НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри

_____Комп'ютерних Наук_____

(назва кафедри)

_____ _Голуб Б. Л._

(підпис) (ПІБ)

"___" _____ 20 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему

«Комп'ютерна система управління віртуальним інвентарем та здібностями ігрового персонажа»

Спеціальність 122 – «Комп'ютерні науки»

Гарант освітньої програми

_____д.е.н., професор_____ _____ Руденський Р.А.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

_____ _____ __Назаренко В.А.__
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав **_____** ____Штонда В.В._
(підпис) (ПІБ студента)

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ

Завідувач кафедри

____Комп'ютерних Наук_____

_____ Голуб Б. Л.

(науковий ступінь, вчене звання) (підпис) (ПІБ)

"____" _____20_____р.

З А В Д А Н Н Я

на виконання бакалаврської кваліфікаційної роботи студенту

_____Штонда Владислав Віталійович_____

(прізвище, ім'я, по батькові)

Спеціальність 122 – «Комп'ютерні науки»

Тема бакалаврської кваліфікаційної роботи __Комп'ютерна система управління віртуальним інвентарем та здібностями ігрового персонажа

Затверджена наказом ректора НУБіП України від 16.12.2024 № 2246 "С"

Термін подання завершеної роботи на кафедру _____

(рік, місяць, число)

Вихідні дані до бакалаврської кваліфікаційної роботи

Розробка та реалізація системи управління віртуальним інвентарем та здібностями ігрового персонажа для застосунку на базі ігрового рушія Unreal Engine 5

Перелік питань, які потрібно розробити:

1) Аналіз предметної області та існуючих аналогів.
2) Проєктування системи інвентарю та здібностей персонажа.
3) Реалізація програмної частини гри на Unreal Engine.
4) Підготовка інсталяційного пакету та рекомендації з експлуатації.

Дата видачі завдання "16"12.2024р.

Керівник бакалаврської кваліфікаційної роботи

_____д-р філос., доц.____ _____ Назаренко В.А.

(науковий ступінь та вчене звання) (підпис) (ПІБ)

Завдання прийняв до виконання _____ Штонда В.В.

(підпис) (прізвище та ініціали студента)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання бакалаврської роботи	Строк виконання етапів бакалаврської роботи	Примітка
1	Видача завдання	16.12.2024	
2	Аналіз предметної області	Лютий 2025	
3	Моделювання предметної області	Лютий 2025	
4	Поставновка завдання	Лютий 2025	
5	Проєктування системи	Лютий 2025	
6	Розробка системи	Березень 2025	
7	Тестування системи	Травень 2025	
8	Оформлення записки	Травень 2025	
9	Проходження нормо контролю	24-25 травня 2025	
10	Перевірка на плагіат	26-27 травня 2025	
11	Попередній захист	2-3 червня 2025	
12	Захист	9-12 червня 2025	

Студент _____ ___Штонда В.В.__
(підпис) (прізвище та ініціали)

Керівник бакалаврської кваліфікаційної роботи _____ Назаренко В.А.
(підпис) (прізвище таініціали)

АНОТАЦІЯ

У бакалаврській кваліфікаційній роботі розроблено комп’ютерну систему управління віртуальним інвентарем та здібностями ігрового персонажа в рамках однокористувацької гри. Система дозволяє гравцеві взаємодіяти з предметами, здійснювати їхнє сортування, екіпірування та зберігання, а також використовувати активні здібності відповідно до визначених умов, таких як кулдауни або стан персонажа.

Основна мета роботи полягає в створенні зручного, гнучкого та розширюваного інтерфейсу, що забезпечує інтуїтивну взаємодію гравця з елементами ігрової системи. Проєкт реалізовано за допомогою ігрового рушія Unreal Engine 5, що дозволило застосувати сучасні підходи до побудови UI/UX, обробки подій, та збереження даних гри.

Документ містить опис структури системи, її функціональних можливостей, основних механік взаємодії з інвентарем і здібностями. Також у роботі наведено діаграми, які відображають логіку функціонування системи, та результати тестування, що підтверджують працездатність реалізованого рішення. Отримана система може бути адаптована для використання в інших проєктах ігрового жанру RPG або Action.

ABSTRACT

This bachelor’s qualification thesis presents the development of a computer system for managing virtual inventory and character abilities within a single-player game. The system enables the player to interact with items, perform sorting, equipping, and storing operations, as well as to use active abilities under defined conditions such as cooldowns or character states.

The primary goal of this project is to create a user-friendly, flexible, and scalable interface that provides intuitive interaction between the player and the game system elements. The implementation is carried out using the Unreal Engine 5, which allows the use of modern approaches for UI/UX design, event handling, and game data storage.

The document describes the system structure, functional capabilities, and core mechanics of inventory and ability management. It also includes diagrams that reflect the system’s logic and the results of testing, which confirm the operability of the developed solution. The resulting system can be adapted for use in other RPG or Action game projects

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

UI – User Interface – користувацький інтерфейс.
UX – User Experience – досвід користувача.
CD – Cooldown – кулдаун – час затримки перед повторним використанням здібності.
HUD – Head-Up Display – графічний інтерфейс, що показується поверх ігрового процесу (наприклад, інвентар, здоров'я, здібності).
RPG – Role-Playing Game – рольова гра.
UML – Unified Modeling Language – уніфікована мова моделювання.
UE – Unreal Engine – рушій для розробки ігор.
.ini / .cfg – текстові конфігураційні файли параметрів.
UFS / non-UFS – Unreal File System – внутрішня система зберігання файлів UE.
Blueprint – візуальна мова програмування в UE.
JSON – JavaScript Object Notation – формат обміну структурованими даними.

ВСТУП

Сучасна відеоігрова індустрія переживає період інтенсивного розвитку, де ключову роль відіграють системи управління ігровими об’єктами, зокрема інвентарем та здібностями персонажа. В умовах зростання очікувань гравців щодо гнучкості, інтуїтивності та візуальної складової інтерфейсів, розробка ефективних рішень для керування цими елементами стає критично важливою.

Важливість створення зручної та функціональної системи управління віртуальним інвентарем і здібностями полягає в забезпеченні комфортного ігрового досвіду, який дозволяє гравцеві швидко орієнтуватися в ресурсах, приймати стратегічні рішення та адаптувати свого героя до нових викликів. Більшість сучасних рольових ігор надають користувачу великий обсяг можливостей для кастомізації персонажа, тому актуальність створення якісного програмного рішення для управління цими можливостями є беззаперечною.

Метою цієї роботи є проєктування та розробка комп’ютерної системи, яка забезпечує ефективне управління віртуальним інвентарем і здібностями ігрового персонажа в умовах однокористувацької гри. Система має бути інтегрованою частиною загального геймплейного процесу, забезпечуючи як технічну функціональність (додавання, сортування, застосування предметів і здібностей), так і привабливий візуальний інтерфейс.

У межах реалізації програмного додатку передбачається використання сучасних інструментів і технологій, зокрема рушія Unreal Engine 5. Цей рушій надає широкі можливості для реалізації графіки високої якості, підтримки складної логіки за допомогою системи Blueprint та модульного підходу до побудови інтерфейсів. Реалізація зберігання даних буде базуватися на файловій системі рушія, без використання традиційних СУБД, що є актуальним рішенням для однокористувацьких ігор.

Апробація. Теза до бакалаврської роботи, доповідь на тему «концепція інформаційної системи управління віртуальним інвентарем та здібностями ігрового персонажа » була опублікована на науково-практичній конференції «Теоретичні та прикладні аспекти розробки комп'ютерних систем 2025» (24 квітня 2025 року)

Структура роботи. Дипломний проект включає чотири основних розділи та додатки. У першому розділі проведено системний аналіз предметної області, розглянуто наявні підходи до реалізації подібних систем та здійснено постановку задачі. Другий розділ присвячено інформаційному забезпеченню – опису логічної структури даних та підходів до організації збереження. У третьому розділі наведено прикладну реалізацію системи: опис архітектури, інструментів розробки та програмну реалізацію модулів. Останній розділ містить рекомендації щодо впровадження та тестування системи, а також опис вимог до середовища її функціонування. Записка обсягом 60 сторінок містить понад 20 використаних джерел та кілька додатків, що включають фрагменти коду, зображення інтерфейсів і структурні діаграми системи.

РОЗДІЛ 1
СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка задачі

Тема: Комп’ютерна система управління віртуальним інвентарем та здібностями ігрового персонаж

У рамках дипломної роботи метою є розробка програмної системи керування інвентарем та здібностями персонажа в однокористувацькій action-RPG грі. Основне призначення цієї системи — забезпечення гравцеві зручного та ефективного інструменту для взаємодії з внутрішньоігровими предметами, здібностями та ресурсами персонажа. Така система покликана підвищити глибину геймплею, надати можливість гнучкої адаптації стилю гри під уподобання користувача, а також забезпечити стабільну й інтуїтивну взаємодію з внутрішньоігровими об’єктами.

Конкретна ціль розробки — створення повноцінного ігрового модуля, що дозволяє:

- зберігати, відображати та керувати інвентарем персонажа;
- активувати та керувати здібностями, що мають параметри (наприклад, тривалість, ефект, перезарядка);
- взаємодіяти з інтерфейсом користувача для інтуїтивного управління ресурсами гри.

Система повинна бути розроблена на рушії Unreal Engine 5 з урахуванням гнучкості для подальшого масштабування, використання внутрішньоігрових механік, візуалізації через UI-компоненти та локального збереження даних.

Задачі, які потрібно вирішити:

Розробка моделі інвентаря та здібностей персонажа.

- Створення структур для зберігання даних про предмети, їх типи, властивості, кількість.
- Реалізація моделей здібностей з ефектами, тривалістю та cooldown-механікою.

Реалізація інтерфейсу користувача (UI) для взаємодії з системою.

- Розробка інтерфейсів для перегляду інвентаря, опису предметів, активації здібностей.
- Інтеграція системи інвентаря в загальний HUD (екран користувача).

Організація логіки керування та обробки подій.

- Застосування системи input'ів гравця для вибору та активації предметів або здібностей.
- Реалізація механізмів cooldown'у та відображення стану здібностей.

Збереження та завантаження стану персонажа.

- Створення механізму збереження інвентаря та стану здібностей у локальні файли.
- Відновлення даних при повторному запуску гри.
-

Тестування системи в рамках ігрового середовища.

- Перевірка працездатності механік у бою, під час дослідження, при переходах між локаціями.

Вхідна інформація:

- Дані про предмети: тип, назва, опис, властивості, кількість.
- Дані про здібності: назва, ефект, тривалість, cooldown, тип (активна/пасивна).

- Дані про персонажа: клас, рівень, доступні здібності, інвентар.
- UI-макети інтерфейсів (створені вручну або за допомогою UE5 UI Editor).

Вихідна інформація:

- Розроблена та інтегрована система керування інвентарем та здібностями.
- Ігровий інтерфейс для перегляду, зміни, використання предметів і здібностей.
- Модуль збереження і завантаження стану персонажа.
- Документація, звіт про тестування, приклади використання.
- Вимоги до розроблювальної системи:

1. Функціональні вимоги:

- Можливість додавання/видалення предметів інвентаря.
- Відображення опису, параметрів, кількості предметів.
- Активація здібностей з урахуванням cooldown'у та умов.
- Можливість сортування/групування інвентаря за типами.
- Збереження/завантаження інвентаря та здібностей у файл.

2. Нефункціональні вимоги:

- Зручний та адаптивний інтерфейс.
- Висока продуктивність (оптимізовані масиви, відсутність затримок).
- Надійність роботи при тривалому використанні.
- Легкість у масштабуванні та модифікації структури.

3. Технічні вимоги:

- Unreal Engine 5 як основна платформа розробки.
- Blueprints/C++ для логіки поведінки.
- UI через UMG (Unreal Motion Graphics).
- Локальні .sav файли або структура SaveGame для збереження даних.

· **Класифікація системи:**

1. За типом використання:

- Вбудована підсистема RPG-гри, що забезпечує управління ресурсами персонажа.

2. За масштабом:

- Локальна (в межах одного гравця і однієї гри).

3. За типом взаємодії:

- Інтерактивна ігрова система з прямою реакцією на дії користувача.

4. За ступенем автоматизації:

- Напівавтоматизована система: частина процесів активується вручну, частина — обробляється автоматично (наприклад, cooldown).

.

1.2 Аналіз існуючих рішень

1.2.1 Існуючі рішення для системи інвентарю

Аналізуючи сучасні ігрові проекти, можна виявити різні підходи до реалізації систем інвентаря. Розглянемо найбільш вдалі приклади з популярних ігор, які можуть стати орієнтиром для нашої розробки.

1. Система інвентаря в The Witcher 3: Wild Hunt

Інвентар Геральта організований у вигляді сітки з обмеженою кількістю слотів, а також має обмеження за вагою, що змушує гравця приймати рішення щодо того, які предмети збирати та носити з собою. Предмети поділяються на категорії: зброя (мечі, арбалети), обладунки, витратні матеріали (зілля, їжа), інгредієнти для алхімії та крафту, квестові предмети та сміття. Кожен предмет має детальний опис, статистику та рідкість. Взаємодія з предметами включає

екіпірування, використання, викидання, продаж торговцям та розбирання на компоненти.

Архітектурні особливості: Ймовірно, використовується компонентний підхід, де інвентар є окремим компонентом персонажа. Предмети, ймовірно, реалізовані як об'єкти з базовим класом Item та похідними класами для різних категорій, що дозволяє легко додавати нові предмети та їхні властивості. Система ваги та слотів, ймовірно, керується атрибутами персонажа або компонентами інвентарю.

Переваги та недоліки:

Переваги: Реалістичне обмеження ваги додає глибини геймплею, змушуючи гравця керувати ресурсами. Чітка категоризація предметів спрощує навігацію. Можливість розбирання предметів стимулює збір "сміття".

Недоліки: Обмеження ваги може бути дратівливим для деяких гравців, особливо на початку гри. Інтерфейс інвентарю, хоча й функціональний, іноді може здаватися перевантаженим інформацією.

Рис. 1. Інвентар гри The Witcher 3: Wild Hunt

2. **Інвентар у The Elder Scrolls V: Skyrim**

Skyrim використовує інвентар на основі списку, який не має обмежень за кількістю слотів, але сильно залежить від загальної ваги. Предмети поділяються на категорії (зброя, броня, зілля, книги, їжа, різне), що полегшує пошук. Кожен предмет має вагу, ціну та опис. Гравці можуть екіпірувати зброю та броню,

використовувати зілля, читати книги, продавати предмети та зберігати їх у скринях.

Архітектурні особливості: Ймовірно, реалізовано через базовий клас Item з великою кількістю властивостей та можливістю додавання скриптів для унікальної поведінки. Система ваги, ймовірно, є глобальним атрибутом персонажа, який впливає на швидкість пересування та можливість швидкої подорожі.

Переваги та недоліки:

Переваги: Простота використання завдяки списковому представленню. Відсутність обмежень за слотами дозволяє збирати велику кількість предметів. Можливість модифікації інвентарю через моди.

Недоліки: Надмірне накопичення предметів може призвести до "захараценого" інвентарю. Вагове обмеження може бути незручним, змушуючи гравця постійно скидати предмети або шукати торговців.

Рис. 2. Інвентар гри The Elder Scrolls V: Skyrim

3. Система у Divinity: Original Sin 2

Original Sin 2 є сітковим, але кожен персонаж у групі має свій окремий інвентар, що додає тактичний елемент розподілу предметів. Існує обмеження за вагою, але не за слотами. Предмети можуть бути переміщені між інвентарями персонажів, а також об'єднані в стаки. Особливістю є можливість "контейнерів в інвентарі" (наприклад, рюкзаки), які мають власні слоти та можуть містити інші предмети.

Архітектурні особливості: Складна система, яка, ймовірно, використовує компонентний підхід для кожного персонажа. Предмети можуть бути реалізовані як об'єкти з ієрархією класів, що підтримують вкладеність (контейнери). Мережева синхронізація інвентарю є ключовою для кооперативного режиму.

Переваги та недоліки:

Переваги: Розподілений інвентар між персонажами групи додає стратегічної глибини. Система контейнерів дозволяє краще організувати предмети.

Недоліки: Управління інвентарем кількох персонажів може бути громіздким, особливо для нових гравців. Незважаючи на сітку, велика кількість предметів все одно може призвести до безладу.

Рис. 3. Інвентар гри Divinity: Original Sin 2

4. Рішення від Resident Evil 4

Система інвентарю: Інвентар у Resident Evil 4 (особливо в оригінальній версії) є унікальним "кейс-інвентарем" у вигляді сітки, де кожен предмет займає певну кількість клітинок, і гравцеві потрібно оптимізувати розташування предметів, щоб вмістити якомога більше. Це створює елемент головоломки. Предмети включають зброю, боєприпаси, лікувальні засоби, ключі та квестові предмети.

Архітектурні особливості: Система, ймовірно, базується на двовимірному масиві або сітці, де кожен елемент інвентарю має свої розміри (ширина x висота). Алгоритми для перевірки вільного місця та додавання предметів є ключовими.

Переваги та недоліки:

Переваги: Унікальний та захопливий елемент головоломки, що додає напруги та стратегії до управління ресурсами. Створює відчуття обмеженості та виживання.

Недоліки: Може бути дратівливим, якщо гравець не може вмістити потрібний предмет. Вимагає постійної "тетріс-оптимізації", що не всім подобається.

Рис. 4. Інвентар гри Resident Evil 4

1.2.2 Існуючі рішення для системи здібностей

Система здібностей — ключовий елемент багатьох рольових та екшен-ігор. Вона відповідає за бойову різноманітність, побудову стилю гри та взаємодію з ігровим світом. Сучасні проекти демонструють різні підходи до реалізації здібностей: від складних дерев вмінь до простих списків активних умінь із певними умовами використання. Нижче розглянемо найбільш релевантні приклади.

1. Система здібностей у Dark Souls / Elden Ring

У цих іграх здібності тісно пов'язані з екіпіруванням (зброєю, щитами, талісманами) та магічними заклинаннями/молитвами. Кожна зброя має унікальну "Попіл Війни" (Ash of War) або "Навичку Зброї" (Weapon Art), яка є активною здібністю. Заклинання та молитви вивчаються та екіпіруються у спеціальні слоти. Використання здібностей витрачає очки концентрації (FP) або витривалість. Система не має традиційних "дерев талантів", натомість розвиток персонажа відбувається через розподіл очок характеристик та вибір екіпірування.

Архітектурні особливості: Ймовірно, здібності реалізовані як окремі об'єкти або компоненти, що прикріплюються до зброї або персонажа. Кожна здібність має свої умови активації, ефекти та витрати. Система атрибутів персонажа (сила, спритність, віра, інтелект) безпосередньо впливає на ефективність здібностей.

Переваги та недоліки:

Переваги: Глибока кастомізація білдів через комбінації зброї, здібностей та заклинань. Кожна здібність відчувається унікальною та важливою.

Недоліки: Відсутність чіткого дерева розвитку може бути незрозумілою для нових гравців. Деякі здібності можуть бути незбалансованими.

Рис. 5. Гра Elden ring (використання обраної здібності)

2. Path of Exile 2 (система «гемів»)

Path of Exile відома своєю унікальною системою "гемів" (каменів умінь). Основні здібності (Skill Gems) вставляються в слоти екіпірування (зброя, броня), а потім поєднуються з "каменями підтримки" (Support Gems), які модифікують їхню поведінку (наприклад, збільшують шкоду, додають ланцюговий ефект, змінюють тип шкоди). Це дозволяє створювати величезну кількість комбінацій та кастомізувати кожну здібність. Також є величезне "дерево пасивних умінь" (Passive Skill Tree), що дозволяє додатково розвивати персонажа.

Архітектурні особливості: Складна модульна система, де кожен гем є окремим об'єктом з унікальними властивостями та логікою. Слоти в екіпіруванні, ймовірно, мають свої правила зв'язування гемів. Дерево пасивних умінь, ймовірно, є великим графом, де кожен вузол представляє пасивне уміння.

Переваги та недоліки:

Переваги: Безпрецедентна глибина кастомізації здібностей та білдів. Висока реграбельність завдяки різноманіттю комбінацій.

Недоліки: Надзвичайно високий поріг входу для нових гравців через складність системи. Потребує багато часу для розуміння всіх механік.

Рис. 6. Гра Path of Exile 2 (вікно обраних скілл гемів)

3. Diablo III / IV

У Diablo III та IV здібності прив'язані до класу персонажа. Гравці вибирають активні та пасивні здібності з доступного набору для свого класу. У Diablo III була система "рун", що модифікували здібності, а в Diablo IV є "дерево навичок" (Skill Tree) та "дошка парагонів" (Paragon Board), що дозволяють поглибити кастомізацію. Здібності мають перезарядку та витрати ресурсів (гнів, мана, дух тощо).

Архітектурні особливості: Ймовірно, використовується система, де кожна здібність є об'єктом, пов'язаним з класом персонажа. Дерево навичок та дошка парагонів, ймовірно, реалізовані як графові структури, що дозволяють розблоковувати та покращувати здібності.

Переваги та недоліки:

Переваги: Зрозуміла та інтуїтивно зрозуміла система для більшості гравців. Чітка роль кожного класу та його здібностей.

Недоліки: Може бути менш гнучкою порівняно з іграми, що пропонують повну свободу в комбінаціях здібностей. Деякі гравці можуть відчувати обмеженість у виборі білдів.

Рис. 7. Гра Diablo 4 (дерево вмінь)

4. Hades (Supergiant Games)

Hades – це **roguelike** гра, де здібності є тимчасовими "Дарами" (Boons) від олімпійських богів, які гравець отримує під час кожного проходження. Кожен Дар надає унікальну активну або пасивну здібність, або модифікує існуючі атаки/здібності. Це створює величезну кількість синергій та комбінацій, які змінюються з кожним новим забігом. Також є "Збройові Аспекти" та "Нічні Апгрейди", що дають постійні бонуси.

Архітектурні особливості: Система, ймовірно, базується на динамічному додаванні та видаленні компонентів або ефектів до персонажа. Кожен Дар є окремим об'єктом, що застосовує певні модифікатори до базових атак та здібностей гравця.

Переваги та недоліки:

Переваги: Висока реграбельність завдяки випадковості та різноманіттю комбінацій Дарів. Кожне проходження відчувається унікальним.

Недоліки: Залежність від випадковості може іноді призвести до "невдалих" білдів. Немає традиційного постійного розвитку здібностей, що може не сподобатися гравцям, які шукають довгострокову прогресію.

Рис. 8. Гра Hades (вікно вибору "благословень" (boons))

Висновки для розробки проєкту на основі аналізу існуючих рішень:

Для системи інвентарю:

- **Гнучкість структури**: Аналіз таких ігор, як The Witcher 3, Skyrim, Divinity: Original Sin 2 та Resident Evil 4, показує, що не існує універсального "ідеального" інвентарю. Кожен підхід (сітковий з вагою, списковий з вагою, сітковий з оптимізацією простору) має свої переваги та недоліки, які визначаються жанром та цілями гри. Для нашого проєкту це означає необхідність створення гнучкої базової структури інвентарю, яка дозволить легко адаптувати його під різні потреби, чи то обмеження за слотами, чи за вагою, чи комбінацію.

- **Категоризація та фільтрація**: Успішні системи інвентарю, як у The Witcher 3 та Skyrim, ефективно використовують категоризацію предметів. Це є критично важливим для зручності користувача, особливо при великій кількості предметів. Наш проєкт повинен передбачати чітку

систему тегів та категорій для предметів, а також можливості фільтрації та сортування для покращення UX.

- **Взаємодія з предметами**: Важливо забезпечити інтуїтивно зрозумілі та різноманітні способи взаємодії з предметами (екіпірування, використання, викидання, продаж, розбирання), що відповідає потребам гравця. Досвід Resident Evil 4 підкреслює, що навіть елемент головоломки в інвентарі може бути цікавим, якщо він відповідає загальній концепції гри.

- **Оптимізація для одного гравця**: Оскільки проєкт є однокористувацьким, відпадає необхідність у складних механізмах мережевої реплікації інвентарю. Це дозволяє зосередитися на локальній продуктивності та стабільності, забезпечуючи швидку та безперебійну роботу системи інвентарю без затримок, що характерні для багатокористувацьких середовищ. Головний акцент робиться на ефективній обробці даних та взаємодії з користувачем на стороні клієнта.

Для системи здібностей:

- **Модульність та розширюваність**: Системи здібностей, як у Path of Exile з її гемами, або Hades з дарами, демонструють величезну цінність модульності. Це дозволяє створювати безліч комбінацій та модифікацій здібностей без зміни базового коду. Наш проєкт повинен використовувати подібний підхід, де здібності та ефекти є окремими, легко модифікованими компонентами.

- **Баланс між гнучкістю та доступністю**: Ігри Dark Souls/Elden Ring пропонують високу гнучкість через екіпірування, тоді як Diablo III/IV – більш структуровані дерева навичок. Вибір залежить від цільової аудиторії. Для нашого проєкту важливо знайти баланс між глибиною кастомізації та легкістю освоєння системи для гравця.

- **Управління ресурсами та перезарядкою**: Всі розглянуті ігри використовують механіки витрат ресурсів (мана, FP, витривалість) та перезарядки (кулдауни) для балансування здібностей. Це є фундаментальним аспектом, який необхідно ретельно спроєктувати та налаштувати.

- **Динамічна поведінка**: Досвід Hades показує, що динамічне додавання та модифікація здібностей під час проходження може значно підвищити реграбельність та унікальність кожного ігрового сеансу. Це може бути цінною можливістю для подальшого розвитку нашої системи.

- **Відмова від складної системи атрибутів**: У контексті даного проєкту, для спрощення та зосередження на базових механіках, інтеграція здібностей з детальною системою атрибутів персонажа не передбачається. Ефективність здібностей буде визначатися фіксованими значеннями або залежати від інших, більш простих параметрів, що дозволить зосередитися на їхній функціональності та взаємодії з інвентарем.

1.3 Моделювання предметної області

У процесі розробки комп’ютерної гри, що включає систему управління віртуальним інвентарем та здібностями ігрового персонажа, важливим етапом є моделювання предметної області. Метою цього етапу є формалізація взаємодії між користувачем та підсистемами гри, визначення основних функціональних елементів системи, а також фіксація бізнес-логіки, що буде реалізована під час програмування. У даному розділі буде розглянуто загальні підходи до моделювання, а також представлено приклади діаграм для двох обраних мною підсистем гри — інвентаря та здібностей.

1.3.1 Загальні відомості про етапи розробки комп’ютерних ігор

Процес створення гри зазвичай включає наступні етапи:

- **Формування ідеї та концепції гри**. На цьому етапі формується загальна ідея, визначається жанр, геймплейні механіки, а також наявність внутрішньоігрових систем, таких як інвентар та здібності.

- **Попереднє проєктування.** Включає розробку документації, технічних вимог, створення діаграм, що дозволяють на ранньому етапі описати систему формально.

- **Прототипування.** Розробляються базові механіки у рушії гри (у моєму випадку Unreal Engine), створюється тимчасовий інтерфейс та перевіряється логіка взаємодії з предметами і здібностями.

- **Реалізація**. Реалізується повна функціональність: інвентар, менеджмент предметів, візуалізація, використання здібностей, умови їх активації тощо.

- **Тестування**. Включає перевірку наявності багів, помилок в логіці, взаємодії компонентів, а також зручності для гравця.

- **Випуск гри та подальша підтримка.** Включає збір зворотного зв’язку, оновлення, балансування механік та додавання нового контенту.

1.3.2 Загальні відомості про процеси моделювання для предметної області

Моделювання дозволяє ще до написання коду зрозуміти, як працюватимуть окремі частини гри, як гравець взаємодіятиме із системами, та які компоненти необхідно реалізувати. У проєкті використано стандартні типи UML-діаграм, зокрема:

- **Діаграми прецедентів**, що показують взаємодію користувача з системою через доступні функції;

- **Діаграми послідовності**, що відображають логіку виконання дій користувача та послідовність процесів;

- **Діаграми активності**, які деталізують внутрішню логіку процесів у рамках однієї функції.

Моделювання було розділене на дві частини: одна стосується системи інвентаря, інша — системи здібностей. Нижче наведено відповідні діаграми та пояснення до них.

1.3.3 Діаграми прецедентів

Діаграма прецедентів для системи інвентаря (Рис. 9)

На діаграмі прецедентів для системи інвентаря зображено основні дії, до яких має доступ гравець: підбір предмету, перегляд інвентарю, дії над предметов у самому інвентарю (видалення, знятя, використання, екіпірування, переміщення та перевірка стану предмету) та сортування інвентарю.

В свою чергу гра (система) виконує автоматизовані операції, зокрема додавання предмету до інвентаря після його підбору, видалення предмету у разі використання або знищення, перевірку обмежень інвентаря за кількістю предметів та доступних слотів для їх зберігання, а також збереження актуального стану інвентаря для подальшого відновлення при наступному запуску гри або зміні ігрового стану.

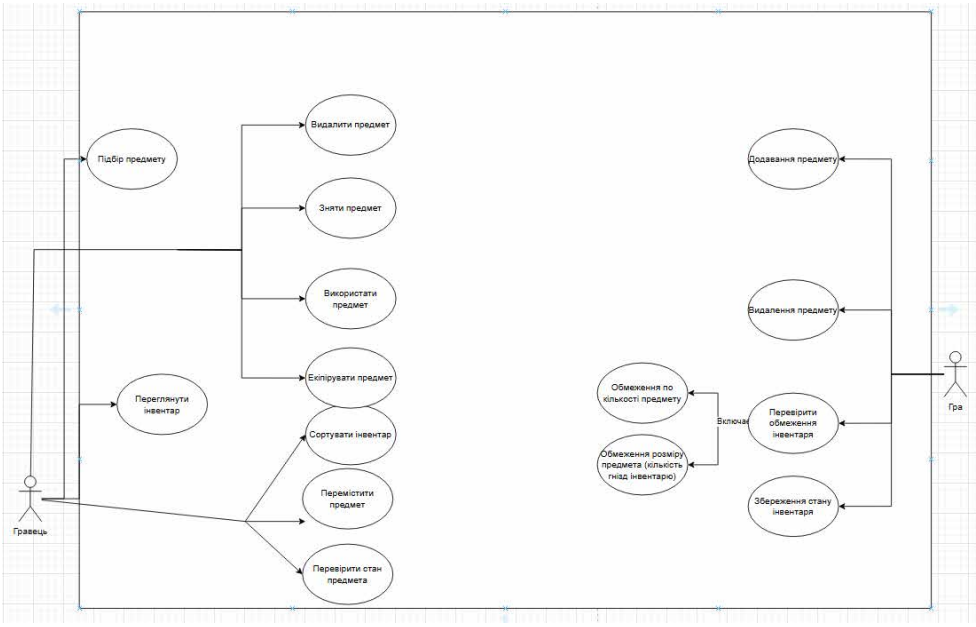

Рис. 9 Діаграма прецедентів взаємодії гравця із системою інвентарю

Діаграма прецедентів для системи здібностей (Рис. 10)

Діаграма прецедентів для системи здібностей відображає ключові дії, які доступні гравцю при взаємодії з активними здібностями персонажа. Основними прецедентами є перегляд списку доступних здібностей, активація обраної здібності під час бою або інших ситуацій, а також скасування або відключення активної здібності.

Крім того, гравець може керувати набором здібностей — додавати нові здібності після їх отримання або відкриття, замінювати існуючі, налаштовувати порядок використання здібностей та переглядати докладну інформацію про кожну з них (ефекти, умови активації, час перезарядки).

Зі сторони системи (гри) реалізовано перевірку умов активації здібностей, наприклад наявність необхідних ресурсів або дотримання затримки перезарядки, автоматичне оновлення стану здібностей після їх використання, а також збереження та відновлення конфігурації здібностей при збереженні та завантаженні ігрового процесу.

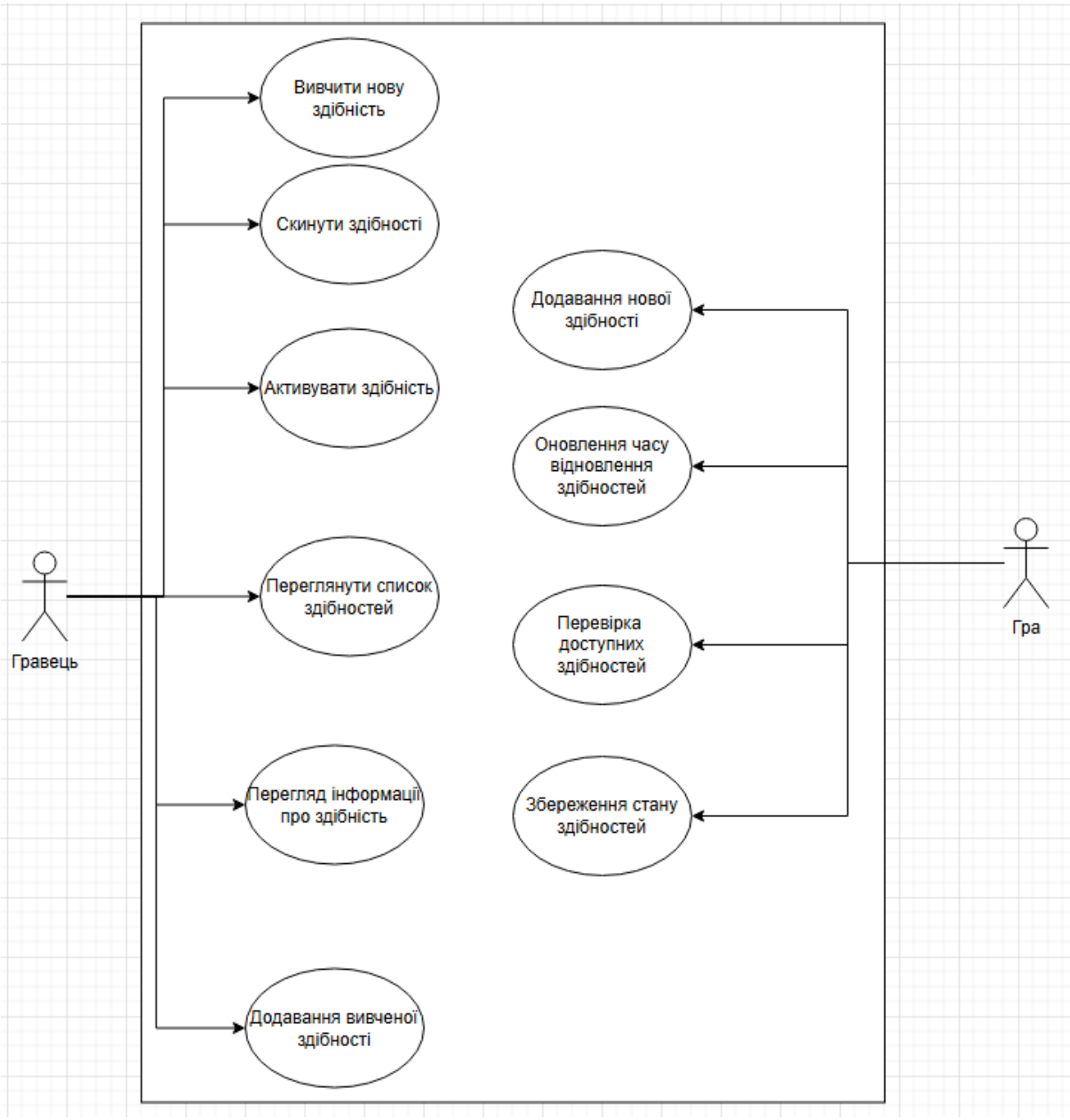

Рис. 10 Діаграма прецедентів взаємодії гравця із системою здібностей

1.3.4 Діаграми послідовності

Діаграма послідовності для інвентаря (рис. 11)

Дана діаграма послідовності ілюструє алгоритм взаємодії гравця з системою інвентарю під час виконання операцій додавання та видалення предметів. Процес відображає послідовність дій, повідомлень і умовних переходів, що забезпечують коректну роботу інвентарю в грі.

1. Ініціація взаємодії

Гравець через інтерфейс гри (UI) ініціює дію, наприклад:

- Натискає кнопку "Додати предмет" або "Видалити предмет".
- Відкриває інвентар ("Відкрити інвентар").

---

2. Обробка запитів системою інвентарю

Додавання предмета:

1. Система перевіряє наявність вільного місця:
   - Якщо місця достатньо, предмет додається ("Предмет додано").
   - Якщо місця недостатньо, гравець отримує повідомлення "Недостатньо місця".
2. У разі переповнення інвентарю може видалити непотрібний предмет, щоб замінити його на новий.

Видалення предмета:

1. Гравець обирає "Видалити предмет".
2. Система підтверджує операцію ("Предмет успішно видалено з інвентарю").

---

3. Відображення результатів

Після кожної операції інтерфейс оновлюється:

- Відображається змінений вміст інвентарю.
- Виводяться повідомлення про статус дії (наприклад, "Місця достатньо").

---

4. Принципи роботи системи

Інтерфейс гри (UI):

- Відповідає за візуалізацію інвентарю та обробку введення гравця.

Система інвентарю:

- Керує логікою додавання/видалення предметів, перевіркою місця та оновленням даних.

Умовні переходи:

- Забезпечують гнучкість роботи (наприклад, обробка помилок при недостатньому місці).

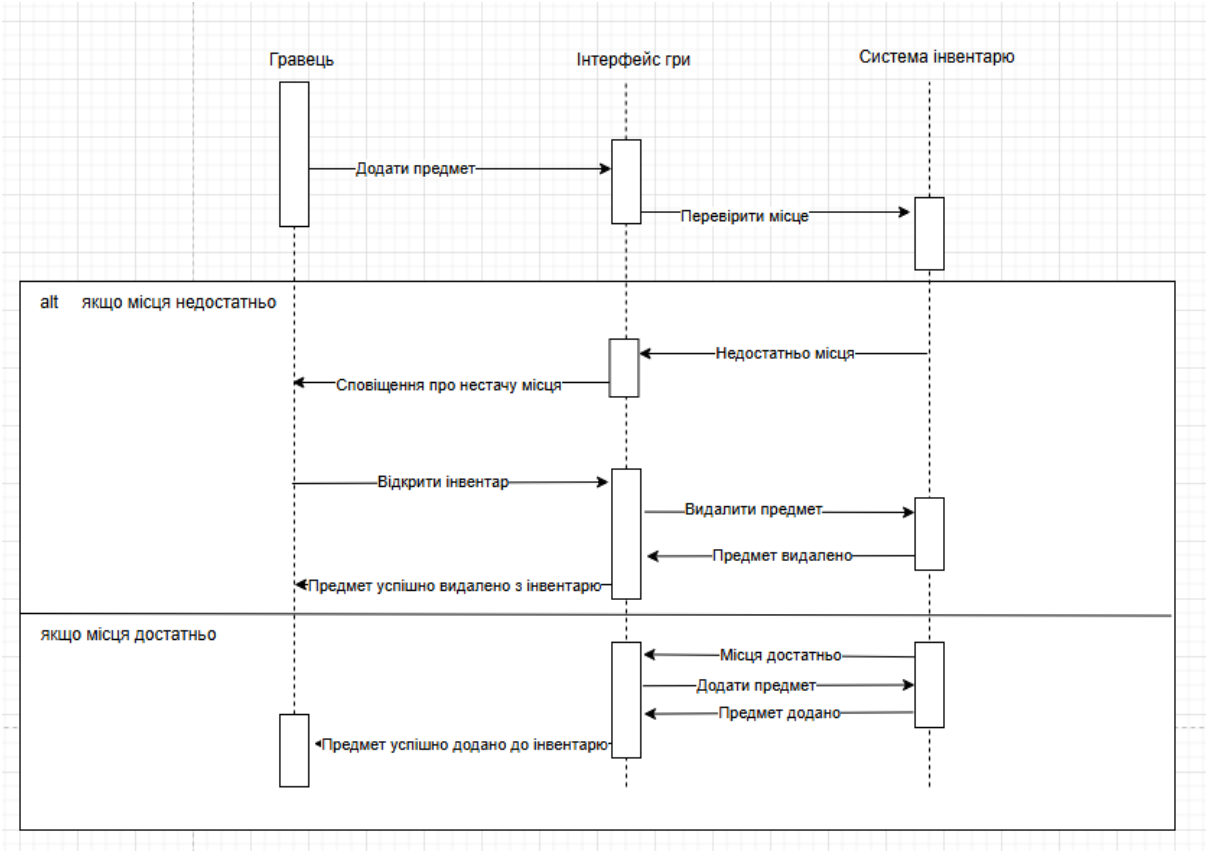

Рис. 11 Діаграма послідовності додавання предмету в інвентар

Діаграма послідовності для здібностей (рис. 12)

Дана діаграма демонструє логіку взаємодії гравця з системою здібностей під час спроби активувати вміння. Процес включає перевірку ресурсів (мани) та стану перезарядки здібності, а також обробку різних сценаріїв у разі успіху чи невдачі.

1. Ініціація дії гравцем

- Гравець через інтерфейс здібностей (UI) намагається активувати обране вміння (натискає кнопку, вибирає комбінацію клавіш тощо).
- Система отримує запит "Намагається активувати здібність".

2. Перевірка умов для активації

Система здібностей і Менеджер ресурсів та станів послідовно перевіряють дві критичні умови:

1. Наявність мани:

- Виконується запит "Перевірити кількість мани".

2. Статус перезарядки здібності:

- Виконується запит "Перевірити відкат уміння".

3. Сценарії обробки

Діаграма описує **три альтернативні гілк**и (позначені alt):

Сценарій 1: Успішна активація

Умови:

- Мани достатньо ("Мани достатньо").
- Здібність готова (завершила перезарядку, "Здібність готова для застосування").

Дії:

- Система активує здібність ("Активація здібності").
- Гравець отримує підтвердження (анімація застосування здібності).

Сценарій 2: Нестача мани

Умови:

- Мани недостатньо ("Мани недостатньо").

Дії:

- Активація блокується ("Активація здібності не успішна").
- Гравець отримує повідомлення про нестачу ресурсів ("Повідомлення про нестачу мани").
- Система може запропонувати чекати на відновлення мани ("Очікується поповнення мани").

Сценарій 3: Здібність на перезарядці

Умови:

- Здібність не перезаряджена ("Здібність не готова для застосування").

Дії:

- Активація відхилена ("Активація здібності не успішна").
- Гравець отримує повідомлення про відкат ("Повідомлення про перезарядку здібності").

- Система вказує час до готовності ("Очікується готовність здібності").

4. Відповідальність компонентів

Інтерфейс здібностей (UI):

- Відображає стан здібностей (час перезарядки, вартість мани) та сповіщення.

Система здібностей:

- Керує логікою активації, перевіряє умови.

Менеджер ресурсів та станів:

- Надає дані про кількість мани та статус перезарядки.

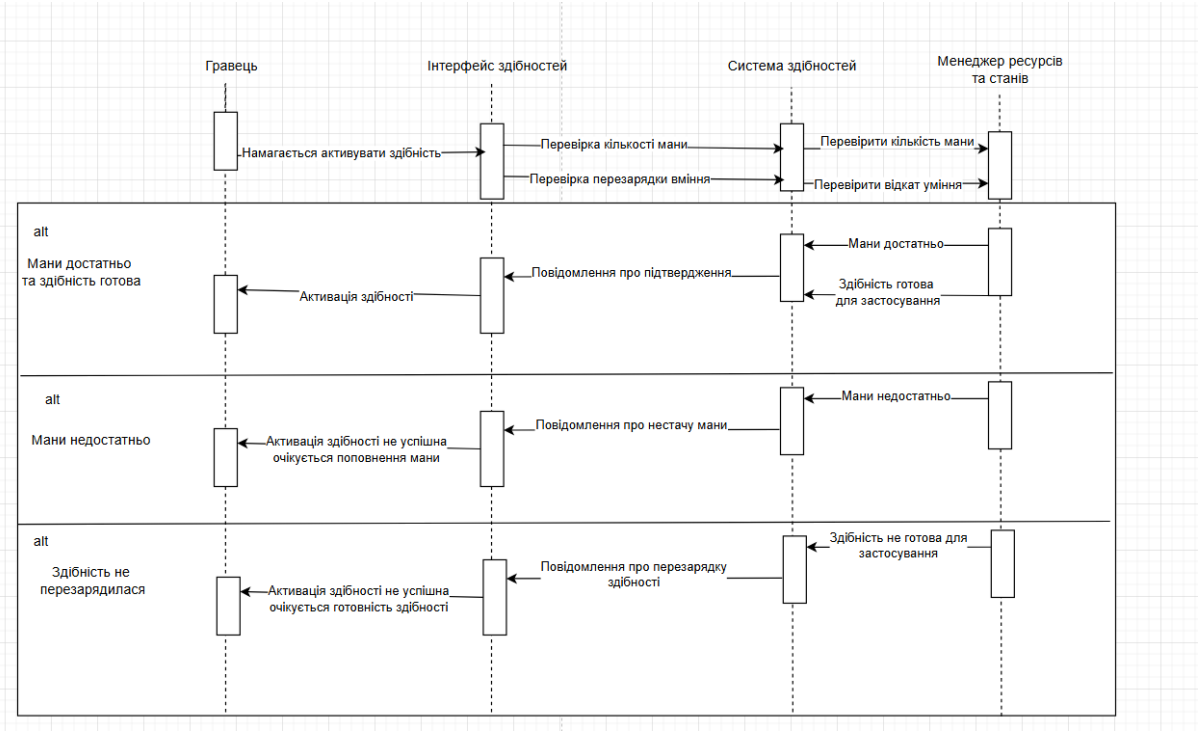

Рис. 12 Діаграма послідовності активації здібності

1.3.5 Діаграми активності

Діаграма активності для інвентаря (рис. 13)

Діаграма активності ілюструє процес взаємодії гравця з інтерфейсом інвентарю у грі. Вона відображає послідовність дій від відкриття вікна інвентарю до додавання предметів, перевірку наявності вільного місця та можливі альтернативні сценарії у випадку його відсутності. У разі нестачі місця система повідомляє гравця, який може прийняти рішення про видалення предмета або відмовитися від додавання нового. Такий підхід дозволяє детально описати як основний сценарій, так і альтернативні гілки поведінки користувача, що забезпечує гнучкість і наочність моделювання.

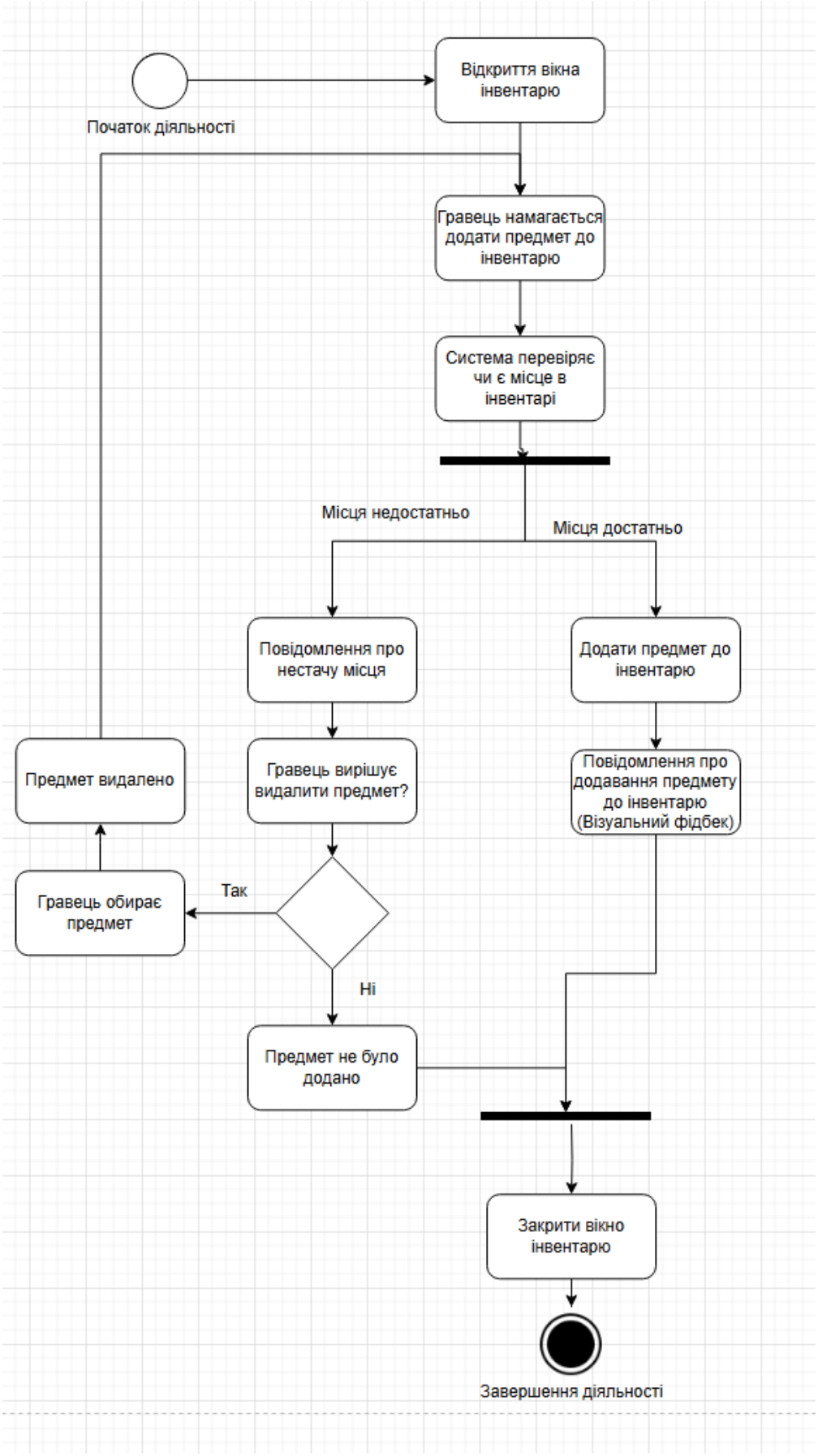

Рис. 13 Діаграма активності додавання предмету в інвентар

Діаграма активності для здібностей(рис. 14)

Діаграма активності ілюструє основні етапи використання здібностей ігровим персонажем. Процес починається з вибору здібності через інтерфейс гравця та спроби активації. Після цього система виконує перевірку наявності достатньої кількості мани та стану кулдауну для обраної здібності. Якщо ресурси достатні, а кулдаун завершено, здібність активується, і відбувається відповідна зміна стану персонажа або середовища гри. У разі, коли мани недостатньо або здібність ще перебуває на перезарядці, система повідомляє гравця про неможливість активації, і він може обрати іншу дію.

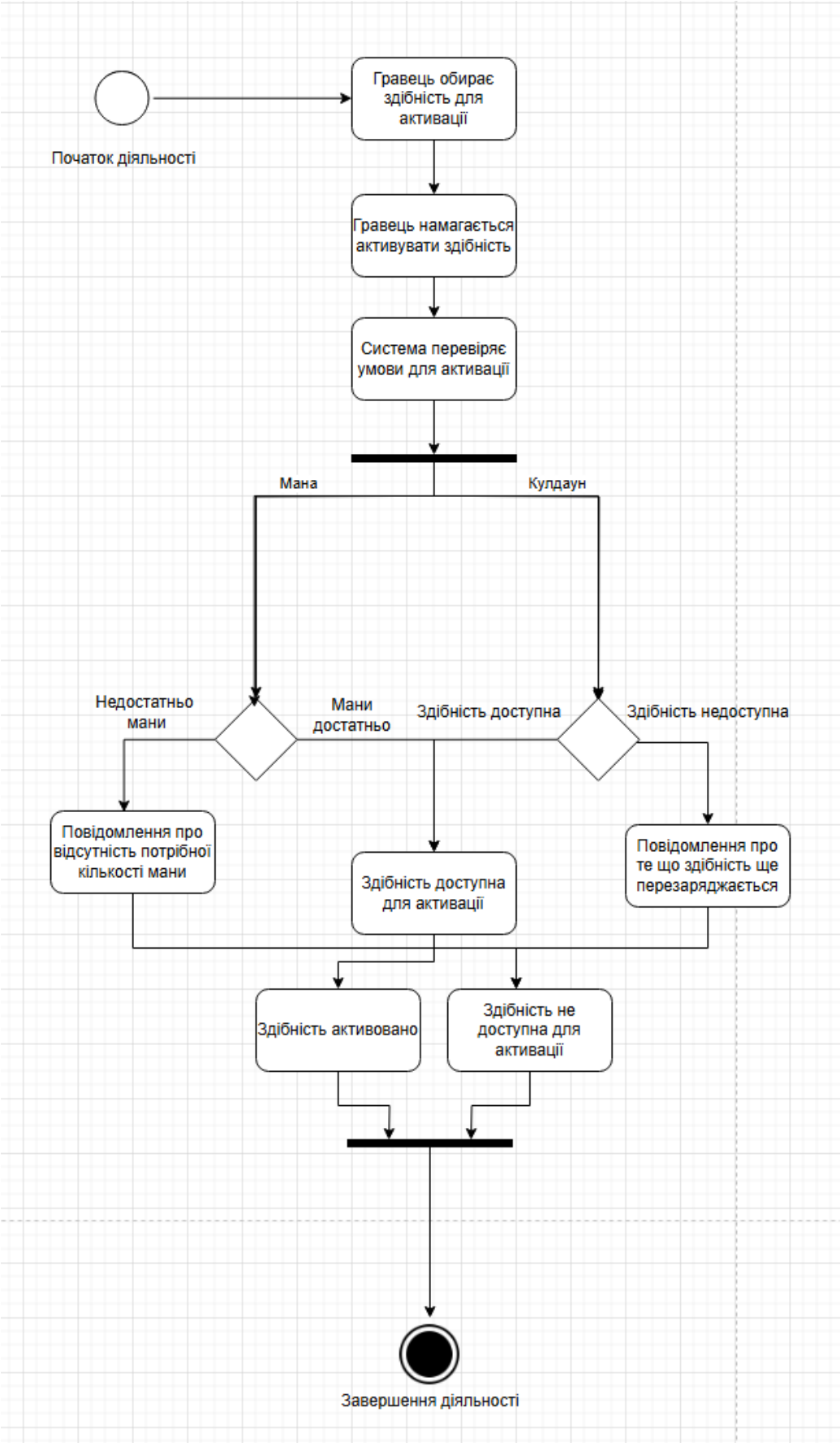

Рис. 14 Діаграма активності активації здібності

РОЗДІЛ 2

ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Побудова логічної моделі

Логічна модель даних є ключовим етапом у проектуванні інформаційної системи, оскільки дозволяє формалізувати структуру об’єктів предметної області, їхні властивості та зв’язки між ними. У контексті даного проєкту — однокористувацької гри з системами інвентаря та здібностей — модель даних забезпечує основу для збереження інформації про гравця, його предмети та активні здібності.

Метою логічного моделювання є створення структурованої, послідовної та придатної до реалізації моделі, що дозволить зручно обробляти ігрові дані в середовищі Unreal Engine 5.

У розробленій системі передбачено такі основні сутності:

- **Player** — основна сутність, яка представляє гравця. Містить загальні дані про гравця, зокрема його ім’я, рівень, список здібностей та інвентар.
- **Inventory** — сутність, яка відповідає за збереження інформації про предмети, що знаходяться у гравця.
- **Item** — окрема одиниця предмета, яка може мати унікальні властивості (тип, розмір, рідкість, ефекти).
- **AbilityList** — список здібностей, прив’язаний до конкретного гравця.
- **Ability** — окрема здібність, яка може бути активною або пасивною, з визначеними параметрами використання та ефектами.

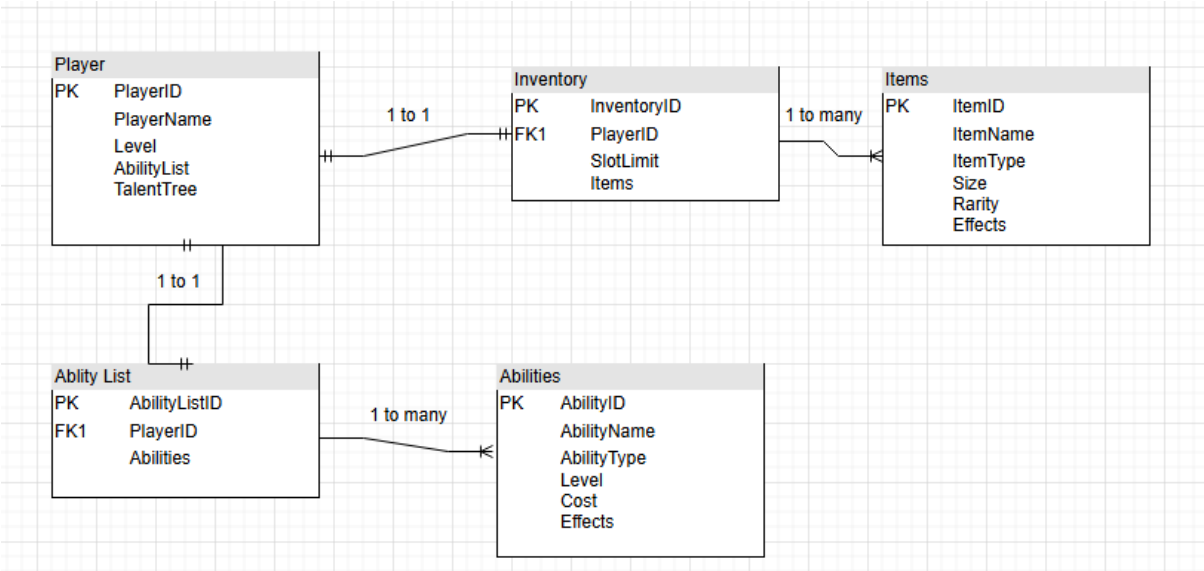

Рис. 15 Логічна модель даних

2.2 Вибір та обґрунтування вибору системи управління базою даних

Під час розробки комп'ютерної гри важливим аспектом є правильний вибір системи управління базами даних (СУБД), яка забезпечить надійне та ефективне збереження і доступ до даних про гравця, інвентар та здібності. Для цього було розглянуто кілька типів СУБД, серед яких реляційні, документно-орієнтовані, а також варіанти локального збереження даних, які пропонує рушій Unreal Engine.

Реляційні СУБД (наприклад, MySQL, PostgreSQL) широко використовуються завдяки своїй структурованості та надійності. Вони дозволяють виконувати складні запити та підтримують цілісність даних за рахунок жорстких правил типів і зв'язків. Проте, їх застосування в контексті однокористувацької гри має низку недоліків. Зокрема, інтеграція таких СУБД потребує розгортання серверної частини або налаштування локального сервера, що значно ускладнює розробку та підвищує вимоги до ресурсів. Також затримки при зверненні до бази можуть негативно вплинути на продуктивність гри.

Документно-орієнтовані СУБД (наприклад, MongoDB) дозволяють зберігати дані у форматі JSON-подібних документів, що дає більшу гнучкість у структурі даних і полегшує збереження вкладених об'єктів. Однак подібні системи також вимагають додаткової інфраструктури і рідко використовуються в однокористувацьких іграх через складність налаштування.

У зв'язку з особливостями гри, що розробляється на Unreal Engine 5, а саме – однокористувацький режим, відсутність потреби у серверній взаємодії та високі вимоги до швидкості завантаження і збереження даних, було прийнято рішення використовувати вбудовані засоби збереження даних рушія. Unreal

Engine підтримує локальне збереження інформації у вигляді файлів (JSON або бінарних), що дає змогу швидко зберігати стан гравця, його інвентар та вибрані здібності без додаткових звернень до зовнішніх систем.

Переваги такого підходу:
Простота реалізації – не потрібно налаштовувати сервер або окрему базу даних, що знижує складність розробки.

Висока швидкість доступу – локальні файли зчитуються та записуються швидко, що позитивно впливає на плавність ігрового процесу.

Автономність – збереження не залежить від інтернет-з'єднання або зовнішніх серверів.

Гнучкість структури – збереження даних у форматі JSON дозволяє легко розширювати модель, додавати нові властивості без складних міграцій.

Недоліки:
Відсутність централізованого сховища, що ускладнює мультиплеєрні сценарії.

Можливі проблеми з безпекою при збереженні локальних файлів.

Обмеження обсягів збережених даних, якщо порівнювати з повноцінними СУБД.

Враховуючи, що основна мета – створення зручної та швидкої системи управління інвентарем і здібностями у однокористувацькій грі, вибір на користь локального збереження в Unreal Engine є оптимальним. Це дозволяє знизити залежність від сторонніх технологій і сфокусуватися на внутрішній логіці гри без ускладнень, пов'язаних із серверною частиною.

2.3 Реалізація моделі даних

2.3.1 Моделі даних здібностей

У проєкті реалізація моделі даних для системи здібностей гравця базується на архітектурі Unreal Engine 5 із застосуванням Gameplay Ability System (GAS). Основна ідея полягає в розділенні логіки здібностей на два рівні: Blueprint актора здібності та Gameplay Ability Blueprint.

Blueprint актора здібності (наприклад, BP_Fireball) відповідає за параметри самої здібності — це місце зберігання таких даних, як величина урону, ефекти, швидкість руху, а також інші характеристики поведінки (наприклад, час дії, радіус вибуху). Тут зосереджена і візуальна частина — модель фаєрболла, ефекти горіння, а також усі ігрові параметри, що безпосередньо впливають на взаємодію здібності з оточенням.

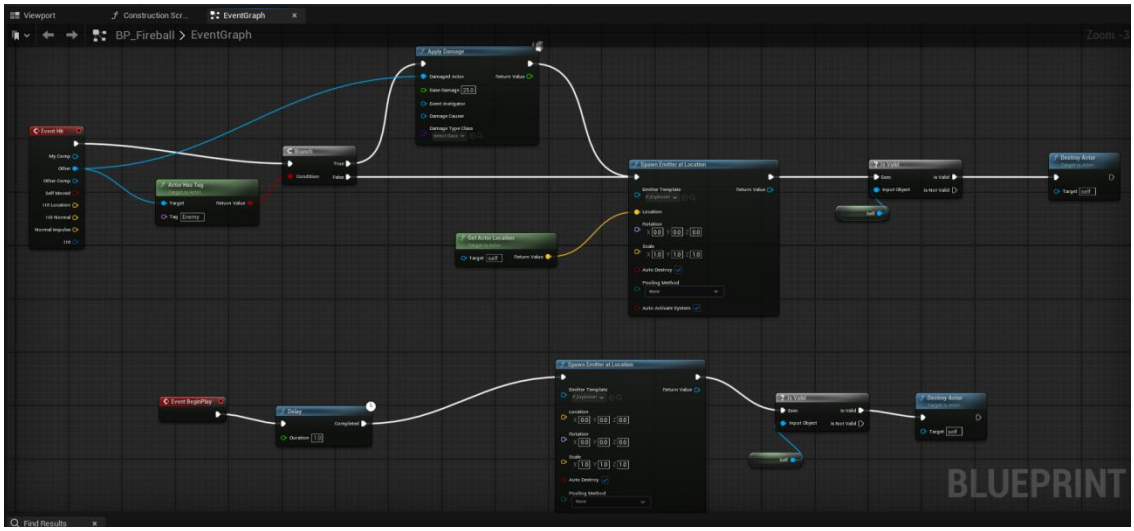

Рис. 16 Blueprint актора здібності “Fireball”

Gameplay Ability Blueprint (наприклад, GA_Fireball) відповідає за логіку спавну та активації здібності: визначає місце й напрямок створення актора здібності у світі, а також інші аспекти механіки активації (анімовані ефекти запуску, умови для активації, тригери тощо). Він не зберігає параметри урону чи ефекти, а лише контролює, коли і як здібність починає діяти.

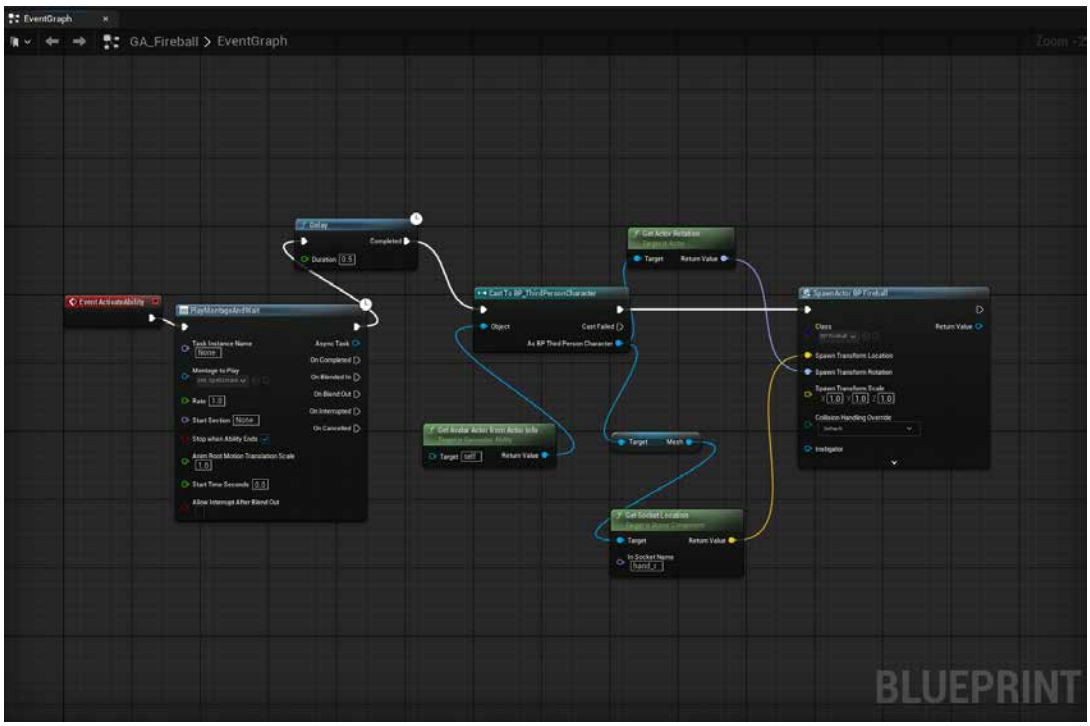

Рис. 17 Gameplay ability для здібності “Fireball”

Таким чином, розділення на BP та GA забезпечує гнучкість та зрозумілу структуру проекту — зміна параметрів здібності не потребує зміни її логіки спавну і навпаки.

Варто відзначити, що у цьому проєкті дані здібностей не зберігаються у традиційній реляційній базі даних, а управляються безпосередньо через внутрішні механізми Unreal Engine 5. Це обумовлено специфікою ігрового процесу, де критично важливо мати швидкий доступ до даних та тісну інтеграцію з механіками рушія.

На рисунках 16 та 17 наведені приклади Blueprint актора здібності і Gameplay Ability Blueprint, що ілюструють основні компоненти моделі даних і механізми взаємодії.

Таким чином, реалізація моделі даних для здібностей у рамках GAS дозволяє ефективно та зручно керувати складною ігровою логікою, одночасно зберігаючи чистоту і модульність архітектури гри.

2.3.2 Моделі даних інвентарю

Реалізація моделі даних інвентарю в системі базується на поєднанні **Data Table** та внутрішніх структур у вигляді масивів, що дозволяє ефективно зберігати та управляти інформацією про предмети гравця. Цей підхід є оптимальним для однокористувацького проєкту, забезпечуючи високу продуктивність та гнучкість.

Data Table використовується для зберігання основних, статичних характеристик всіх доступних предметів у грі. Це дозволяє централізовано керувати даними про предмети, що значно спрощує їхнє створення, модифікацію та балансування. Кожен рядок у Data Table представляє унікальний тип предмета і містить такі параметри, як:

- **Назва предмета (DisplayName):** Зрозуміла назва для відображення в інтерфейсі.
- **Тип предмета (ItemType)**: Категорія предмета (наприклад, "Зброя", "Броня", "Зілля", "Квестовий предмет"), що важливо для фільтрації та логіки використання.
- **Рідкість (Rarity):** Впливає на візуальне оформлення та потенційні властивості предмета.
- **Іконка (Icon):** Шлях до текстури або посилання на іконку для відображення в інвентарі.
- **Опис (Description):** Текстовий опис предмета, що надає гравцеві інформацію про його властивості та призначення.

- **Максимальний розмір стаку (MaxStackSize):** Визначає, скільки одиниць цього предмета може бути в одному слоті інвентарю.

- **Вага (Weight):** Впливає на загальну вагу інвентарю персонажа, якщо така система використовується.

- **Базові параметри ефектів (BaseEffectValue):** Початкові значення для ефектів, які предмет може застосовувати (наприклад, кількість відновленого здоров'я для зілля, базова шкода для зброї).

- **Посилання на Gameplay Effect (GameplayEffectClass):** Якщо предмет застосовує ефекти через GAS, тут може зберігатися посилання на відповідний Gameplay Effect.

Такий підхід дозволяє дизайнерам швидко додавати нові предмети та налаштовувати їхні властивості без потреби у зміні коду, забезпечуючи високу ітеративність розробки.

Інвентар гравця реалізований як масив структур або об'єктів у блюпринтах Unreal Engine. Цей масив зберігається в пам'яті під час виконання гри і представляє поточний стан інвентарю персонажа. Кожен елемент цього масиву (наприклад, структура FInventorySlot або об'єкт UInventoryItemInstance) містить:

- **Посилання на конкретний предмет із Data Table (ItemID або FDataTableRowHandle):** Це дозволяє отримати доступ до всіх статичних характеристик предмета.

- **Кількість одиниць предмета (CurrentStackSize):** Для стакованих предметів.

- **Унікальні властивості екземпляра (UniqueInstanceID, Durability, CurrentCharges, CustomStats):** Параметри, які можуть змінюватися для конкретного екземпляра предмета під час гри (наприклад, міцність зброї, кількість зарядів у зілля, випадкові бонуси).

Такий підхід забезпечує динамічне керування інвентарем у процесі гри: додавання, видалення або оновлення предметів відбувається безпосередньо у пам’яті, що гарантує високу швидкість обробки та відсутність затримок.

Відсутність використання традиційної системи керування базами даних (СУБД) пояснюється кількома ключовими факторами, що особливо актуальні для однокористувацького проєкту на Unreal Engine:

- **Особливості Unreal Engine:** Двигун забезпечує гнучкість через роботу з Data Table та Data Assets, які є вбудованими та оптимізованими механізмами для зберігання ігрових даних. Вони інтегровані безпосередньо в редактор, що спрощує робочий процес.

- **Спрощення розробки:** Використання внутрішніх механізмів Unreal Engine значно спрощує розробку, дозволяючи інтегрувати логіку

інвентарю без додаткових накладних витрат на налаштування, синхронізацію та обробку зовнішніх СУБД (наприклад, MySQL, PostgreSQL, SQLite). Це усуває необхідність у додаткових бібліотеках, драйверах та логіці взаємодії з базою даних.

- **Продуктивність для одного гравця:** Для однокористувацьких ігор, де всі дані знаходяться локально, продуктивність вбудованих Data Table та масивів у пам'яті є більш ніж достатньою. Завантаження всіх необхідних даних на старті гри та подальша робота з ними в оперативній пам'яті забезпечує миттєвий доступ до інформації про предмети та високу швидкість виконання операцій з інвентарем. Зовнішні СУБД могли б додати зайві затримки та складність без значних переваг для даного типу проєкту.

Загалом, обрана модель даних для інвентарю поєднує у собі надійність зберігання статичних даних через Data Table та зручність, динамічність і високу продуктивність управління екземплярами предметів у грі через масиви структур/об'єктів у пам'яті, що відповідає вимогам продуктивності та простоти реалізації для даного однокористувацького проєкту.

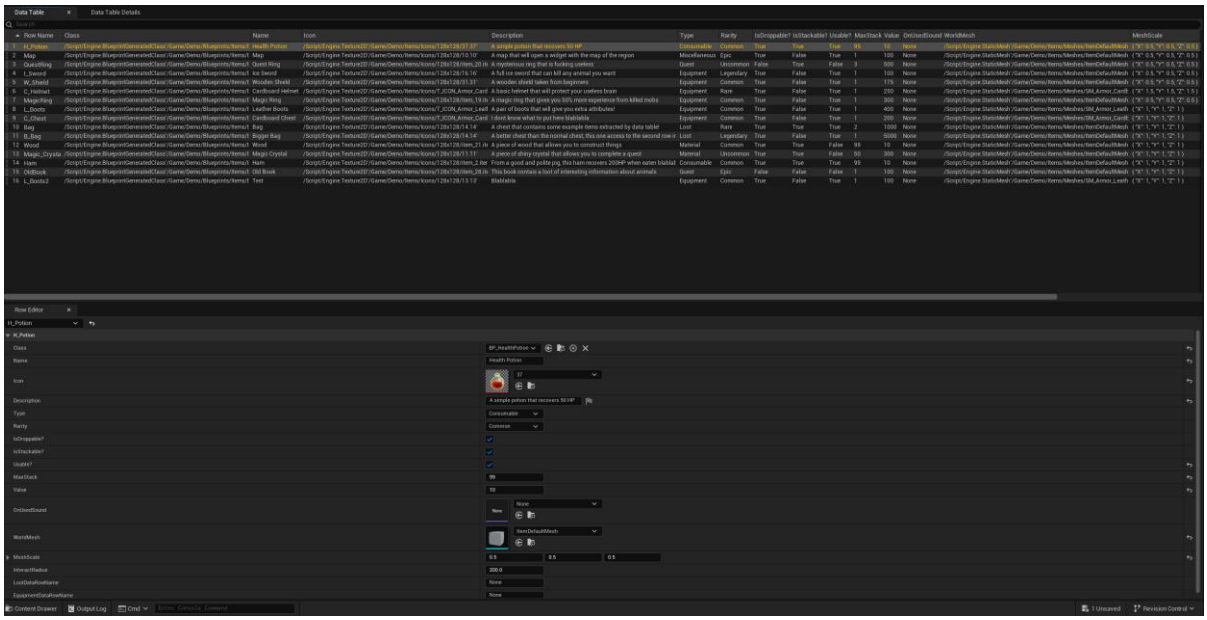

Рис. 18 Data Table предметів

2.4 Проєктування інтерфейсу користувача

Детальний опис елементів інтерфейсу:

Інтерфейс інвентарю (Inventory UI):

- **Загальний вигляд**: Інтерфейс інвентарю представлений у вигляді двох основних панелей: "Equipment" (Екіпірування) та "Inventory" (Інвентар), розташованих у правій частині екрана. Це забезпечує чітке розділення між екіпірованими предметами та тими, що знаходяться в сумці.

- **Панель "Equipment":** Містить слоти для екіпірування різних типів предметів, таких як броня (голова, торс, руки, ноги), зброя та аксесуари. Кожен слот візуально відображає тип предмета, який може бути в ньому розміщений. Наприклад, на наданому зображенні видно слот для шолома та броні. Це дозволяє гравцеві швидко переглядати та змінювати екіпіровку персонажа.

- **Панель "Inventory":** Реалізована у вигляді сітки слотів, що дозволяє візуально організувати предмети. Кожен слот може містити один предмет або стек ідентичних предметів. На іконках предметів чітко відображається кількість одиниць у стеку (наприклад, "x50", "x40", "x16"), що є важливим для управління витратними матеріалами та ресурсами. Приклади предметів включають зілля, кристали, факели та інші ігрові об'єкти.

- **Індикатори**: Під панеллю інвентарю відображається поточна кількість ігрової валюти (наприклад, "¥ 50"), що є важливим елементом для взаємодії з торговцями.

- **Взаємодія**: Передбачається підтримка механіки перетягування (drag-and-drop) для переміщення предметів між слотами інвентарю та екіпірування, а також для викидання предметів. Додаткові дії (використати, розібрати, продати) можуть бути доступні через контекстне меню (наприклад, по правому кліку миші).

- **Дизайн:** Елементи інтерфейсу мають заокруглені кути та темний, контрастний фон, що забезпечує хорошу читабельність іконок та тексту.

Рис. 19 Інвентар персонажа

Панель здібностей (Skill Bar):

- **Загальний вигляд:** Хоча на наданих зображеннях немає повноцінного скілл-бару, типовий дизайн передбачає його розташування в нижній частині екрана, забезпечуючи швидкий доступ до активних здібностей персонажа.

- **Слоти для здібностей:** Скілл-бар складається з ряду слотів, кожен з яких відображає іконку певної здібності. Ці слоти можуть бути прив'язані до клавіш швидкого доступу (наприклад, 1, 2, 3, 4).

- **Візуалізація стану:** Кожна іконка здібності повинна візуально відображати її поточний стан: активна, на перезарядці (кулдауні), недостатньо ресурсів для використання. Це може бути реалізовано за допомогою затемнення іконки, таймера зворотного відліку або індикатора витрат ресурсів.

- **Приклад іконки здібності:** На зображенні показано приклад іконки, яка може бути використана для представлення здібності або предмета на скілл-барі. Яскравий, чіткий дизайн іконок є ключовим для швидкого розпізнавання.

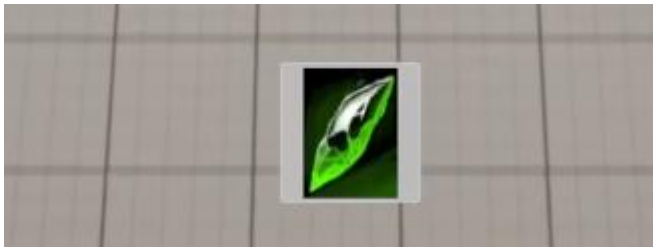

Рис. 20 Приклад здібності в панелі здібностей

Індикатор здоров'я (HP Bar):

- Загальний вигляд: Індикатор здоров'я є одним з найважливіших елементів HUD, що надає гравцеві миттєвий зворотний зв'язок про стан його персонажа. Він зазвичай розташовується у верхній або нижній частині екрана.

- Візуалізація: На зображенні видно горизонтальну смугу червоного кольору, яка візуально відображає поточний рівень здоров'я. Зменшення заповнення смуги вказує на отриману шкоду.

- Числове значення: Поряд зі смугою відображається числове значення поточного та максимального здоров'я (наприклад, "Health: 300 / 500"). Це надає гравцеві точну інформацію та дозволяє краще планувати свої дії.

- Дизайн: Чіткий, контрастний дизайн смуги здоров'я забезпечує її помітність навіть у динамічних сценах.

Рис. 21 Індикатор здоров’я

РОЗДІЛ 3

ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Організація програмного забезпечення

Розробка прикладного програмного забезпечення (ППЗ) для комп'ютерної гри передбачає створення чітко структурованої системи, що складається з окремих модулів, які виконують специфічні функції. Правильна організація ППЗ є одним із ключових факторів, що визначають якість, зручність підтримки та масштабованість проєкту.

У розглянутій системі, що реалізує управління віртуальним інвентарем та здібностями ігрового персонажа, програмне забезпечення поділено на декілька логічних пакетів:

1. Інтерфейс користувача (UI)

Цей пакет відповідає за взаємодію гравця з грою, візуалізацію інформації про інвентар, навички та інші елементи геймплею. В Unreal Engine 5 UI створений із застосуванням системи віджетів (UMG – Unreal Motion Graphics), що забезпечує гнучкі можливості дизайну та адаптивність під різні роздільні здатності екранів.

Інтерфейс розділено на кілька підсистем: основний HUD (інформація про здоров'я та ману), меню інвентарю, вікно здібностей. Така деталізація дозволяє зручно модифікувати кожен елемент незалежно, не порушуючи цілісність UI.

2. Модуль управління інвентарем

Логіка інвентарю реалізована через масив структур даних, де кожен елемент містить посилання на конкретний предмет, описаний у Data Table. Цей підхід є оптимальним для ігор на UE5, оскільки Data Table дозволяє

централізовано зберігати інформацію про предмети, а блюпрінти забезпечують динамічне маніпулювання інвентарем під час гри.

У модулі реалізовано функції додавання та видалення предметів, обмеження максимального розміру інвентарю, сортування та фільтрацію за категоріями.

3. Модуль управління здібностями

Для реалізації системи активних здібностей використовується Gameplay Ability System (GAS) Unreal Engine, що є потужним фреймворком для створення ігрових механік. У GAS кожна здібність представлена як окремий об’єкт (Gameplay Ability), який містить інформацію про анімації, умови активації, ефекти та логіку нанесення шкоди.

GAS дозволяє гнучко керувати життєвим циклом здібностей, їх накладанням, а також масштабувати систему для майбутніх доповнень. Використання GAS забезпечує відокремлення логіки здібностей від інших систем гри, що підвищує модульність та зручність тестування.

4. Модуль збереження та завантаження

Збереження прогресу є необхідною складовою сучасних ігор. У нашій системі стан гравця (включно з інвентарем, активними здібностями, позицією в світі) зберігається у локальні файли. Використовуються формати JSON або Binary, що дозволяє зберігати структури даних у компактному вигляді та забезпечувати швидке завантаження.

Реалізація збереження базується на вбудованих класах UE5 для серіалізації даних, що дозволяє легко масштабувати систему та уникати помилок при читанні-записі.

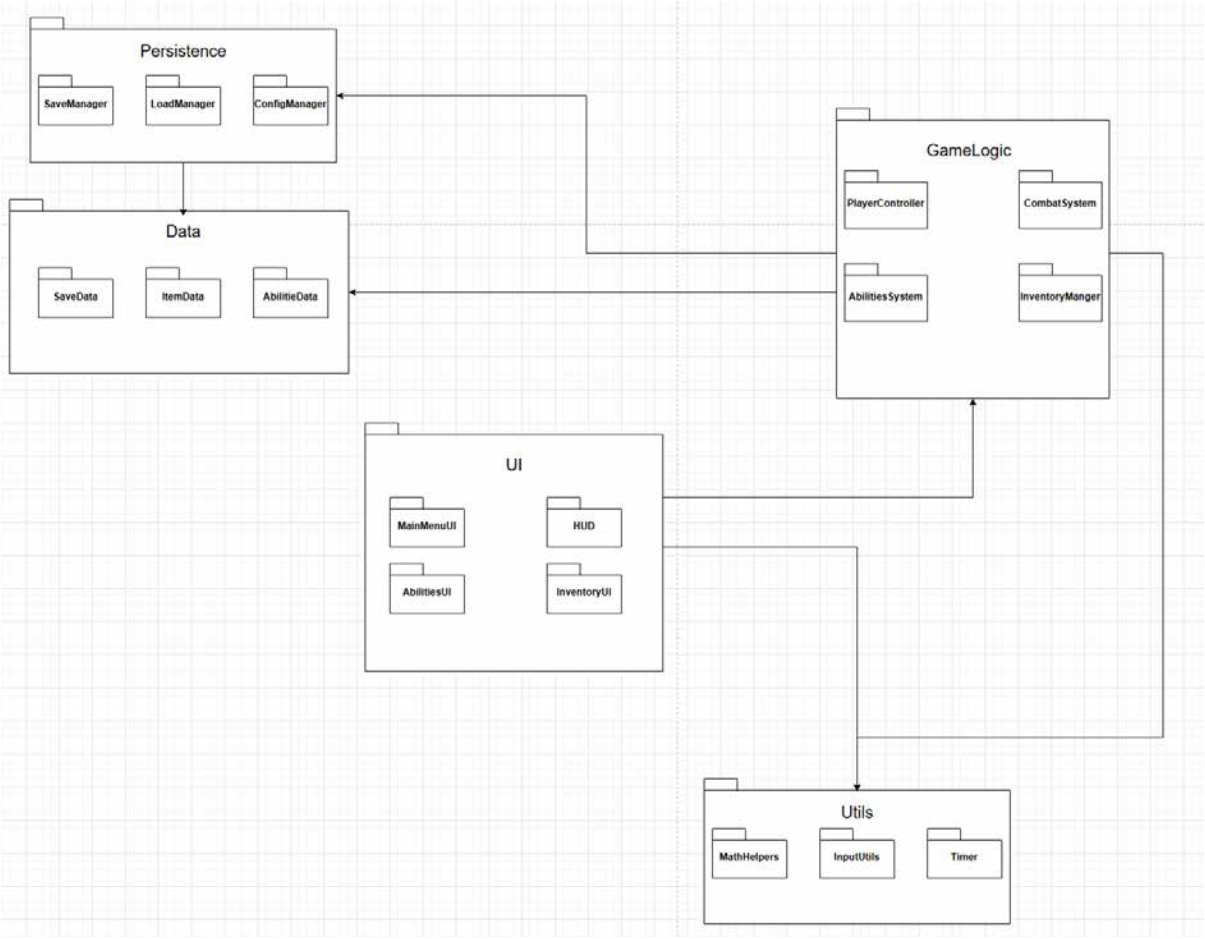

Рис. 19 Діаграма пакетів

Взаємодія між модулями

Взаємодія між описаними модулями відбувається через чітко визначені інтерфейси та події. Наприклад, UI отримує дані про інвентар від відповідного модуля через публічні методи або за допомогою системи делегатів Unreal Engine, що забезпечує реактивність інтерфейсу. Здібності через GAS повідомляють інші підсистеми про активацію, закінчення дії або зміну стану.

Також між модулями підтримується розділення відповідальності, що знижує зв’язність коду та підвищує його повторне використання.

Переваги такої організації

- Модульність – кожен пакет відповідає за конкретну сферу функціональності, що спрощує розробку і тестування.

- Масштабованість – нові функції можна додавати, міняючи лише відповідний модуль без впливу на всю систему.

- Зручність підтримки – логічна структура полегшує пошук і виправлення помилок.

- Використання сучасних технологій UE5 – застосування GAS, Blueprint та Data Table забезпечує ефективність і швидкість розробки.

Недоліки та обмеження

- Залежність від специфічних технологій Unreal Engine може ускладнити перенос проекту на інші платформи.

- Високий рівень абстракції GAS вимагає додаткового часу на вивчення і налаштування.

- Використання Blueprint в деяких випадках може впливати на продуктивність при великій кількості логіки.

3.2 Вибір мови програмування та середовища розробки

При розробці такого проекту одним із ключових рішень є вибір мови програмування та середовища розробки. Цей вибір впливає на продуктивність, гнучкість, зручність розробки, а також подальшу підтримку і масштабування проєкту.

У нашому випадку основним середовищем розробки обрана платформа Unreal Engine 5 (UE5), а мова програмування — C++ у поєднанні з Blueprints і

Gameplay Ability System (GAS). Нижче наведено обґрунтування такого вибору, а також порівняння з іншими популярними рішеннями.

3.2.1 Переваги Unreal Engine 5

Unreal Engine 5 — це потужний і сучасний ігровий рушій, який надає широкий набір інструментів для створення високоякісних 3D-ігор з розвиненою графікою та геймплейними механіками. Основні переваги UE5:

- Візуальна якість: підтримка Nanite (віртуальна геометрія), Lumen (реалістичне освітлення), що забезпечує фотореалістичні сцени.
- Інтегровані інструменти: готові редактори для анімації, UI, частинок, візуальних ефектів.
- Blueprints: візуальна мова програмування, яка дозволяє швидко створювати прототипи та логіку без написання коду.
- Gameplay Ability System (GAS): фреймворк для реалізації комплексних здібностей та ефектів, що ідеально підходить для RPG.
- Масштабованість: підтримка різних платформ (ПК, консолі, мобільні).
- Активна спільнота: велика кількість ресурсів, документації та готових рішень.

3.2.2 Використання C++ у UE5

C++ є основною мовою програмування для Unreal Engine і дає розробнику доступ до найнижчого рівня рушія. Основні переваги:

Таблиця 1

Переваги використання C++

Переваги C++	Пояснення
Висока продуктивність	Нативний код працює швидко, що важливо для ігор
Гнучкість	Можливість створювати складну логіку, оптимізувати код
Підтримка ООП	Полегшує підтримку великого проєкту
Глибока інтеграція з UE5	Дозволяє працювати з усіма можливостями рушія
Контроль над пам’яттю	Важливо для ресурсомістких ігор

3.2.3 Blueprints — візуальне програмування

Blueprints — це система візуального скриптингу в UE5, що дозволяє створювати логіку без коду. Вона:

- Прискорює розробку, особливо для прототипів.
- Дозволяє художникам і дизайнерам працювати з логікою.
- Добре підходить для простих і середніх за складністю завдань.

Водночас, Blueprints мають обмеження щодо продуктивності і підтримки складних алгоритмів, тому в поєднанні з C++ використовуються для UI, анімації, ігрових подій.

3.2.4 Gameplay Ability System (GAS)

GAS — це фреймворк UE5, який спеціально створений для ігор жанру RPG:

- Дає готові механізми для реалізації здібностей з різними станами.

- Підтримує мультиплеєр і реплікацію.
- Забезпечує масштабованість і легкість додавання нових ефектів.
- Взаємодіє з інвентарем, системою станів та іншими ігровими системами.
- Використання GAS дозволяє зменшити кількість «ручної» роботи при реалізації складних механік.

3.2.5 Порівняння з іншими мовами та рушіями

Таблиця 2

Порівняння UE5 з іншими ігровими рушіями

Критерій	Unreal Engine 5 (C++/Blueprints)	Unity (C#)	Godot (GDScript/C#)	CryEngine (C++)
Продуктивність	Висока	Середня	Нижча	Висока
Підтримка AAA-графіки	Дуже добра	Добра	Середня	Дуже добра
Складність освоєння	Висока	Середня	Низька	Висока
Візуальне програмування	Blueprints	Visual Scripting (Bolt)	Вбудоване в редактор	Обмежено
Спеціалізація для GAS	Потрібно додатково	Потрібно	Обмежено	

Критерій	Unreal Engine 5 (C++/Blueprints)	Unity (C#)	Godot (GDScript/C#)	CryEngine (C++)
RPG	реалізовувати		допрацьовувати	
Ліцензійна політика	Безкоштовно до $1 млн доходу	Безкошт овно/Pro	Відкрите джерело	Безкоштовно/Платно

Таблиця 2 (закінчення)

3.2.6 Висновки

Вибір Unreal Engine 5 з використанням C++, Blueprints та GAS є оптимальним для розробки проєкту з високими вимогами до якості графіки, складності ігрової механіки та масштабованості. Поєднання продуктивності C++ з гнучкістю Blueprints і потужністю GAS забезпечує ефективність розробки, зручність налагодження і подальшої підтримки.

3.3 Діаграма класів

У процесі розробки прикладного програмного забезпечення для гри жанру Action RPG в Unreal Engine 5 було реалізовано об'єктно-орієнтовану структуру на основі класів. Далі наведено опис основних модулів у вигляді діаграми класів, що відображає ключові компоненти та зв’язки між ними.

Основні класи:

1. **PlayerCharacter**
   - **Атрибути:**
     - Health: float

- Stamina: float
- Inventory: InventoryManager
- Abilities: AbilitySystemComponent
- SaveData: SaveManager

- **Методи:**
  - TakeDamage(amount: float): void
  - UseAbility(id: int): void
  - PickUpItem(item: Item): void
  - SaveGame(): void
  - LoadGame(): void

Опис:

Головний ігровий персонаж, який взаємодіє з іншими модулями: інвентарем, здібностями та системою збереження. Усі дії гравця ініціюються саме через цей клас.

2. InventoryManager

- **Атрибути:**
  - Items: List<Item>
- **Методи:**
  - AddItem(item: Item): void
  - RemoveItem(item: Item): void
  - GetItemByID(id: int): Item

Опис:

Клас, що відповідає за зберігання та управління предметами в інвентарі. Забезпечує додавання, вилучення та пошук об'єктів.

3. Item

- **Атрибути:**
  - ID: int
  - Name: string

- Description: string
- Type: EItemType
- Quantity: int

- **Методи:**
  - Use(): void

Опис:

Узагальнений клас предмета. Може бути використаний як для зброї, так і для витратних матеріалів.

4. **AbilitySystemComponent**

- **Атрибути:**
  - Abilities: List<Ability>
- **Методи:**
  - ActivateAbility(id: int): void
  - CooldownTick(): void

Опис:

Компонент, що реалізує систему здібностей за допомогою GAS. Відповідає за активацію, затримки, ефекти тощо.

5. **Ability**

- **Атрибути:**
  - ID: int
  - Name: string
  - Cooldown: float
  - EffectData: EffectSpec
- **Методи**:
  - Execute(): void

Опис:

Представляє окрему здібність гравця. Містить логіку використання та прикріплені візуальні/ігрові ефекти.

6. **SaveManager**

- **Атрибути:**
  - SaveData: SaveGameData
- **Методи:**
  - Save(): void
  - Load(): void

Опис:

Клас, що відповідає за збереження та завантаження ігрового прогресу. Працює з файлами локально на пристрої.

7. **SaveGameData**

- **Атрибути:**
  - PlayerStats: PlayerStats
  - InventoryState: List<Item>
  - AbilitiesState: List<Ability>
- **Методи:**
  - Serialize(): string
  - Deserialize(data: string): SaveGameData

Опис:

Структура, в якій серіалізуються всі важливі дані, необхідні для відновлення стану гри.

8. **Utils**

- MathHelpers

- InputUtils
- Timer

Опис:

Допоміжні класи для реалізації спільної логіки: математичні обчислення, обробка вводу, таймери, інші утиліти.

Взаємозв'язки між класами:

- **PlayerCharacter** має асоціації з **InventoryManager**, **AbilitySystemComponent** та **SaveManager**.
- **InventoryManager** управляє колекцією об’єктів **Item**.
- **AbilitySystemComponent** активує об’єкти **Ability**.
- **SaveManager** оперує об’єктом **SaveGameData**, який містить інформацію про стани Item та **Ability**.
- Класи **Utils** використовуються в різних частинах гри, але не мають жорсткої прив’язки.

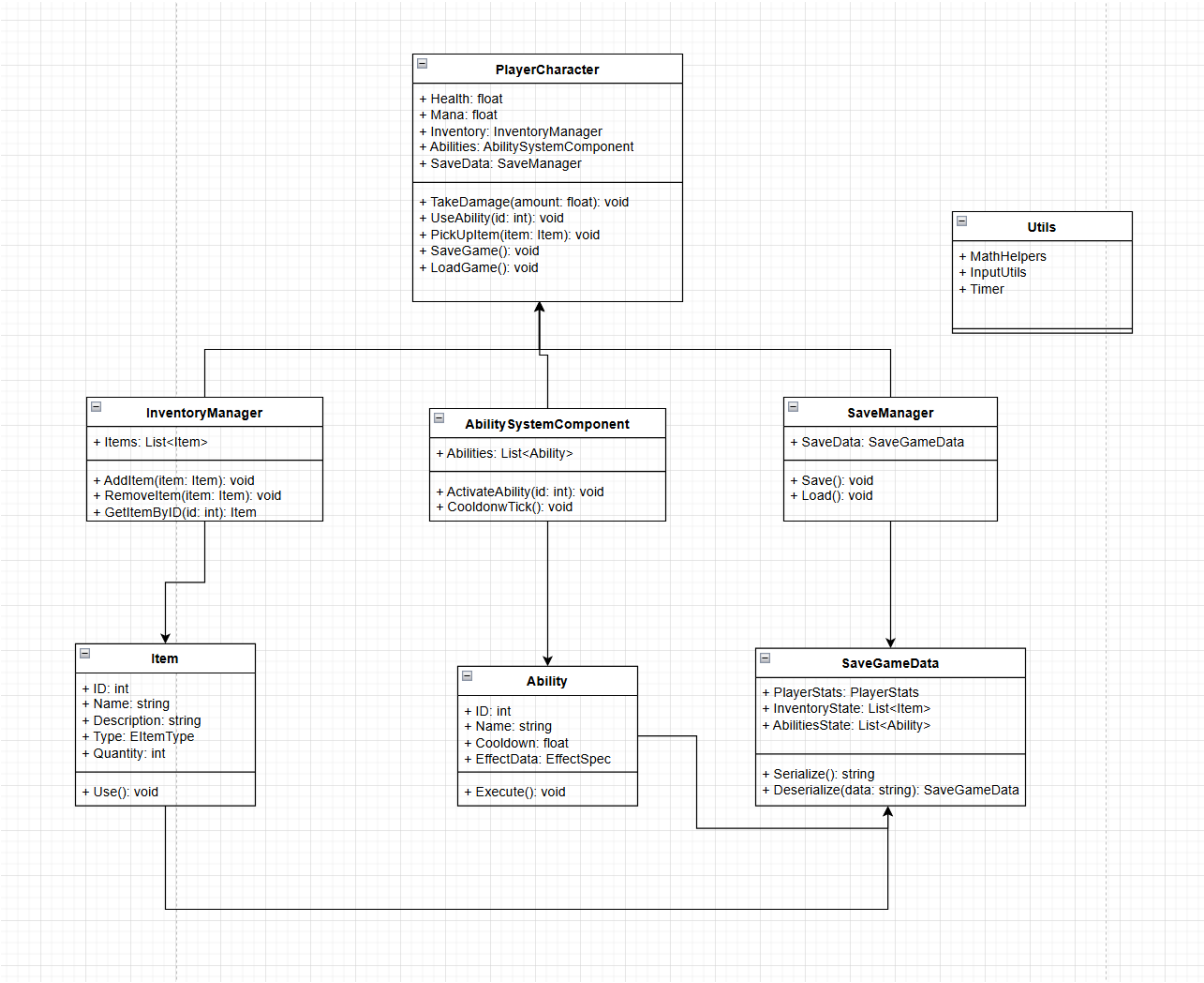

Рис. 20 Діаграма класів

3.4 Приклади роботи та результати

Представлення скриншотів виконання програми

Рис. 21 Персонаж на тестовому рівні для перевірки інвентарю

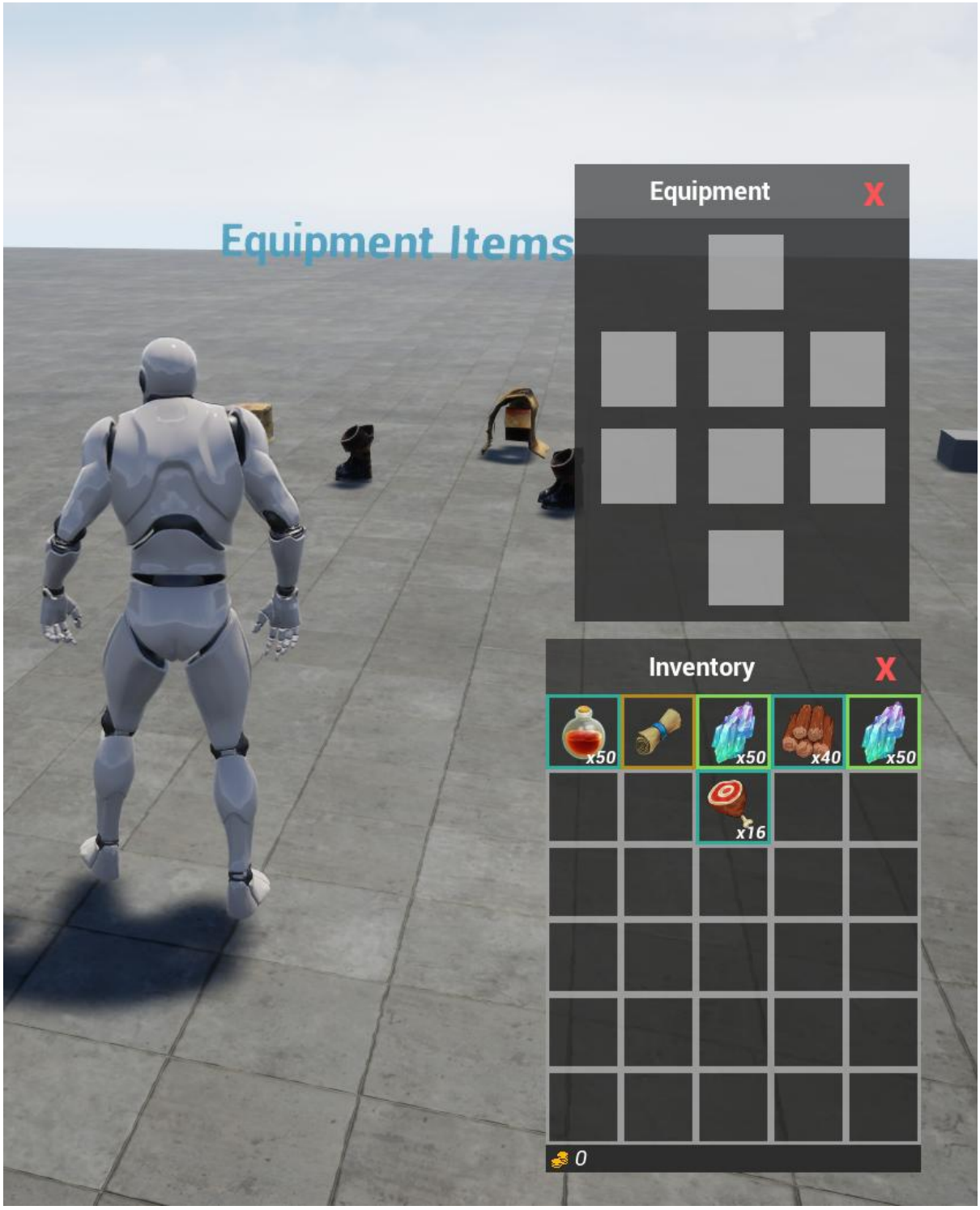

Рис. 22 інтерфейс вікна інвентарю та спорядження

Рис. 23 Демонстрація надітого спорядження та опису предмета

Рис. 24 Тестовий рівень для перевірки здібностей

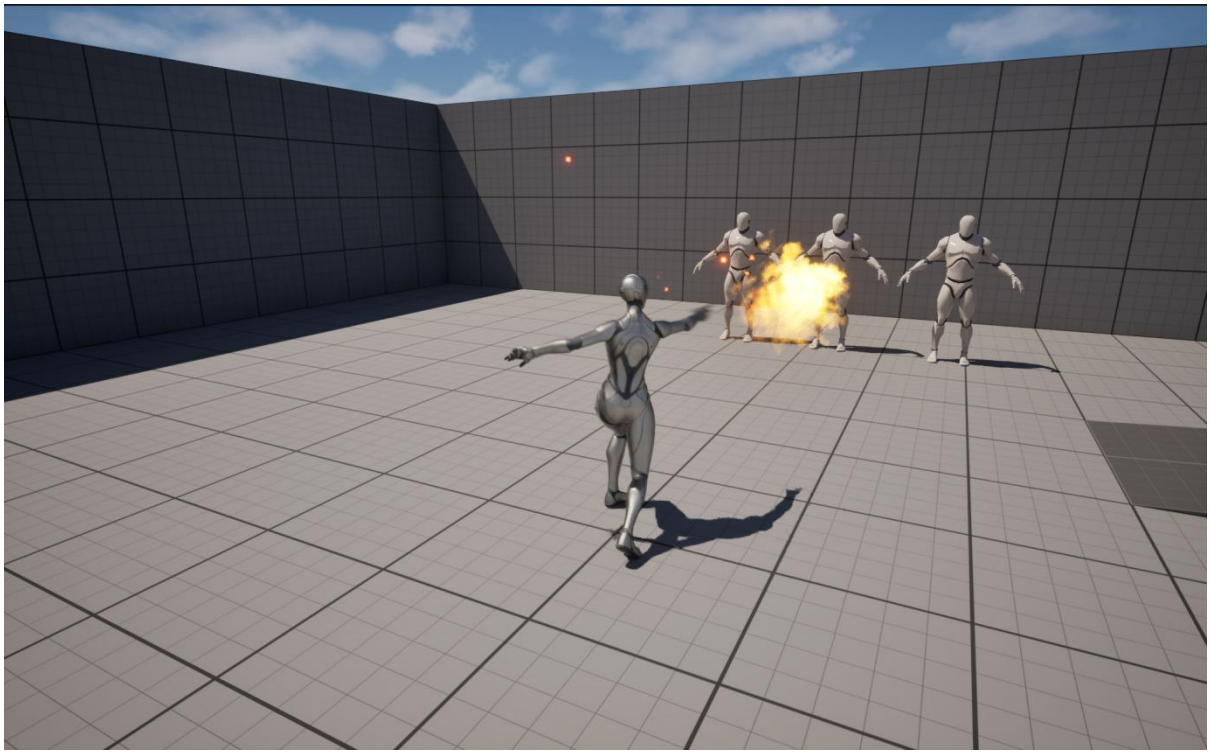

Рис. 25 Активація вміння “Fireball”

Рис. 26 Активація вміння “WaterStorm”

Висновки до розділу

Організація програмного забезпечення гри на Unreal Engine 5 є ефективною завдяки чіткому розподілу на модулі (UI, інвентар, здібності, збереження). Використання C++ разом із Blueprints та GAS дозволило поєднати продуктивність із гнучкістю розробки. Запропонована структура класів забезпечує зручну підтримку та масштабування проекту, що підтверджують робочі приклади з гри.

РОЗДІЛ 4

РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

1.1 Вимоги до апаратного і програмного забезпечення

Для успішного запуску та комфортної роботи з проєктом на Unreal Engine 5 необхідно врахувати наступні вимоги до апаратного та програмного забезпечення. Ці вимоги можуть варіюватися залежно від складності вашої гри, її графіки та функціоналу.

Компонент	Мінімальні	Рекомендовані
Процесор	Чотириядерний процесор з тактовою частотою 2.5 ГГц або вище (Intel Core i5 7-го покоління або AMD Ryzen 5 1-го покоління)	Восьмиядерний процесор з тактовою частотою 3.0 ГГц (Intel Core i7 10-го покоління або AMD Ryzen 7 3-го покоління)
Оперативна пам’ять	16 ГБ DDR4	32 ГБ DDR4 або більше
Графічний процесор	NVIDIA GeForce RTX 2060, AMD Radeon RX 6600 XT або аналогічна з 8 ГБ VRAM та підтримкою DirectX 12	NVIDIA GeForce RTX 3080, AMD Radeon RX 6800 XT або новіші моделі з 12 ГБ VRAM або більше (для використання Nanite та Lumen на основі трасування променів)
Накопичувач	SATA SSD 250 GB	NVMe SSD 1 TB
Операційна система	Windows 10 64-bit (версія 1909 або новіша)	Windows 11 64-bit

Проект на Unreal Engine 5 підтримує запуск на різних конфігураціях апаратного забезпечення, що дозволяє адаптувати гру під різні категорії користувачів.

Мінімальні вимоги забезпечують можливість запуску гри з базовою графікою і функціоналом. На комп’ютерах з мінімальними характеристиками гра працюватиме з обмеженим рівнем деталізації, зниженими параметрами освітлення, текстур і ефектів. Це дозволяє охопити широку аудиторію користувачів, у яких немає потужних ігрових систем.

Рекомендовані та максимальні вимоги дозволяють повністю розкрити потенціал рушія Unreal Engine 5, включаючи підтримку трасування променів, високої роздільної здатності текстур, складних візуальних ефектів і плавності роботи. На таких системах користувачі отримають максимальну якість графіки, комфортне управління і стабільний високий FPS.

4.2 Вимоги до запуску ПЗ

4.2.1 Апаратне забезпечення

4.2 Інструкція впровадження та експлуатації системи

Перед тим як передати гру кінцевим користувачам або демонструвати її як завершений програмний продукт, необхідно провести правильне пакування проєкту. Це забезпечує, що всі файли, ресурси, логіка гри та налаштування будуть зібрані у виконуваний інсталяційний пакет, готовий до запуску на платформі Windows.

Налаштування стартової мапи

Однією з обов’язкових умов перед пакуванням є встановлення мапи за замовчуванням, яка буде завантажена при старті гри. Якщо цього не зробити, під час запуску користувач побачить чорний екран, оскільки рушій не знатиме, з якої сцени починати.

Для цього в редакторі Unreal Engine переходимо:
Edit > Project Settings > Maps & Modes
і у полях **Editor Startup Map** та **Game Default Map** вказуємо основну карту гри (ThirdPersonExampleMap), яка ініціалізує всі необхідні системи: інвентар, здібності, UI тощо.

П а к у в а н н я г р и

Далі здійснюється безпосереднє пакування гри. У головному меню Unreal Editor обираємо:
Platforms > Windows > Package Project

Після цього відкриється вікно вибору директорії, куди буде збережено зібраний проєкт. Після підтвердження починається процес пакування (build), який триває кілька хвилин залежно від обсягу контенту гри.

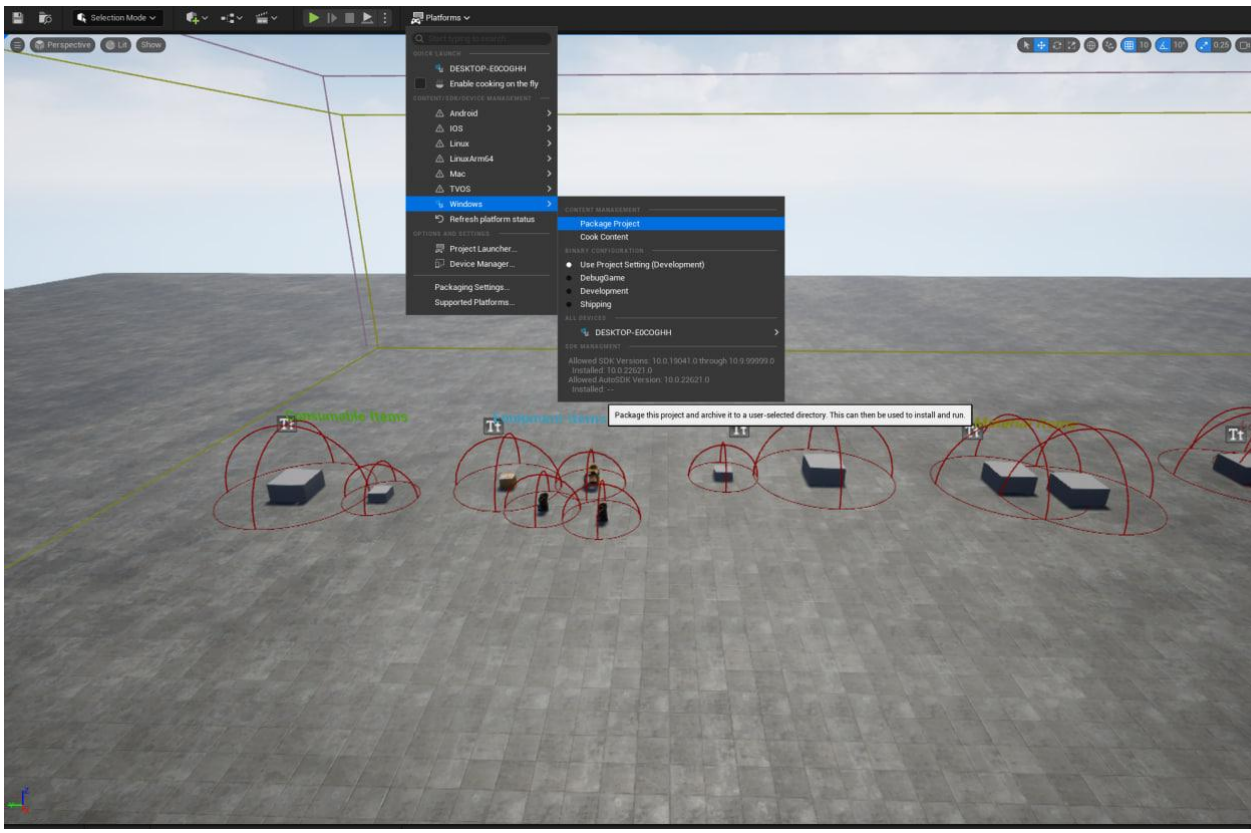

Рис. 26 Пакування проекту

4.2.2 Інсталяційний пакет

Після пакування гри, створеної в Unreal Engine 5, формується інсталяційний пакет, що містить усі необхідні файли для запуску проєкту на цільовій платформі.

Інсталяційний пакет гри включає такі основні компоненти:

Виконуваний файл гри (MyRPGGame.exe) – головний файл, який запускає гру. Саме з нього починається завантаження всіх підсистем: управління інвентарем, здібностями, UI тощо.

Директорія Engine – містить необхідні компоненти рушія Unreal Engine. Включає низку системних бібліотек, модулів та залежностей, без яких запуск гри неможливий на машині, де Unreal Engine не встановлено.

Директорія Content – одна з ключових папок. Містить усі ресурси гри, створені під час розробки:

- моделі предметів інвентаря та персонажів;
- ефекти активних здібностей;
- анімації, текстури, ігрові рівні;
- інтерфейс користувача (UI).

Директорія Config – включає конфігураційні .ini файли, які відповідають за налаштування графіки, клавіш керування, звуку, режиму вікна, локалізації тощо.

Проміжна директорія Saved (може з’явитися після першого запуску) – зберігає локальні дані користувача: збереження, лог-файли, кеши UI та налаштування користувача.

ВИСНОВОК

У ході виконання дипломної роботи була розроблена комп’ютерна система управління віртуальним інвентарем та здібностями ігрового персонажа на основі сучасного ігрового рушія Unreal Engine 5. Метою проєкту було створення ефективної, гнучкої і зручної у використанні системи, що забезпечує основні функції роботи з предметами та активними здібностями персонажа.

На початковому етапі роботи було проведено аналіз предметної області, сформовано постановку завдання та визначено функціональні й нефункціональні вимоги до системи. Для кращого розуміння майбутньої системи були розроблені діаграми прецедентів, які наочно відображають основні сценарії взаємодії користувача з інвентарем і здібностями персонажа. Цей етап дозволив чітко окреслити функціонал, необхідний для подальшої розробки.

Наступним кроком стала розробка логічної моделі даних. За допомогою діаграми «Сутність-Зв’язок» (ERD) було структуровано інформацію про предмети інвентарю, характеристики здібностей та їхні взаємозв’язки. Це допомогло організувати базу даних і закласти фундамент для подальшої реалізації системи. Крім того, були створені діаграми активності та послідовності, які деталізували послідовність дій гравця і взаємодію між різними компонентами системи.

Вибір Unreal Engine 5 як платформи для реалізації системи був зумовлений його широкими можливостями, зокрема підтримкою гібридного програмування на C++ та Blueprints, що дозволяє поєднувати низькорівневий код з візуальним сценарієм. Це дало змогу ефективно реалізувати ключові функції — додавання і видалення предметів, їх використання, а також управління активацією і перезарядкою здібностей. Особлива увага приділялася інтеграції системи з користувацьким інтерфейсом, щоб забезпечити інтуїтивно зрозумілу взаємодію для гравця.

Програмне забезпечення було спроєктоване з урахуванням модульності та компонентного підходу, що значно спрощує подальше розширення та підтримку коду. Основні алгоритми, такі як облік предметів у інвентарі, розрахунок ефектів від використання здібностей і управління станами персонажа, були детально опрацьовані і успішно реалізовані. Завдяки цьому система демонструє динамічну поведінку і адаптивність до різних ситуацій у грі.

У заключному етапі роботи проведено комплексне тестування системи, яке підтвердило її стабільність і відповідність вимогам. Результати свідчать про

те, що розроблена система готова до використання і може стати основою для подальших досліджень і вдосконалень.

У перспективі планується розширити функціонал системи, зокрема додати можливість розвитку здібностей персонажа через дерево талантів, впровадити мультиплеєрний режим, а також оптимізувати продуктивність для різних апаратних платформ. Крім того, можливе інтегрування з більш складними ігровими механіками, що дозволить створити більш глибокий і насичений ігровий досвід.

Таким чином, виконана робота демонструє комплексний підхід до розробки ігрових систем управління інвентарем та здібностями, що є важливою частиною сучасних комп’ютерних ігор. Отримані результати можуть бути використані як база для подальшого розвитку ігрового проекту та створення нових ігрових механік.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. McShaffry M., Graham S. Game Coding Complete. – 4th ed. – CRC Press, 2012. – 960 p.

2. Gregory J. Game Engine Architecture. – 3rd ed. – CRC Press, 2018. – 1240 p.

3. Шевченко В.П. Основи розробки комп’ютерних ігор. – К.: Видавництво Ліра-К, 2021. – 276 с.

4. Unreal Engine 5 Documentation. – https://docs.unrealengine.com/

5. Brackeys. Unity vs Unreal: Game Engine Comparison. – https://brackeys.com/unity-vs-unreal

6. Krivoruchko D. Blueprint Visual Scripting for Unreal Engine. – Unreal Engine Blog, 2021.

7. Олійник О.В., Степаненко О.В. Проєктування програмного забезпечення. – Х.: ХНУРЕ, 2020. – 152 с.

8. Офіційна документація Gameplay Ability System (GAS). – https://docs.unrealengine.com/GameplayAbilitySystem/

9. Petricoin M. Unreal Engine C++ Developer: Learn C++ and Make Video Games. – Udemy Course, 2023.

10. Чернов С.І. Методи та засоби збереження ігрового прогресу. – Запоріжжя: ЗНТУ, 2021. – 112 с.

11. Anderson S. UX for Games. – CRC Press, 2021. – 340 p.

12. Паранюк П.М. Архітектура програмного забезпечення: курс лекцій. – Львів: ЛНУ, 2020. – 240 с.

13. Felicia S. Blueprints Visual Scripting for Unreal Engine 5: Beginner's Guide. – Packt Publishing, 2022. – 352 p.

14. Karlin R. The Elements of User Experience: User-Centered Design for the Web and Beyond. – 2nd ed. – New Riders, 2014. – 256 p.

15. Rama T. Learning C++ by Creating Games with Unreal Engine 4. – Packt Publishing, 2015. – 342 p.

16. Unreal Engine Forums. – https://forums.unrealengine.com/

17. Томіліна Н.І., Гриневич І.В. Проектування інтерфейсів користувача. – Львів: Видавництво ЛНУ, 2020. – 148 с.

18. Unreal Engine Save Game System. – https://docs.unrealengine.com/en-US/API/Runtime/Engine/GameFramework/USaveGame/

19. Doran J., Patterson N. Mastering Unreal Engine: A Beginner's Guide. – Packt Publishing, 2023. – 496 p.

20. Крячко М.О. Основи проєктування інформаційних систем у середовищі ігрових рушіїв. – Одеса: ОНПУ, 2021. – 134 с.

Додаток А

Діаграма розгортання

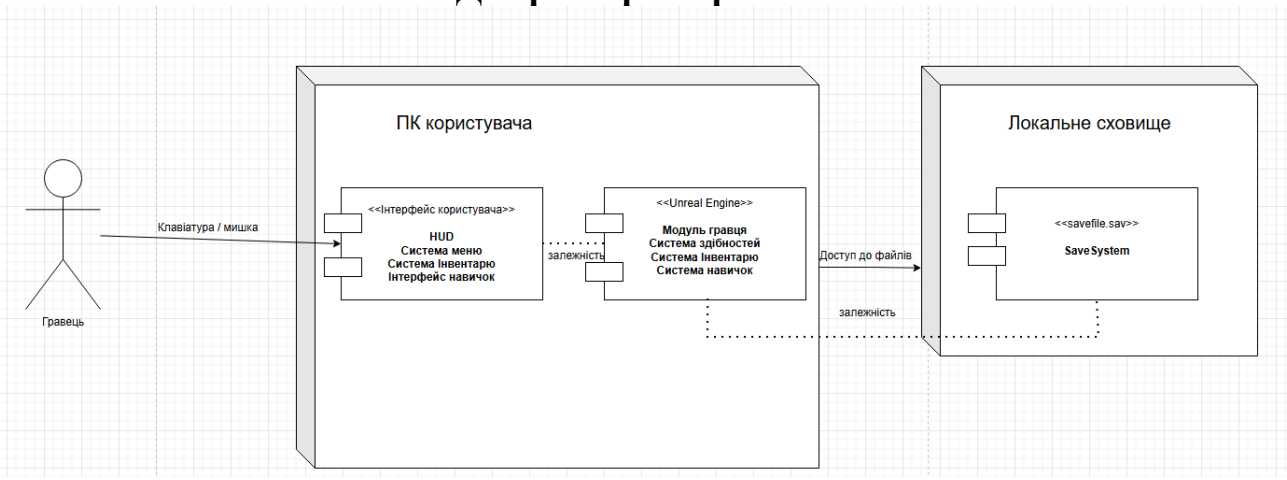

Рис. 27 Діаграма розгортання

Додаток Б

Блюпрінт «BPC_PlayerInventory»

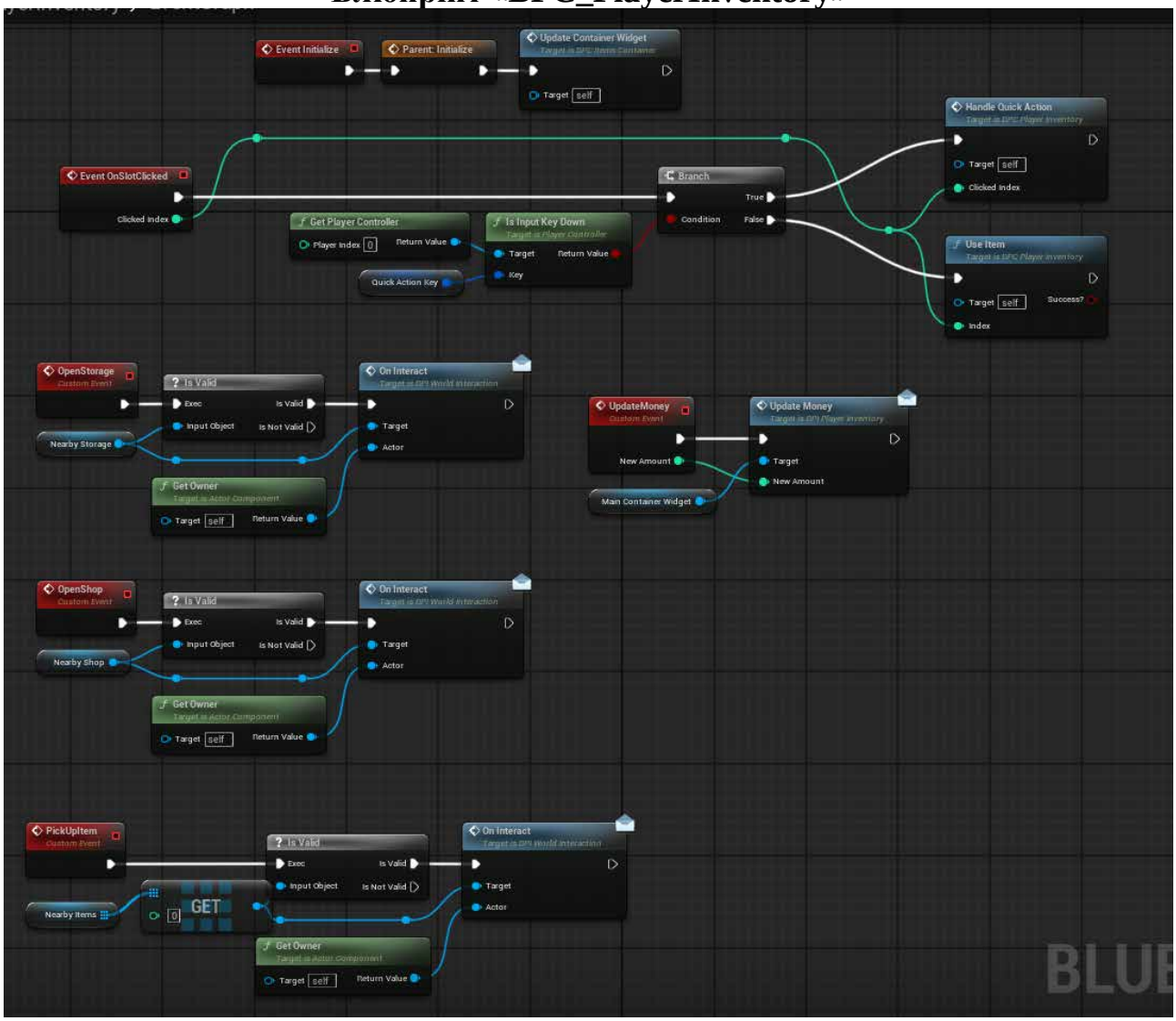

Рис. 28 Компонент інвентарю гравця

Додаток В

Блюпрінт «BPC_ItemsContainer»

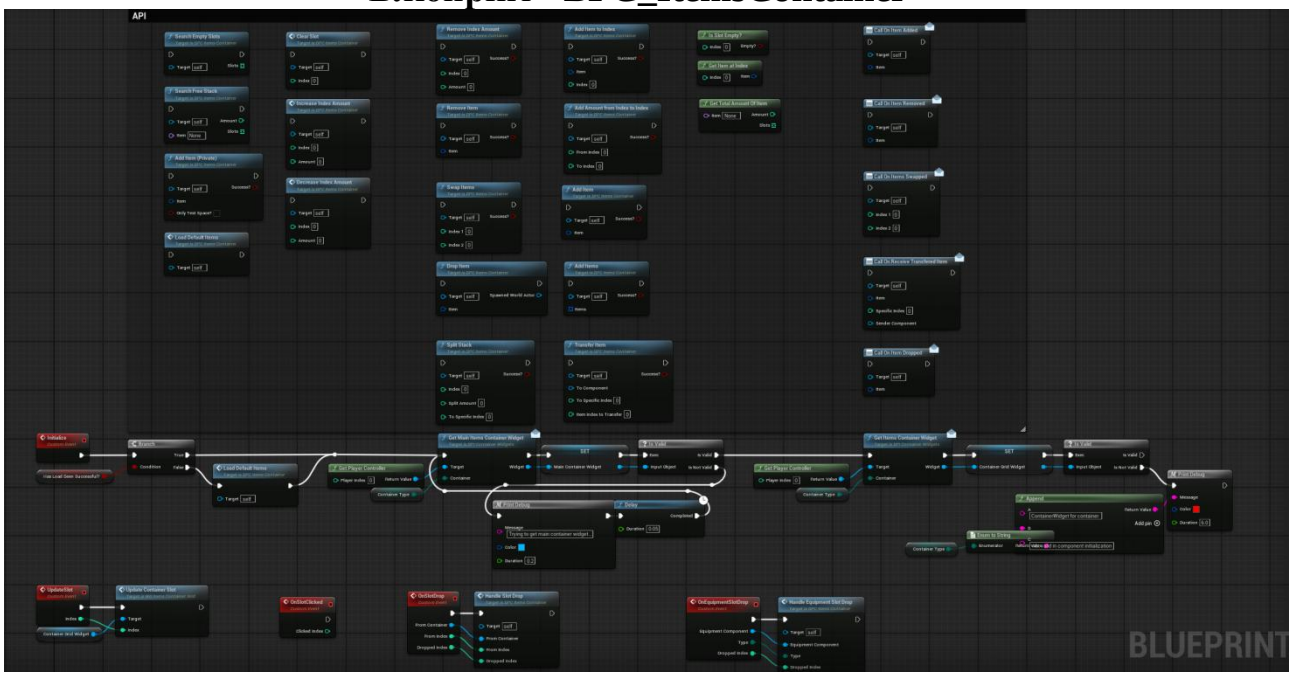

Рис. 29 Компонент контейнера предметів

Додаток Г

Блюпрінт «BPC_Equipment»

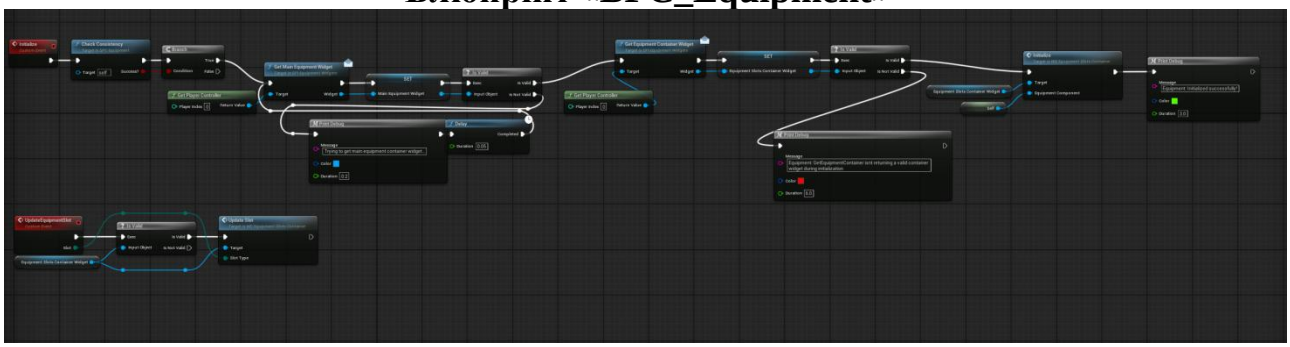

Рис. 30 Компонент екіпірування предметів для персонажа

Додаток Ґ

Блюпрінт «GA_WaterStorm»

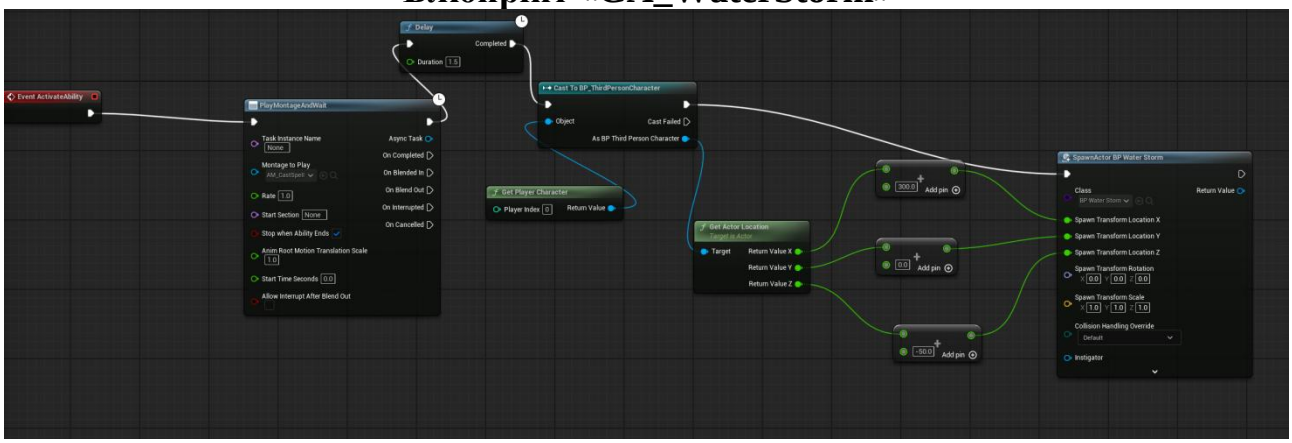

Рис. 31 Логіка активації здібності «WaterStorm»