битві беруть участь декілька героїв, кожен з яких відзначається власними бойовими властивостями, озброєнням та призначенням у команді. Головним персонажем є Нарукамі – мечник. Різе – лучниця, Йосуке використовує ніж, Чіє – боксерка, виконує функцію цілителя та підтримки. Кандзі – мутант з фізичною силою, він виконує роль важкого бійця, що витримує великі обсяги шкоди. Ці персонажі доступні гравцеві на вибір в світовому інтерфейсі (рис. 2.10).

Рисунок 2.10 – Відображення персонажів в інтерфейсі

Бойові персонажі базуються на класі BP_jRPG_Character_Battle_Base (рис. 2.11), що визначає всю бойову логіку: анімації, поточний стан здоров'я, використання навичок, плюс пересування на бойовій арені (табл. 2.4). Цей клас виступає основою, з якої успадковуються всі персонажі, незалежно від того, союзники вони чи противники. В ньому реалізована система бойових станів, таких як «готовий до бою», «виконує атаку», «отримує пошкодження», «переможений», яка контролює їхні дії в рамках покрокового бою. Завдяки централізованій логіці в одному базовому класі, поведінка всіх бійців узгоджується. Логіку бойового персонажа продемонстровано в додатку Б.

Таблиця 2.4 – Компоненти класу BP_jRPG_Character_Battle_Base

Назва компоненту	**Тип**	**Призначення**
Capsule Component	Collision Component	Визначає фізичні межі персонажа під час бою
Arrow Component	Scene Component	Орієнтир у просторі, не використовується в бойовій логіці
Mesh	Skeletal Mesh	Модель персонажа з анімаціями руху та атаки
TargetPointer	Billboard Component	Відображає візуальний індикатор наведення на персонажа
Camera_ChildActor	Camera Component	Камера, яка активується при ході персонажа
Character Movement	Movement Component	Компонент для контролю фізики переміщення
CharacterStats	Custom Component	Містить дані про навички, параметри та характеристики персонажа
CombatMovement	Custom Component	Відповідає за рух персонажа по бойовому полю до цілі чи обраної позиції

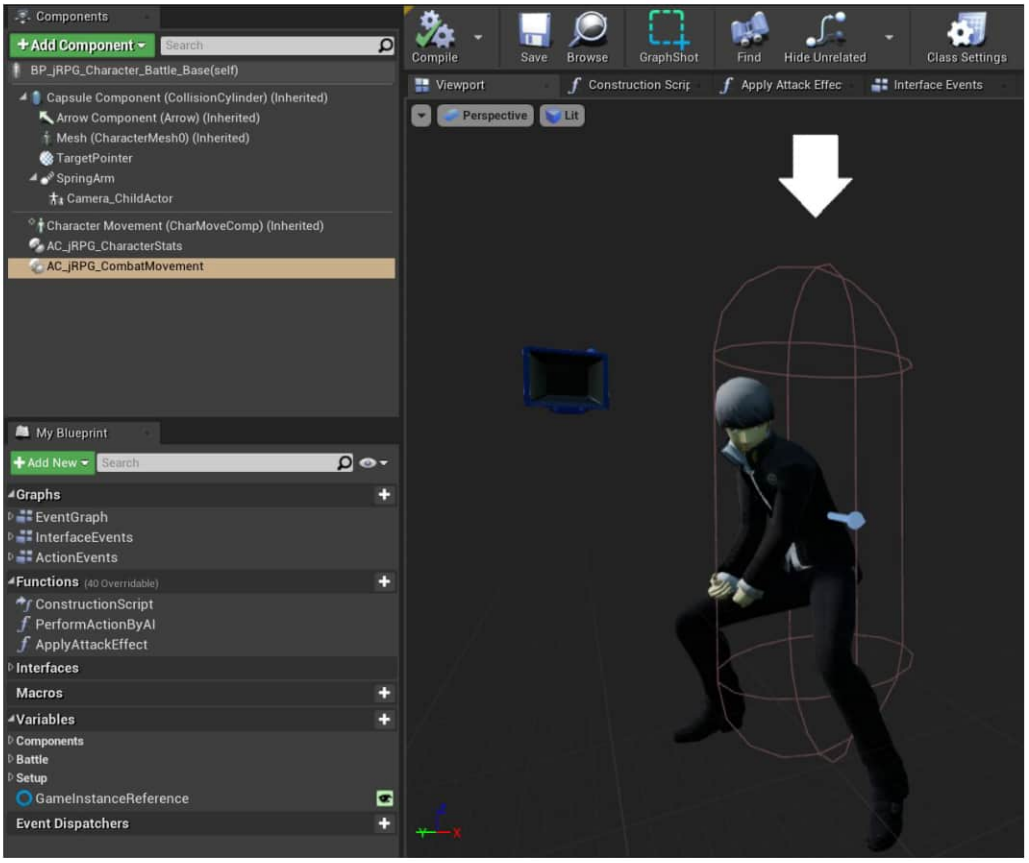


Рисунок 2.11 – Компоненти класу бойового персонажа

2.3.3 Система розвитку персонажів

У грі присутня система прокачки героїв, завдяки якій можна покращувати їх бойові вміння, збільшувати рівень та підбирати екіпірування. Основна логіка цього процесу зосереджена в компоненті AC_jRPG_CharactersManager, який керує підвищенням рівня, порівнянням поточних показників, оснащенням новими предметами, а також оновленням інтерфейсу при змінах. Інтерфейс WBP_jRPG_WorldInterface показує інформацію про актуальний стан героїв та надає гравцю можливість взаємодії з системою розвитку (рис. 2.12).

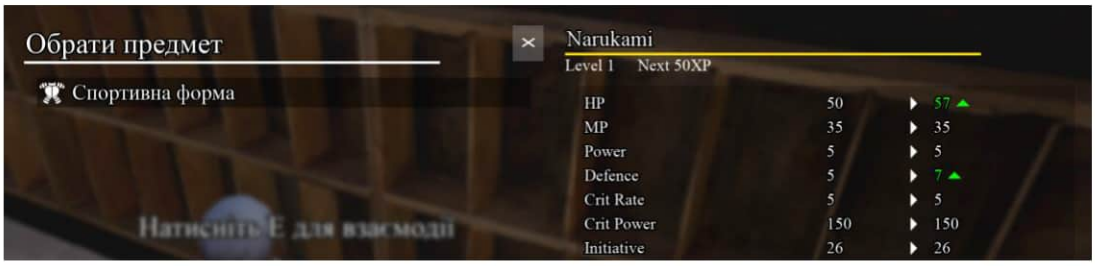


Рисунок 2.12 – Покращення характеристик при одяганні предмета

Дані про кожного героя отримуються з таблиці DT_jRPG_Characters, де зібрано основні характеристики та початкові параметри, які завантажуються автоматично під час створення персонажа (рис. 2.13). Це забезпечує централізоване управління ігровим балансом і полегшує коригування властивостей героїв у процесі подальшої розробки. Для кожного персонажа визначені індивідуальні характеристики, що мають прямий вплив на ігровий процес: вони задіяні під час бою для обчислення завданих пошкоджень, визначення черговості дій у ході битви, а також розрахунку витрат ресурсів, зокрема магії (MP) та здоров'я (HP), відповідно до обраних навичок і обставин на полі бою (табл. 2.5).

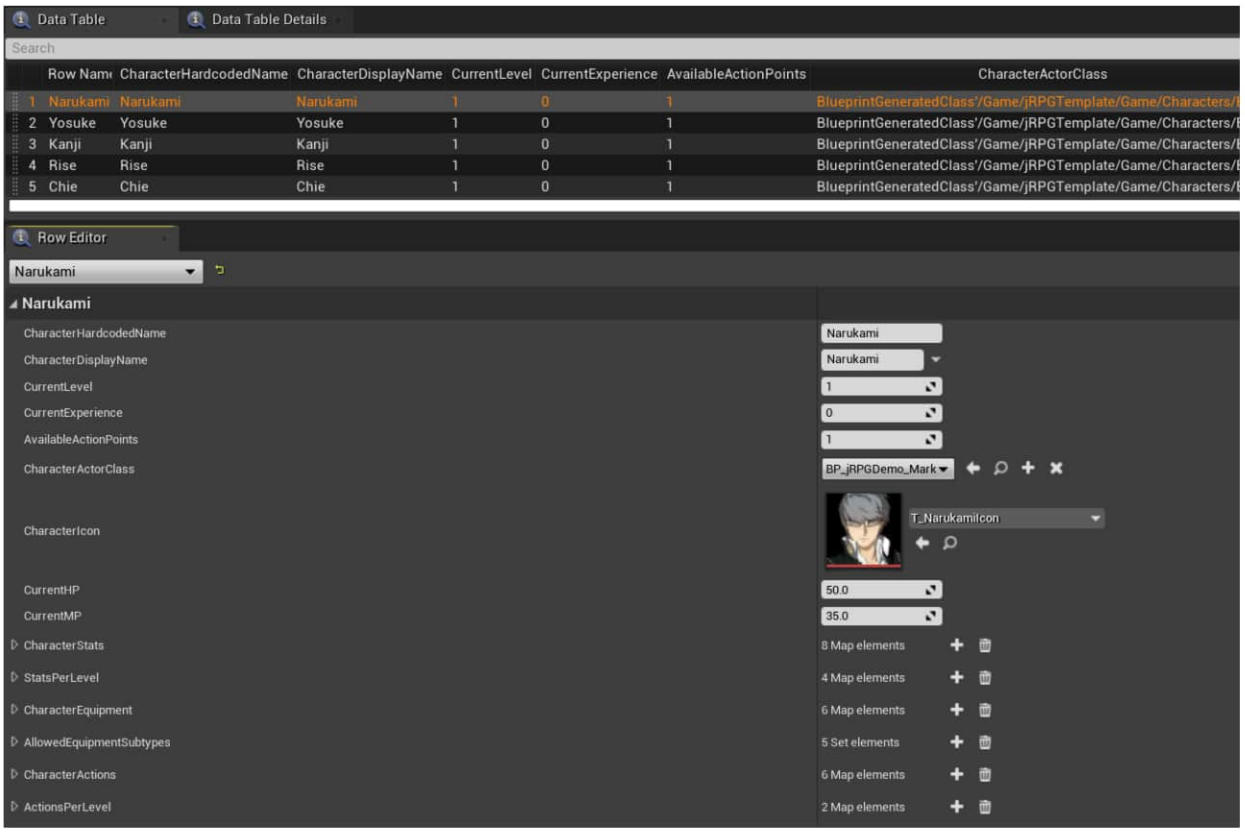


Рисунок 2.13 – Таблиця з даними о персонажах

Таблиця 2.5 – Параметри персонажів

Параметр	**Опис**
HP	Одиниця здоров’я персонажа
MP	Одиниця магічної енергії, використовується для навичок
Power	Фізична сила, яка впливає на шкоду від атак
Defence	Захист, що знижує вхідну шкоду
Crit Rate	Ймовірність нанесення критичного удару
Crit Power	Множник шкоди при критичному ударі
Initiative	Швидкість дій у бою, впливає на черговість ходів

2.4 Бойова система

2.4.1 Логіка бойової системи та інтерфейс

Бойова система втілена в окремій сцені, яка активується, коли гравець стикається з ворогом. Відбувається завантаження спеціальної бойової локації. Кількість і типи ворогів визначаються у класі Enemy_World_Base, з яким

зіткнувся ворог. Як тільки сцена завантажилася, зчитується поточний склад команди гравця, і всі активні персонажі додаються до бойової черги, враховуючи їх швидкість. Компоненти битви відображено в додатку В.

За більшу частину логіки бою відповідає компонент AC_BattleManager. Він управляє послідовністю дій, обчислює чергу на основі параметру швидкості, та ініціює початок або завершення ходу. Компонент також відповідає за відображення екрану смерті, вибір цілі для атаки або використання навичок, а також за перехід до наступного учасника бою.

Паралельно працює AC_ActionsManager, який обробляє вибір доступних дій, параметри навичок та їх активацію. Він слугує посередником між інтерфейсом та логікою бою: отримує вибір гравця та викликає відповідні події.

Основний бойовий інтерфейс реалізовано у вигляді віджету WBP_BattleInterface (рис. 2.14). Він показує імена персонажів, їхній рівень здоров’я, запас магії, перелік доступних дій та чергу ходів. Гравець бачить, хто з персонажів зараз активний, і може обрати одну з опцій - атаку, використання навички чи іншу дію. Інтерфейс динамічно оновлюється після кожної дії, щоб відображати поточний стан бою.

Рисунок 2.14 – Бойовий інтерфейс

2.4.2 Прогрес битви

Усі бойові сценарії втілюються через анімаційні монтажі (Anim Montage). Вони використовуються для програвання атак, застосування вмінь або реалізації захисних дій. Anim Montage (рис. 2.15) – це особливий різновид анімаційної послідовності в Unreal Engine, що надає змогу контролювати ключові моменти виконання дії за допомогою AnimNotify. У ці саме точки інтегруються виклики функцій, які завдають шкоди (рис 2.16), відновлюють здоров'я, активують спецефекти або змінюють поточний стан персонажа.

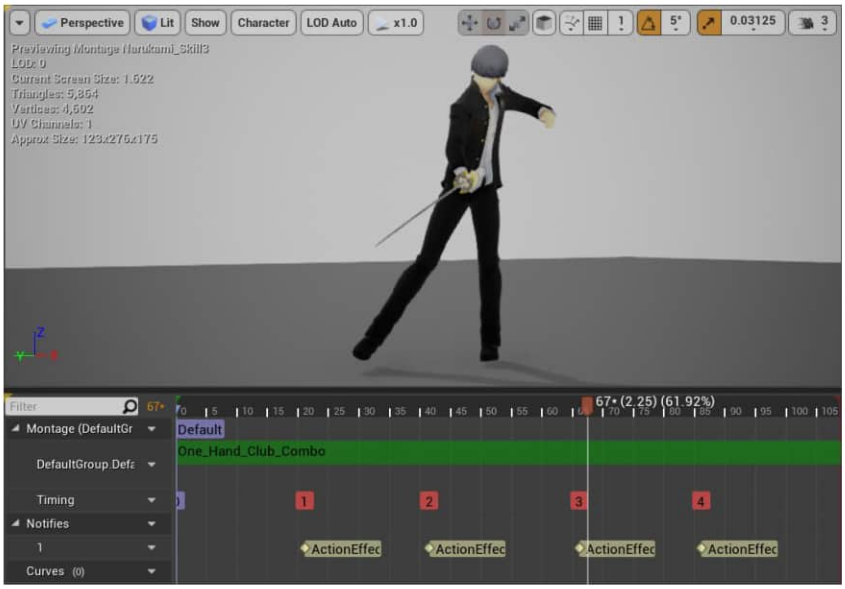


Рисунок 2.15 – Anim Montage з точками активації атаки

Рисунок 2.16 – Активація ефекту атаки за допомогою AnimNotify

У процесі бою гравець має змогу вибирати одну з декількох доступних дій. Кожна з них реалізована як елемент вибору в бойовому інтерфейсі та супроводжується відповідними анімаціями та візуальними ефектами (табл. 2.6)

Таблиця 2.6 – Типи дій

Назва дії	**Опис**
Атака	Базова фізична атака, шкода залежить від параметра сили персонажа.
Захист	Анімація захисту персонажа, що знижує вхідну шкоду на 50%. Завершується пропуском ходу.
Навички	Спеціальні дії, які витрачають магію. Можуть завдати шкоди або вилікувати.
Предмети	Можливість використання предметів: лікування, відновлення магії, воскресіння.
Допоміжні навички	Ефекти, які не потребують магії. Наприклад, удар ногою.
Втеча	Спроба покинути бій. У разі успіху повертає гравця назад у дослідницьку зону.

У момент виконання будь-якої дії активується спеціальна камера, реалізована через BP_Camera. Перемикання камери відбувається автоматично, залежно від того, хто є активним у грі - гравець або ворог. У момент атаки гравця камера фокусується на всіх персонажах на полі битви. Під час ходу ворога камера перемикається у вид з його позиції (рис. 2.17)

Рисунок 2.17 – Вид камери під час атаки ворога

2.4.3 Завершення бою

Після завершення бою гра проводить аналіз результату, враховуючи стан команди гравця та супротивників. У випадку знищення усіх ворогів, на екрані з’являється вікно перемоги. Гравцеві демонструються здобутий досвід, нараховані кошти та список предметів, що випали з ворогів (рис. 2.18).

Рисунок 2.18 – Результат бою після перемоги

У випадку програшу, якщо головний герой або весь його загін зазнають поразки, гра автоматично перемикає камеру, фокусуючи її на тілі героя. Після цього на екрані з’являється інтерфейс поразки, який сповіщає про завершення бою та автоматично переходить до головного меню (рис. 2.19)

Рисунок 2.19 – Результат бою після поразки

2.5 Система взаємодії з об’єктами

2.5.1 Головний клас взаємодії з об’єктами

Всі об'єкти, з якими гравець може взаємодіяти, досліджуючи ігровий світ, базуються на логіці класу BP_jRPG_InteractionObject_Base. Цей клас визначає основну поведінку, яка є спільною для взаємодії з різними об'єктами: дверима, скринями, неігровими персонажами, магазинами та іншими інтерактивними елементами. Об’єкти класу взаємодії відображено в додатку Г.

Центральним елементом цього класу є InteractionRange, представлений компонентом типу Sphere Component (сферичний компонент). Ця сфера визначає зону, в межах якої можлива взаємодія. Коли гравець потрапляє в цю зону, активується віджет InteractInfo3D, який відображає підказку на екрані. Ця підказка інформує гравця про можливість взаємодії з об'єктом (рис. 2.20).

Рисунок 2.20 – Приклад взаємодії з предметом

Особливістю реалізації є збереження стану взаємодії. Після першого використання об'єкта, наприклад, після відкриття скрині, він додається до масиву Interacted Objects. У цьому масиві зберігається унікальний ідентифікатор (ID) об'єкта. Це забезпечує контроль за тим, щоб повторна

взаємодія не призвела до повторення ефекту. Наприклад, щоб не було повторного отримання того самого предмету.

2.5.2 Скрині та торгівля

Скрині побудовані на основі базового класу взаємодії. Ключовою функціональною особливістю є змінна ItemsInChest, що застосовується у компоненті InventoryManager. Ця змінна визначає, які саме предмети та в якій кількості гравець зможе забрати зі скрині. Після першого відкриття вміст передається до інвентарю гравця (рис. 2.21), а сам об'єкт додається до списку взаємодій, аби уникнути повторного отримання. При наступному відкритті скриня буде вже пустою, і гра повідомить про відсутність будь-яких предметів (рис. 2.22).

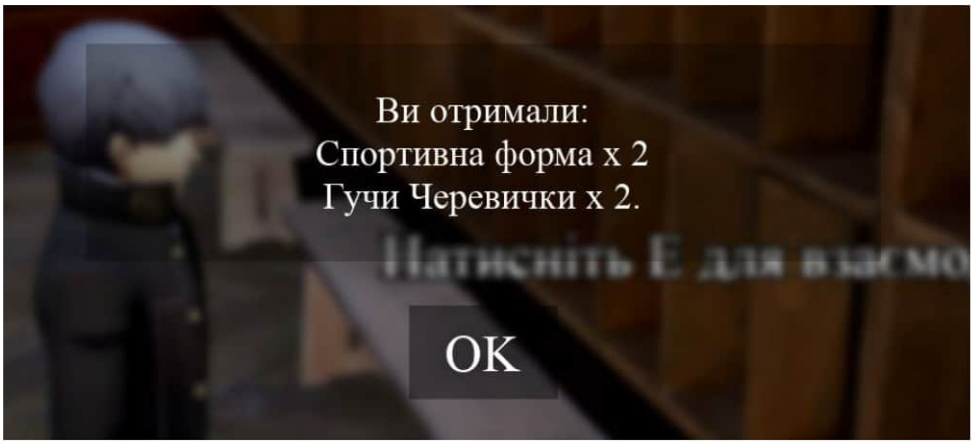


Рисунок 2.21 – Перша взаємодія з скринькою

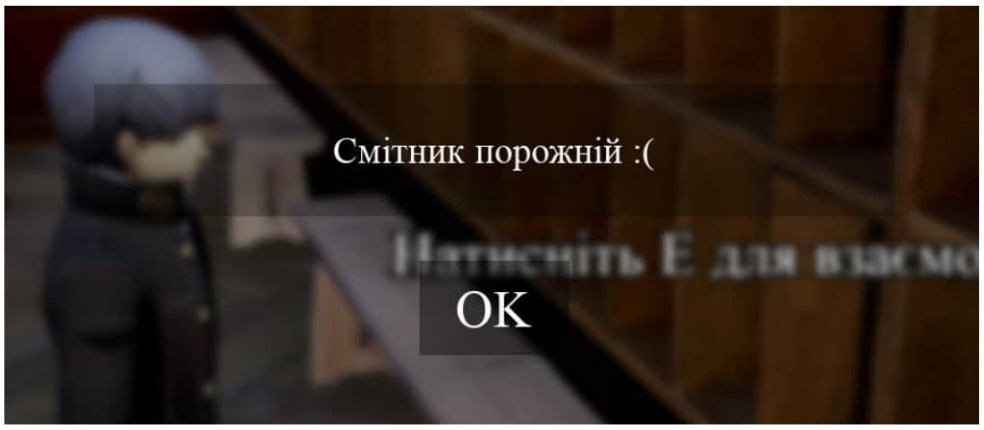


Рисунок 2.22 – Друга взаємодія з скринькою

Механіка торгівлі реалізується через взаємодію з неігровими персонажами (NPC). При взаємодії з торговцем відкривається спеціальне вікно для купівлі чи продажу предметів (рис. 2.23). У торговця зберігається змінна AvailableItems, яка містить перелік предметів, доступних для придбання. Гравець може продати будь-які непотрібні предмети зі свого інвентарю, зазвичай за нижчою ціною, ніж ціна купівлі (рис. 2.24).

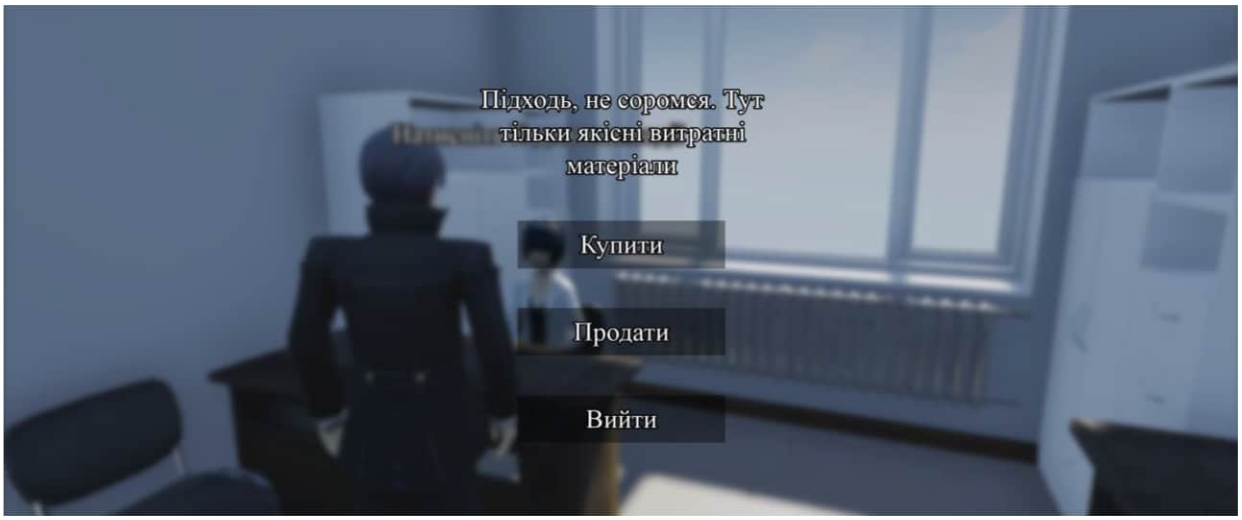


Рисунок 2.23 – Взаємодія з торговцем

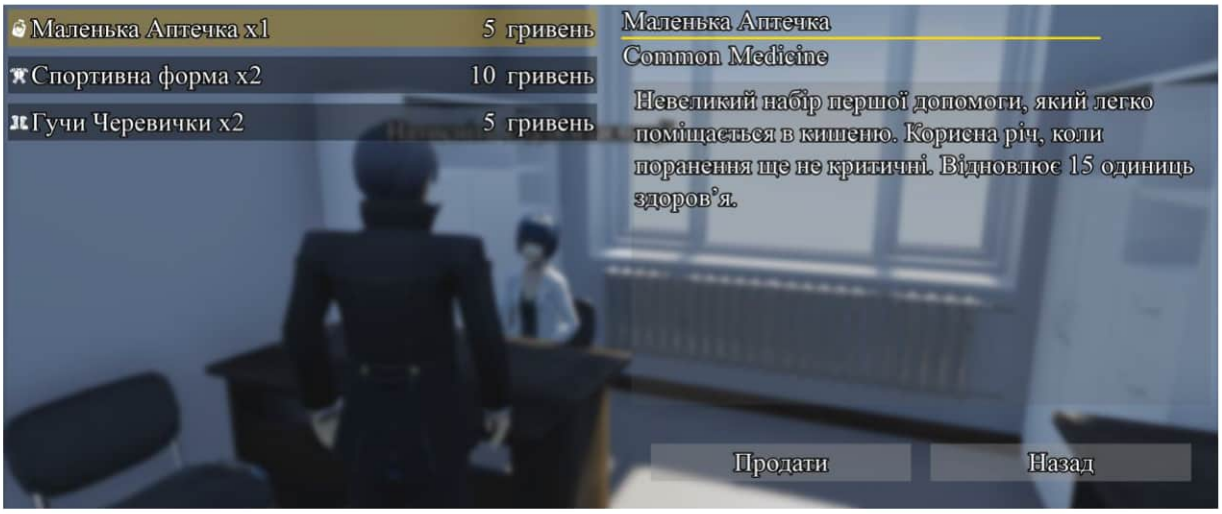


Рисунок 2.24 – Вікно продажу

Інформація про всі ігрові предмети зберігається в таблиці DT_Items, де кожен об'єкт має набір параметрів. Серед них – можливість екіпірування,

використання в бою, вартість покупки та продажу, а також бонуси до характеристик, які застосовуються, якщо це зброя чи обладунки (рис. 2.25).

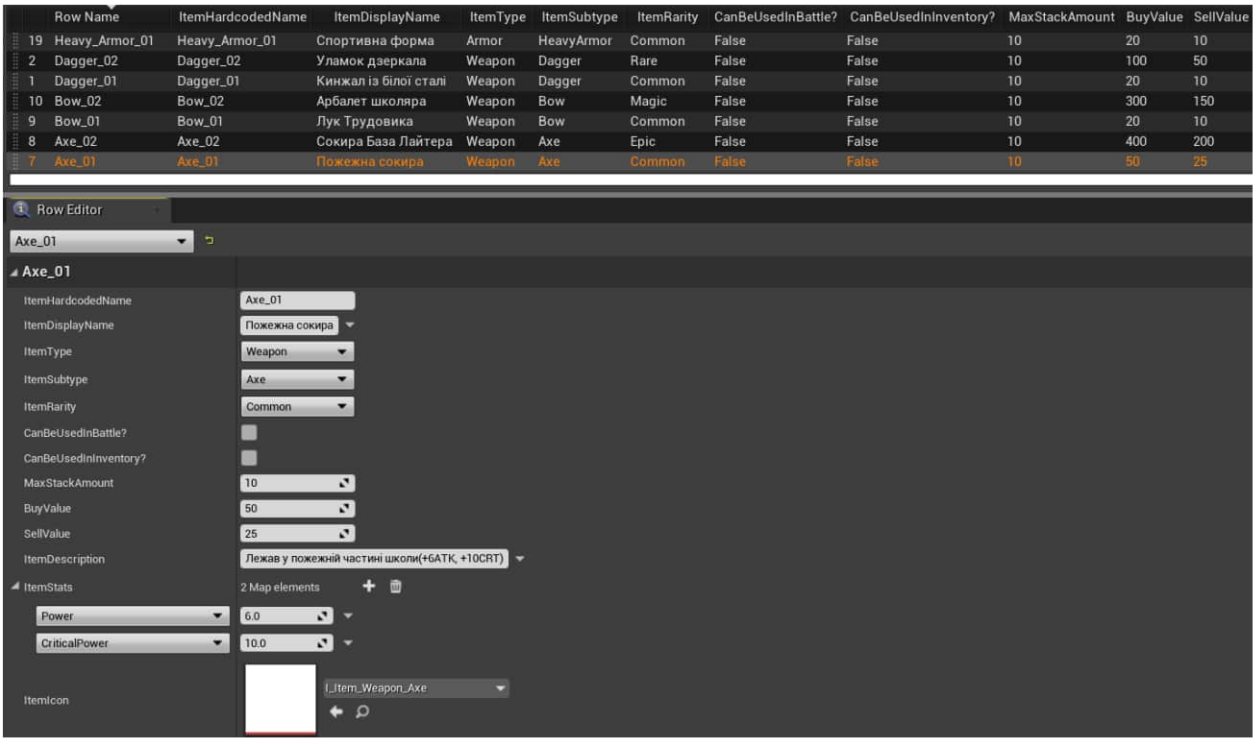


Рисунок 2.25 – Дані про предмет з таблиці DT_Items

2.5.3 Двері

Взаємодія з дверима у грі це ключовий елемент геймплею, що надає змогу пересуватися між просторами, відкривати нові ділянки. Усі двері як об’єкти реалізовані як актори типу BPA_InteractObj, з компонентами, котрі стежать за можливістю взаємодії та коригують свій статус відповідно до умов гри.

У початковому стані, коли гравець наближається до дверей, висвічується спеціальний індикатор, який сповіщає про можливість взаємодії (рис. 2.26). Якщо двері замкнені, система видає відповідне сповіщення, скажімо, «Зачинено» або «Потрібен ключ». У такій ситуації гравець мусить знайти необхідний предмет, щоби відкрити доступ.

Рисунок 2.26 – Двері перед взаємодією

Коли вимоги доступу виконано, двері відчиняються, ініціюючи завантаження нової сцени чи перенесення персонажа у нову область (рис. 2.27). Зміна стану дверей відбувається разом з анімацією затемнення екрану, звуком та тимчасовим блокуванням керування, що гарантує плавний перехід.

Рисунок 2.27 – Нова локація після взаємодії з дверима

2.5.4 Зберігання ігрового прогресу

Для збереження досягнень у геймплеї впроваджено спеціального актора, який виконує роль контрольної точки. Ззовні він постає як звичайний куб, що помітно відрізняється серед решти об'єктів навколишнього середовища.

При наближенні до куба та активації взаємодії відкривається world interface – інтерфейс користувача, котрий з'являється безпосередньо у світі гри. Відразу відобразить відповідну категорію збереження, де можна створити новий слот (рис. 2.28). Після збереження гра фіксує поточний стан прогресу: позицію гравця, гроші, предмети, рівень персонажів, відкриті сундуки та вбиті вороги.

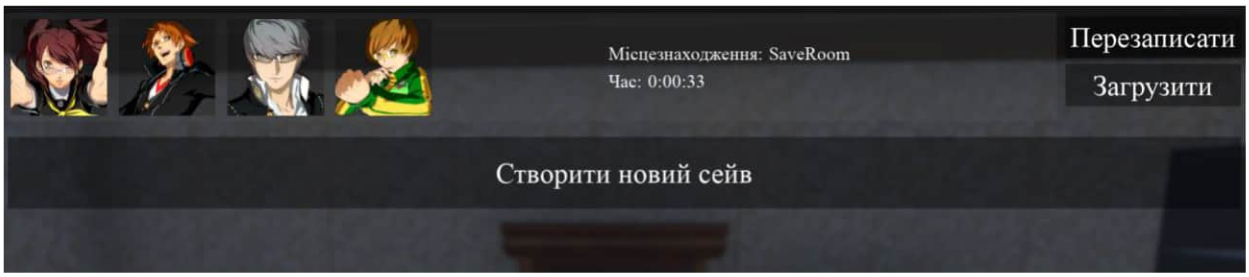


Рисунок 2.28 – Інтерфейс зберігання

2.6 Система діалогів

2.6.1 Реалізація компоненту та інтерфейсу

Система діалогів втілена через компонент BPC_Conversant (табл. 2.7), що відповідає за запуск, завершення та обробку подій діалогу. При активації діалогу створюється інтерфейс, який блокує взаємодію з об'єктами світу, відображає UI та дає змогу обрати відповідь. Логіка діалогів відображено в додатку Д.

Подія Start Conversation запускає початок діалогу. Вона активує введення для гравця, створює та додає на екран віджет WBP_ConversationMain, перемикає режим управління на UI, а також активує курсор миші. Усі об'єкти взаємодії типу BPA_InteractObj приховують свої

компоненти UInteractionComponent, що унеможливлює взаємодію гравця з навколишнім світом до завершення діалогу.

Подія End Conversation повертає гру до звичайного ігрового стану. Камера повертається до гравця, видаляється UI інтерфейс, знову активується введення та геймплейний режим. Приховані компоненти взаємодії стають видимими, дозволяючи гравцеві відновити взаємодію з ігровим середовищем.

Механізм Dialogue Event Trigger дає змогу запускати додаткові події під час діалогу на основі іменованих тригерів. Це, наприклад, може бути запуск анімації, програвання звуку, або активація катсцени.

Для реалізації інтерфейсу використовуються віджети WBP_Conv_Main, WBP_Conv_Answer та WBP_Conv_NPC, які відповідають за відображення реплік NPC, варіантів відповідей гравця та загальну логіку діалогу. У WBP_Conv_Main також реалізовано flipbook-анімацію, яка змінюється в залежності від активного персонажа, дозволяючи візуально відобразити того, хто говорить.

Уся структура діалогу зчитується з таблиці DT_Conversant (рис. 2.29), яка містить ієрархію реплік, їхній текст, ID, співрозмовника та зв'язки між відповідями.

Таблиця 2.7 – Компоненти системи діалогу

Компонент	**Тип**	**Призначення**
BPC_Conversant	Actor Component	Головна логіка запуску/завершення діалогу та тригерів
WBP_Conv_Main	User Widget	Основний UI-інтерфейс діалогу, координує репліки, відповіді та Flipbook-анімацію
WBP_Conv_Answer	User Widget	Кнопки-відповіді гравця з можливістю вибору і звуковим відгуком
WBP_Conv_NPC	User Widget	Відображає репліку NPC у діалоговому інтерфейсі
DT_Conversant	Data Table	Таблиця, що містить усі репліки, їхній текст, ID, зв’язки та мета-дані

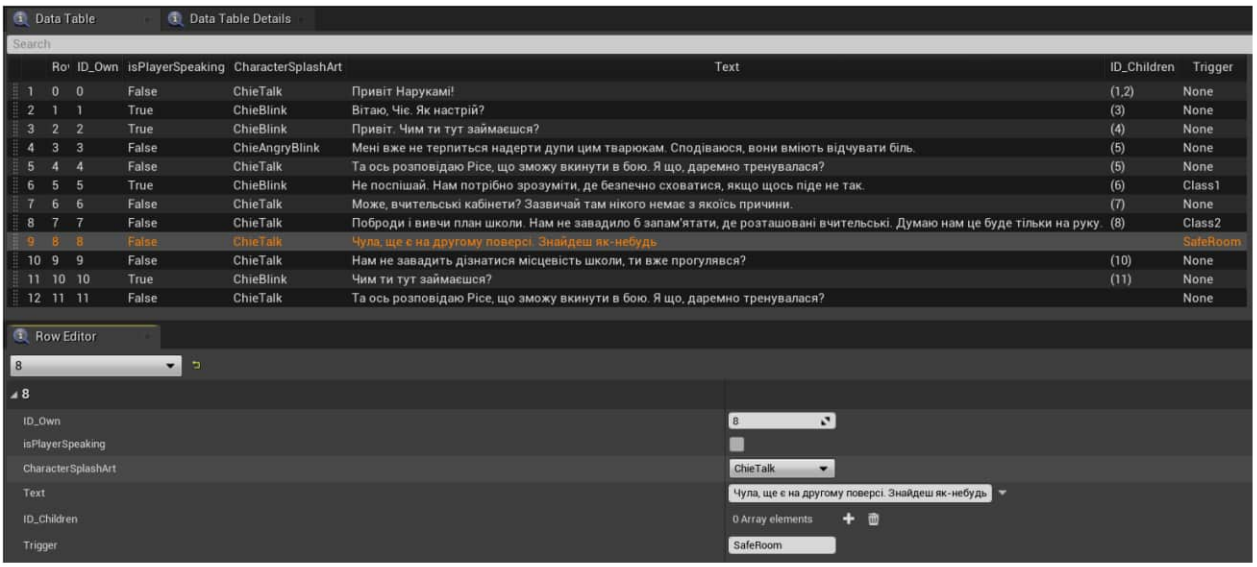


Рисунок 2.29 – Приклад таблиці з діалогом

2.6.2 Демонстрація діалогу

Перш ніж взаємодіяти, гравець бачить персонажа у грі - NPC розташований у визначеному місці та очікує. Коли гравець наближається на потрібну відстань, над NPC з'являється знак (індикатор), який сповіщає про доступність взаємодії (рис. 2.30).

Коли гравець розпочинає діалог, на екрані з'являється діалогове вікно (рис. 2.31). Система автоматично переключає ракурс камери, блокує інші дії в оточенні, показує інтерфейс WBP_ConversationMain з текстом репліки NPC. В цьому вікні також відображається персонаж, що говорить, за допомогою flipbook-анімації.

Коли настає час відповіді гравця, з'являється набір можливих реплік, з яких можна обрати одну (рис. 2.32). Після вибору варіанту гравець впливає на подальший перебіг розмови. Взаємодія супроводжується звуковими ефектами та передачею ідентифікаторів в систему для завантаження наступної гілки діалогу.

В діалогах можуть бути передбачені події, які впливають на ігровий процес. Наприклад на момент, коли під час репліки активується подія, що змінює позицію камери - це реалізовано через Dialogue Event Trigger (рис. 2.33).

Рисунок 2.30 – Персонаж із позначкою доступності діалогу

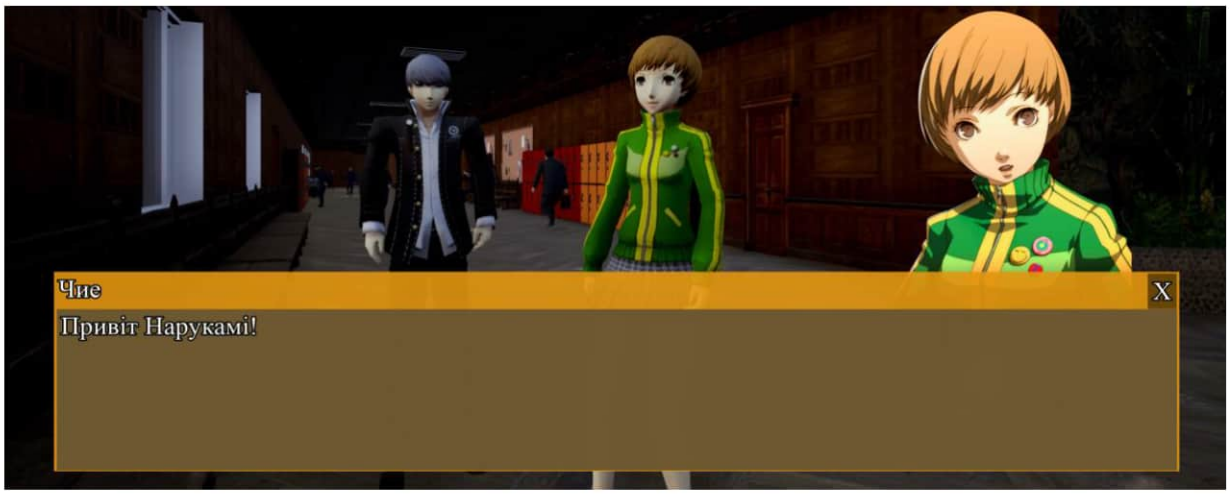


Рисунок 2.31 – Інтерфейс діалогу після початку взаємодії

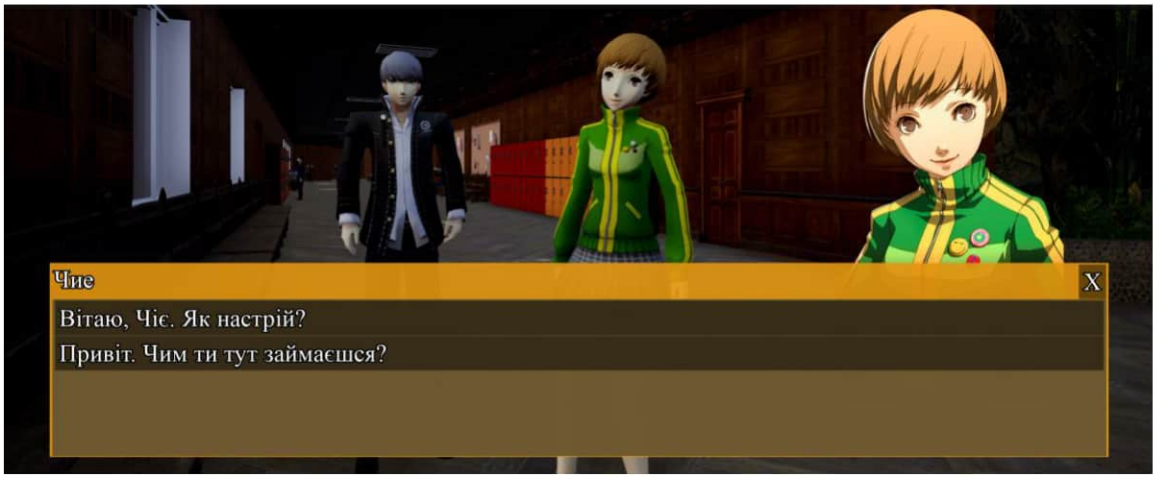


Рисунок 2.32 – Обирання репліки гравцем

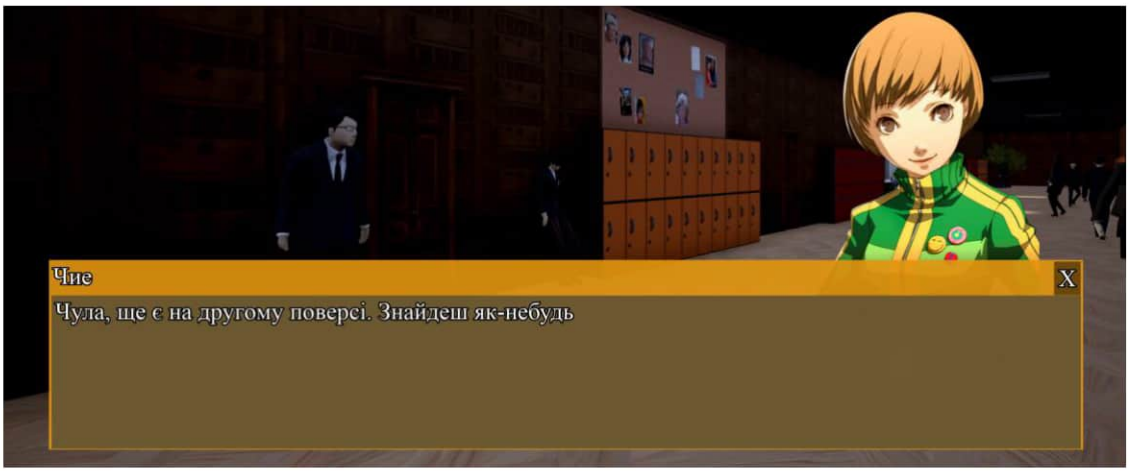


Рисунок 2.33 – Зміна камери під час діалогу

2.7 Штучний інтелект NPC

2.7.1 Концепція роботи ШІ

Штучний інтелект NPC у грі втілений за допомогою гнучкої поведінкової моделі, яка дає змогу персонажам пристосовувати свою поведінку до різних обставин. Основний принцип роботи базується на простих, але дієвих станах: патрулювання, вільне пересування, переслідування гравця та пошук після зникнення з поля зору.

Коли гравець відсутній у полі зору, NPC діє згідно з визначеною поведінкою: якщо є маршрут, він патрулює по точках; якщо маршруту немає, вільно переміщується територією. Якщо ворожий NPC помічає гравця, він починає активне переслідування. У випадку втрати візуального контакту, ворожий NPC прямує до останньої відомої позиції гравця, намагаючись його відшукати.

Не всі неігрові персонажі виявляють ворожість. Звичайні мешканці віртуального світу найчастіше не реагують на появу гравця, займаючись своїми справами або рухаючись заданими маршрутами, що створює відчуття живої локації. Вони можуть ходити, зупинятися, змінювати напрямок руху, але уникають взаємодії, якщо це не передбачено сценарієм конкретної ситуації. Втім, гравець може вплинути на їхню поведінку, перекривши шлях. Тоді NPC автоматично змінить траєкторію, обминаючи перешкоду, аби

дістатися до місця призначення. Загальний алгоритм логіки поведінки ворожих NPC зображено на рисунку 2.34. Реалізоване дерево рішень за цим алгоритмом зображено на рисунку 2.41

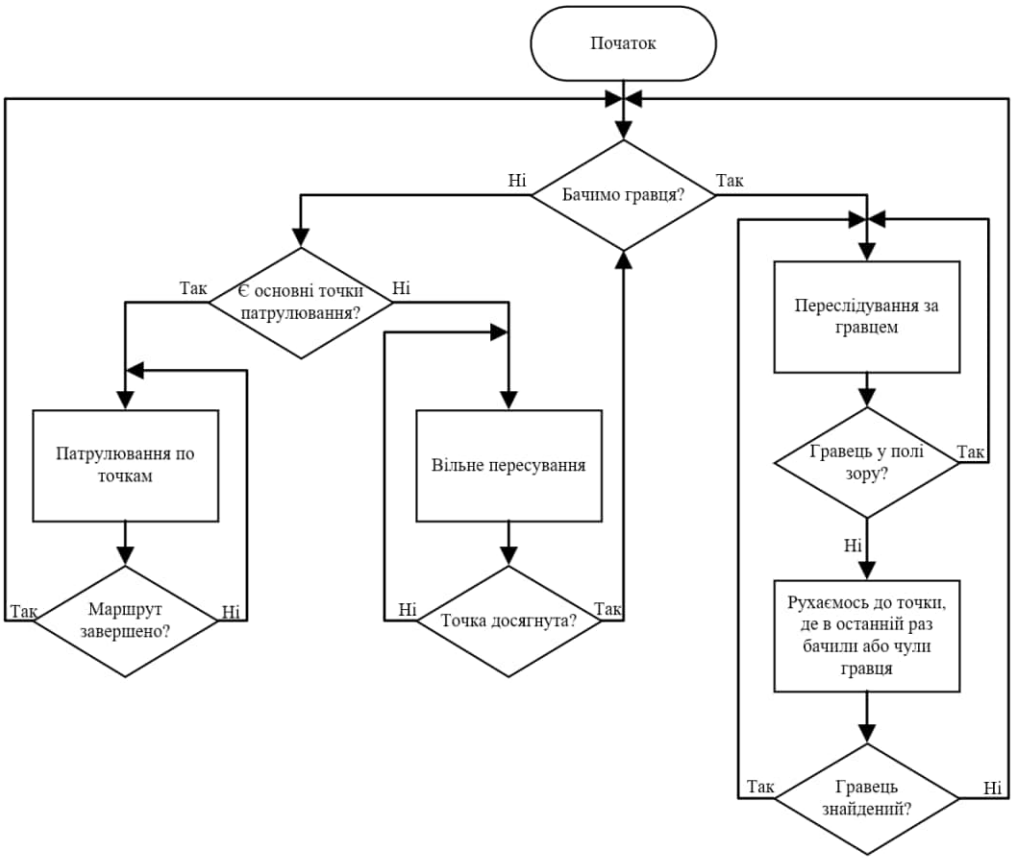


Рисунок 2.34 – Алгоритм поведінки NPC

2.7.2 Інструменти реалізації

Для втілення штучного інтелекту неігрових персонажів у грі, використано стандартні засоби Unreal Engine – Behavior Tree, Blackboard, AI Perception, а також набір власних Task-завдань.

Behavior Tree (дерево поведінки) – це основа для прийняття рішень NPC. Воно функціонує за принципом ієрархії: враховуючи умови та результати виконання попередніх дій, NPC переходить до наступного логічного блоку. Саме дерево визначає послідовність дій персонажа: патрулювання, рух до точки, переслідування гравця тощо.

Blackboard – це структура даних, що містить змінні, важливі для дерева поведінки. Наприклад, гравець як змінна та остання відома позиція гравця. Blackboard виступає посередником між ігровим середовищем та логікою в Behavior Tree.

Завдання (Task) – це базові класи, що визначають конкретні дії у Behavior Tree. Реалізація відбувається через певні події: Event Receive Execute AI (запуск задачі) та Finish Execute (завершення, з показником успіху чи провалу). Task-и дають змогу NPC, наприклад, переміщатися до певної цілі, очікувати, повертатися до початкової точки або виконувати інші дії, що визначаються логікою дерева поведінки.

AI Perception – це система сенсорного сприйняття в Unreal Engine, що дозволяє неігровим персонажам реагувати на зміни у навколишньому світі. У проєкті передбачено використання двох основних типів сенсорів: зору (Sight) та слуху (Hearing) (рис. 2.35).

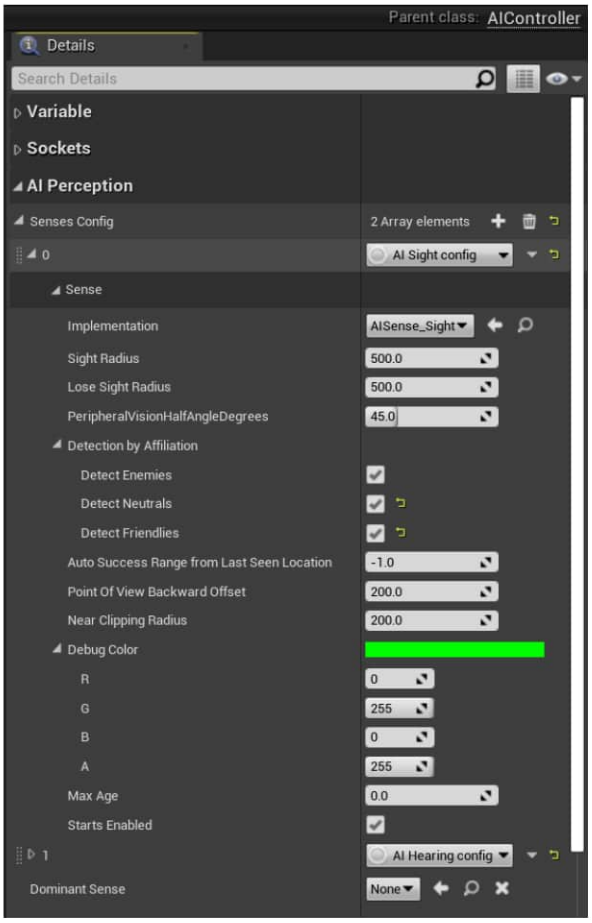


Рисунок 2.35 – Налаштування компоненту AIPerception

Сенсор Sight відповідає за виявлення інших об'єктів у полі зору NPC. Можна регулювати його параметри, включаючи дальність огляду, кут огляду, тривалість утримання цілі після зникнення з поля зору та інше. За допомогою зорового сенсора можна слідкувати за гравцем, іншими NPC або об'єктами в полі зору.

Сенсор Hearing реагує на звуки, які генеруються у світі гри, як-от під час бігу, пострілу або активації певних подій. NPC може отримати інформацію про джерело звуку і направитись до його місцезнаходження, навіть якщо сам об'єкт невидимий.

AI Perception тісно пов’язаний з Blackboard: дані, отримані від сенсорів, записуються у відповідні ключі, що дозволяє дереву поведінки приймати рішення, базуючись на побаченому або почутому.

До NPC можуть бути додані додаткові компоненти, наприклад, компонент патрулювання точками. Якщо NPC має цей компонент патрулювання, Behavior Tree автоматично підлаштовується під це: включається логіка циклічного руху між визначеними точками.

2.7.3 Реалізація штучного інтелекту

Для реалізації логіки поведінки неігрових персонажів розроблено спеціальні задачі (Tasks) у вигляді Blueprint-класів. Ці задачі виконують конкретні дії, що запускаються через події Event Receive Execute AI та завершуються викликом Finish Execute. Основні задачі для штучного інтелекту представлені в таблиці 2.8 та додатку Е.

Для керування неігровими персонажами розроблені спеціалізовані AI Controller-и та класи Character (рис. 2.36). Вони виконують функції пересування та реагування на навколишнє середовище. Ці компоненти встановлюють взаємодію між поведінковим деревом та безпосередньою логікою руху, реакціями на події, а також реалізацією дій в межах ігрового простору. Опис класів BP_NPC та AIController_NPC представлено в таблиці 2.9 – 2.10 та в додатку Е. Структуру класів зображено на рисунках 2.37 – 2.38.

Таблиця 2.8 – Задачі ШІ

Назва задачі	**Тип**	**Опис**
BTTTask_MoveRandomLocation	Користувацький Blueprint-клас	Переміщення до випадкової точки в межах заданого радіусу.
BTTask_Patrol	Користувацький Blueprint-клас	Патрулювання по заздалегідь визначеним точкам згідно з логікою AiBehavior.
BTTask_ClearBlackboardValue	Користувацький Blueprint-клас	Очищення вказаних ключів у Blackboard для скидання стану поведінки.
Move To	Вбудований (Unreal Engine)	Переміщення до вказаної цілі або точки на мапі.
Wait	Вбудований (Unreal Engine)	Затримка на певний час перед виконанням наступної дії.

Рисунок 2.36 – Контролер та класи Character

Таблиця 2.9 – Опис BP_NPC

Назва Компонента	**Тип**	**Призначення**
Capsule Component	Collision Component	Визначає фізичні межі персонажа, використовується для зіткнень
Arrow Component	Arrow Component	Служить для візуального відображення напряму
Mesh	Skeletal Mesh	Відображає тривимірну модель персонажа з анімацією
Character Movement	Movement Component	Керує пересуванням персонажа. Задає швидкість, гравітацію, прискорення
NavigationInvoker	Navigation Component	Активує побудову навігаційної сітки поблизу персонажа під час руху
AIBehavior	Blueprint Component	Дозволяє задавати точки патрулювання для NPC
OptimizationProxy	Blueprint Component	Відповідає за оптимізацію продуктивності, відключаючи логіку ШІ поза полем зору гравця.

Таблиця 2.10 – Опис AI_Controller_NPC

Назва Компонента	**Тип**	**Призначення**
Transform Component (Inherited)	Scene Component	Містить інформацію про позицію, обертання та масштаб контролера у світі.
Path Following Component (Inherited)	AI Path Component	Відповідає за пересування агента за побудованим маршрутом навігації.
Actions Comp (Inherited)	Pawn Actions Component	Використовується для керування діями AI (наприклад, MoveTo, Wait, Sequence тощо).
AIPerception	AI Perception Component	Обробляє зорові та слухові стимули, реагуючи на появу гравця або звуки.

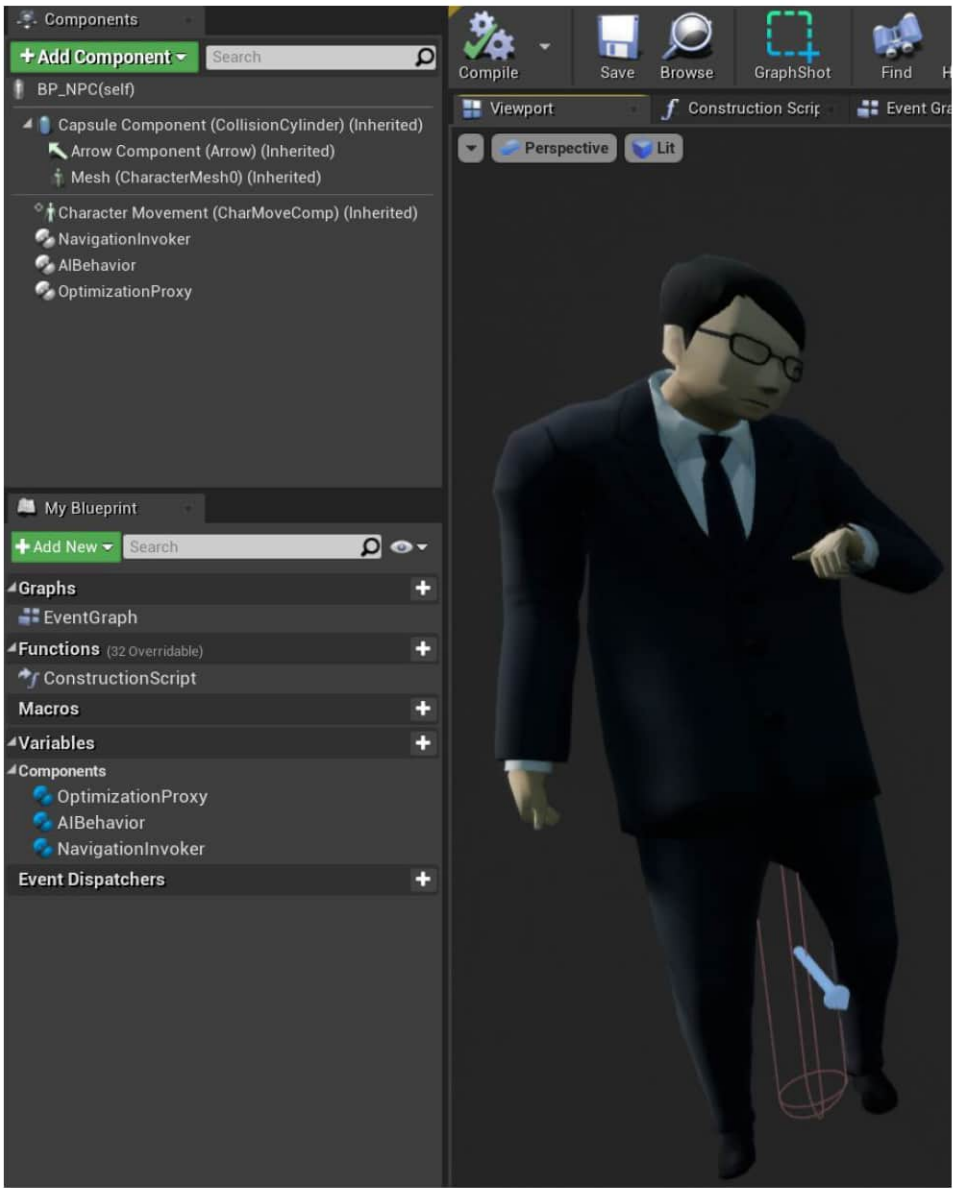


Рисунок 2.37 – Клас BP_NPC

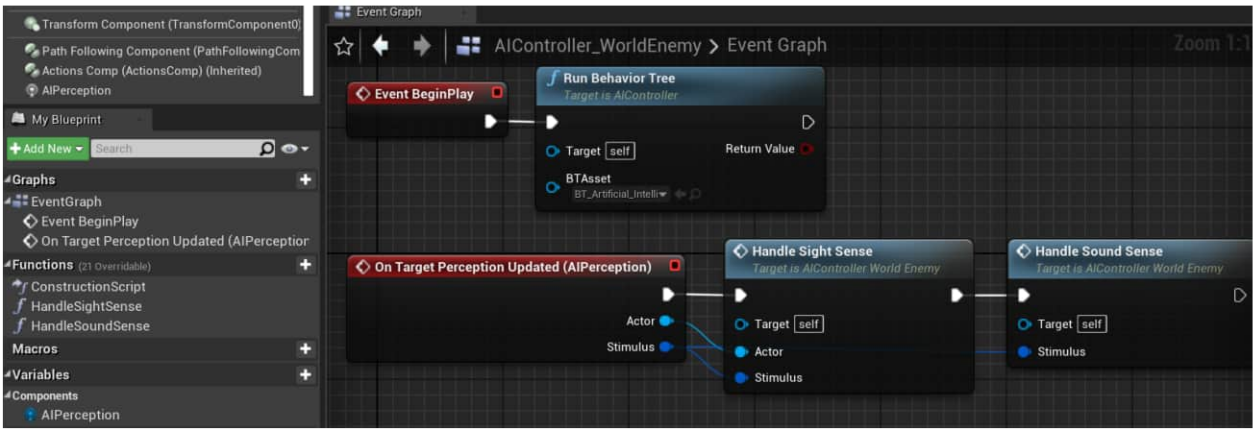


Рисунок 2.38 – Клас AI_Controller_NPC

Окремою складовою реалізації є компонент AiBehavior, який дозволяє визначати шляхи патрулювання для NPC, а також налаштовувати їх поведінку на цих маршрутах . У режимі Loop персонаж рухається визначеним маршрутом циклічно, безперервно повторюючи його. Після досягнення останньої точки, він стартує спочатку. У випадку Return, NPC патрулює маршрут у двох напрямках: після досягнення кінцевої точки, рух відбувається у зворотному порядку. Режим None передбачає одноразове проходження маршруту, після чого персонаж зупиняється і припиняє рух. Також впроваджено налаштування часу для конкретної точки, скільки NPC буде очікувати на різних частинах маршруту. Налаштування компоненту зображено на рисунку 2.39.

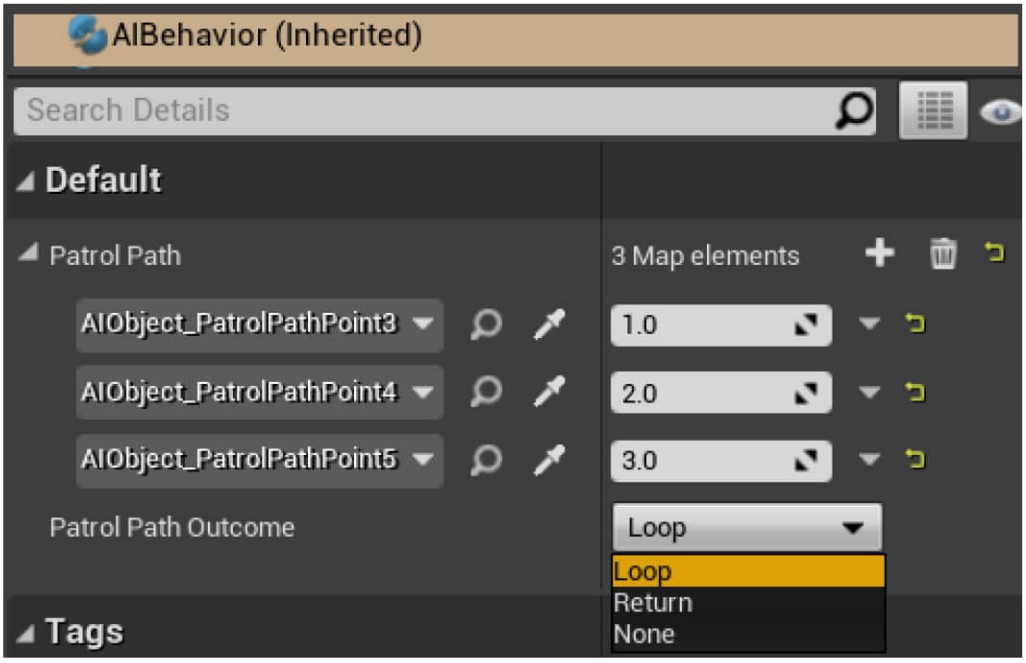


Рисунок 2.39 – Налаштування компоненту AIBehavior

Для встановлення взаємодії між логікою поведінки дерева та зовнішнім середовищем, розроблено Blackboard Keys. Вони виступають як змінні, що зберігають важливі дані про об'єкти, цілі чи координати. Ці ключі є складовою частиною дерева рішень та широко застосовуються при прийнятті рішень NPC, під час патрулювання, виявлення гравців та вибору відповідної поведінки. Огляд ключів зображено в таблиці 2.11 та на рисунку 2.40.

Таблиця 2.11 – BlackBoard ключі

Назва ключа	**Тип**	**Призначення**
SelfActor	Object	Стандартний ключ, посилання на самого себе, що автоматично додається Unreal Engine
TargetLocation	Vector	Координати нашого гравця, які створюються під час кроків гравця
TargetActor	Object	Об'єкт типу гравця (BP_Character_World), на який NPC фокусує свою увагу при виявленні
LastKnownLocation	Vector	Остання відома позиція гравця, яку NPC зафіксував перед втратою з поля зору

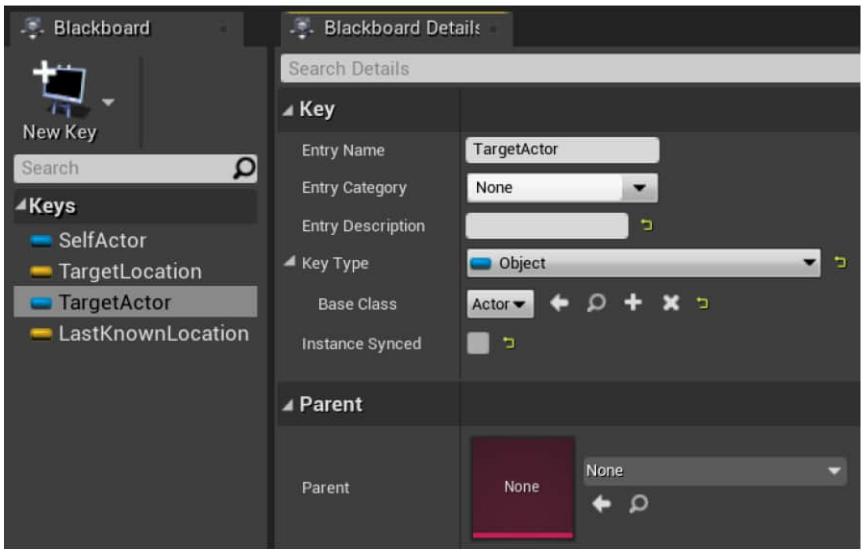


Рисунок 2.40 – Ключі в BlackBoard

У дереві рішень, що зображена на рисунку 2.41, для ворогів реалізовано розгалужену структуру, що бере початок з вузла Root. Цей вузол виконує роль відправної точки, звідки активується вся логіка ШІ. Далі дерево розгортається до основних вузлів, серед яких Sequence, Selector та Blackboard Based Condition, що забезпечують NPC можливість приймати адаптивні рішення залежно від поточної ситуації в ігровому просторі.

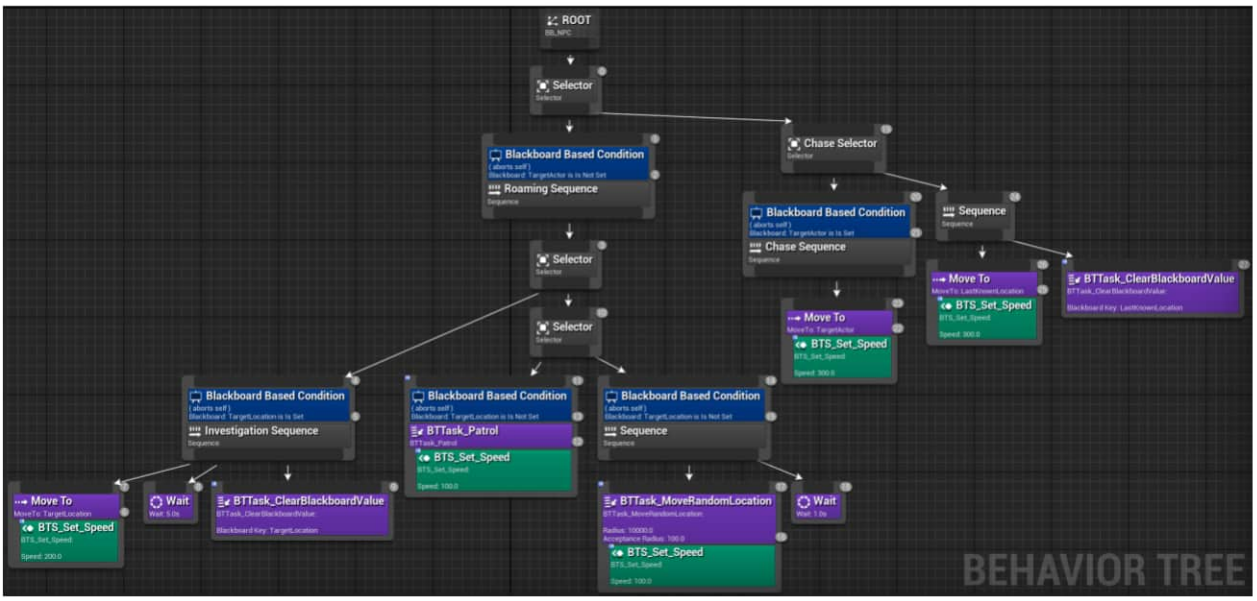


Рисунок 2.41 – Дерево рішень ворожих NPC

Вузол Sequence застосовується для послідовного виконання завдань: кожне наступне завдання запускається лише після успішного завершення попереднього. У випадку, якщо хоча б одне із завдань повертає результат Failed, весь Sequence завершується невдало, і ШІ перемикається на іншу гілку логіки.

Selector, на відміну від Sequence, призначений для вибору однієї з можливих дій. Він проходить через свої дочірні елементи до тих пір, поки один з них не завершиться успішно. Це дозволяє NPC реагувати на зміну обставин: якщо гравець помітний – атакувати, якщо тільки чутно – перевірити джерело звуку, якщо жодних змін – продовжувати патрулювання.

Blackboard Based Condition використовується для перевірки даних у Blackboard. Це дозволяє запускати конкретні дії тільки за наявності певних умов. Наприклад, якщо в ключі TargetActor збережено об’єкт гравця, активується поведінка переслідування. Якщо ж цей ключ відсутній або порожній, відповідна гілка дерева не активується. У розробленому дереві цей тип перевірки застосовується майже перед кожним завданням, забезпечуючи постійний моніторинг стану NPC протягом усієї гри.

На відміну від ворожих персонажів, що керуються ШІ, звичайні NPC не вдаються до алгоритмів сприйняття. Їхній функціонал обмежений простою

структурою дій, що відповідає виключно за переміщення локацією за заданим шляхом. Їхня програмна логіка не включає активної взаємодії з гравцем, проте гравець може вплинути на маршрут неігрового персонажа, перекривши дорогу. Спрощену версію дерева для дружніх NPC зображено на рисунку 2.42.

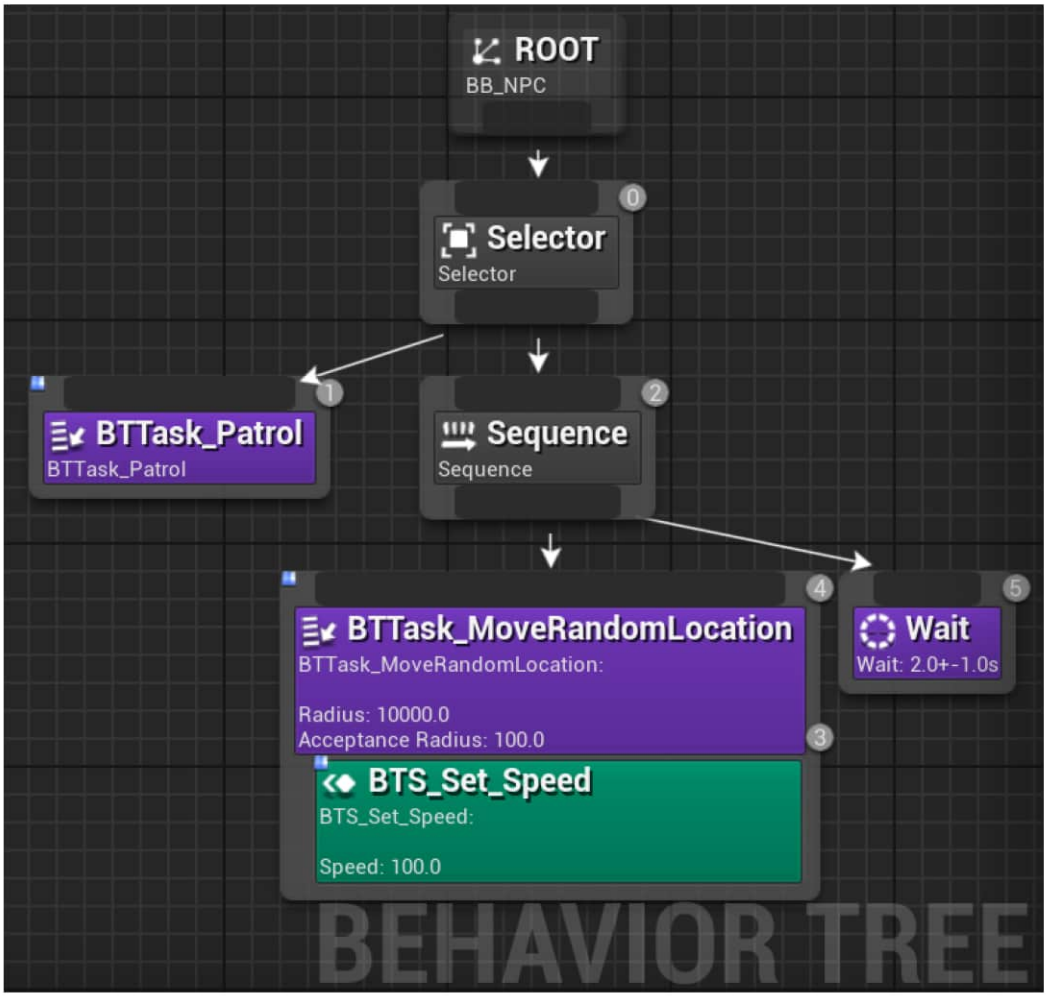


Рисунок 2.42 – Дерево рішень звичайних NPC

2.7.4 Демонстрація роботи ШІ

Персонажі використовують навігаційну сітку NavMesh (Мережа Навігації). Це спеціалізована система, що будується на основі геометрії рівня та визначає ділянки, де персонажі здатні рухатися. У вікні редактора, при активації режиму "Navigation", вона зображається у вигляді зеленої сітки. Саме на основі цієї сітки NPC обирають найкращий шлях до своєї цілі,

оминаючи перешкоди та враховуючи особливості руху. На рисунку 2.43 можна побачити, як звичайні NPC пересуваються, використовуючи NavMesh.

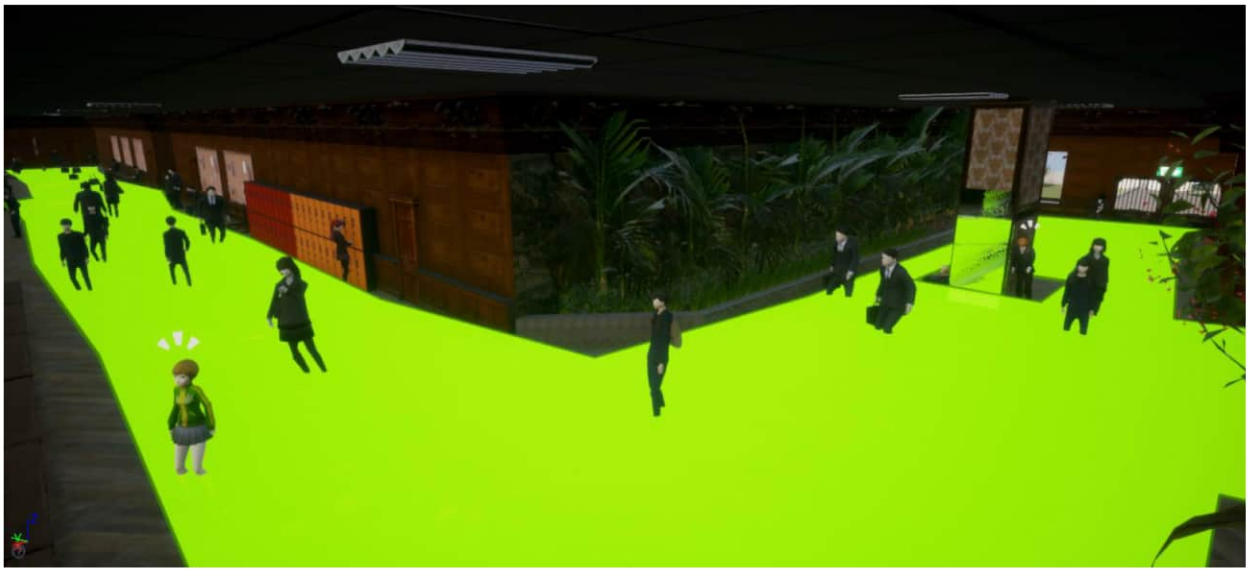


Рисунок 2.43 – Неігрові персонажі рухаються по NavMesh

Розглянемо приклад ворога, котрий використовує систему AI Perception для реагування на гравця. Навколо нього розміщено кольорові сфери, котрі демонструють зони сприйняття. Зелена сфера показує область зорового сприйняття (Sight), тобто поле зору NPC. Жовта сфера представляє зону слуху (Hearing), де AI здатний чути звуки, наприклад, кроки гравця. Якщо гравець потрапляє в одну з цих зон, система реєструє його як ціль для спостереження або переслідування.

При активації debug-режиму через AI Debug (наприклад, клавіша “`” або комбінація клавіш), розробник може побачити повну статистику обраного неігрового персонажа: яка поведінка виконується, чи бачить він гравця, які саме сенсори спрацювали, а також візуалізацію зон зору та слуху. Рисунок 2.44 демонструє зони сприйняття ворога за допомогою AI Perception.

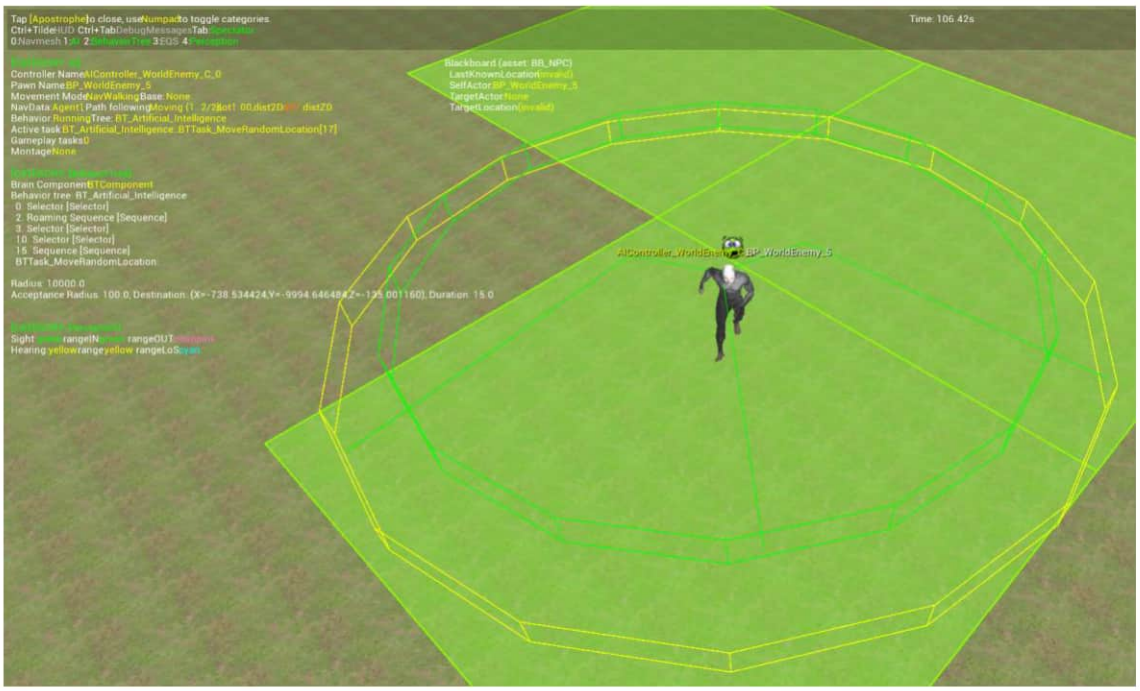


Рисунок 2.44 – Зони сприйняття AI Perception

Коли гравець опиняється у полі зору або в зоні чутності, AI Perception активується, виділяючи ціль спеціальною сферою – зеленою (зір) або жовтою (слух), в залежності від того, який канал сприйняття зафіксував гравця. Після виявлення гравця, NPC отримує інформацію про його положення та запускає поведінкове дерево для переслідування. На рисунку 2.45 показано момент, коли спрацьовує сприйняття і розпочинається переслідування гравця.

Рисунок 2.45 – Реагування ворога за допомогою AI Perception

2.8 Система звукового супроводу

У процесі створення гри впроваджено повноцінну систему аудіосупроводу, яка включає в себе музику. Unreal Engine надає підтримку різних форматів аудіо, серед яких найпоширеніші - це .wav, .ogg та .flac. Для досягнення найвищої якості та сумісності, всі звуки використовуються у форматі .wav.

Ключовим елементом системи є Sound Cue – специфічний аудіо-ресурс, який дозволяє конструювати складні ланцюжки програвання звуку за допомогою візуального інтерфейсу. Зокрема, за допомогою Sound Cue втілено динамічне перемикання між музичними фрагментами. Перша частина композиції відтворюється один раз, після чого відбувається перехід до другої частини, яка програється в циклічному режимі (рис. 2.46).

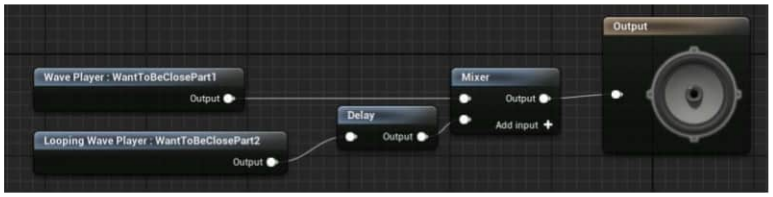


Рисунок 2.46 – Фрагмент музики через Sound Cue

Для організації відтворення музики згідно з локацією, використовується спеціальний Blueprint-актор, який з'являється у світі під час завантаження рівня. У його рамках реалізована логіка вибору треку: з масиву доступних музичних тем випадково обирається одна композиція, яка запускається для конкретної області гри (рис. 2.47). Головний клас відтворення музики відображенно в додатку Ж.

Рисунок 2.47 – Випадкове відтворення музики через масив

Окремо реалізовано звуки інтерфейсу. При натисканні на кнопки, відкритті меню або зміні налаштувань UI елементи видають характерні звуки, що сприяють зворотньому зв'язку з користувачем і покращують загальне відчуття від взаємодії.

Втілено функцію Footstep (рис. 2.48), що відповідає за відтворення звуків кроків. Цю функцію інтегровано безпосередньо в Anim Blueprint головного персонажа, що забезпечує точну синхронізацію звуку з фазою анімації ходьби. Звуки змінюються в залежності від поверхні, по якій рухається персонаж (скажімо, плитка, дерево, трава). Додатково, кожен крок активує Pawn Noise Emitter, який генерує відповідний шум та передає його системі сприйняття штучного інтелекту – AI Perception. Відповідно, якщо ворог розташований в межах радіусу чутності, він здатний почути кроки гравця та почати переслідування.

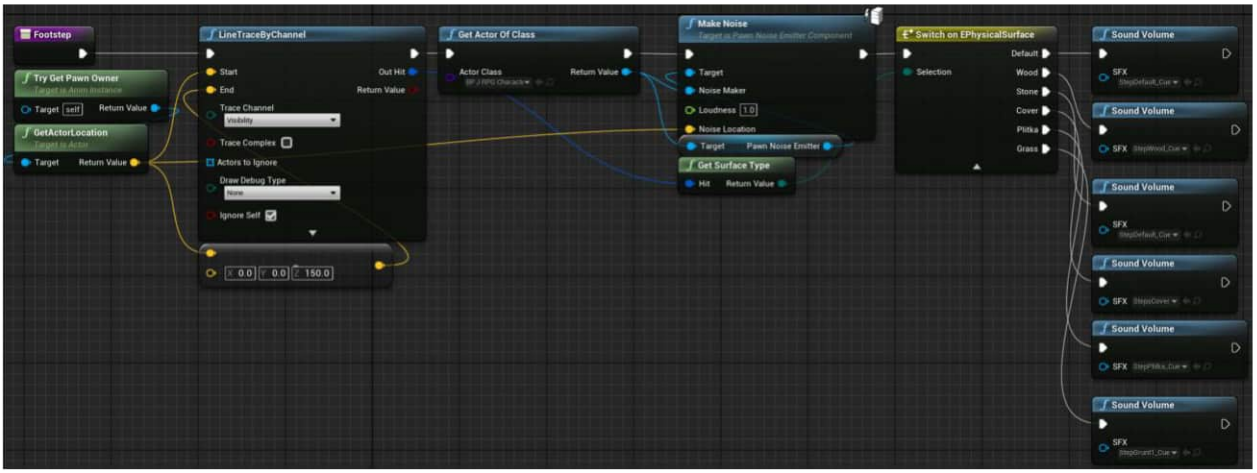


Рисунок 2.48 – Функція звукового супроводу кроків

2.9 Тестування гри

2.9.1 Мінімальні системні вимоги

Щоб гра коректно функціонувала без проблем, необхідно, аби комп’ютер користувача відповідав мінімальним технічним параметрам. Дотримання цих вимог допомагає запобігти критичним помилкам, що можуть

виникнути через низьку продуктивність, невідповідність компонентів або брак ресурсів.

Враховуючи, що гра базується на рушії Unreal Engine 4, обов’язково потрібно встановити додаткові бібліотеки – Microsoft Visual C++ Redistributable Packages for Visual Studio 2015–2022. Без цих компонентів буде неможливо запустити готовий програмний продукт. Мінімальні системні вимоги зазначені в таблиці 2.12

Таблиця 2.12 – Мінімальні системні вимоги для запуску гри

Компонент	**Характеристика**
Операційна система	Windows 10 64-bit
Процесор	AMD FX 8350 або Intel Core I5-6500
Оперативна пам’ять	8 ГБ
Відеокарта	NVIDIA GTX 660 або AMD Vega 10. Мінімум 2 ГБ VRAM
DirectX	Версія 11
Вільне місце на диску	5 Гб

2.9.2 Основні засоби тестування

Під час створення гри надзвичайно важливим етапом є тестування, тобто процес виявлення помилок, непередбачуваної роботи чи некоректної логіки. Unreal Engine 4 пропонує розробникам значний арсенал засобів, що суттєво полегшують налагодження ігрових механік.

Одним з найбільш простих та корисних засобів для елементарного тестування у Unreal Engine є Print String. Ця функція надає можливість відображати текстові повідомлення прямо на екрані під час гри. До неї можливо приєднувати змінні різноманітних типів, як-от текстові (String), числові (Float, Integer), логічні (Boolean), вектори (Vector), об'єкти та інші. Unreal автоматично трансформує значення змінної в текстовий формат. Це дозволяє виводити у реальному часі, наприклад, позицію персонажа, стан сприйняття штучного інтелекту, показники здоров'я або результат логічного виразу, що сприяє контролю правильності виконання коду та загальної логіки

поведінки гри. Приклад використання функції зображено на рисунках 2.49 та 2.50

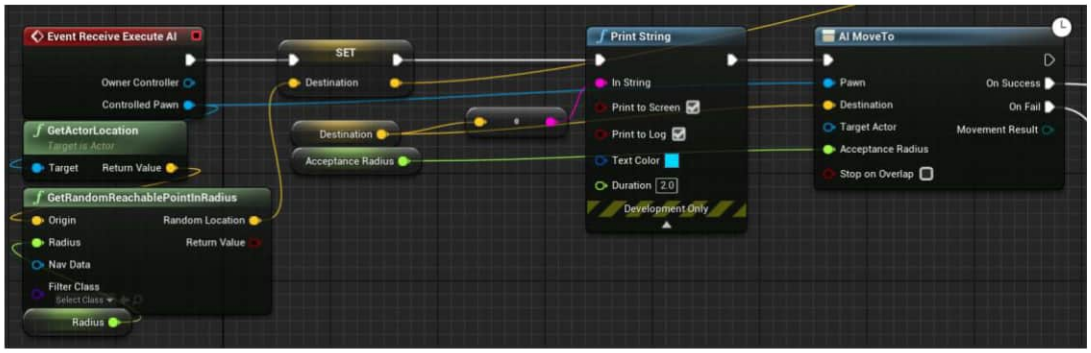


Рисунок 2.49 – Приклад використання функції Print String

Рисунок 2.50 – Виведення координат точки за допомогою Print String

Не менш активно використовується Draw Debug (рис. 2.51) – набір функцій для візуалізації елементів безпосередньо у ігровому світі (лінії, сфери, стрілки та інші фігури). Наприклад, можна показати трасування перед персонажем або ж наочно відобразити радіус огляду ворога.

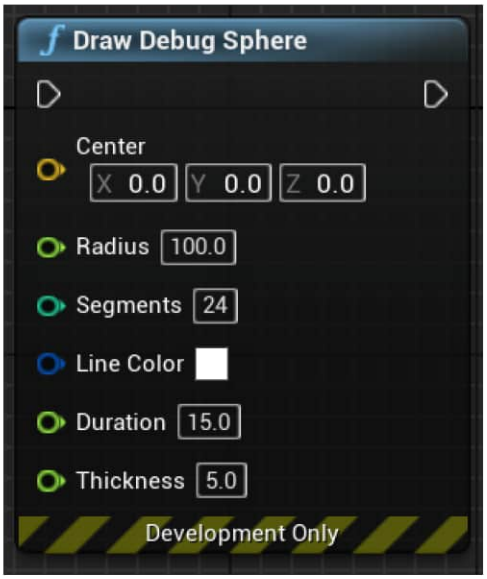


Рисунок 2.51 – Функція Draw Debug

Важливо привернути увагу на режимах перегляду (View Mode) у вікні Viewport. Вони надають можливість перемикатися між різноманітними варіантами візуалізації сцени, що є надзвичайно корисним при діагностуванні графіки та оптимізації. Наприклад, режим Lit демонструє сцену з усім освітленням, тінями й текстурами, тобто максимально реалістично, наближено до фінального зображення гри. У режимі Unlit показуються лише базові кольори матеріалів, без будь-якого освітлення, що дозволяє сфокусуватися на геометрії об’єктів, без впливу тіней і світла. Wireframe дає змогу побачити полігональну структуру моделей, що важливо при перевірці геометрії та щільності сітки. Також доступні режими Light Complexity, Shader Complexity, Reflections та інші, які показують складність освітлення, навантаження від шейдерів та роботу систем відображення відповідно. Ці режими зображені на рисунку 2.52

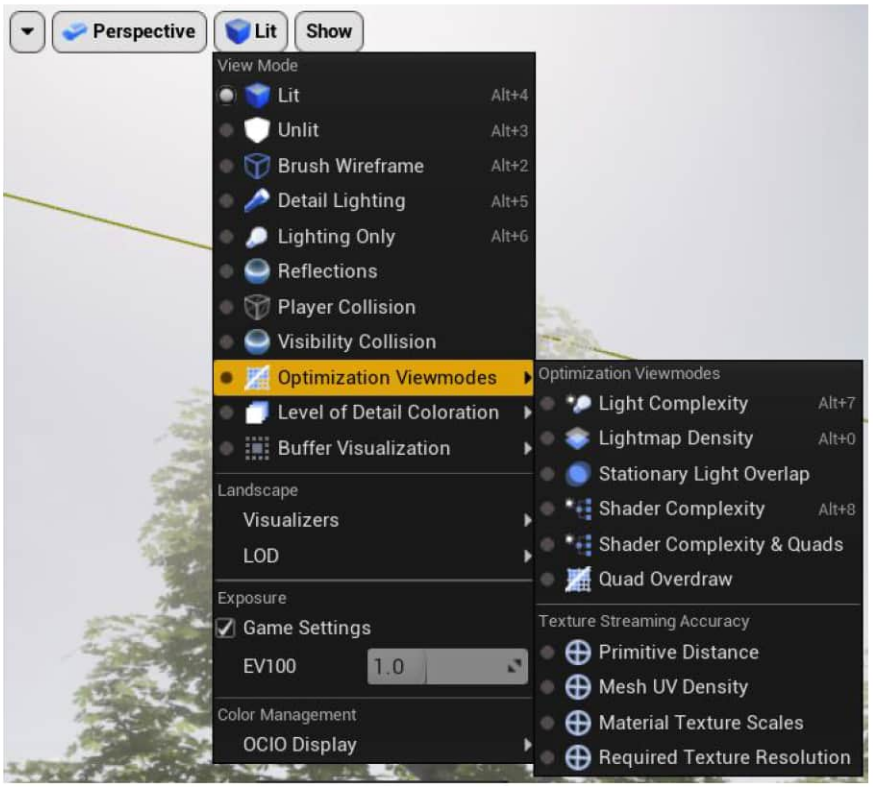


Рисунок 2.52 – Режими відображення во Viewport

У фінальній версії гри (Shipping), весь набір інструментів розробника, зокрема Print String, візуалізація Debug, варіанти View Mode, а також

діагностичні команди, є неактивними. Таке рішення спрямоване на досягнення максимальної ефективності та гарантування безпеки кінцевого продукту. Якщо ж виникає необхідність використати ці інструменти під час тестування або показу гри, гру можна зібрати в режимах Debug або Development. У цих режимах усі згадані функції залишаються активними, що суттєво полегшує виявлення та виправлення помилок. Варіанти збірки гри зображено на рисунку 2.53.

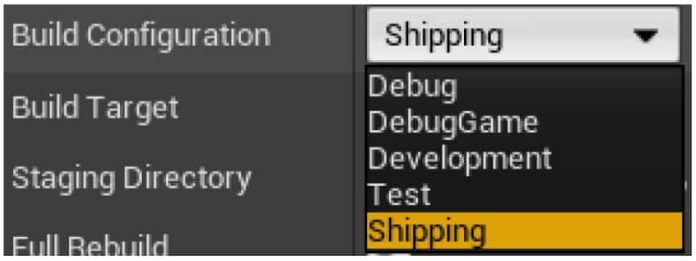


Рисунок 2.53 – Режими збірки ігрового додатку

2.9.3 – Тестування ШІ

Головний метод перевірки логіки штучного інтелекту полягає у візуалізації створених цілей та контролі виконання дій через повідомлення. З цією метою широко використовувалися інструменти Print String та Draw Debug Sphere. Коли ШІ отримував нову ціль, наприклад, під час патрулювання чи переміщення до випадкового місця, в цій точці з'являлась кольорова сфера, що показувала позицію, до якої має дістатися NPC.

Разом з тим через Print String виводилась інформація про сам процес, підтвердження успішного досягнення. Як тільки NPC досягав заданої позиції, сфера автоматично зникала, сигналізуючи про завершення дії та очищення стану у Blackboard. У випадку, якщо ШІ не досягнув заданої точки, функція повертало провал. Приклад та результати тестування зображено на рисунках 2.54 – 2.57.

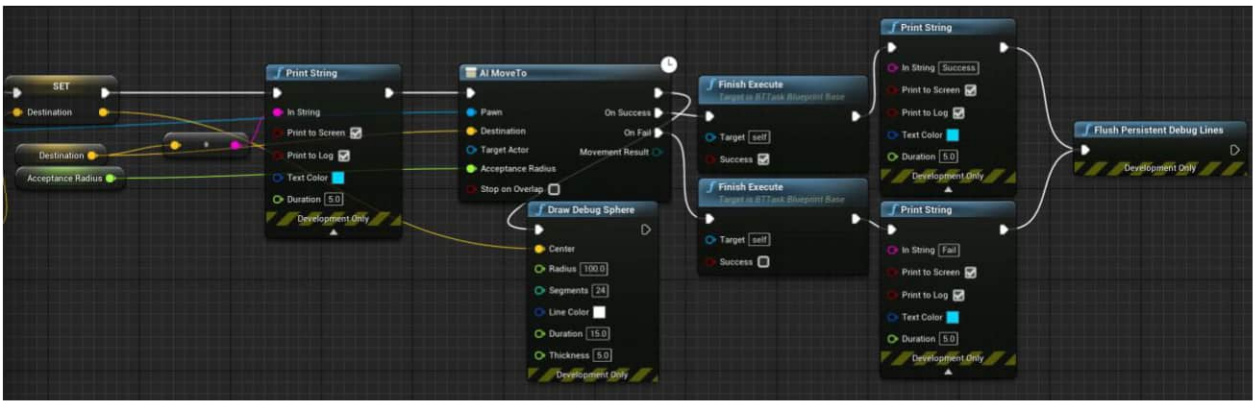


Рисунок 2.54 – Приклад функцій тестування

Рисунок 2.55 – Згенерована точка та її кординати

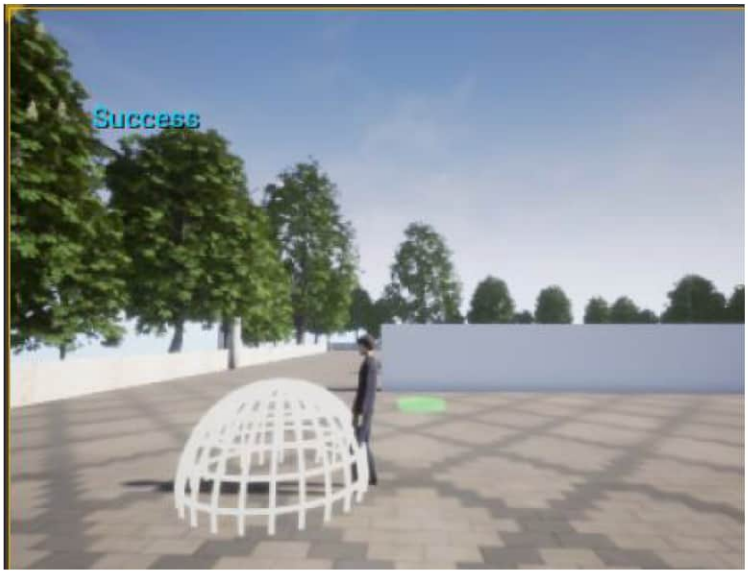


Рисунок 2.56 – Результат успішного досягнення точки

Рисунок 2.57 – Результат провального досягнення точки

2.9.4 Оптимізація ігрового додатку

Оптимізація - це ключовий етап розробки ігор, оскільки вона допомагає знизити навантаження на комп'ютер, забезпечити стабільну частоту кадрів та покращити продуктивність навіть на слабших машинах. Певні процеси в ігровому світі можуть споживати багато ресурсів, якщо не підходити до них з розумом.

Перший приклад - це подія Tick, яка запускається кожен кадр, тобто десятки разів на секунду, залежно від кількості кадрів на секунду (FPS). Хоча Tick зручний для постійного оновлення логіки, надмірне його використання у багатьох об'єктах може суттєво навантажувати процесор. Особливо це стосується об'єктів, які не потребують частого оновлення або не є критичними. У таких випадках краще використовувати систему таймерів (Timer), яка дозволяє викликати функцію один раз або через певний проміжок часу, коли це дійсно необхідно. Це зменшує кількість обчислень та економить ресурси. Приклад реалізації логіки через Timer зображено на рисунку 2.58

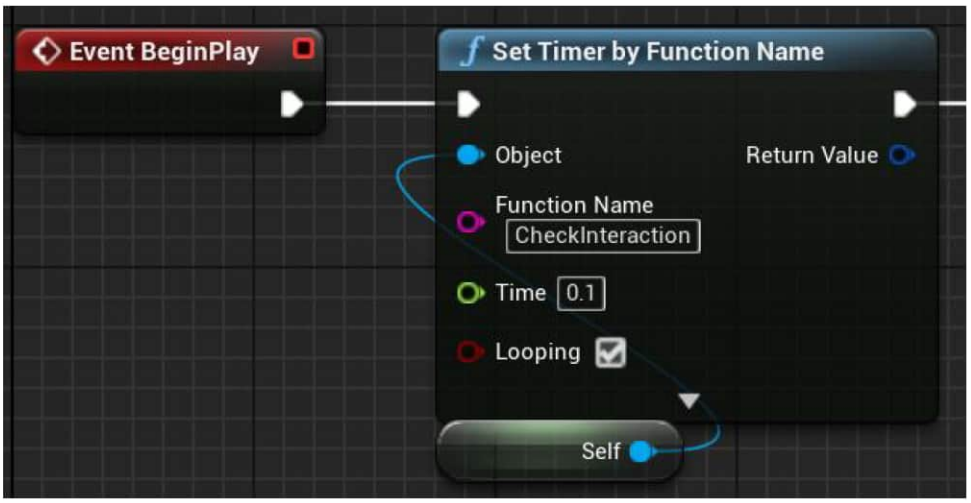


Рисунок 2.58 – Використання функції взаємодії через Timer

Другим способом оптимізації є використання рівнів деталізації (Level of Detail, LOD). Ця технологія дозволяє призначити для кожної 3D-моделі кілька версій з різною кількістю полігонів, які автоматично перемикаються залежно від відстані до камери. Таким чином, об'єкти, що знаходяться далеко від гравця, відображаються з меншою деталізацією, що значно знижує навантаження на графічний процесор. Налаштування LOD для статичних мешів (Static Mesh) розташоване у відповідному вікні властивостей об'єкта. Приклад використання LOD зображено на рисунках 2.59 та 2.60.

Рисунок 2.59 – Дерево без оптимізації LOD

Рисунок 2.60 – Дерево з оптимізацією LOD

2.10 Інтерфейс користувача

2.10.1 Головне меню та налаштування гри

Після запуску гри користувач опиняється у головному меню – це його перша точка контакту з грою. Меню зроблене у вигляді окремого екрану, де розташовані інтерактивні кнопки для різних дій: старт нової гри, відновлення попереднього прогресу, перехід до налаштувань та вихід з гри. Всі ці функції реалізуються за допомогою Blueprint Widget, а обробка кліків та натискань відбувається через стандартні події OnClicked. Демонстрація головного меню та їх варіації зображено на рисунках 2.61 – 2.63.

Рисунок 2.61 – Головне меню гри

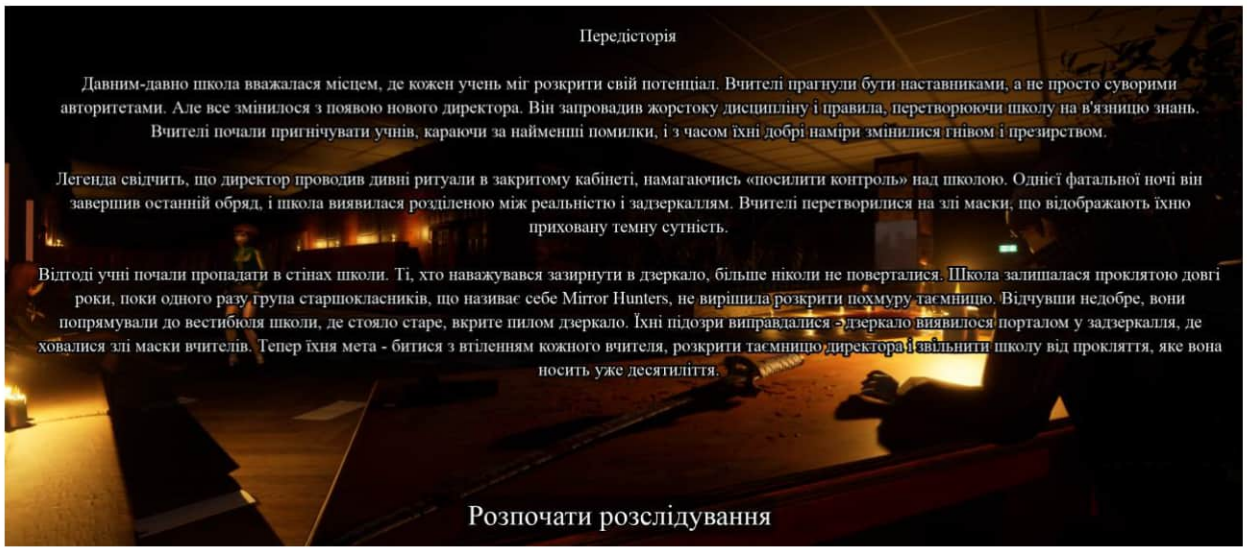


Рисунок 2.62 – Початок нової гри

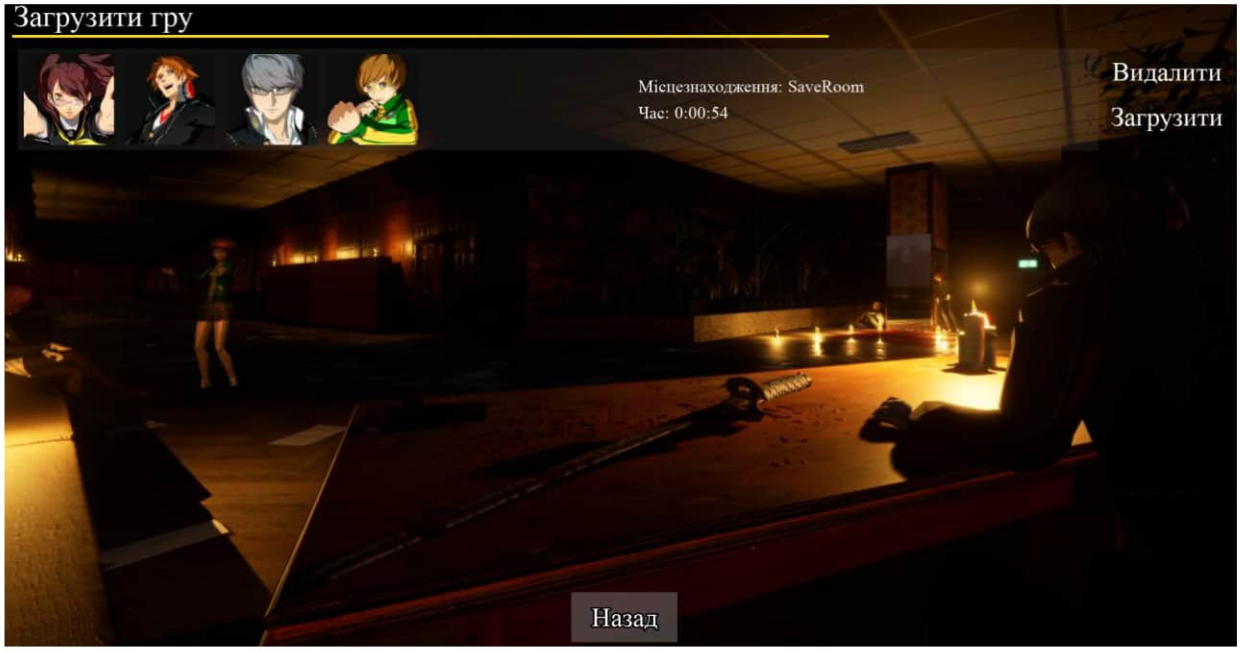


Рисунок 2.63 – Інтерфейс завантаження стану гри

Система налаштувань створена з використанням плагіна Auto Settings [12], який можна знайти в Unreal Engine Marketplace. Цей плагін надає простий та універсальний спосіб управління параметрами гри. Він автоматично пов'язує інтерфейс з системними змінними і зберігає вибрані налаштування між ігровими сесіями. Інтерфейс налаштувань зображено рисунку 2.64.

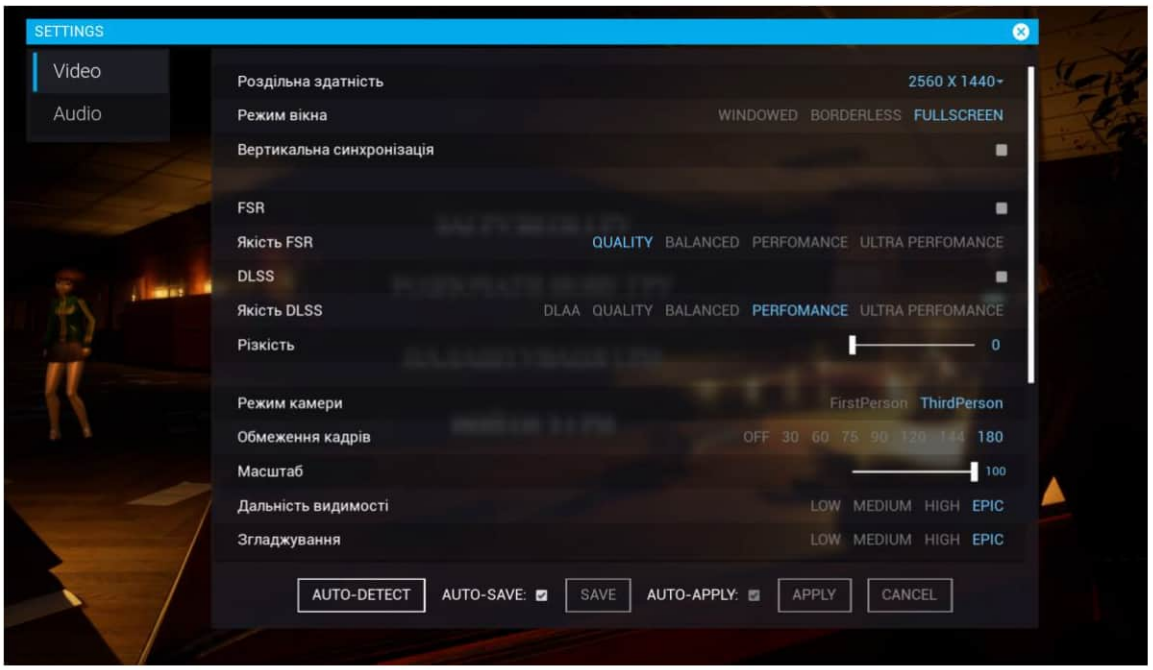


Рисунок 2.64 – Інтерфейс плагіну налаштувань Auto Settings

У меню налаштувань реалізовано детальний контроль над графікою: змінювати можна роздільну здатність, якість освітлення, текстур, тіней, пост-обробки та інші параметри. Окрему увагу приділено підтримці сучасних технологій масштабування, таких як DLSS (Deep Learning Super Sampling) від NVIDIA та FSR (FidelityFX Super Resolution) від AMD.

DLSS [13] використовує машинне навчання для підвищення частоти кадрів, не втрачаючи при цьому якості зображення. FSR [14], у свою чергу, є відкритою альтернативою, що дозволяє оптимізувати продуктивність навіть на менш потужному залізі. Обидві технології інтегровані в налаштування гри та можуть бути активовані гравцем за його бажанням.

2.10.2 Інтерфейс дослідження

Інтерфейс дослідження з'являється під час гри після натискання клавіші ESC. Відкривається вікно з розширеними можливостями керування персонажами, ресурсами та налаштуваннями. У верхній частині екрана відображається ключова інформація: поточне розташування гравця, кількість грошей та час, проведений у грі. У нижній частині – розміщена підказка, що

пояснює призначення вибраної вкладки. Крім основних вкладок, інтерфейс також містить кнопки для виходу з гри, повернення до дослідження світу, а також доступу до налаштувань безпосередньо під час гри. Головний інтерфейс зображено на рисунку 2.65.

Рисунок 2.65 – Головний інтерфейс дослідження

Головна вкладка - це «Огляд». У ній представлені активні персонажі поточної команди, а також їхня базова статистика: рівень, здоров'я, атака, захист та предмети.

Вкладка «Персонажі» дає змогу детально розглянути кожного з бійців, переглянути їхні характеристики, рівень, наявні навички. Через цей інтерфейс можна змінювати спорядження персонажів, зокрема зброю, броню та аксесуари, що безпосередньо впливає на їхню ефективність у бою. Також вкладка дозволяє редагувати склад активної команди, обираючи, хто саме братиме участь у майбутніх битвах, залежно від стратегії гравця. Вкладку «Персонажі» зображено на рисунку 2.66.

Рисунок 2.66 – Вкладка «Персонажі»

У вкладці «Предмети» зібрано весь інвентар гравця. Тут можна використовувати лікувальні зілля для відновлення здоров’я або інших характеристик, переглядати й застосовувати предмети а також видаляти непотрібні речі для звільнення місця в інвентарі. Вкладку «Предмети» зображено на рисунку 2.67.

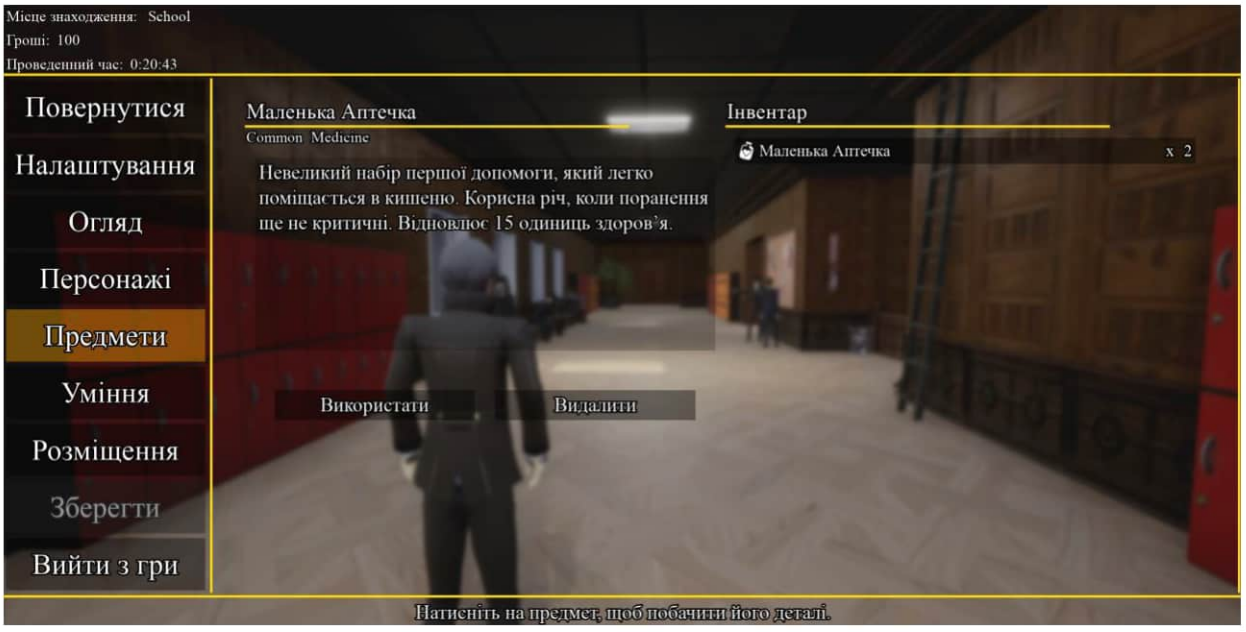


Рисунок 2.67 – Вкладка «Предмети»

Вкладка «Уміння» показує набір навичок кожного з персонажів. У цьому розділі можна переглядати опис кожної навички, її поточний рівень та вплив на ігровий процес. Також тут доступна система покращення умінь за допомогою очок прокачування, які нараховуються персонажам після досягнення нового рівня. Це дозволяє гравцеві формувати індивідуальний стиль бою для кожного героя. Вкладку «Уміння» зображено на рисунку 2.68.

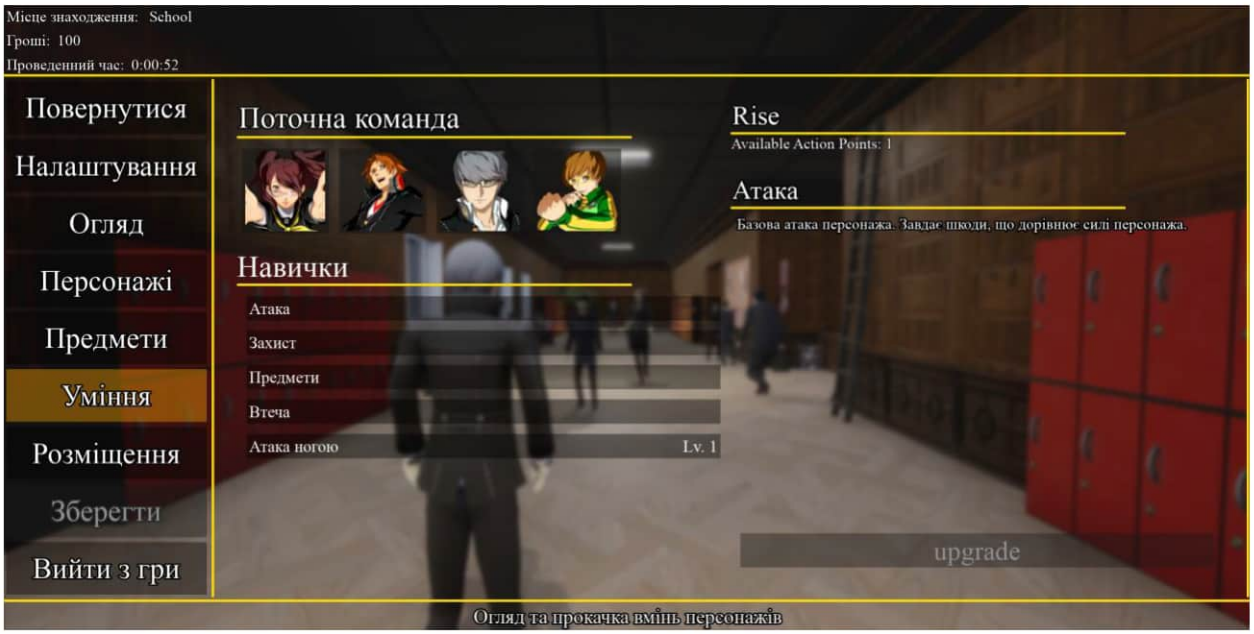


Рисунок 2.68 – Вкладка «Уміння»

У вкладці «Розміщення» можна змінювати бойове положення персонажів у загоні відповідно до обраної тактики. Гравець вирішує, хто з бійців перебуватиме на передовій – ближче до ворога для завдання фізичних ударів, а хто – залишатиметься на задній лінії, щоб підтримувати команду за допомогою магії чи лікування. Це надає додаткову глибину плануванню бою. Вкладку «Розміщення» зображено на рисунку 2.69.

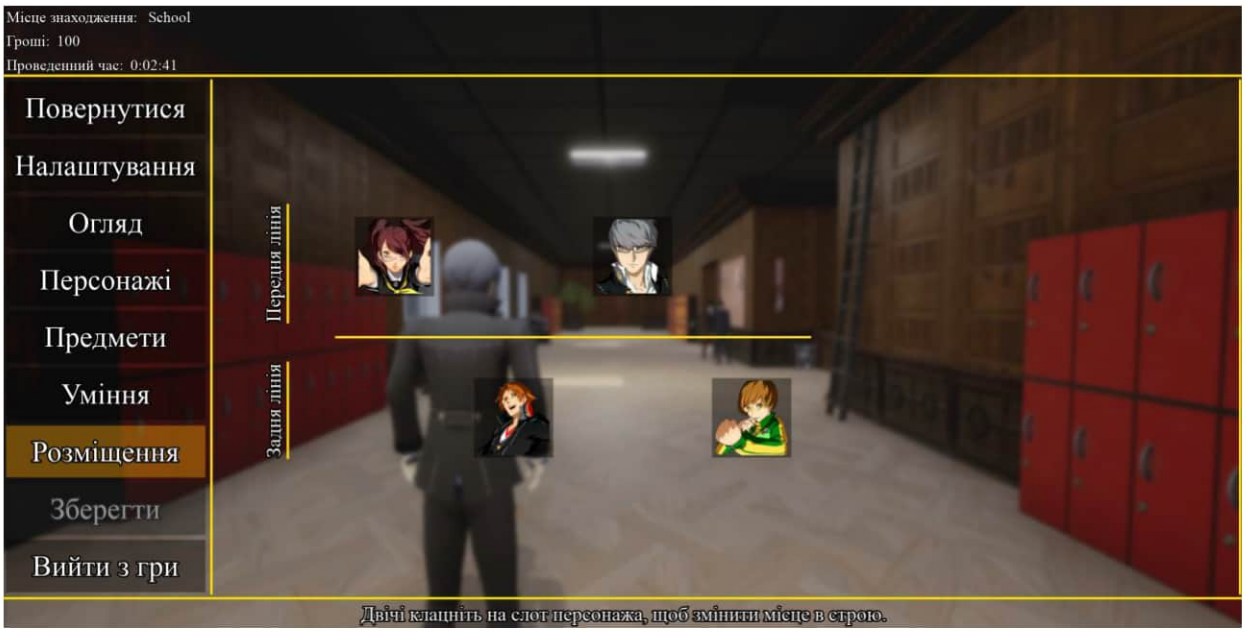


Рисунок 2.69 – Вкладка «Розміщення»

Вкладка «Зберегти» активується лише поблизу спеціально відведених точок збереження, що розташовані в безпечних зонах гри. Вона дозволяє гравцеві зафіксувати поточний прогрес, включно з рівнем персонажів, пройденими подіями та зібраними ресурсами. Завдяки збереженню гравець може надалі повернутися до цього моменту й уникнути втрати досягнень. Вкладку «Зберегти» зображено на рисунку 2.70.

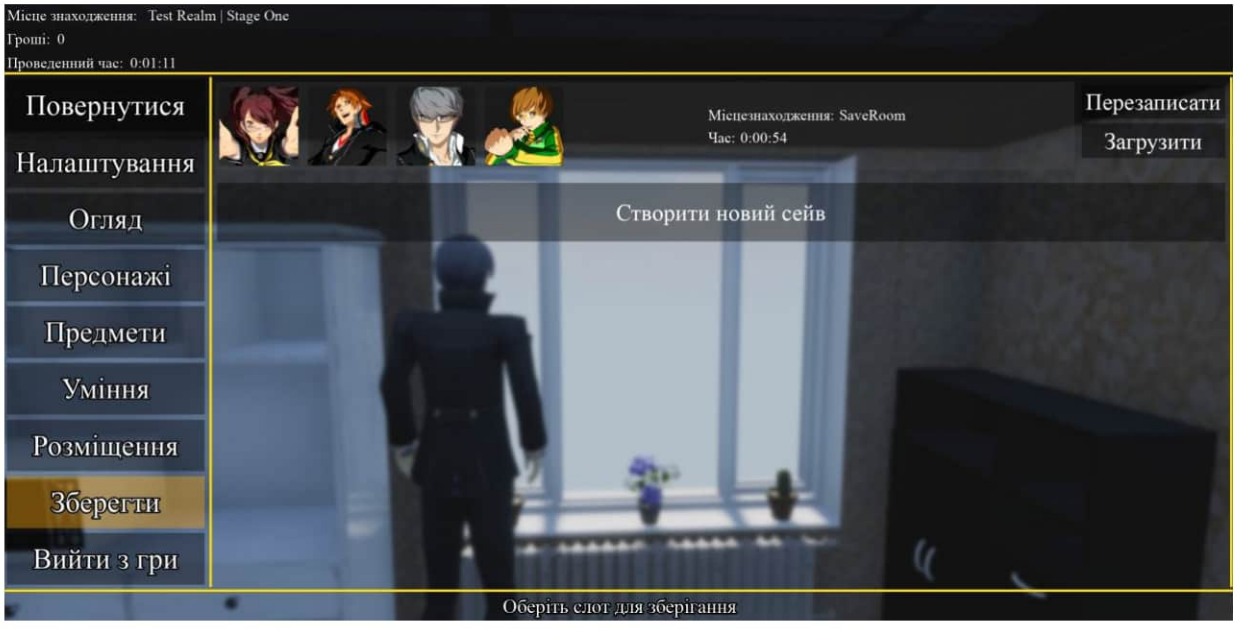


Рисунок 2.70 – Вкладка «Зберегти»

2.11 Висновки до другого розділу

У другому розділі здійснено проєктування та реалізацію основних систем ігрового застосунку у жанрі JRPG з використанням інструментів Unreal Engine 4. У результаті виконання розділу:

1) Визначено концепцію гри, цільову аудиторію, жанрову специфіку та ключові геймплейні особливості.
2) Розроблено архітектуру проєкту, описано структуру файлів та взаємодію між основними модулями гри.
3) Реалізовано систему персонажів, включаючи головного героя, бойових супутників та механізм їхнього розвитку.
4) Побудовано покрокову бойову систему з елементами інтерфейсу, відображенням прогресу та завершенням бою.
5) Створено систему взаємодії з об’єктами ігрового середовища, як-от скрині, двері та збереження прогресу.
6) Реалізовано інтерактивну діалогову систему для NPC з динамічним інтерфейсом.
7) Побудовано систему штучного інтелекту NPC з використанням Behavior Tree, AI Controller та BlackBoard.
8) Налагоджено базовий звуковий супровід та протестовано гру на відповідність мінімальним технічним вимогам.
9) Впроваджено адаптивний користувацький інтерфейс для головного меню, дослідження та бою.

ВИСНОВКИ

В результаті виконання роботи даної кваліфікаційної роботи втілено ігровий додаток у жанрі JRPG, застосовуючи можливості ігрового рушія Unreal Engine 4 разом із елементами штучного інтелекту. Проведено аналіз актуальних підходів до розробки ігор даного жанру, зокрема вивчено приклади успішних проєктів, як-от Shin Megami Tensei та Persona.

Розроблений додаток має завершену архітектуру, яка містить систему управління персонажем, дослідження ігрового світу, бойову механіку, інтерфейс користувача, систему інвентарю, збереження прогресу, а також обробку подій з використанням Blueprint-логіки. В рамках проєкту розроблено кілька поведінкових алгоритмів NPC за допомогою системи AI Perception, враховуючи зір та слух ворогів, що дозволяє створити динамічну модель реагування на дії гравця.

Використано плагін Auto Settings для гнучкого налаштування графіки, а також інтегровано сучасні технології масштабування зображення (DLSS та FSR), що дозволяє пристосувати гру під різні технічні характеристики пристроїв. Особливу увагу приділено оптимізації продуктивності, яка забезпечує стабільну частоту кадрів, швидке завантаження рівнів та мінімізацію затримок навіть на менш потужних системах.

Результатом роботи є прототип ігрового застосунку з цілісною структурою, налаштованими механіками та продуманою візуальною складовою. Представлений проєкт демонструє можливість реалізації складних ігрових сценаріїв із використанням штучного інтелекту, а також слугує прикладом практичного застосування сучасних інструментів розробки в інтерактивному середовищі.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Persona 3 Reload Sells. [Електронний ресурс] URL: https://gameworldobserver.com/2024/02/08/persona-3-reload-sales-1-million-copies-atlus-record? (дата звернення: 22.04.2025).

2. Persona 3 Reload Successful launch. [Електронний ресурс] URL: https://gameworldobserver.com/2024/02/02/persona-3-reload-42k-concurrent-players-steam-atlus-launch (дата звернення: 22.04.2025).

3. Persona 5 Royal Review. [Електронний ресурс] URL: https://www.ign.com/articles/persona-5-royal-review (дата звернення: 22.04.2025).

4. Persona 3 Reload Review. [Електронний ресурс] URL: https://www.ign.com/articles/persona-3-reload-review (дата звернення: 22.04.2025).

5. Shin Megami Tensei V: Vengeance Review. [Електронний ресурс] URL: https://www.ign.com/articles/persona-3-reload-review (дата звернення: 23.04.2025).

6. Unity Real-Time Development Platform [Електронний ресурс] URL: https://unity.com/ (дата звернення: 24.04.2025).

7. Godot Engine - Free and open source 2D and 3D game engine. [Електронний ресурс] URL: https://godotengine.org/ (дата звернення: 25.04.2025).

8. Unreal Engine: The most powerful real-time 3D creation tool. [Електронний ресурс] URL: https://www.unrealengine.com/en-US (дата звернення: 24.04.2025).

9. Unreal Engine:Introduction to Blueprints. [Електронний ресурс] URL: https://www.unrealengine.com/fr/blog/introduction-to-blueprints (дата звернення: 25.04.2025).

10. Behavior Trees in Unreal Engine. [Електронний ресурс] URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/behavior-trees-in-unreal-engine (дата звернення: 26.04.2025).

11. Coding in Unreal Engine: Blueprint vs. C++. [Електронний ресурс] URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/coding-in-unreal-engine-blueprint-vs.-cplusplus (дата звернення: 26.04.2025).

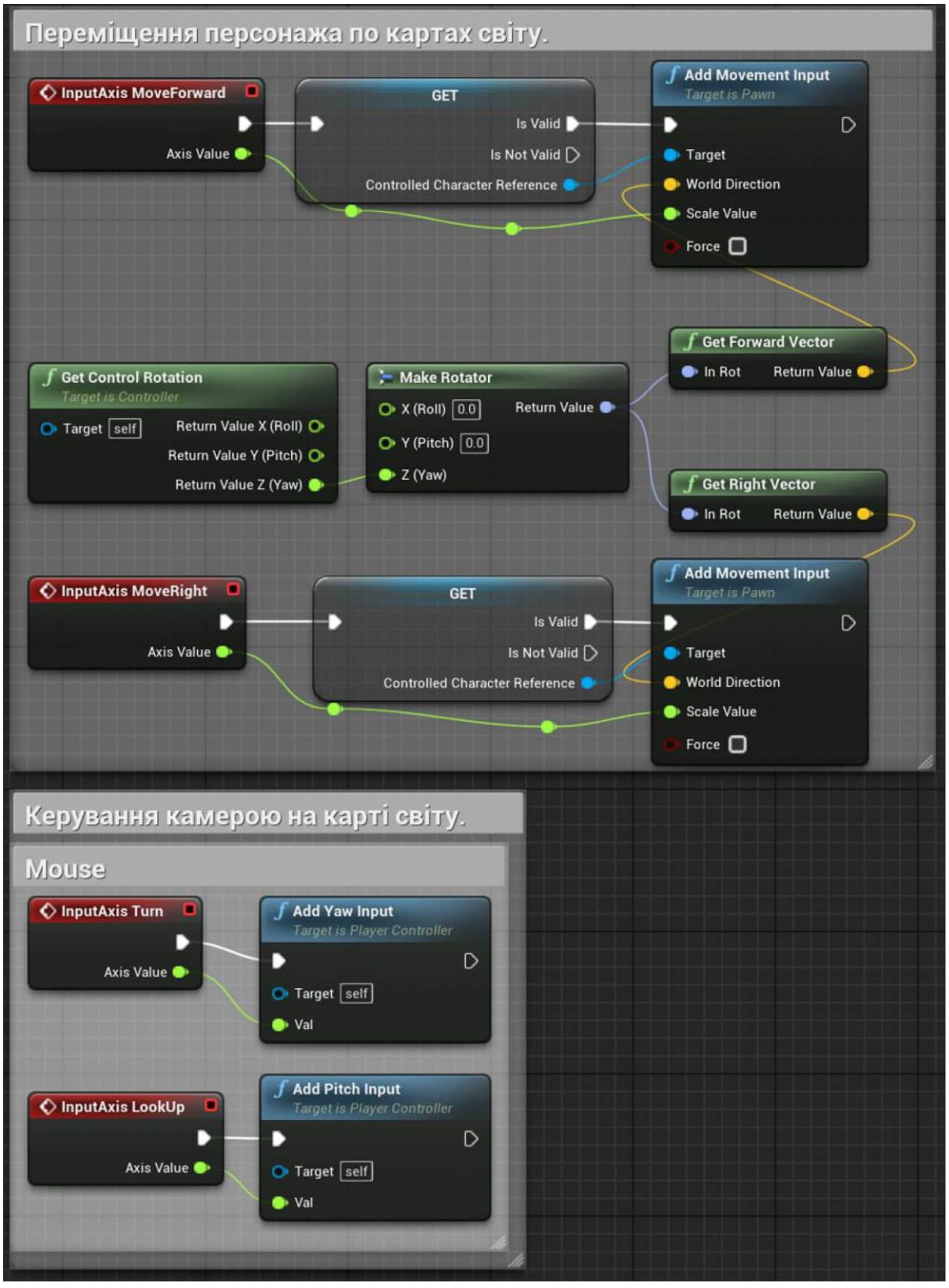
12. Fab: Auto Settings [Електронний ресурс] URL: https://www.fab.com/listings/6166897a-d04b-46d5-b2cc-a692a74b0698 (дата звернення: 29.04.2025).

13. NVIDIA DLSS 4 Technology [Електронний ресурс] URL: https://www.nvidia.com/en-us/geforce/technologies/dlss/ (дата звернення: 13.05.2025).

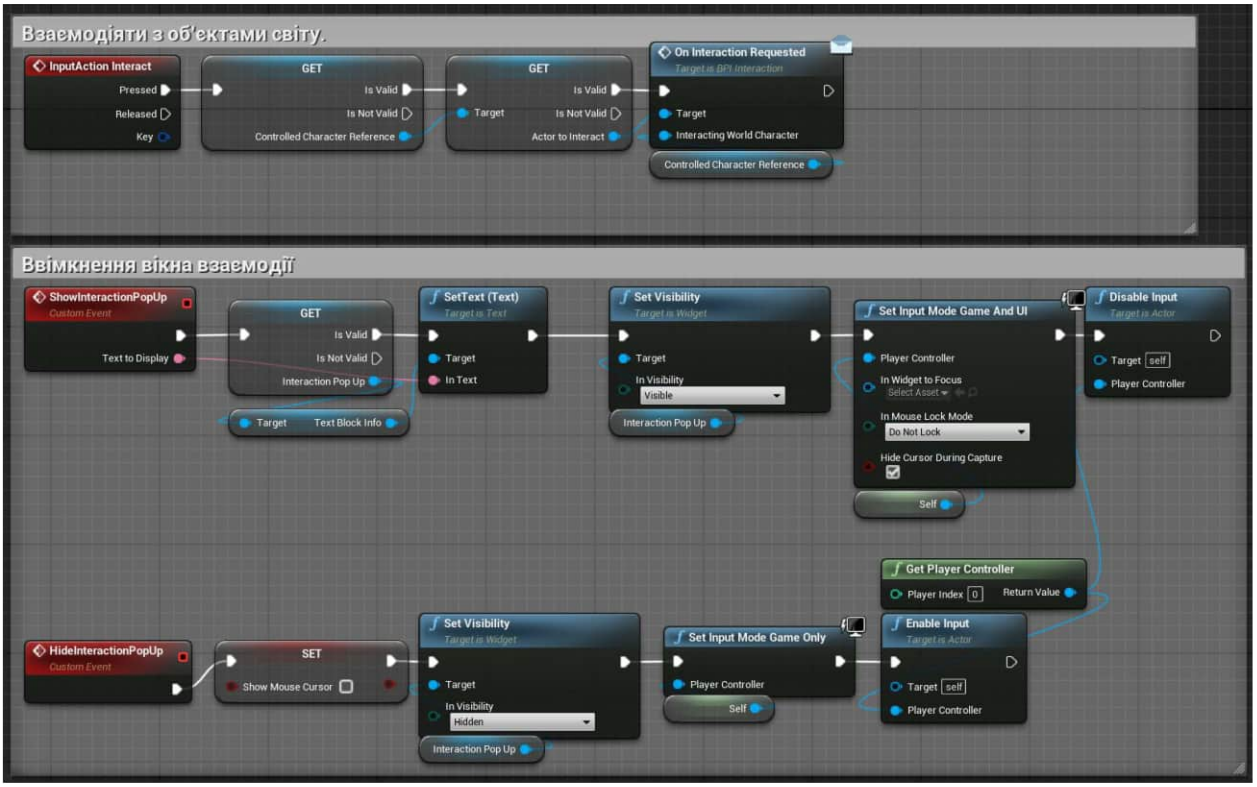
14. AMD FidelityFX Super Resolution [Електронний ресурс] URL: https://www.fab.com/listings/6166897a-d04b-46d5-b2cc-a692a74b0698 (дата звернення: 13.05.2025).

ДОДАТОК А.
ФРАГМЕНТИ КЛАСУ ГОЛОВНОГО ГЕРОЯ

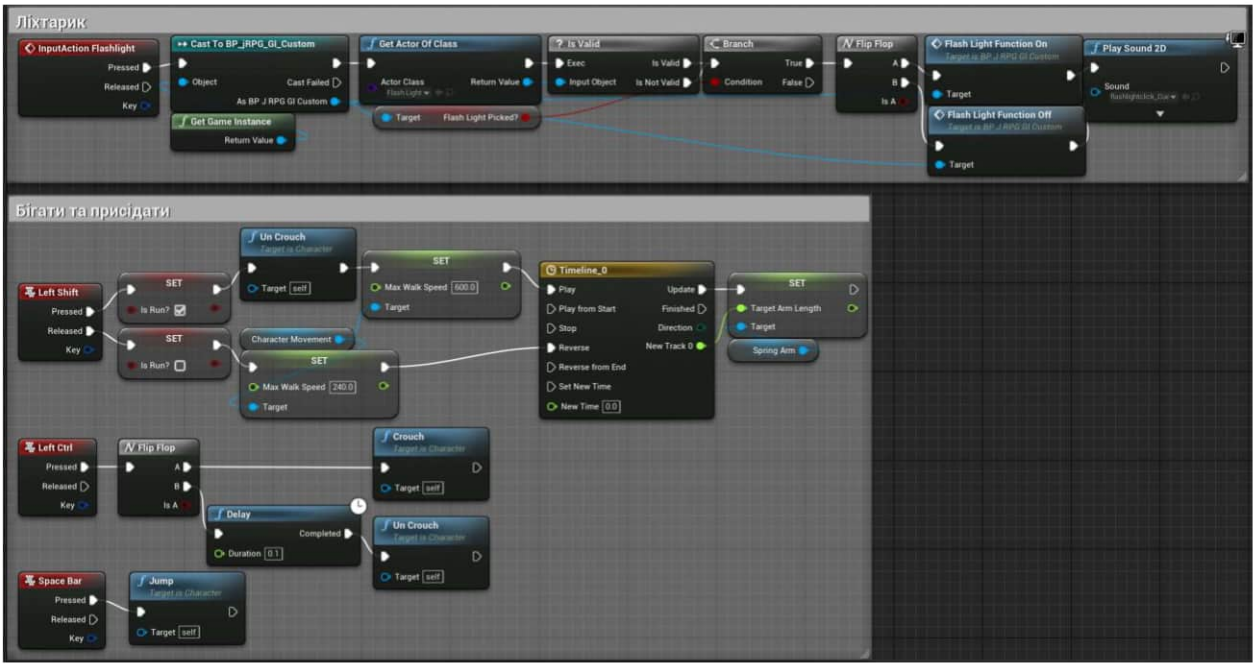
A.1 Character Controls в BP_jRPG_Controller_World



A.2 Interaction Graph в BP_jRPG_Controller_World

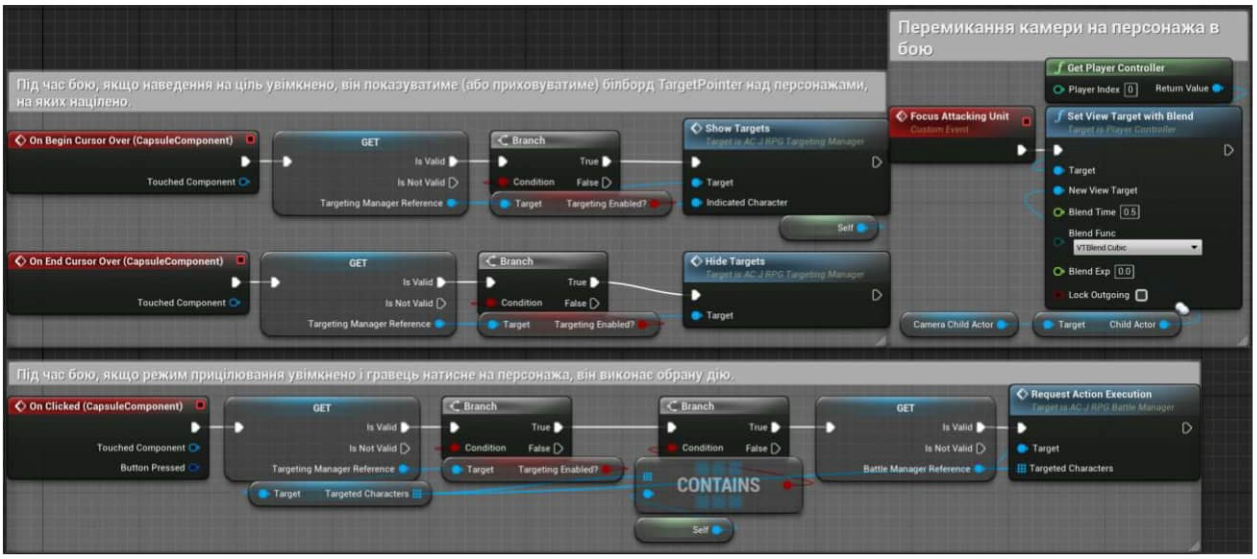


A.3 Event Graph в BP_jRPG_Character_World



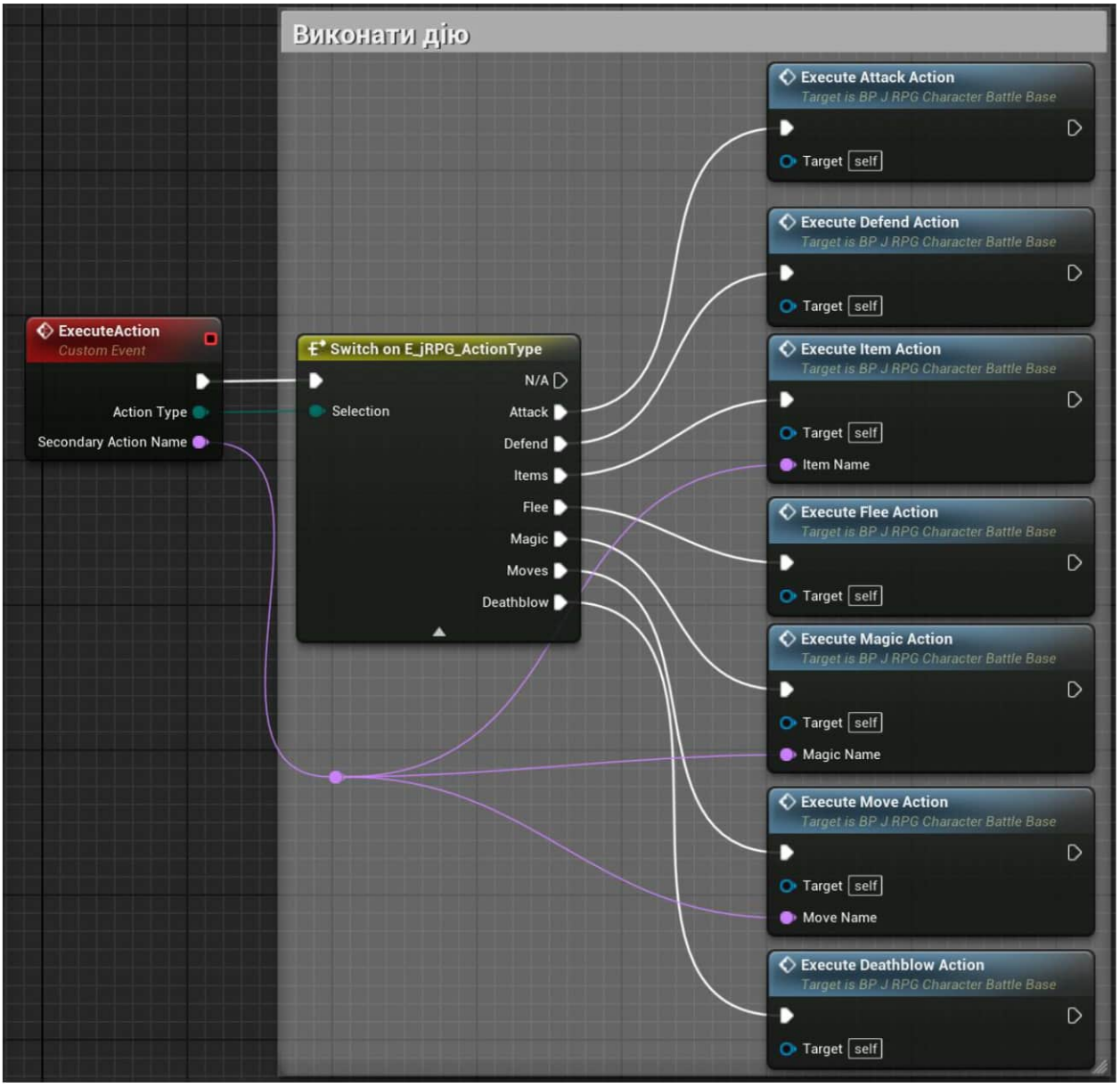
ДОДАТОК Б.
ФРАГМЕНТИ КЛАСУ БОЙОВОГО ПЕРСОНАЖА

Б.1 Event Graph в BP_jRPG_Character_Battle_Base

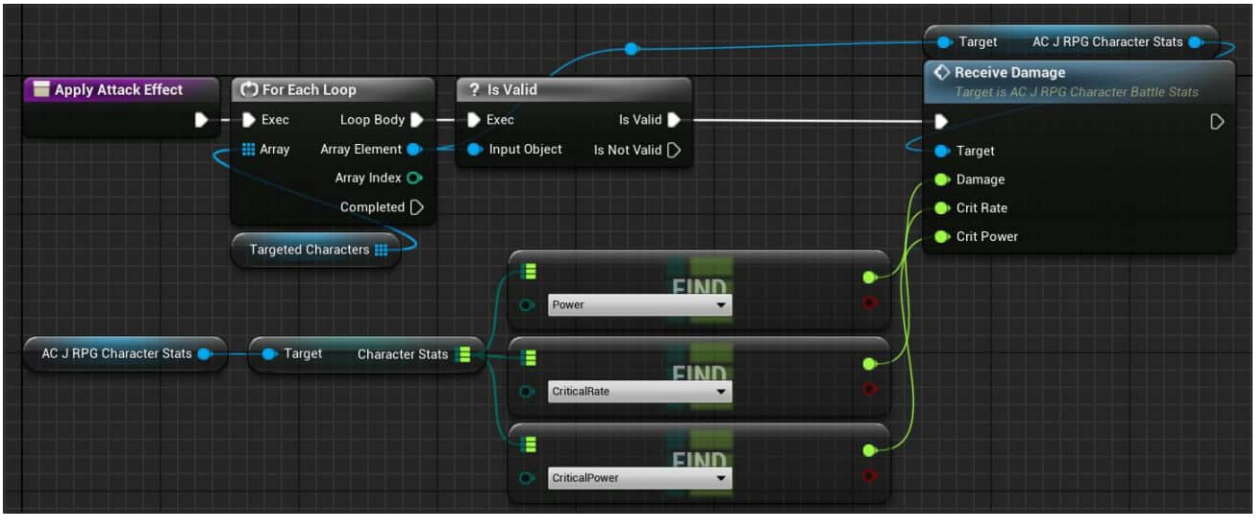


Б.2 Interface Events в BP_jRPG_Character_Battle_Base

Б.3 Action Events в BP_jRPG_Character_Battle_Base

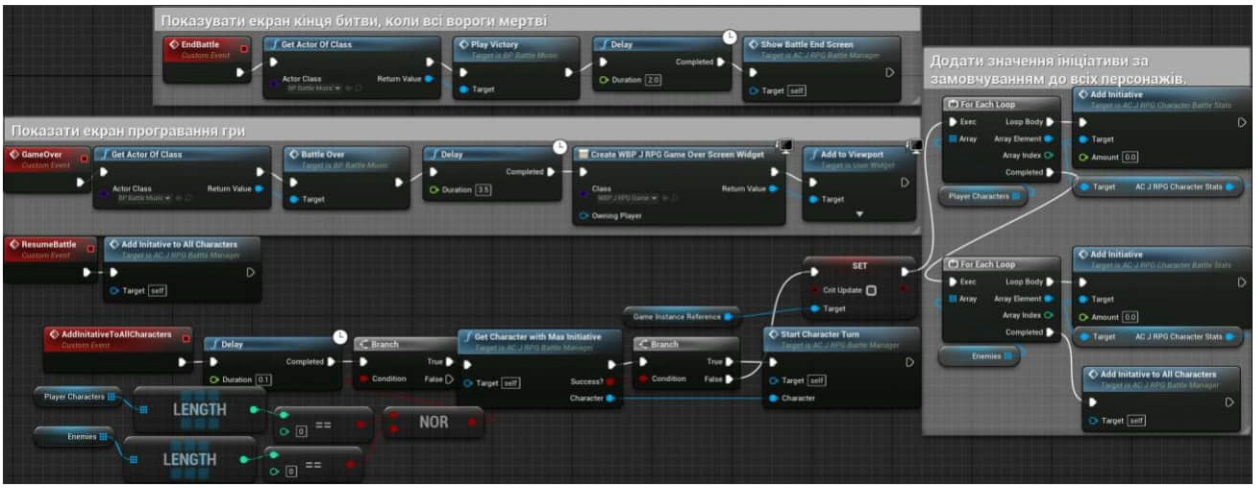


Б.4 Функція Apply Attack Effect в BP_jRPG_Character_Battle_Base

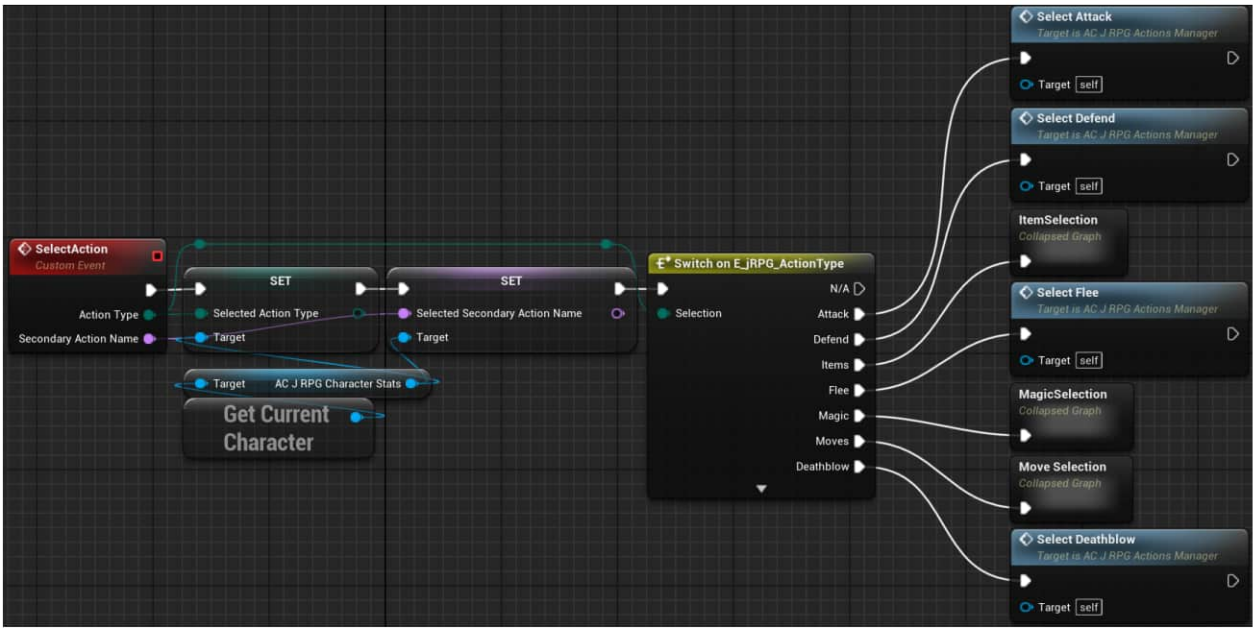


ДОДАТОК В.
КОМПОНЕНТИ БИТВИ

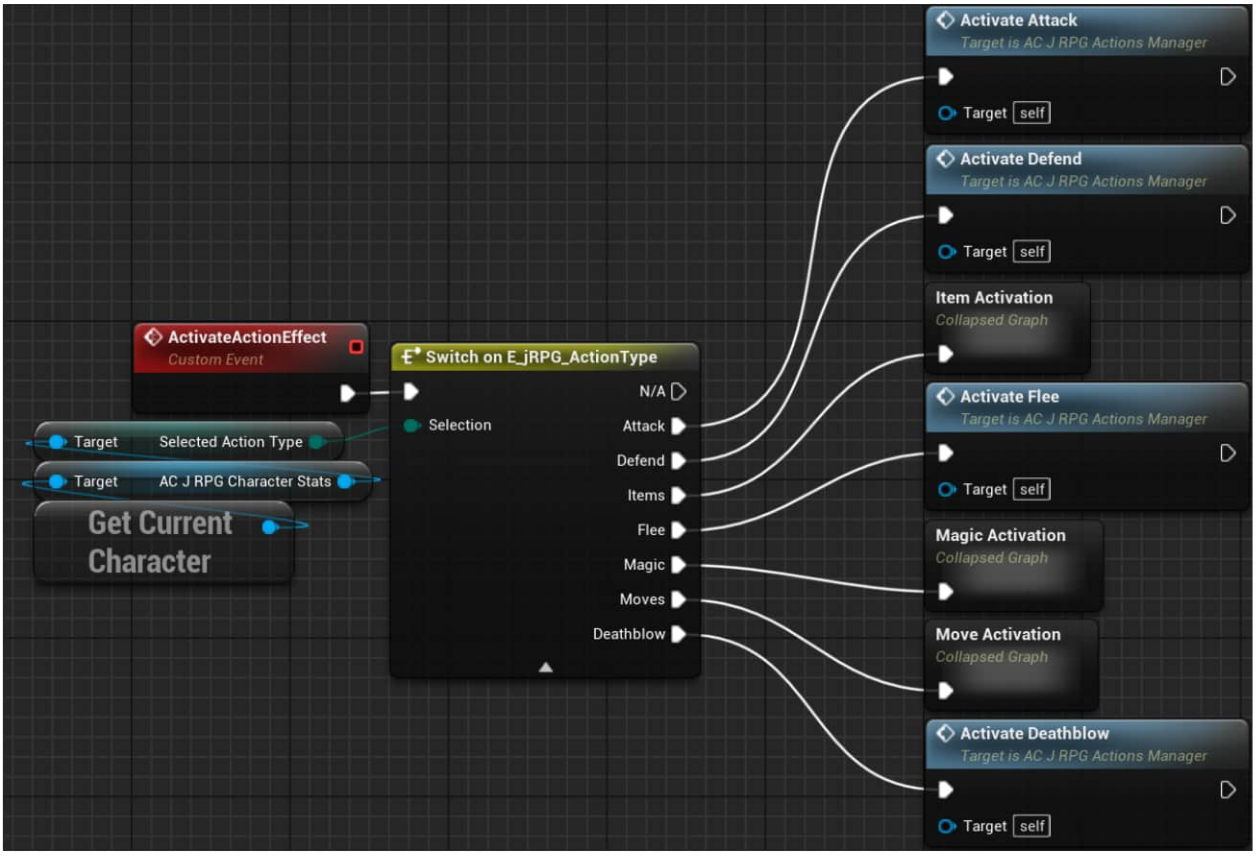
B.1 Battle Graph в AC_jRPG_BattleManager



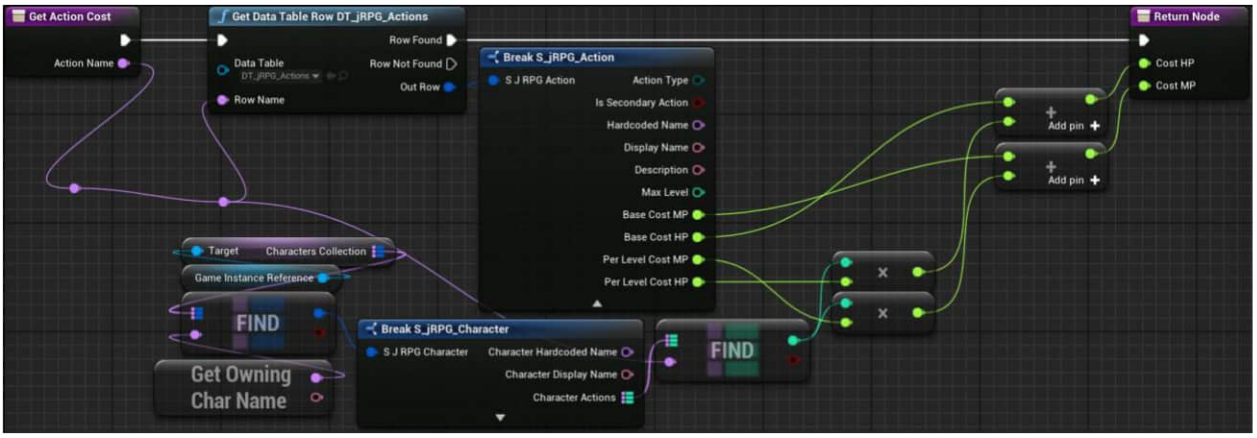
B.2 Selection Graph в AC_jRPG_ActionsManager



B.3 Selection Graph в AC_jRPG_ActionsManager

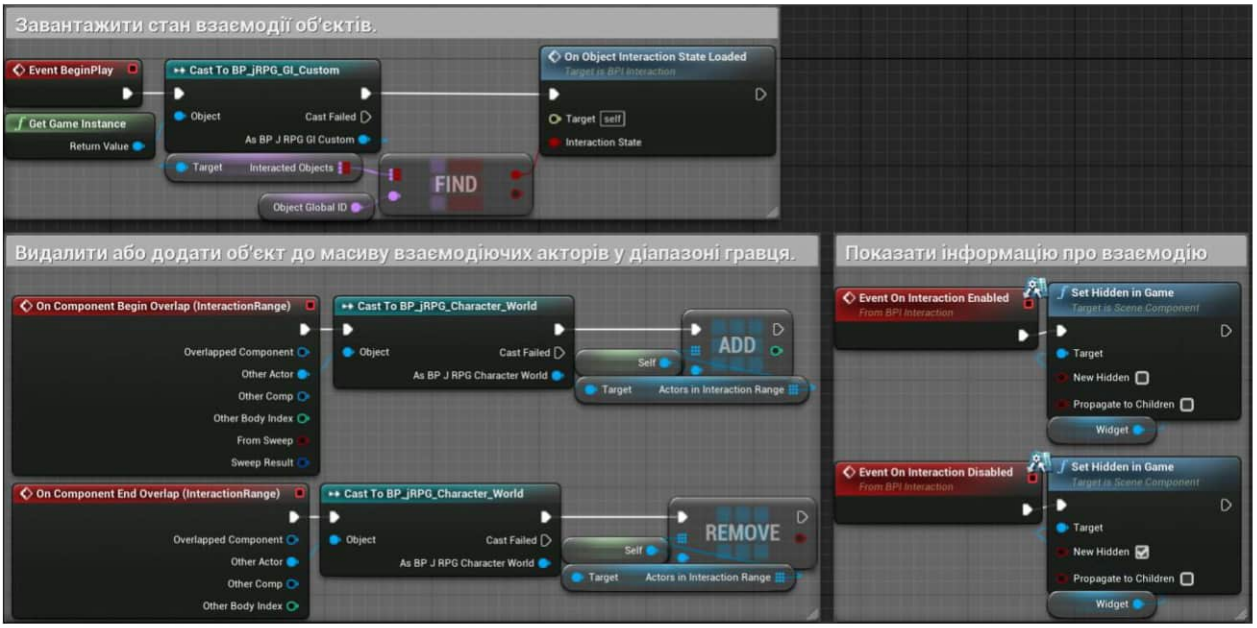


B.4 Функція Get Action Cost в AC_jRPG_ActionsManager

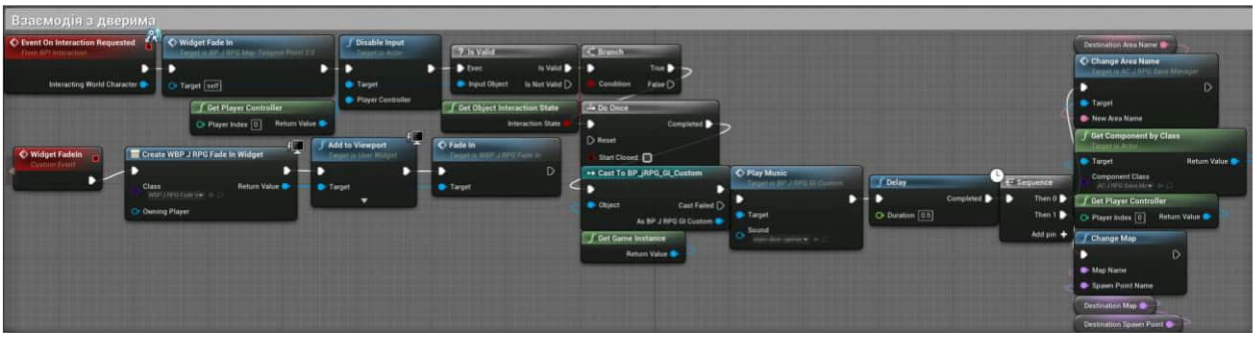


ДОДАТОК Г.
СИСТЕМА ВЗАЄМОДІЇ

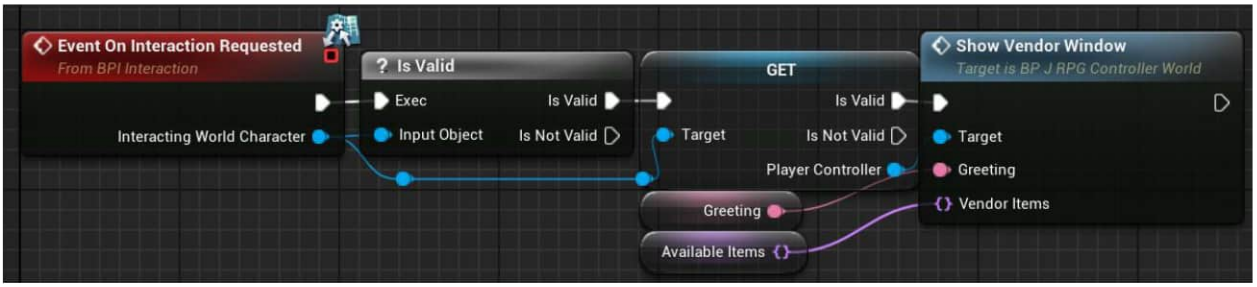
Г.1 Головний клас взаємодії BP_jRPG_InteractionObject_Base



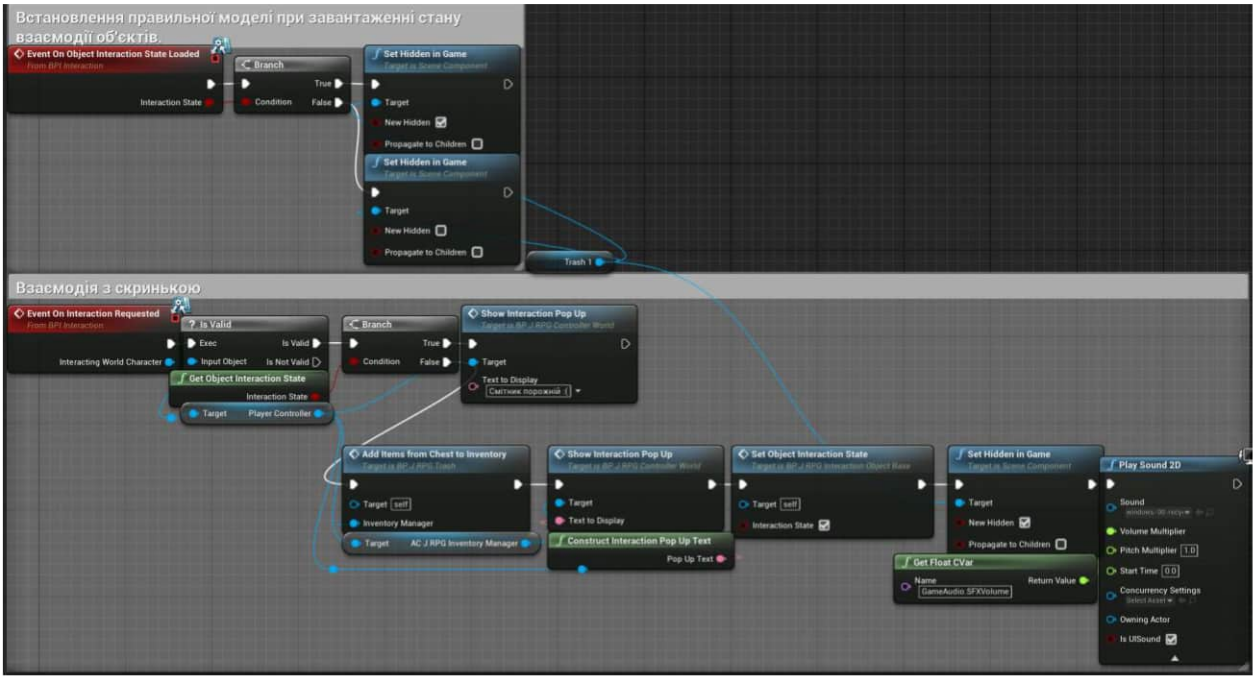
Г.2 Клас дверей BP_jRPG_MapTeleportPoint



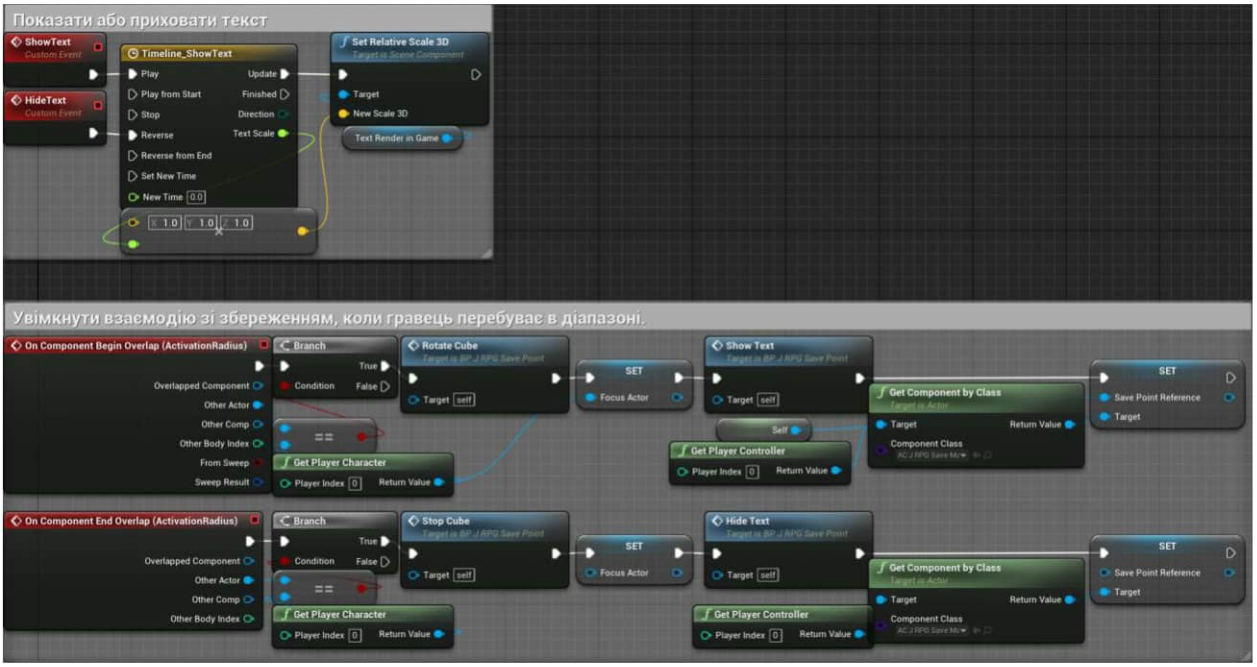
Г.3 Клас торговця BP_jRPG_Vendor_NPC



Г.4 Клас скриньки BP_jRPG_InteractionObject_Chest

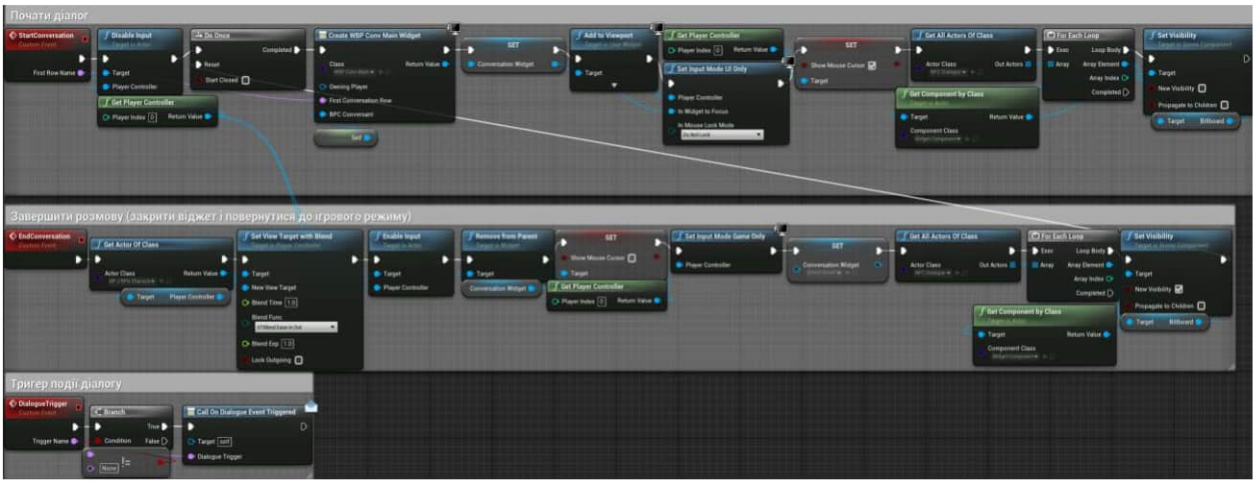


Г.5 Клас збереження BP_jRPG_Save_Object

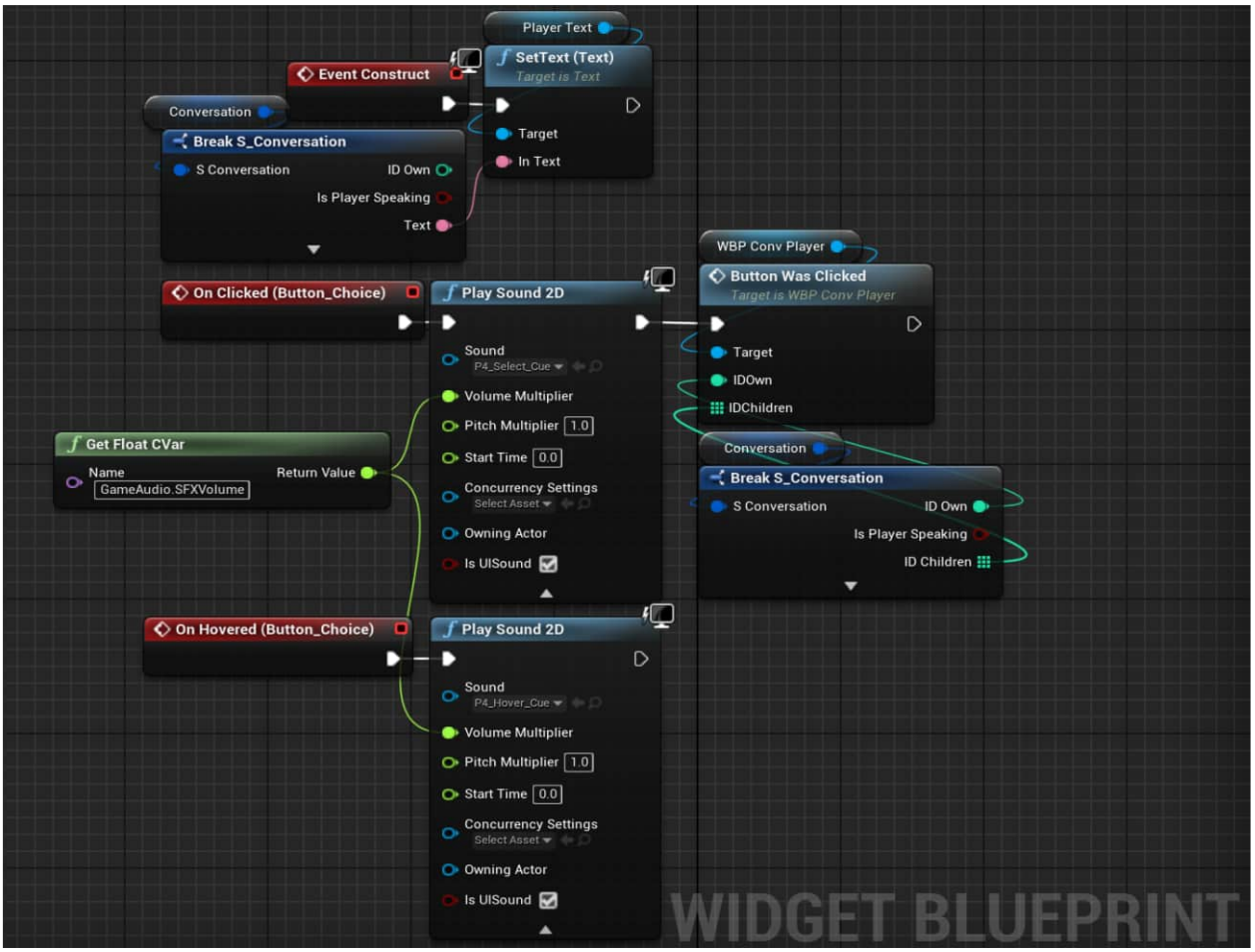


ДОДАТОК Д.
СИСТЕМА ДІАЛОГІВ

Д.1 Компонент BPC_Conversant



Д.2 Віджет WBP_Conv_Answer



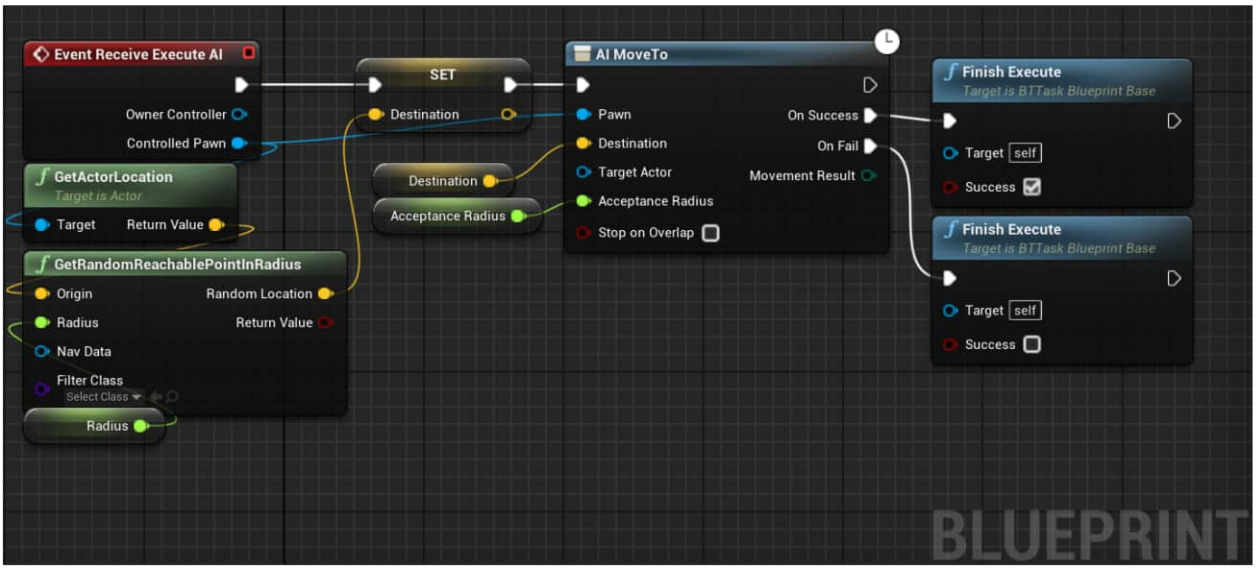
Д.3 Віджет WBP_Conv_Plaver

Д.4 Віджет WBP_Conv_NPC

Д.5 Віджет WBP_Conv_Main

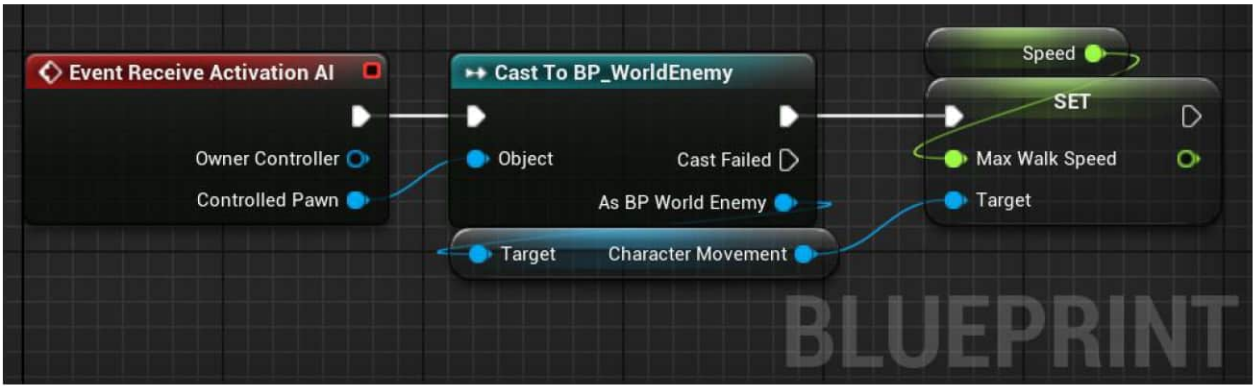
ДОДАТОК Е.
КОМПОНЕНТИ ШТУЧНОГО ІНТЕЛЕКТУ

Е.1 Задача BTTask_MoveRandomLocation



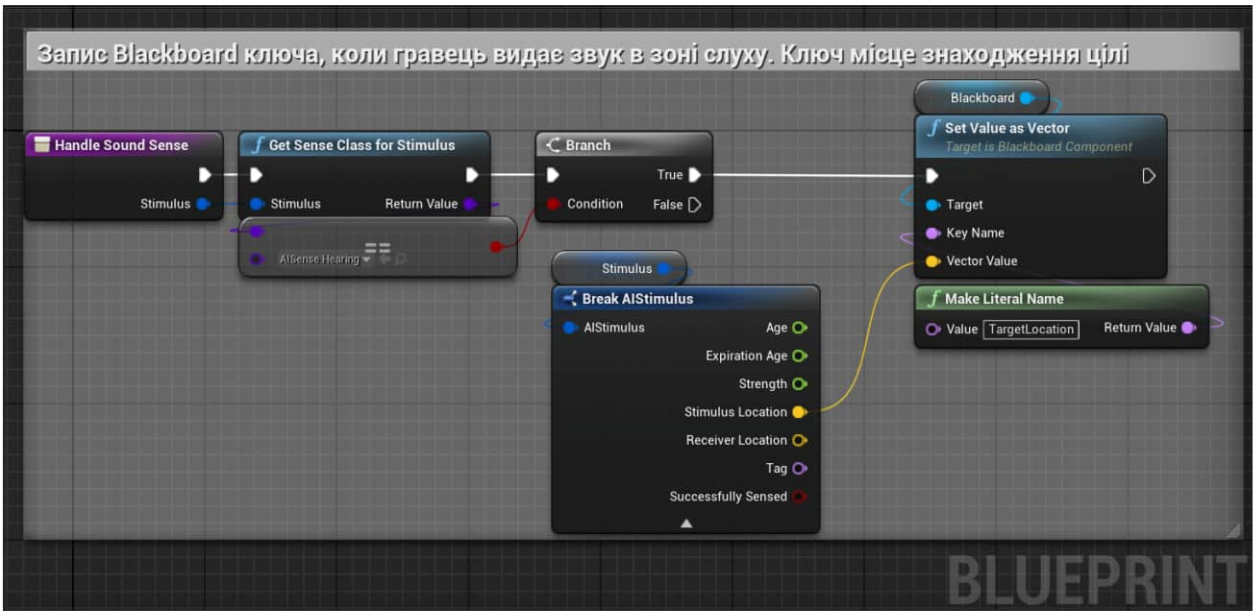
Е.2 Задача BTTask_ClearBlackboardValue

Е.3 Сервіс BTS_Set_Speed

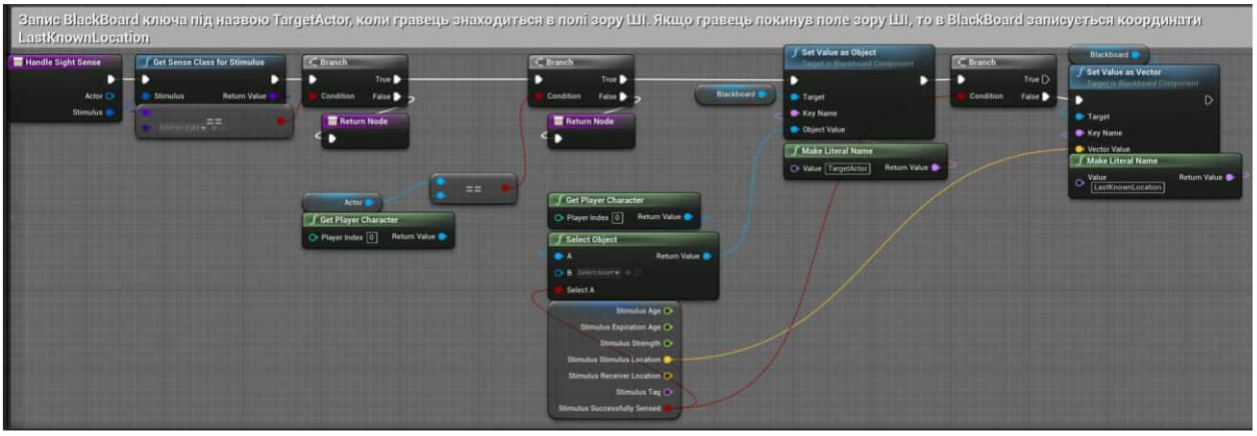


Е.4 Задача BTTask_Patrol

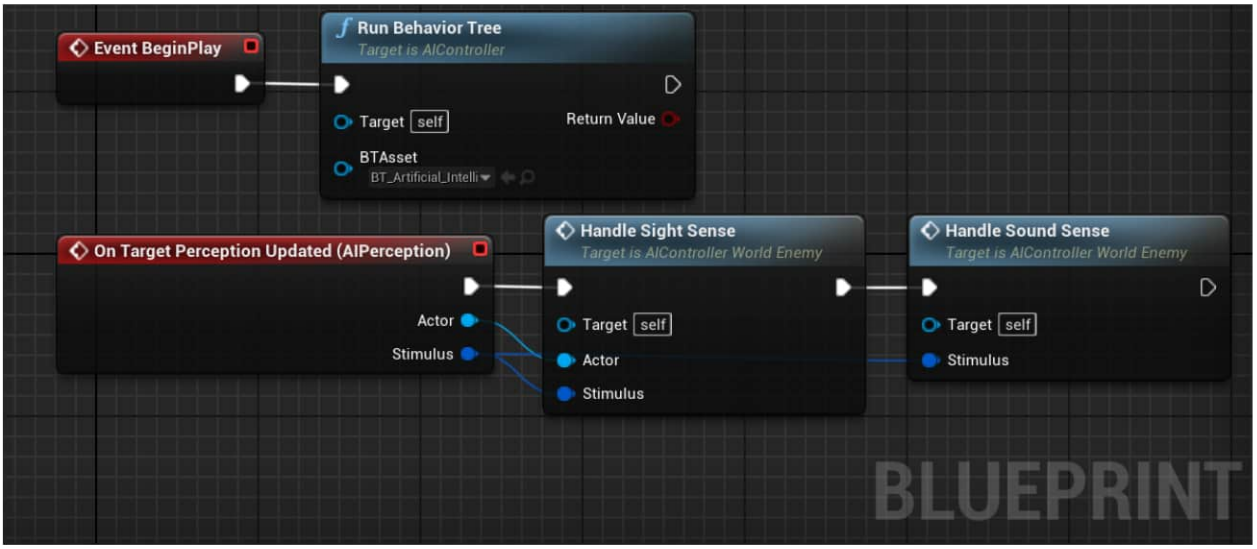
Е.5 Функція Handle Sound Sense



E.6 Функція Handle Sight Sense



E.7 Event Graph в AI_Controller



ДОДАТОК Ж.
ГОЛОВНИЙ КЛАС МУЗИЧНОГО СУПРОВОДУ

Ж.1 Event Graph в BP_Music