Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Ігровий програмний застосунок у жанрі Dungeon Crawler
(тема)

Виконав:
здобувач 4 року навчання
групи ПЗПІ-21-10

Вадим БЕХОВ
(Власне ім'я, ПРІЗВИЩЕ)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник ст. викл. кафедри ПІ Дмитро МАТВЄЄВ
(посада, Власне ім'я, ПРІЗВИЩЕ)

Допускається до захисту
Зав. кафедри _____ Кирило СМЕЛЯКОВ
(підпис) (Власне ім'я, ПРІЗВИЩЕ)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Бехову Вадиму Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Ігровий програмний застосунок у жанрі Dungeon Crawler.
Затверджена наказом по університету від 19.05.2025р. № 397 Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 13.06.2025
3. Вихідні дані до роботи *Розробити ігровий програмний застосунок у жанрі Dungeon Crawler. Реалізацію системи виконати на ігровому двигуні Unreal Engine 5 з використанням мови програмування C++ та Blueprint*
4. Перелік питань, що потрібно опрацювати в роботі
Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	19.04.2025	*виконано*
2	Створення специфікації ПЗ	26.04.2025	*виконано*
3	Проектування ПЗ	03.05.2025	*виконано*
4	Розробка ПЗ	09.05.2025	*виконано*
5	Тестування ПЗ	25.05.2025	*виконано*
6	Оформлення пояснювальної записки	30.05.2025	*виконано*
7	Підготовка презентації та доповіді	31.05.2025	*виконано*
8	Попередній захист	08.06.2025	*виконано*
9	Нормоконтроль, рецензування	08.06.2025	*виконано*
10	Здача роботи у електронний архів	10.06.2025	*виконано*
11	Допуск до захисту у зав. кафедри	12.06.2025	*виконано*

Дата видачі завдання 7 квітня 2025р.

Здобувач (ка) _____ Бехов В. С.
(підпис)

Керівник роботи _____ ст. викл. кафедри ПІ Матвєєв Д. І.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра: 66 с., 20 рис., 9 джерел, 6 таблиць.

ДЕКОРАЦІЇ, ГЕНЕРАЦІЯ РІВНІВ, ІГРОВИЙ РУШІЙ, ПРОЦЕДУРНА ГЕНЕРАЦІЯ, BLUEPRINT, C++, DUNGEON CRAWLER, UNREAL ENGINE 5

У цій кваліфікаційній роботі розглянуто процес створення системи процедурної генерації рівнів і декорацій для гри в жанрі dungeon crawler. Метою дослідження є розробка інструментарію, що дозволяє генерувати унікальні підземелля з інтерактивними елементами для забезпечення варіативного ігрового досвіду.

Проєкт реалізовано з використанням рушія Unreal Engine 5 та мов програмування C++ і Blueprint. Система генерації підтримує створення рівнів зі змінною структурою кімнат, розміщенням об'єктів і тригерів, а також забезпечує інтеграцію з основними геймплейними механіками (рух, бій, взаємодія з оточенням).

Проведено аналіз існуючих ігор і підходів до процедурної генерації. На основі цього спроєктовано архітектуру системи, розроблено UML-діаграми класів, прецедентів та активностей. Реалізовано модульну структуру, яка дозволяє легко масштабувати та налаштовувати систему генерації для нових сценаріїв гри.

Результатом роботи є ефективна, адаптивна та розширювана система, яка забезпечує динамічне створення локацій і підтримує високу продуктивність. Розробка може бути використана як основа для майбутніх ігрових проєктів, що передбачають змінність та унікальність ігрового простору.

ABSTRACT

DECORATIONS, LEVEL GENERATION, GAME ENGINE, PROCEDURAL GENERATION, BLUEPRINT, C++, DUNGEON CRAWLER, UNREAL ENGINE 5

This qualification work examines the process of creating a procedural generation system for levels and scenery for a dungeon crawler game. The goal of the research is to develop a toolkit that allows generating unique dungeons with interactive elements to provide a variable gaming experience.

The project was implemented using the Unreal Engine 5 engine and the C++ and Blueprint programming languages. The generation system supports the creation of levels with a variable room structure, object placement and triggers, and also provides integration with the main gameplay mechanics (movement, combat, interaction with the environment).

An analysis of existing games and approaches to procedural generation was conducted. Based on this, the system architecture was designed, UML diagrams of classes, precedents and activities were developed. A modular structure was implemented that allows easy scaling and customization of the generation system for new game scenarios.

The result of the work is an effective, adaptive and extensible system that provides dynamic location creation and supports high performance. The development can be used as a basis for future game projects that involve variability and uniqueness of the game space.

ЗМІСТ

ВСТУП

У сучасну епоху цифрових технологій, коли інтерактивні розваги стають невіддільною частиною повсякденного життя, комп’ютерні ігри продовжують відігравати значну роль у формуванні культурного та технологічного ландшафту. Одним із жанрів, який привертає увагу гравців завдяки атмосфері, глибокому зануренню та елементам дослідження, є Dungeon Crawler. Цей жанр пропонує гравцям спуститися в загадкові підземелля, де на них чекають численні небезпеки, головоломки, приховані скарби та противники, які вимагають уважності та стратегічного мислення.

Ігри жанру Dungeon Crawler часто поєднують в собі елементи бойової системи, дослідження локацій, управління ресурсами та розвиток персонажа. Вони створюють унікальне середовище, де кожен крок може мати вирішальне значення, а ігровий процес базується на поступовому розкритті структури рівня. Саме цей жанр дозволяє гравцеві відчути справжній дух пригод, який підтримується як візуальною складовою, так і геймплейною логікою.

Завдяки розвитку сучасних інструментів розробки та графічних рушіїв, створення подібних застосунків стало доступнішим як для великих студій, так і для незалежних розробників. Використання новітніх підходів до побудови середовища, реалізації освітлення, а також створення систем взаємодії з оточенням дозволяє зробити ігровий процес ще глибшим та емоційно насиченим.

Жанр Dungeon Crawler має багатий історичний бекграунд. Його витоки сягають перших рольових ігор, а еволюція охоплює десятки років розвитку ігрової індустрії. Від текстових підземель до повноцінних 3D-світів — цей жанр завжди залишався вірним своїй головній ідеї: дати гравцеві відчуття дослідження та боротьби за виживання в невідомому середовищі.

Метою даної дипломної роботи є розробка ігрового застосунку в жанрі Dungeon Crawler з урахуванням актуальних вимог до сучасних ігор: гнучкої архітектури, візуальної привабливості, ефективної роботи систем освітлення та

механік взаємодії з оточенням. Проєкт охоплює створення ігрового середовища, налаштування ігрової логіки, впровадження базових механік та загальної структури застосунку.

У процесі роботи буде проведено аналіз жанрових особливостей, обрано відповідні технології та підходи, реалізовано основні функціональні компоненти гри, а також оцінено їхню ефективність та перспективи подальшого розвитку. Особлива увага приділяється якості візуального представлення, ергономіці управління та гнучкості архітектурного рішення, що є ключовими критеріями успішного ігрового проєкту.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Жанр Dungeon Crawler є різновидом рольових або пригодницьких ігор, в яких гравець досліджує лабіринтоподібні підземелля, стикається з ворогами, розгадує головоломки та поступово відкриває нові частини ігрового світу [1]. Для цього жанру характерними є обмежений простір, покрокове або реальне пересування, інтенсивна взаємодія з оточенням, а також акцент на виживання та тактичний підхід до бою.

Ключовим елементом, який формує досвід гравця у подібних іграх, є дизайн рівнів. Успішні представники жанру демонструють, як ретельно продумана структура підземель може викликати відчуття напруги, цікавості та досягнення. Один із яскравих прикладів — Legend of Grimrock (див. рис. 1.1), який поєднує покроковий рух по сітці, систему бою в реальному часі та безліч інтерактивних головоломок [2].

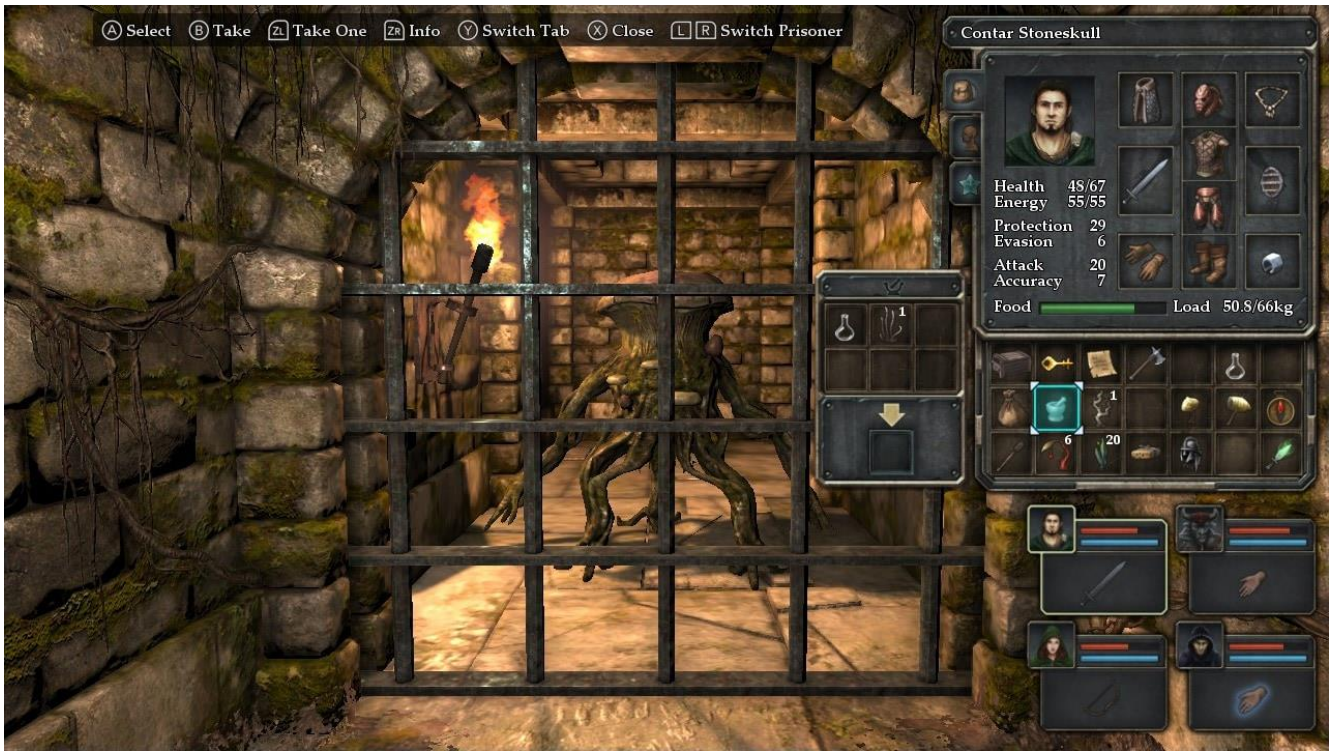

Рисунок 1.1 – Геймплей Legend of Grimrock (зроблено власноруч)

Його продовження, Legend of Grimrock 2 (див. рис. 1.2), розширює цей підхід, додаючи відкриті території, більш складні рівні та варіативність підходів до

проходження. Ці ігри демонструють важливість збереження балансу між боєм, дослідженням і головоломками, а також ролі візуального стилю та освітлення в створенні атмосфери.

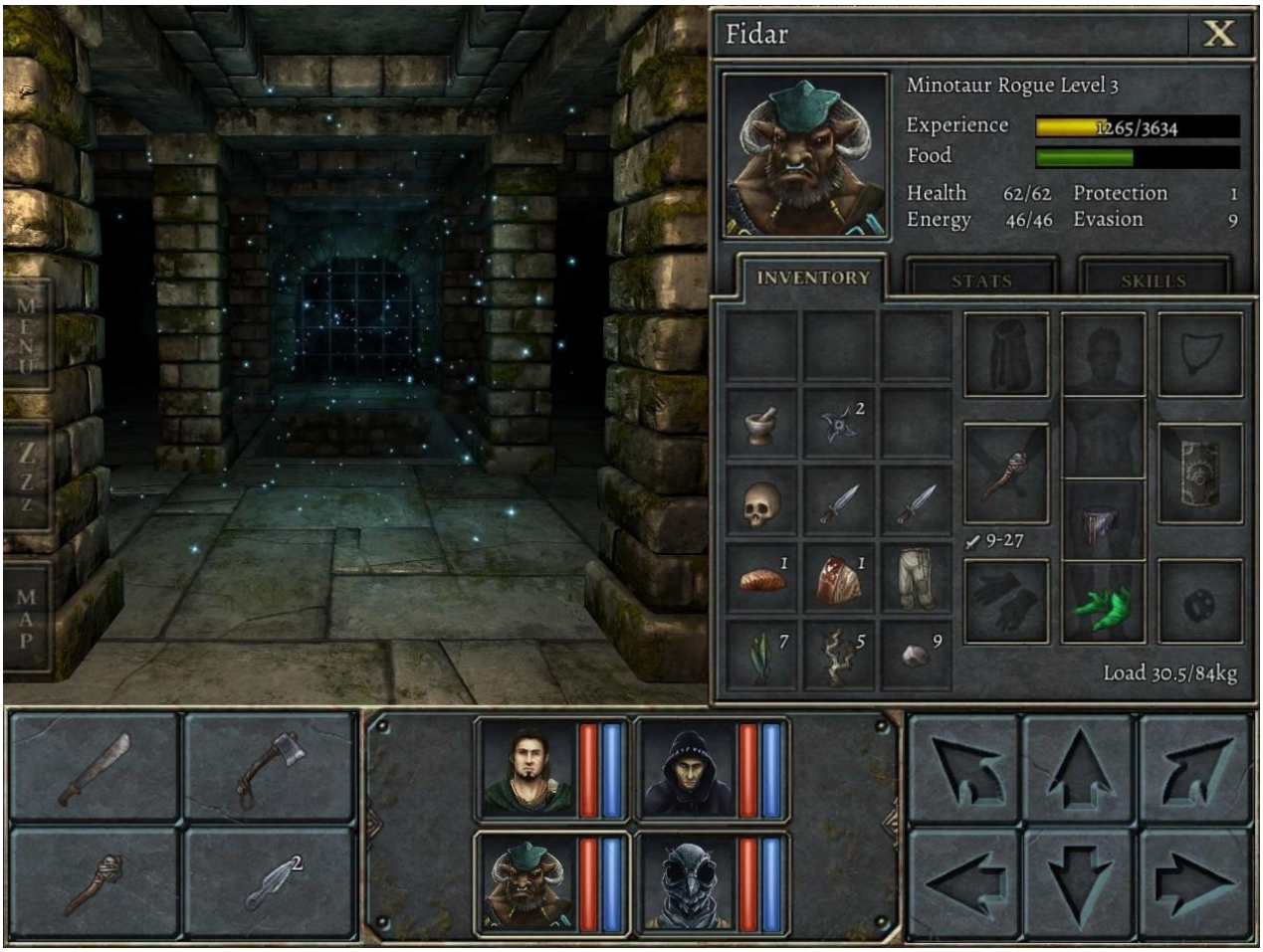

Рисунок 1.2 – Геймплей Legend of Grimrock 2 (зроблено власноруч)

Інший приклад — Path of Exile (див. рис. 1.3), хоч і належить до жанру Action RPG, все ж містить елементи dungeon crawler, зокрема випадково згенеровані рівні, ізольовані зони, значну кількість ворогів, пасток та обмежених ресурсів [3]. Це показує, як навіть у сучасних масштабних проектах зберігається структура «підземного дослідження» з акцентом на небезпеки, які чатують на гравця у кожному кутку. Така структура підтримує інтерес і створює відчуття постійної загрози.

Рисунок 1.3 – Геймплей Path of Exile (зроблено власноруч)

Minecraft Dungeons (див. рис. 1.4) — ще один сучасний приклад гри, яка адаптує dungeon crawler-механіки до доступнішого формату. Вона використовує процедурну генерацію рівнів, що дозволяє створювати унікальні конфігурації підземель для кожної сесії гри [4]. Це, у свою чергу, підвищує вартість повторного проходження та робить геймплей більш динамічним, незважаючи на простіші механіки бою та відсутність складних головоломок.

Рисунок 1.4 – Геймплей Minecraft Dungeons (зроблено власноруч)

Усі ці приклади демонструють, що незалежно від технічної реалізації, фундаментом dungeon crawler-застосунку є:

- структуроване та логічно побудоване середовище;
- обмежена видимість (туман війни, темні приміщення, коридори);
- елементи стратегічного планування (збереження ресурсів, підготовка до бою);
- відчуття ізоляції, небезпеки та нагороди за дослідження;
- гнучка система прогресу персонажа (розвиток навичок, здобуття спорядження).

У деяких сучасних реалізаціях використовуються системи процедурної генерації для створення динамічних підземель, як-от у Minecraft Dungeons або Path

of Exile [3][4]. Це дозволяє значно скоротити час розробки рівнів та збільшити варіативність ігрового процесу. Однак у класичних представниках жанру, таких як Legend of Grimrock, акцент робиться на вручну створеному контенті [2], що дає змогу точніше контролювати ритм гри та якість дизайну.

Завдяки поєднанню цих елементів, ігри жанру Dungeon Crawler створюють глибокий, напружений і захоплюючий досвід, який і буде реалізовано в рамках даного дипломного проєкту.

1.2 Виявлення проблем та актуалізація рішень

Після аналізу предметної галузі стає очевидним, що жанр dungeon crawler вимагає від розробника створення цікавого, глибокого ігрового світу, який стимулює дослідження та винагороджує гравця за уважність, стратегічне мислення та взаємодію з навколишнім середовищем [1]. Успішні проєкти цього жанру зазвичай поєднують атмосферу таємничості, візуальну глибину, варіативність ворогів і підземель, а також систему поступового розвитку персонажа [2][3][4].

Проте створення такого досвіду пов’язане з рядом проблем. Однією з ключових задач є створення глибокої та нелінійної структури рівнів, яка залишатиметься цікавою навіть при повторному проходженні. Фіксована структура карт може швидко стати передбачуваною, тому виникає потреба у використанні процедурної або модульної генерації підземель [5][6][7]. Проблема полягає в тому, щоб забезпечити баланс між випадковістю і дизайнерським контролем: надмірна випадковість може зруйнувати ігровий ритм, а занадто суворий контроль — позбавити гру реіграбельності.

Ще однією проблемою є ефективне розміщення декорацій та об’єктів взаємодії. Для жанру dungeon crawler важливо, щоб ігрове середовище не тільки виглядало атмосферно, але й впливало на геймплей. Наприклад, вузькі проходи можуть змінювати поведінку ворогів або вимагати від гравця іншої тактики [2][4]. Також слід уникати надмірного повторення елементів, що знижує ефект занурення.

Виклик становить і збереження оптимальної продуктивності при високій щільності об’єктів на сцені. Dungeon crawler-и часто передбачають темне

освітлення, об’ємні тіні, частинки (дим, пил), що навантажує систему. Щоб уникнути проблем із продуктивністю, необхідно реалізувати оптимізаційні техніки, як-от LOD-системи, динамічне завантаження частин рівня або використання інстансингу для повторюваних об’єктів [8].

Окрему увагу слід приділити розміщенню ворогів, пасток, загадок і скарбів. Вони мають бути логічно вплетені в середовище та мати естетичне обґрунтування. Наприклад, розміщення скарбу за пасткою має виглядати природно, а не випадково. Для цього доцільно створити систему тегів або зонування, що дозволяє генератору враховувати контекст [6][7].

Ще одна актуальна тенденція — адаптивність рівнів. Це означає, що структура підземель може змінюватись відповідно до дій гравця (наприклад, відкриватися нові шляхи, змінюватись складність). Впровадження такої функціональності потребує продуманої логіки та системних тригерів, які не порушують ігрову цілісність [7].

Таким чином, розробка dungeon crawler-гри вимагає вирішення складного комплексу завдань: створення варіативного, атмосферного простору, що одночасно залишається структурованим, логічним, геймплейно насиченим та технічно ефективним. Раціональне поєднання процедурної генерації [5][6][7], ручного дизайну, оптимізаційних підходів [8] і сучасних методів побудови середовища є запорукою створення якісного і конкурентного продукту.

1.3 Постановка задачі

Проєкт передбачає розробку системи процедурної генерації ландшафту та декорацій для гри у жанрі dungeon crawler, де гравець досліджує підземелля, стикається з пастками, монстрами та шукає ресурси [1][2]. Основною задачею є створення алгоритму, здатного автоматично будувати рівні з функціонально завершеною структурою: ізольовані або пов’язані між собою кімнати, коридори, розвилки та секретні ділянки, що зберігають баланс між варіативністю та навігаційною зручністю [5][6][7].

Особливу увагу необхідно приділити логіці побудови простору: генерація

повинна забезпечити проходження рівня без «мертвих кутів», виключаючи ситуації, де гравець або противник опиняється в пастці через помилку генератора. Водночас важливо зберегти напругу дослідження: неочікувані повороти, розгалуження, тупики мають з’являтися контрольовано — як частина геймдизайну [4][7].

Крім геометрії рівня, необхідно реалізувати генерацію декорацій, які виконують не лише естетичну, а й функціональну роль: візуальні орієнтири, перешкоди, місця для засідок, інтерактивні об’єкти. Ці елементи повинні узгоджуватись із темою підземелля — від сирих печер до руїн храму — і підтримувати атмосферу світу [2][4][5].

Також задачі включають:

- побудову системи параметрів, які дозволять розробникам впливати на характер генерації (розмір, складність, щільність декорацій, стиль тощо) [5][6];
- забезпечення коректної роботи навігаційної сітки для ШІ противників, з урахуванням змінної структури рівня [7][8];
- оптимізацію процесу генерації з урахуванням продуктивності в ігровому середовищі Unreal Engine 5 [8].

Загальна мета — створити гнучку, надійну систему генерації, яка дозволить формувати унікальні й насичені середовища, придатні як для бойових ситуацій, так і для дослідження, з мінімальним ручним втручанням з боку розробника.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Постановка мети

Метою проекту є розробка гри жанру dungeon crawler, з використанням процедурної генерації локацій і декорацій для забезпечення унікальності кожного ігрового досвіду. Гра буде розроблена на рушії Unreal Engine 5 із застосуванням мови програмування C++ та Blueprint.

Проект передбачає створення ігрової механіки, що включає генерацію підземель, їх елементів та взаємодію гравця з оточенням. Генерація локацій буде враховувати не тільки рандомізацію розташування, але й інтеграцію таких елементів як вороги, пастки, предмети, що взаємодіють з прогресією персонажа. Гравець зможе проходити через різноманітні локації, кожна з яких буде мати унікальний вигляд і атмосферу, створену за допомогою процедурної генерації.

Основна увага приділяється створенню механік, що будуть підтримувати динамічний ігровий процес, гармонійно інтегруючи генерацію локацій в загальну структуру гри. Генерація ландшафту і декорацій буде орієнтована на те, щоб забезпечити баланс між візуальною складовою та вимогами ігрового процесу.

Ціль — створити захопливу гру, де кожна нова гра пропонує унікальний досвід для гравця, який розвивається на основі випадкових, але виважених рішень щодо створення локацій.

2.2 Загальний опис

Система генерації локацій та декорацій для гри в жанрі dungeon crawler є основою для створення захоплюючого ігрового досвіду. Вона дозволяє гравцям потрапити в унікальні, динамічні підземелля, де кожна локація має свою атмосферу та виклики. Система базується на алгоритмах процедурної генерації, що створюють неповторні локації, які адаптуються до стилю гри та прогресу персонажа.

Компоненти системи. Система включає набір модулів та контролерів, які взаємодіють з іншими частинами гри, такими як механіка бою, рух персонажа та інші. Контролери налаштовують параметри генерації локацій, дозволяють регулювати рівень складності, розташування ворогів, пасток і предметів. Генерація

ландшафтів та декорацій адаптується до вимог геймплею, підтримуючи баланс між виглядом та ігровими механіками.

Взаємодія з ігровими механіками. Система генерації локацій інтегрована з основними ігровими механіками, такими як рух, бій та прогресія персонажа. Генерація локацій має враховувати не тільки візуальні аспекти, але й ефекти на геймплей, наприклад, створення нових шляхів для просування через підземелля або влаштування перешкод, які можуть впливати на стратегію гравця.

Модульність та розширюваність. Система спроектована з урахуванням модульності, що дозволяє легко додавати нові функціональні можливості без необхідності змінювати основний код. Конфігураційні файли і бібліотеки модулів відповідають за різні аспекти генерації локацій, що дозволяє легко налаштовувати, масштабувати та адаптувати систему до нових вимог.

2.3 Загальні обмеження

Система генерації локацій та декорацій повинна бути реалізована на мові програмування C++ та Blueprint з використанням рушія Unreal Engine 5. Гра повинна працювати на платформах Windows 10/11. Усі інструменти для розробки повинні бути сумісними з останніми версіями Unreal Engine і забезпечувати оптимальну продуктивність навіть для складних процедурних генерацій.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

Діаграма прецедентів є засобом візуалізації, що використовується для опису функціональних можливостей системи з точки зору користувача.

Для розроблюваного ігрового застосунку було розроблено діаграму прецедентів, що відповідають діям користувача у грі (див. рис. 3.1).

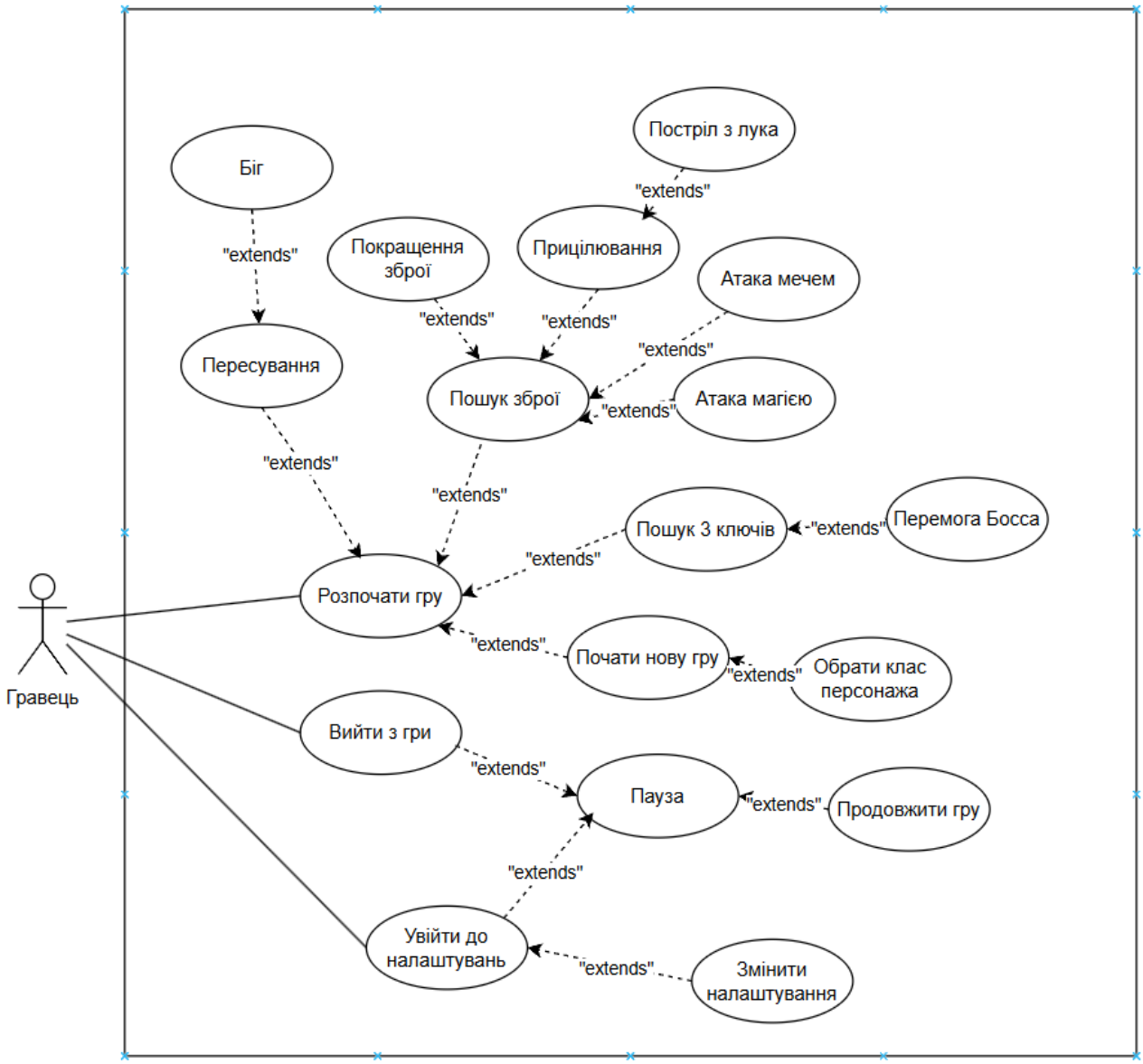

Рисунок 3.1 – Діаграма прецедентів (зроблено власноруч)

Ця діаграма відображає взаємодію між ігровим застосунком та гравцем через ідентифікацію основних дій, які можуть бути виконані в системі. Цей тип діаграми

допомагає зрозуміти потреби користувачів та визначити основні функції системи з їхнього погляду.

3.2 Проектування архітектури ПЗ

Діаграма класів ігрового застосунку демонструє взаємозв’язки між основними класами програмного забезпечення, що відповідають за створення рівнів, розміщення декорацій, а також інтеграцію ігрових механік у жанрі Dungeon Crawler (див. рисунок 3.2).

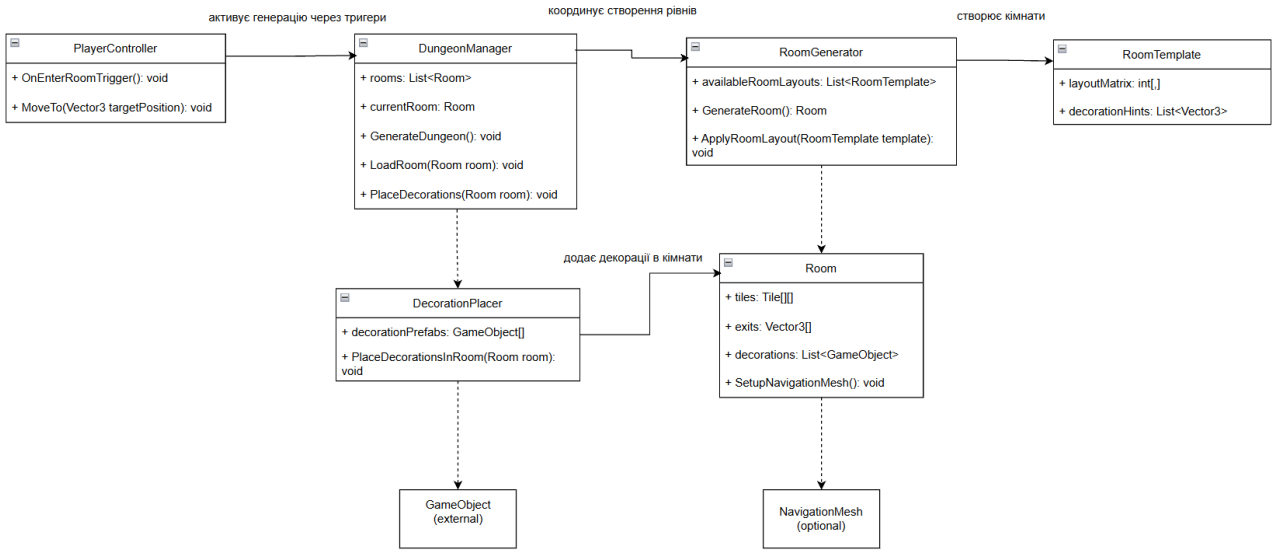

Рисунок 3.2 – Діаграма класів (зроблено власноруч)

Ця діаграма класів створена для демонстрації взаємодії між основними компонентами ігрового застосунку та гравцем, а також показує, як відбувається створення і розміщення декорацій на рівні. Використання класу PlayerController зумовлене тим, що гравець взаємодіє з елементами середовища, активуючи через тригери процеси генерації та оновлення ігрового простору.

3.3 Проектування алгоритму генерації ландшафтів та декорацій

Розроблено діаграму активності для демонстрації поведінки системи процедурної генерації рівня та створення нових кімнат у грі жанру dungeon crawler (див. рисунок 3.3).

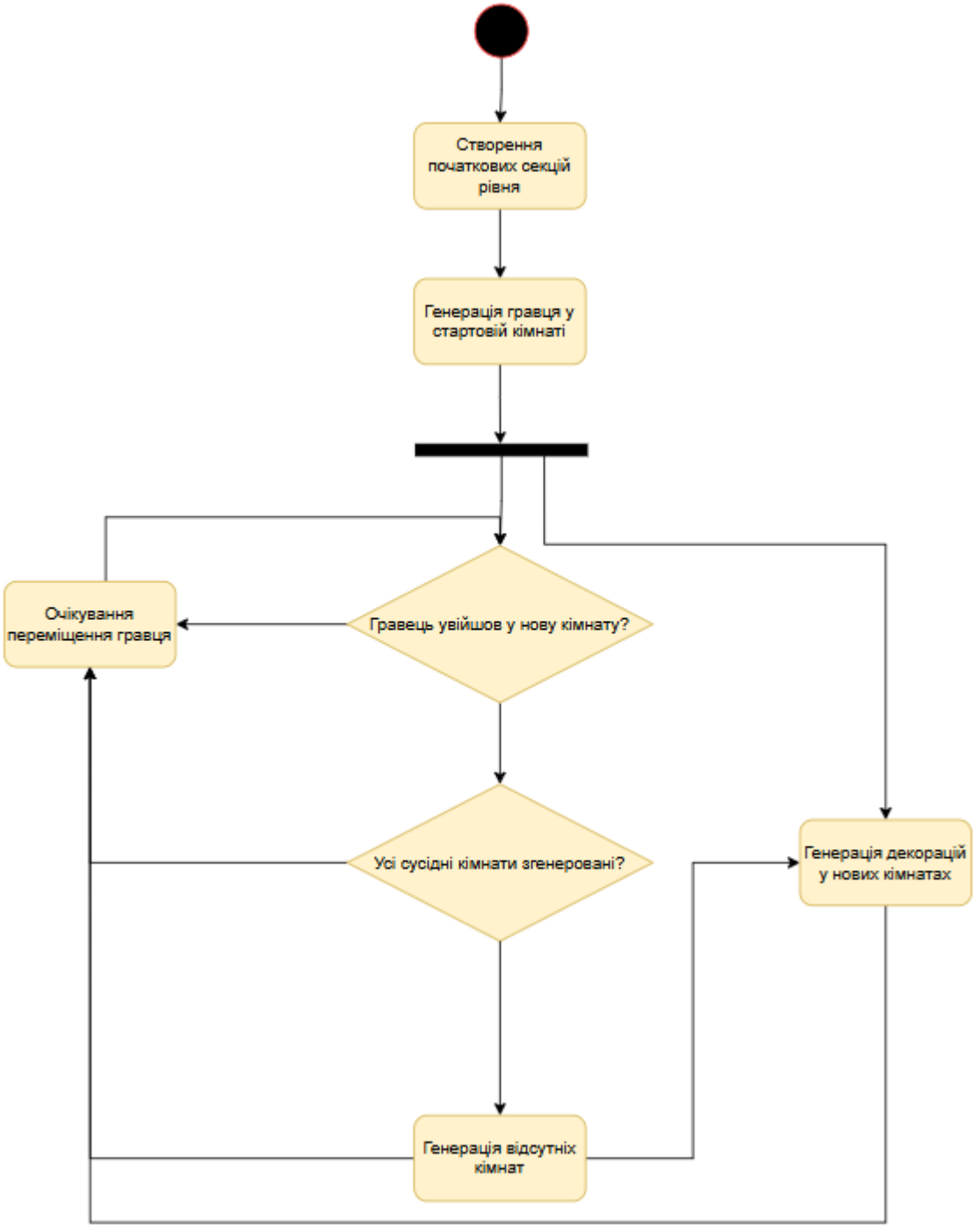

Рисунок 3.3 – Діаграма активності системи генерації ландшафту та декорацій (зроблено власноруч)

Ця діаграма ілюструє процес створення та процедурної генерації рівня навколо гравця на початку гри та під час його переміщення.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Створення візуальної частини системи ігрового інвентаря

Напочатку всього було створено папку Inventory, в якій буде знаходитись вся логіка пов’язана з системою ігрового інвентаря. Далі в папці Inventory було створено ще 4 папки (рис. 4.1):

– BaseItems: папка в якій знаходяться файли, що відносяться до всіх предметів в грі;

– EquipmentSystem: папка в якій знаходяться файли, що відносяться до предметів екіпірування;

– UI: папка з файлами візуальної частини ігрового інвентаря;

– UsingSystem: папка в якій знаходяться файли, що відносяться до предметів, які можна використати.

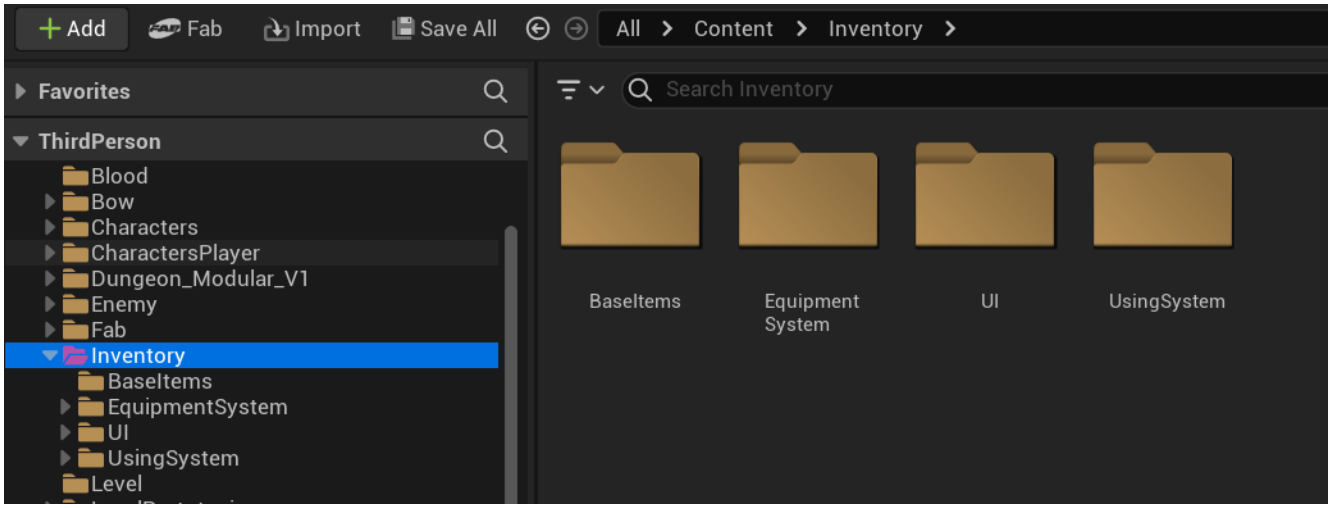

Рисунок 4.1 – Створена папка системи ігрового інвентаря (зроблено власноруч)

Після того, як було створено папку для ігрового інвентаря, потрібно створити файли WidgetBlueprint (рис. 4.2), які відповідають за візуальну частину гри, і будуть відображати ігровий інвентар.

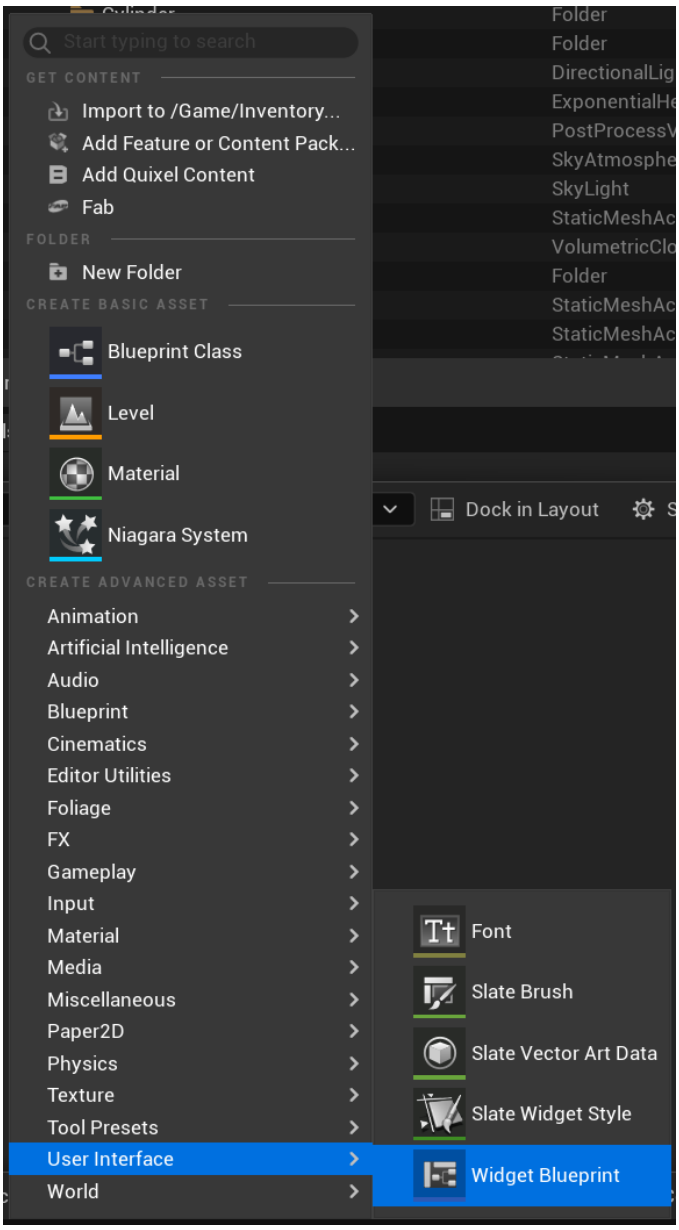

Рисунок 4.2 – Створення віджетів для ігрового інвентаря (зроблено власноруч)

Було створено 10 файлів WidgetBlueprint, а також 2 папки Textures та Tint (рис. 4.3):

- UI_ActionMenu: віджет для взаємодії з предметами ігрового інвентаря;
- UI_BackgroundSlot: віджет, що відображає задній фон інвентаря;
- UI_DragWidget: віджет, що відповідає за відображення DragAndDrop операцій;
- UI_EquipmentSlotBow: віджет, що відображає слот екіпірування для лука;
- UI_EquipmentSlotSword: віджет, що відображає слот екіпірування для меча;

– UI_EquipmentWindow: віджет, який відображує вікно екіпірування;
– UI_Grid: сітка слотів для ігрового інвентаря;
– UI_Inventory: віджет, який включає в себе всі елементи інвентаря і відображає їх на екрані;
– UI_Slot: віджет, який відповідає за відображення одного слота ігрового інвентаря;
– UI_ToolTipInventory: віджет, що відповідає за відображення повної інформації про предмет в ігровому інвентарі;
– Textures: папка, для зберігання текстур ігрового інвентаря;
– Tint: папка, для зберігання шрифтів ігрового інвентаря.

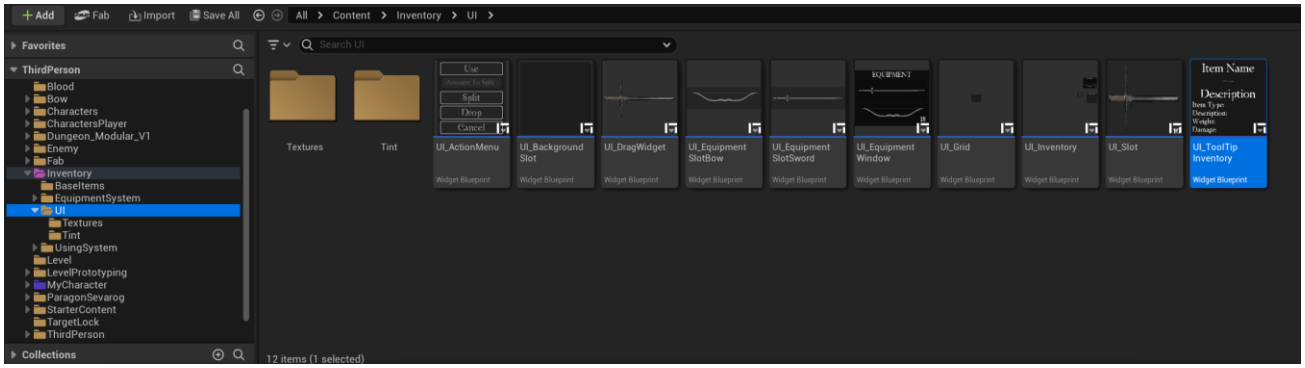

Рисунок 4.3 – Папка зі створеними файлами WidgetBlueprint (зроблено власноруч)

Оскільки розроблений ігровий інвентар складається з багатьох слотів 1 на 1, то розробимо функцію, яка буде відображати сітку інвентаря, в залежності від вказаних розмірів (рис. 4.4)

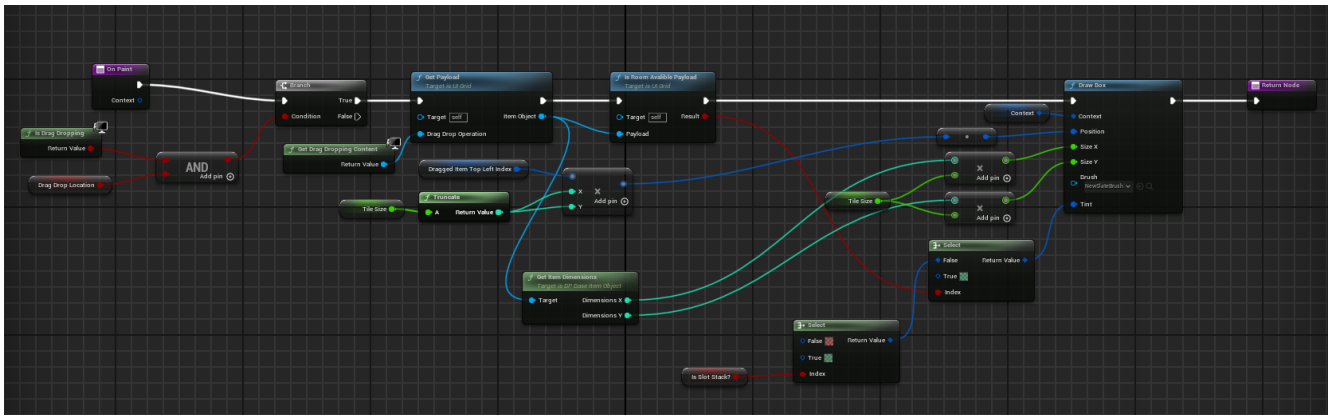

Рисунок 4.4 – Функція для відображення сітки інвентаря (зроблено власноруч)

Оптимальним розміром було обрано 7 колонок на 9 рядків.

Розробивши всі віджети, додамо їх в головний віджет UI_Inventory (рис. 4.5)

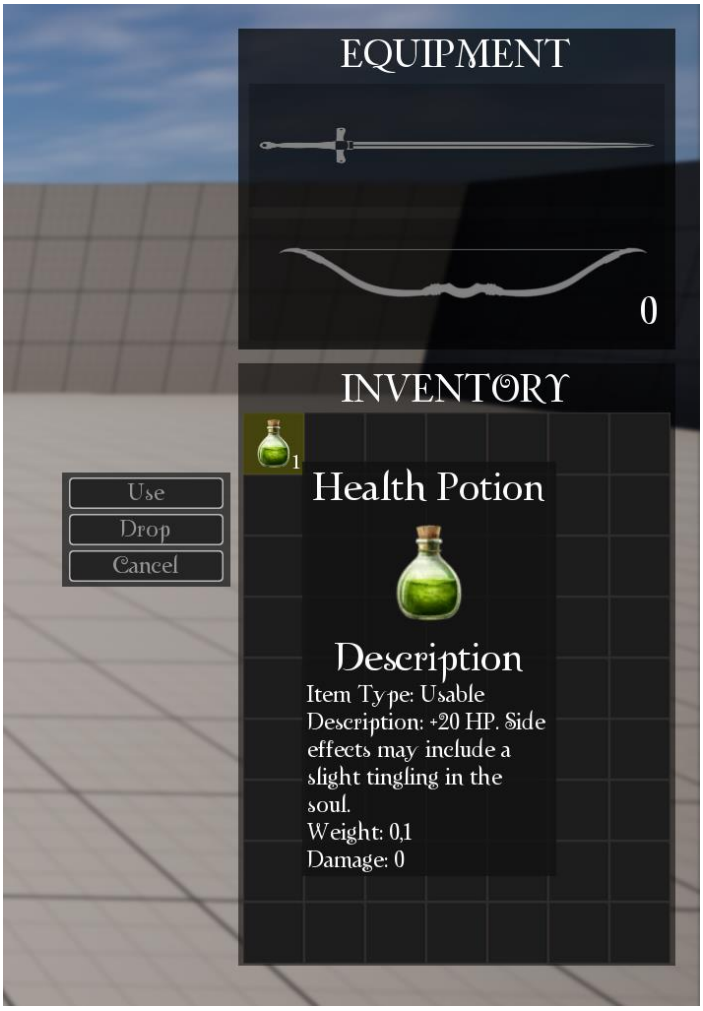

Рисунок 4.5 – Розроблена візуальна частина ігрового інвентаря (зроблено власноруч)

4.2 Створення ігрових предметів

Коли ми вже створили візуальну частину ігрового інвентаря, нам потрібно додати предмети, які можна поміщати в наш інвентар.

Предмети можуть бути:

- Екіпіруванням: луки, одноручні мечі, дворучні мечі;
- Розхідники: зілля життя, зілля мани, стріли.

Також деякі предмети можна складати, тобто стакати. Така можливість є тільки у розхідників.

Предмети повинні мати розмір, який вони займають в ігровому інвентарі, наприклад, зілля життя займає 1 на 1 слот, а одноручний меч 1 на 4 слоти.

В папці BaseItems створимо 6 файлів:

– BP_BaseInventoryItem: файл предмета, який включає в себе Mesh для відображення предмета в грі;
– BP_BaseItemObject: файл з всіма даними предмета;
– BPC_Inventory: контроллер, для взаємодії гравця з інвентарем;
– BPI_Interact: інтерфейс, для додавання предмета в ігровий інвентар;
– ItemInfo: інформація про предмет;
– ItemType: тип предмета.

В файл BP_BaseItemObject додамо перемінні для зберігання даних предметів(рис 4.6)

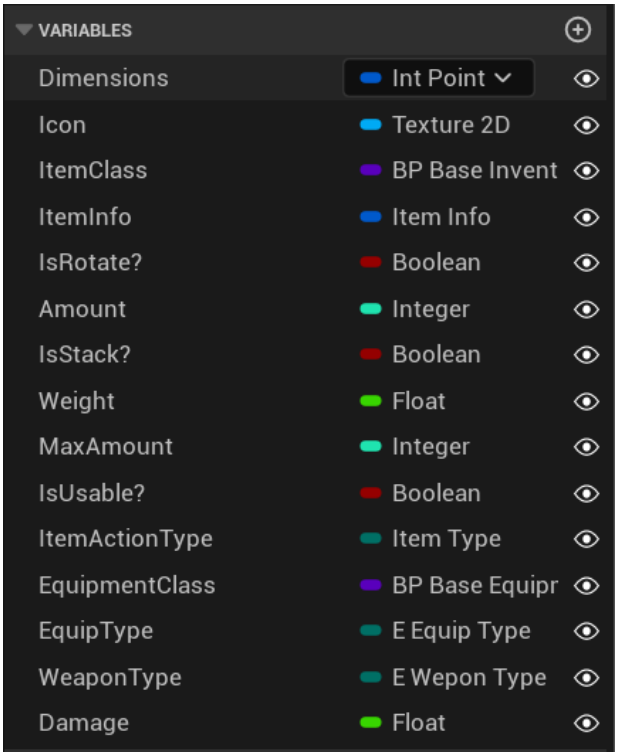

Рисунок 4.6 – Перемінні, які належать ігровому предмету (зроблено власноруч)

Далі створимо файл BP_BaseInventoryItem, та в ньому створимо функцію GetDefoultItemObject і підключимо всі перемінні, які належать предметам (рис. 4.7)

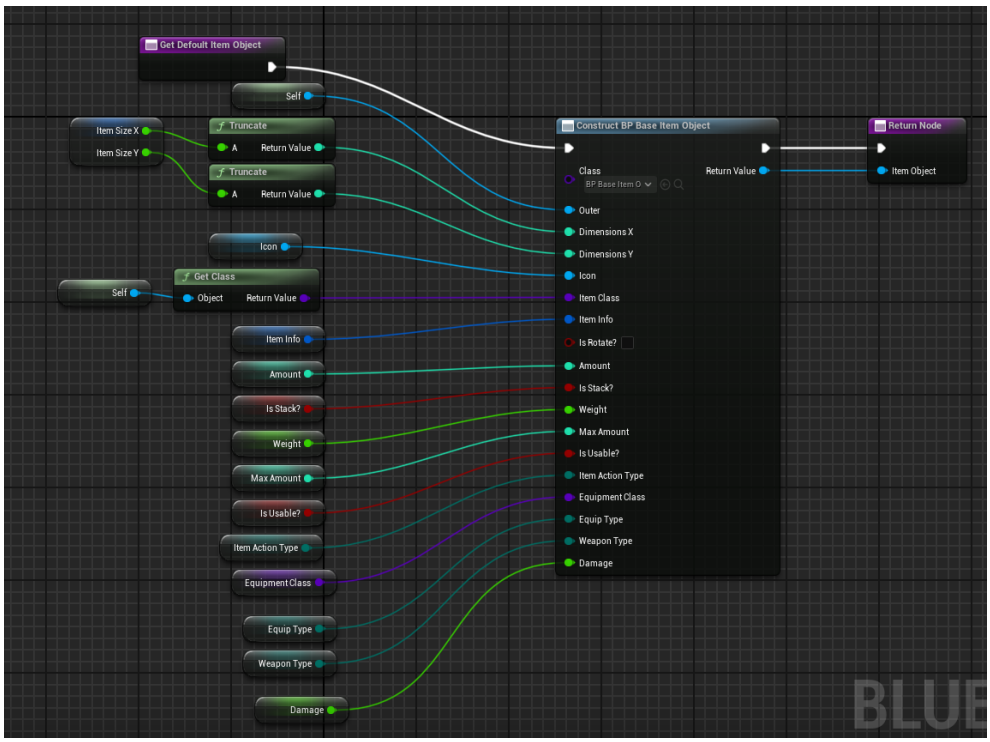

Рисунок 4.7 – Функція GetDefoultItemObject і підключення перемінних предметів (зроблено власноруч)

Таким чином, ми можемо створювати нові предмети і налаштовувати інформацію про них(рис. 4.8)

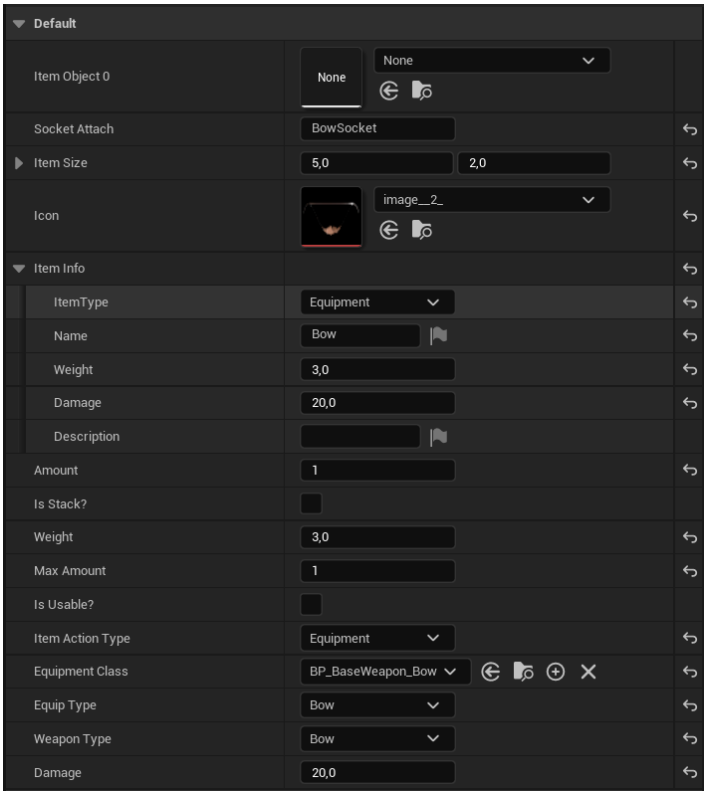

Рисунок 4.8 – Додавання інформації про предмет (зроблено власноруч)

4.3 Додавання предметів до ігрового інвентаря

Було обрано взаємодію з предметами за допомогою SphereTrace (рис. 4.9), тобто якщо у предмета підключено інтерфейс для взаємодії BPI_Interact, то він буде додаватися в інвентар.

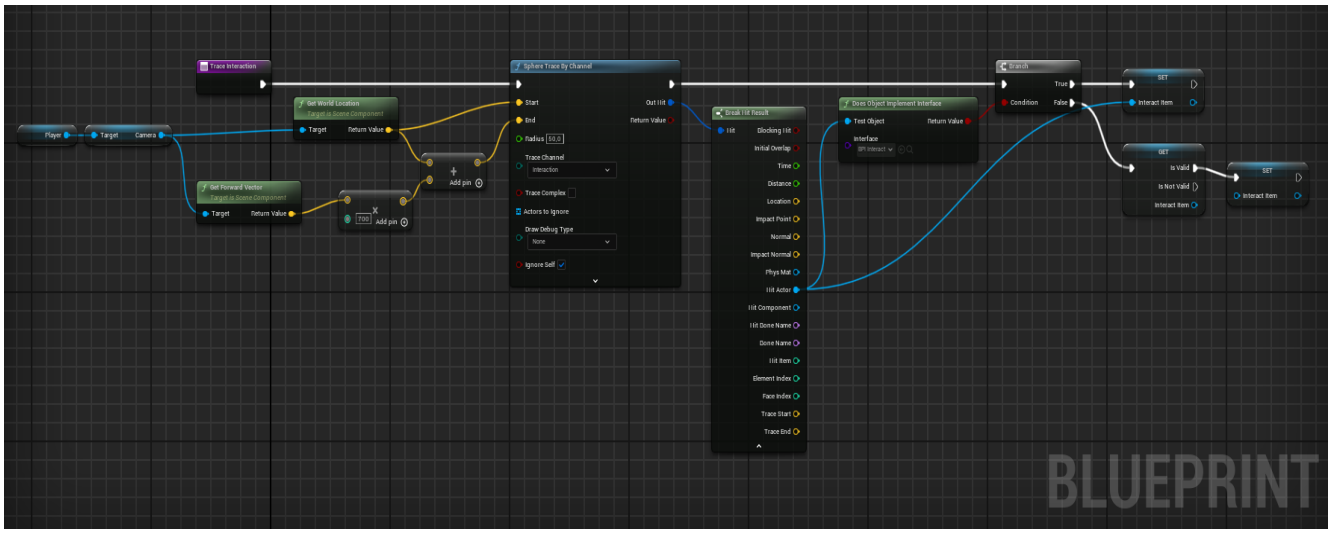

Рисунок 4.9 – Функція взаємодії з предметами за допомогою SphereTrace (зроблено власноруч)

Функція Trace Interaction виконує сферичний трасінг від камери гравця вперед на 700 одиниць з радіусом 50, використовуючи спеціальний канал "Interaction". Вона перевіряє, чи об'єкт, у який потрапляє трасінг, реалізує інтерфейс BPI_Interact. Якщо так — зберігає цей об'єкт у змінну Interact Item для подальшої взаємодії, інакше скидає попереднє значення. Таким чином реалізується система виявлення інтерактивних об'єктів у полі зору гравця.

Далі виконується функція TryAddItem (рис. 4.10), яка додає предмет в ігровий інвентар.

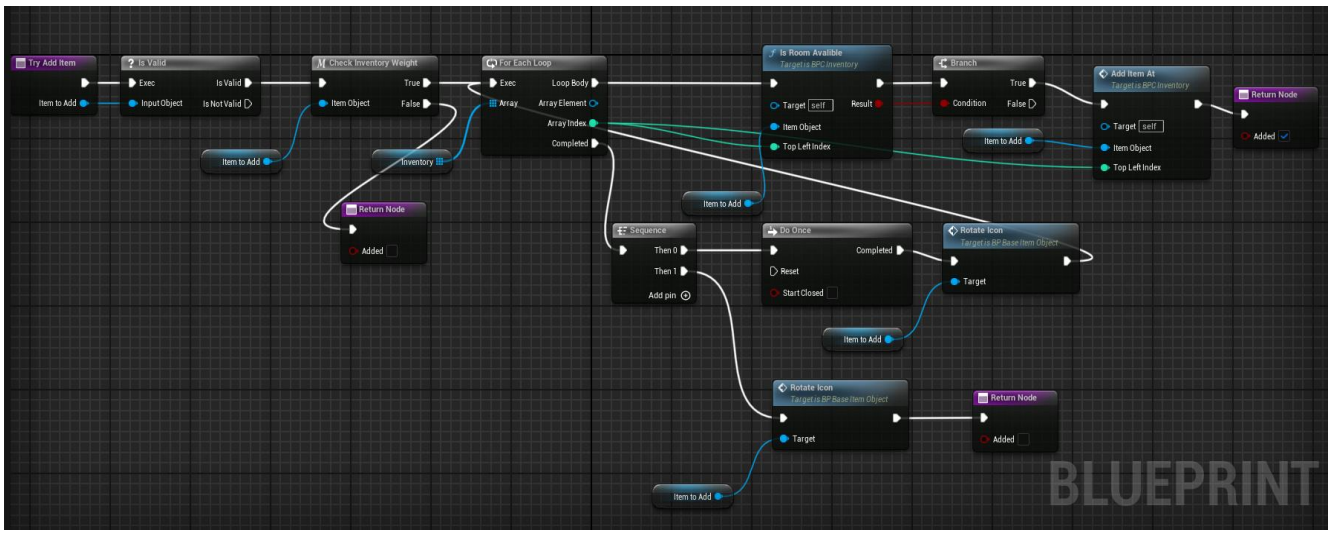

Рисунок 4.10 – Функція TryAddItem, для додавання предметів до інвентаря (зроблено власноруч)

Функція TryAddItem відповідає за додавання предмета до інвентаря. Спочатку перевіряється, чи дійсний об’єкт предмета, після чого викликається перевірка на допустиму вагу інвентаря. Якщо вага не перевищена, запускається цикл, який проходиться по всіх слотах інвентаря і за допомогою функції IsRoomAvailable перевіряє, чи є вільне місце. Якщо таке місце знайдено — предмет додається функцією AddItemAt, і повертається значення "Added = true". Якщо місця немає, запускається блок Sequence, який двічі намагається розмістити предмет з поворотом (через RotateIcon), і якщо після обох спроб вільного місця не знайдено — функція завершується з "Added = false". Таким чином реалізується додавання предметів з урахуванням ваги, простору та можливої ротації для розміщення.

4.4 Екіпірування та використання предметів

У ігрових предметів є 2 типи Equipment та Usable, тож потрібно розробити функції використання та екіпірування предметів в залежності від їх типу.

Спочатку розробимо функцію для використання розхідників(рис. 4.11).

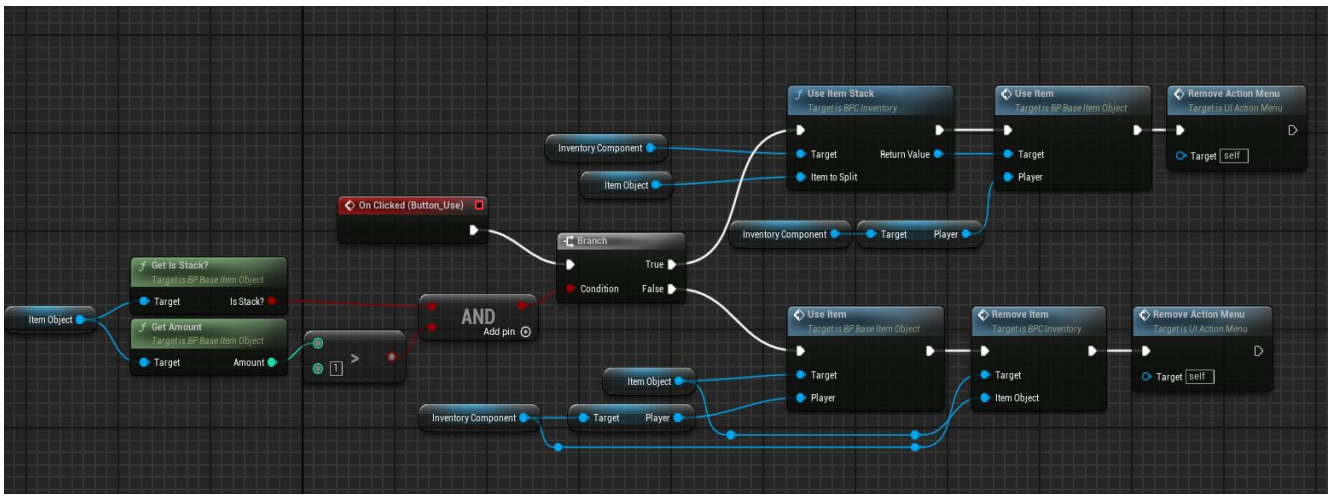

Рисунок 4.11 – Функція використання предметів (зроблено власноруч)

Ця функція в Unreal Engine 5 реалізує логіку використання предмета з інвентаря при натисканні кнопки "Use". Спочатку вона перевіряє, чи є предмет стековим (Is Stack?) і чи його кількість більша за 1. Якщо обидві умови виконуються, викликається функція розділення стеку (Use Item Stack) через інвентар, після чого виконується використання нового предмета та закривається меню дій. Якщо ж предмет не є стековим або його лише один, предмет використовується напряму (Use Item), видаляється з інвентаря та також закривається меню дій.

Далі розробимо функцію екіпірування предметів(рис. 4.12).

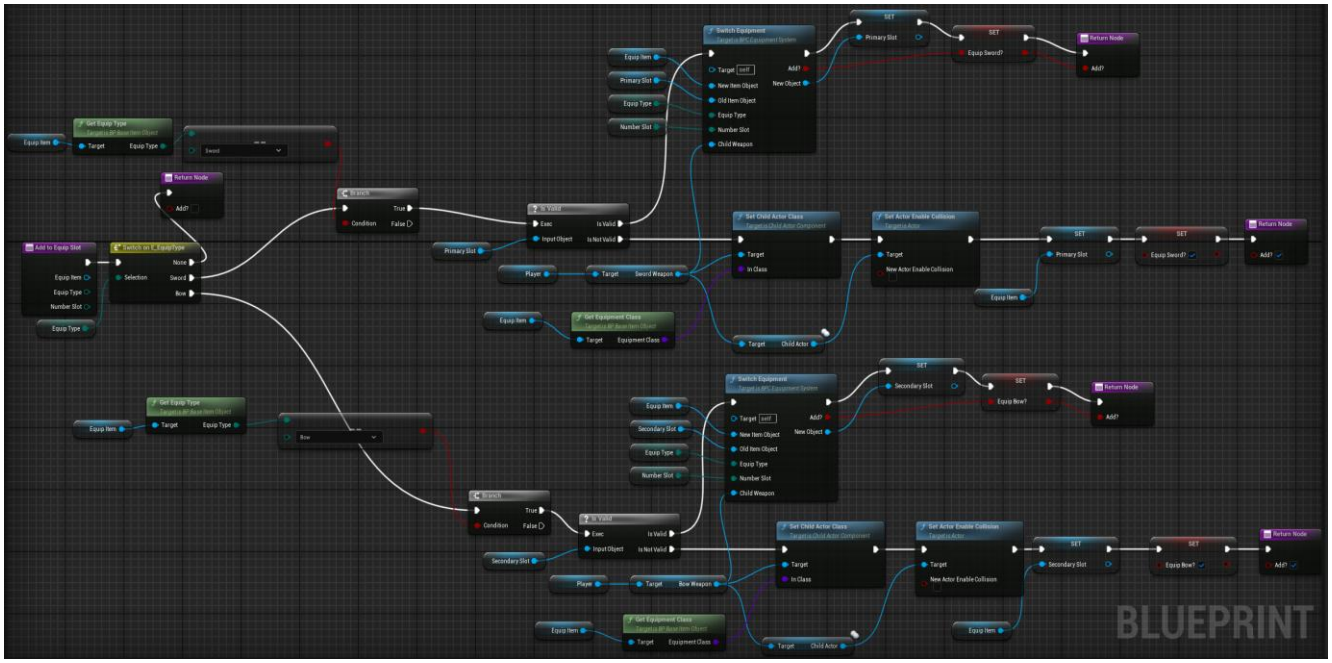

Рисунок 4.11 – Функція екіпірування предметів (зроблено власноруч)

Ця функція реалізує систему екіпірування предметів у слот зброї гравця в залежності від типу (меч або лук). Визначається тип предмета, після чого через Switch on E_EquipType виконується логіка для відповідного слота: Primary для меча, Secondary для лука. Якщо слот вже містить зброю, вона замінюється через Switch Equipment. Новий предмет активується, додається в слот, встановлюється в якості дитини для Child Actor Component, вмикається колізія, і зберігається посилання на нього. У підсумку функція оновлює екіпіровану зброю та повертає статус додавання (Add?).

У процесі розробки системи інвентаря особливу увагу було приділено зручності взаємодії гравця з предметами, адже саме інтуїтивно зрозумілий інтерфейс та логіка дій формують позитивний користувацький досвід. Реалізація поділу предметів на типи (Equipment та Usable) дозволяє ефективно структурувати логіку взаємодії з ними, а також спрощує розширення функціоналу у майбутньому. Наприклад, за потреби можна легко додати нові типи предметів, як-от магічні артефакти або квестові об’єкти, не змінюючи вже наявну систему, а лише доповнюючи її новими гілками логіки.

Важливо також зазначити, що обидві функції — використання та екіпірування — реалізуються з урахуванням можливих виключень, таких як відсутність предмета, спроба використання непридатного об’єкта або конфлікт екіпірування. Такий підхід сприяє підвищенню стабільності гри та зменшенню кількості помилок під час ігрового процесу. Загалом, розроблені функції є базовою, але необхідною частиною загальної системи інвентаря, яка є невід’ємною складовою будь-якого dungeon crawler-проєкту, що претендує на повноцінний геймплей.

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Тестування ігрового застосунку

Ігровий програмний застосунок було протестовано згідно з попередньо підготовленим тестовим планом. На рисунку 5.1 представлено Mind Map, що слугує основою цього плану. Для перевірки функціональності застосовувалася методика Blackbox-тестування, відома також як тестування «чорної скриньки».

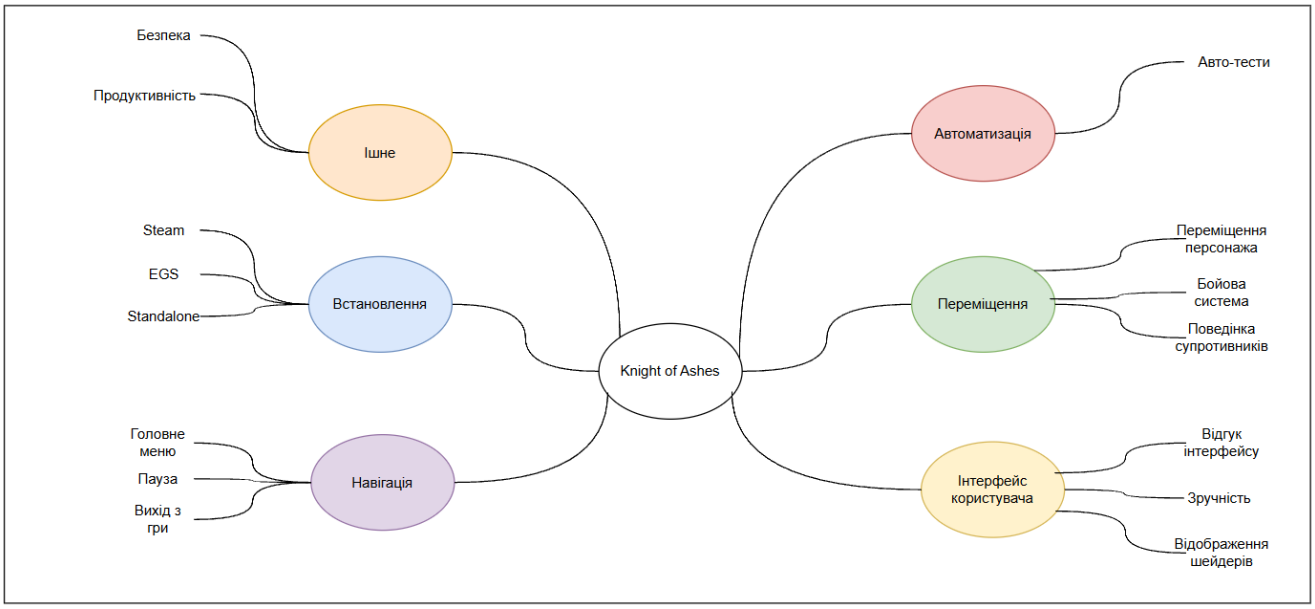

Рисунок 5.1 – Mind Map для гри «Knight of Ashes» (зроблено власноруч)

За допомогою цієї карти ми можемо поділити тестування проекту на окремі частини:

а) Інше;

б) Встановлення;

в) Навігація;

г) Автоматизація;

д) Переміщення;

е) Інтерфейс користувача.

Було проведено тестування в процесі розробки додатку. Тестування функціональності, навігації та інтерфейсу користувача.

5.2 Розробка тестових випадків

На основі класифікації, наведеної у вигляді діаграми Mind Map, було сформовано набір тестових сценаріїв для детального тестування системи. Кожен сценарій містить опис умов проведення, очікуваний і фактичний результати, а також додаткові коментарі. У таблицях 5.1–5.6 наведено сценарії, які були використані під час тестування програмного продукту. Пріоритет сценарію визначається за шкалою від P1 (найвищий) до P4 (найнижчий). Рівень критичності можливої помилки оцінюється від S1 (незначна критичність) до S4 (критична помилка, що може зруйнувати ігровий процес).

Таблиця 5.1 – Баг 1 (таблиця виконана самостійно)

Назва багу	Неправильна колізія предмета при екіпіруванні
Короткий опис	При екіпіруванні предмета його колізія пересікається з колізією головного героя
Компонент додатку	BP_BaseEquipmentItem
Важливість	S3 Висока
Пріоритет	P2 Високий
Кроки відтворення	1. Підібрати та екіпірувати предмет
Фактичний результат	Персонаж самостійно хаотично рухається
Очікуваний результат	Персонаж може вільно переміщатися з екіпірованим предметом

Як ми бачимо у таблиці 5.1, колізія екіпірованого предмета заважає персонажеві ходити, порушується геймплей гри.

Таблиця 5.2 – Баг 2 (таблиця виконана самостійно)

Назва багу	Неправильне розміщення предметів в ігровому інвентарі
Короткий опис	Предмети не повертаються автоматично при додаванні їх в ігровий інвентар
Компонент додатку	BPC_Inventory
Важливість	S2 Середня
Пріоритет	P3 Середній

Кінець таблиці 5.2 (таблиця виконана самостійно)

Кроки відтворення	1. Підібрати декілька предметів 2. Спробувати підібрати ще один предмет, якому для розміщення потрібний поворот
Фактичний результат	Предмет не додається до ігрового інвентаря
Очікуваний результат	Предмет автоматично повернувся та додався до інвентаря

Баг 2, з таблиці 5.2, звісно, не сильно заважає гравцю, але створює проблему додавання предметів в інвентар.

Таблиця 5.3 – Баг 3 (таблиця виконана самостійно)

Назва багу	Проходження скрізь предмети
Короткий опис	Гравець може проходити крізь тверді предмети
Компонент додатку	BP_Knight
Важливість	S1 Блокуюча
Пріоритет	P1 Високий
Кроки відтворення	1. Знайти будь-який предмет 2. Спробувати пройти крізь нього
Фактичний результат	Гравець може пройти крізь предмет
Очікуваний результат	Гравець не може пройти крізь предмет

У багу 3, з таблиці 5.3, знайдено типовий баг ігрових додатків, коли гравець може проходити крізь тверді об’єкти. Зазвичай, це порушує ігрове враження.

Таблиця 5.4 – Баг 4 (таблиця виконана самостійно)

Назва багу	Неправильна стрільба з лука
Короткий опис	Випущені стріли летять не в те місце, в яке цілився гравець
Компонент додатку	BP_Knight
Важливість	S3 Середня
Пріоритет	P2 Середній
Кроки відтворення	1. Зробити вистріл з лука

Кінець таблиці 5.4

Фактичний результат	Стріла летить не в центр екрану
Очікуваний результат	Стріла летить в центр екрану

Баг 4, типовий баг при стрільбі з лука, коли стріли летять повз ціль, попри наведення гравця. Зазвичай, це порушує ігрове враження.

Таблиця 5.5 – Баг 5 (таблиця виконана самостійно)

Назва багу	Освітлення при використанні магії
Короткий опис	Дуже сильне освітлення при використанні магії
Компонент додатку	BP_Fireball
Важливість	S2 Висока
Пріоритет	P2 Середній
Кроки відтворення	1. Використати магію
Фактичний результат	Ігрова сцена занадто засвітлена
Очікуваний результат	Освітлення природне

Як видно з таблиці 5.5, виявлені помилки пов’язані з освітленням, яке суттєво впливає на загальне враження від гри.

Таблиця 5.6 – Баг 6 (таблиця виконана самостійно)

Назва багу	Надто сильне освітлення на сцені
Короткий опис	Від джерел світла на сцені занадто багато світла
Компонент додатку	DUN_Dungeon_MAP
Важливість	S2 Висока
Пріоритет	P3 Високий
Кроки відтворення	1. Знайти джерело світла на сцені
Фактичний результат	Освітлення занадто сильне
Очікуваний результат	Освітлення природне

Баг з таблиці 5.6, також відноситься до проблем з світлом. Коректна робота зі світлом є надзвичайно важливою під час розробки нашого застосунку, оскільки неправильне освітлення може знизити атмосферність і реалістичність сцен. Тому усунення цих помилок має стати пріоритетним завданням для покращення користувацького досвіду. Виправлення проблем з освітленням не лише підвищить візуальну якість, але й допоможе уникнути можливого розчарування гравців через технічні недоліки. Реалістичне та плавне освітлення є ключовим елементом, що сприяє створенню захопливого ігрового середовища та значною мірою формує загальне сприйняття гри.

6 ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

6.1 Соціальне впровадження проекту

Під час розробки нашого застосунку було створено акаунти на популярних соціальних платформах для просування гри. Ми використовували Telegram, YouTube, Instagram та Twitter/X. Найвищу ефективність показав Telegram, оскільки саме там зосереджена більшість представників нашої цільової аудиторії. Instagram та YouTube також дали позитивний результат, сприяючи залученню нових користувачів. Водночас Twitter/X виявився менш результативним, адже платформа не є популярною серед нашої аудиторії.

У майбутньому ми плануємо зосередити просування на Telegram, Instagram та YouTube. Також розглядаємо можливість створення спільнот на спеціалізованих платформах, таких як Steam Community або Fandom, щоб розширити охоплення та комунікацію з гравцями.

6.2 Реліз гри на платформі Steam

Одним із ключових етапів реалізації нашого проекту є запуск гри на платформі Steam. Це рішення продиктоване великою популярністю Steam серед геймерів у всьому світі, а також широким набором інструментів для розробників. Платформа надає можливість швидкого доступу до гри широкому загалу користувачів.

Для випуску гри на Steam ми використовуємо модель Standalone, яка дозволяє гравцям завантажувати та запускати гру без потреби у додаткових програмних компонентах. Ми впевнені, що це підвищить зручність використання та приверне більше користувачів. Крім того, ми плануємо інтегрувати нашу гру з екосистемою Steam Community, де користувачі зможуть обговорювати гру, ділитися враженнями та отримувати підтримку.

6.3 Подальші плани та розвиток

У нас є амбітні плани щодо подальшого розвитку гри. Після виходу на Steam ми передбачаємо регулярні оновлення, які включатимуть нові рівні, функціонал і

виправлення виявлених помилок. У перспективі — порт гри на інші платформи, зокрема Epic Games Store і сучасні ігрові консолі.

Ми також продовжимо активне просування проекту через соціальні мережі та професійні платформи, з метою підтримки інтересу до гри та розширення аудиторії. Наша мета — сформувати стабільну спільноту гравців, яка не лише гратиме, а й братиме участь у розвитку проєкту.

У підсумку, ми переконані, що наша гра має значний потенціал і здатна зайняти гідне місце на ринку відеоігор.

ВИСНОВКИ

У ході виконання цієї кваліфікаційної роботи було розроблено систему процедурної генерації рівнів та декорацій для ігрового застосунку в жанрі dungeon crawler. Розробка здійснювалася на базі рушія Unreal Engine 5 з використанням візуального програмування через Blueprints та підтримкою ОС Windows 10/11.

Було проведено всебічний аналіз предметної галузі, зокрема досліджено ігри, які реалізують схожі механіки підземель, процедурного створення локацій, та інтерактивної взаємодії з елементами оточення. Вивчення ринку та жанрових особливостей дозволило сформувати чітке бачення вимог до системи генерації, яка відповідала б очікуванням гравців та тенденціям ігрової індустрії.

На основі цього аналізу було спроєктовано архітектуру системи, що підтримує модульність, масштабованість і можливість легкого налаштування параметрів генерації. Розроблено діаграми класів і діаграми активності, які демонструють взаємодію окремих компонентів системи, включно з гравцем, тригерами генерації, об’єктами та логікою побудови підземель.

Реалізована система генерації дозволяє створювати динамічні рівні зі змінною структурою кімнат, проходів і декоративних об’єктів. Вона підтримує інтерактивні тригери, які реагують на переміщення гравця та забезпечують поступову генерацію нових ділянок підземелля в реальному часі.

Гнучкість системи забезпечує її адаптивність до зміни геймплейних сценаріїв та дозволяє в майбутньому розширювати її можливості — наприклад, додавати нові типи кімнат, ворогів, пасток або варіанти освітлення. Завдяки оптимізації логіки генерації вдалося зберегти стабільність продуктивності навіть при складних структурах рівня.

Отже, розроблена система процедурної генерації рівнів для гри в жанрі dungeon crawler є ефективною, масштабованою та зручною для подальшої розробки. Вона дозволяє створювати унікальні ігрові світи з високим рівнем занурення, що робить її цінним інструментом для подальших проєктів у цьому жанрі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Dungeon crawler – пояснення [Електронний ресурс] – URL: https://uk.wikipedia.org/wiki/Dungeon_crawler (дата звернення: 15.04.2025).

2. Legend of Grimrock – опис гри [Електронний ресурс] – URL: https://store.steampowered.com/app/207170/Legend_of_Grimrock/ (дата звернення: 15.04.2025).

3. Path of Exile [Електронний ресурс] – URL: https://store.steampowered.com/app/238960/Path_of_Exile/ (дата звернення: 15.04.2025).

4. Minecraft Dungeons [Електронний ресурс] – URL: https://www.minecraft.net/en-us/about-dungeons (дата звернення: 15.04.2025).

5. Procedural Generation in Games [Електронний ресурс] – URL: https://en.wikipedia.org/wiki/Procedural_generation (дата звернення: 15.04.2025).

6. Як працює процедурна генерація рівнів у dungeon crawler [Електронний ресурс] – URL: https://gamedevelopment.tutsplus.com/tutorials/how-to-generate-a-dungeon-crawler-map--cms-29652 (дата звернення: 15.04.2025).

7. Програмна генерація рівнів: алгоритми та підходи [Електронний ресурс] – URL: https://habr.com/ru/articles/278333/ (дата звернення: 15.04.2025).

8. Unreal Engine 5 – офіційна документація [Електронний ресурс] – URL: https://docs.unrealengine.com/5.0/en-US/ (дата звернення: 15.04.2025).

9. GitHub репозиторій дипломного проекту – URL: https://github.com/NureBekhovVadym/2025_B_PI_PZPI-21-10_Bekhov_V_S.git