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 3D Rogue-like шутер від третьої особи. Механіки бойової системи, штучного інтелекту, генерації рівнів, економіки та інтерфейсу бою
(тема)

Виконав:
студент 4 курсу, групи ПЗПІ-20-7

Фурсов Д.С.
(прізвище, ініціали)

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

Тип програми освітньо-професійна

Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник старший викладач Новіков Ю.С.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 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)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студентові Фурсову Данилу Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Ігровий програмний застосунок у жанрі 3D Rogue-like шутер від третьої особи. Механіки бойової системи, штучного інтелекту, генерації рівнів, економіки та інтерфейсу бою
Затверджена наказом по університету від 20 05 2024 р. № 471Ст

2. Термін подання студентом роботи до екзаменаційної комісії 06.06.2024

3. Вихідні дані до роботи Розробити ігровий програмний застосунок в жанрі 3D Rogue-like шутер від третьої особи, а саме механіки бойової системи, штучного інтелекту, генерації рівнів, економіки та інтерфейсу бою, за допомогою ігрового рушію Unreal Engine 5 та мови програмування Blueprints.

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	12.05.2024	*виконано*
2	Створення специфікації ПЗ	14.05.2024	*виконано*
3	Проектування ПЗ	16.05.2024	*виконано*
4	Розробка ПЗ	20.05.2024	*виконано*
5	Тестування ПЗ	22.05.2024	*виконано*
6	Оформлення пояснювальної записки	28.05.2024	*виконано*
7	Підготовка презентації та доповіді	29.05.2024	*виконано*
8	Попередній захист	01.06.2024	*виконано*
9	Нормоконтроль, рецензування	29.05.2024	*виконано*
10	Здача роботи у електронний архів	02.06.2024	*виконано*
11	Допуск до захисту у зав. кафедри	03.06.2024	*виконано*

Дата видачі завдання 8 квітня 2024р.

Студент (ка)_____ Фурсов Д.С.
(підпис)

Керівник роботи _____ старший викладач Новіков Ю.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, стор. 115, табл. 1, рис. 90, джерел 15.

3Д ГРА, ІГРОВИЙ ПРОГРАМНИЙ ЗАСТОСУНОК, ШУТЕР ВІД ТРЕТЬОЇ ОСОБИ, ROGUE-LIKE, UNREAL ENGINE 5

Об’єкт розробки – Ігровий програмний застосунок у жанрі 3D Rogue-like шутер від третьої особи в стилістиці кіберпанку, а саме механіки бойової системи, штучного інтелекту ворогів, генерації рівнів, економіки та інтерфейсу бою.

Мета розробки – створення ігрового програмного застосунку, який матиме цікаву та різноманітну бойову систему шутера, процедурну генерацію рівнів, штучний інтелект ворогів та систему зберігання прогресу гравця. Дана гра повинна бути виконана у стилістиці кіберпанку.

Метод рішення – середовище розробки та ігровий рушій Unreal Engine 5, мова програмування Blueprint Visual Scripting та середовище 3D моделювання Blender.

У результаті розробки створено ігровий програмний застосунок, котрий має в собі бойову систему, процедурну генерацію рівнів, штучний інтелект ворогів та систему зберігання прогресу гравця.

3D GAME, GAMING SOFTWARE APPLICATION, THIRD-PERSON SHOOTER, ROGUE-LIKE, UNREAL ENGINE 5

Object of development – game software application in the genre of 3D Rogue-like third-person shooter in the style of cyberpunk, in particular the mechanics of the combat system, artificial intelligence of enemies, level generation, economy and combat interface.

The goal of the development – to create a gaming software application that will have an interesting and diverse shooter combat system, procedural level generation, artificial intelligence of enemies and a system for storing player progress. This game should be made in the style of cyberpunk.

The solution method – development environment and game engine Unreal Engine 5, programming language Blueprint Visual Scripting and 3D modeling environment Blender.

As a result of the development, a gaming software application was created that includes a combat system, procedural level generation, artificial intelligence of enemies, and a system for storing player progress.

Я, Фурсов Данило Сергійович, студент гр. ПЗПІ-20-7, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Ігровий програмний застосунок у жанрі 3D Rogue-like шутер від третьої особи», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

Жанри Rogue-like та шутер від третьої особи є захоплюючими напрямками в сучасній ігровій індустрії. Rogue-like характеризується процедурно згенерованими рівнями, перманентною смертю персонажа та високим рівнем складності, що забезпечує високу реіграбельність і постійний виклик гравцям. Шутери від третьої особи, у свою чергу, надають динамічний бойовий досвід та інтенсивну тактичну боротьбу, де гравці спостерігають за персонажем з перспективи третьої особи.

Поєднання жанрів Rogue-like та шутер від третьої особи дозволяє створити глибокий ігровий досвід, який задовольнить попит на складність і реіграбельність, а також на динамічні та насичені події. Поєднання цих жанрів дає можливість створити унікальний ігровий досвід, який поєднує в собі швидкоплинні та захоплюючі бойові дії зі стратегією і тактичним плануванням. Ігри в цьому стилі вимагають від гравців не лише вміння швидко реагувати на події в грі, а й ефективно керувати ресурсами та розробляти стратегії для подолання перешкод.

Стилістика кіберпанку є одним із популярних напрямків у сучасній культурі та ігровій індустрії. Вона поєднує в собі тематику майбутнього з високим рівнем технологічного розвитку та занепадом суспільства. Кіберпанк часто описує світ, де високі технології співіснують з соціальною та економічною нерівністю, а також з кримінальними елементами. Візуальні особливості кіберпанку включають неон, темні тони, контрастні кольори та футуристичні міські пейзажі.

Темою кваліфікаційної роботи є розробка ігрового програмного застосунку у жанрі 3D Rogue-like шутер від третьої особи в стилістиці кіберпанку. Основне завдання такої гри – створити унікальний і захоплюючий ігровий досвід, поєднуючи швидкоплинні бойові дії шутера від третьої особи з процедурно згенерованими рівнями та механіками жанру Rogue-like. Гра виконана в стилістиці кіберпанку, яка поєднує футуристичний дизайн, темні тони та неонові акценти, що додає глибини та унікальної атмосфери ігровому світу.

Метою роботи є розробка комп’ютерної гри, яка має бойову систему, процедурну генерацію рівнів, штучний інтелект ворогів, систему зберігання

прогресу гравця та буде виконана у стилістиці кіберпанку. Для розробки продукту використовувався ігровий рушій Unreal Engine 5 та мова програмування Blueprint Visual Scripting. В процесі розробки використовуються технології освітлення Lumen та оптимізації графіки Nanite.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

За останнє десятиліття ринок комп'ютерних ігор зазнав значного зростання та розвитку, що сприяло появі нових жанрів і категорій ігор. Серед них дуже велику популярність мають такі жанри, як шутер та Rogue-like.

Rogue-like (читається «роуґ-лайк»), тобто rogue-подібні ігри – це піджанр рольових відеоігор, визначними особливостями якого є випадкове, процедурне створення рівнів, перманентна смерть персонажа у випадку поразки [1].

Rogue-like – це жанр ігор, який отримав свою назву від класичної комп'ютерної гри "Rogue", випущеної в 1980 році (рисунок 1.1). "Rogue" стала першопрохідцем у впровадженні процедурної генерації рівнів, де кожен сеанс гри був унікальним, завдяки різноманітним випадковим подіям і розміщенню елементів. Гра також включала перманентну смерть персонажа, що значило, що кожна спроба гри була незалежною і потребувала обережності та планування.

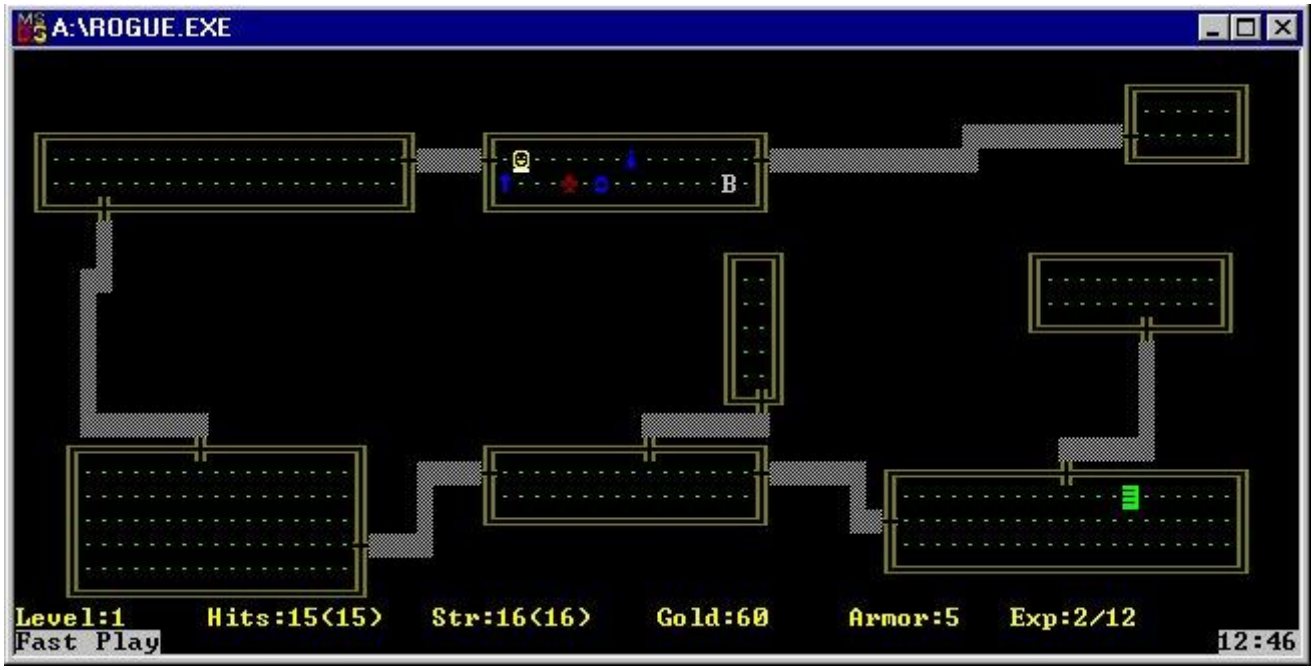

Рисунок 1.1 – гра Rogue (1980 рік)

Історія "Rogue" та її спадок призвели до появи цілого ряду ігор, відомих як "Rogue-like", які наслідують основні риси цієї класичної гри. Сучасні Rogue-like ігри використовують ці концепції, додаючи нові механіки та можливості. Це можуть бути різні типи процедурної генерації рівнів, випадкові події та динамічні зміни в грі, які тримають гравця в стані невідомості та стимулюють стратегічний підхід.

Шутер – це жанр, в якому гравець контролює персонажа та бере участь у бойових діях з використанням вогнепальної або іншої зброї. У шутерах від першої особи гравці бачать світ через очі персонажа, що надає відчуття глибокого занурення в гру. У шутерах від третьої особи гравець бачить свого персонажа ззаду, що дозволяє краще контролювати рухи та позиціювання у просторі.

Шутери можуть пропонувати різноманітні види зброї, від класичних вогнепальних видів до футуристичних та фантастичних. Часто шутери містять різні режими гри, такі як одиночні кампанії, кооперативні місії та багатокористувацькі режими. Важливою особливістю жанру є акцент на швидкій реакції, точності та вміння вести бойові дії ефективно.

Стилістика кіберпанку – це популярний напрямок в ігровій індустрії та поп-культурі, який поєднує футуристичний дизайн з елементами занепаду і соціальної нерівності. Головними рисами кіберпанку є використання неонових кольорів, темних тонів і контрастного освітлення, а також створення віртуальних світів, де високі технології переплітаються з руйнуванням суспільства. У таких іграх часто представлені мегаполіси з хмарочосами, переповненими вулицями та підпільними організаціями. Кіберпанк також зосереджується на темах, пов'язаних з інтеграцією людини і машини, штучним інтелектом та боротьбою за свободу в умовах контролю та цифрової реальності. Цей стиль додає атмосфері гри унікального вигляду, занурюючи гравців у світ майбутнього, сповнений таємниць, небезпек та можливостей.

Поєднання жанрів Rogue-like та шутер від третьої особи, а також стилістики кіберпанку, дозволяє створити ігри, що забезпечують захоплюючий досвід обох жанрів і цікавий візуальний дизайн. Гравці мають можливість досліджувати процедурно згенеровані рівні, стикатися з непередбачуваними викликами, боротися

з різними видами ворогів, використовуючи широкий спектр зброї та здібностей. Це створює унікальний ігровий процес, який утримує гравців у стані постійної напруги та захоплення, доповнений атмосферою кіберпанку, що занурює гравців у футуристичний, насичений високими технологіями світ.

1.2 Аналіз конкурентів

У рамках аналізу конкурентів розглянемо чотири гри, які відносяться до жанрів шутеру та Rogue-like, а також враховують стилістику кіберпанку або подібні елементи.

Cyberpunk 2077 – відеогра в жанрі action RPG в стилі кіберпанку, розроблена польською студією CD Projekt RED [2]. Гравці поринають у світ майбутнього, де високі технології та соціальні конфлікти переплітаються в місті Night City. Гра пропонує багатий світ із відкритим простором для дослідження, вражаючу графіку та глибокий сюжет (рисунок 1.2).

Рисунок 1.2 – Cyberpunk 2077

Переваги:

- вражаючий світ кіберпанку з деталізованою графікою та глибоким дизайном;
- великий вибір зброї та бойових механік для гравця;
- сильний сюжет з розгалуженими діалогами та численними виборами.

Недоліки:

- деякі проблеми з продуктивністю та стабільністю, особливо на старих платформах;
- процес проходження локацій є дуже лінійним;
- майже повністю відсутні елементи шутеру від третьої особи.

Deus Ex: Human Revolution – відеогра жанру стелс-екшен/RPG, розроблена компанією Eidos Montreal і випущена компанією Square Enix в 2011 році [3]. Гравці беруть на себе роль Адама Дженсена, агента, який бореться з корупцією в світі, де кібернетичні модифікації є звичайним явищем. Гра пропонує можливості для стелсу, дослідження світу та взаємодії з персонажами (рисунок 1.3).

Рисунок 1.3 – Deus Ex: Human Revolution

Переваги:

– глибока історія в стилістиці кіберпанку з цікавими персонажами;
– різноманітні можливості для підходу до бойових ситуацій, включаючи стелс і злом;
– сильна взаємодія з оточенням та елементами RPG.

Недоліки:

– основна увага приділена саме сюжету та діалогам, ніж бойовій системі;
– в бойовій системи основною частиною є саме стелс та непомітність, аніж активні бойові дії;
– елементи шутеру від третьої особи присутні в дуже малій кількості.

Enter the Gungeon — відеогра в жанрі bullet hell, розроблена компанією Dodge Roll і випущена компанією Devolver Digital [4]. Гра характеризується високою реіграбельністю, інтенсивним геймплеєм та випадковим розміщенням рівнів, що додає викликів і непередбачуваності (рисунок 1.4).

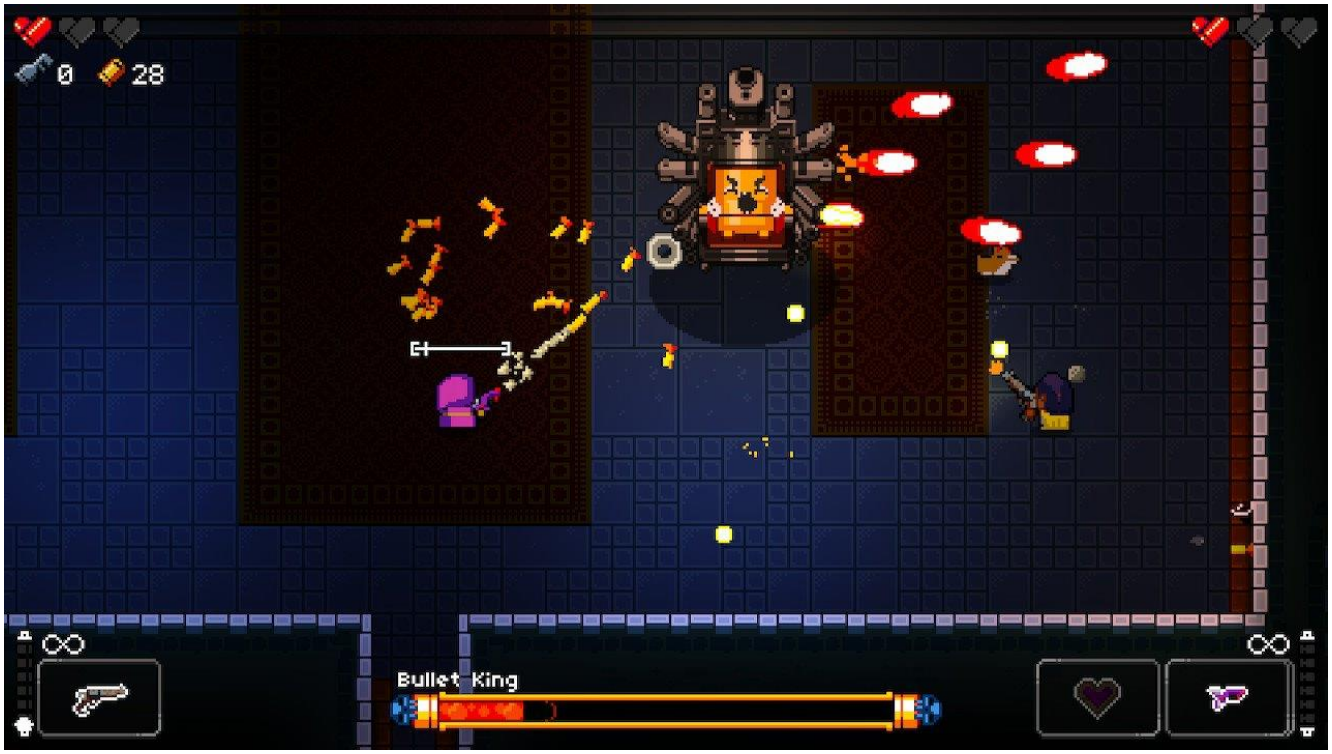

Рисунок 1.4 – Enter the Gungeon

Переваги:

– висока реіграбельність завдяки процедурно згенерованим рівням;
– великий вибір зброї та цікавих здібностей;
– інтенсивний ігровий процес та бойові механіки, які тримають гравця в напрузі.

Недоліки:

– відсутність розширеної історії або глибоких сюжетних елементів.
– візуальний стиль дуже простий у порівнянні з кіберпанком;
– обмежена можливість розвитку персонажа.

Dead Cells — французька відеогра в жанрі метроїдванії, видана незалежною ігровою студією Motion Twin та випущена 7 серпня 2018 року [5]. Це Rogue-like гра, яка пропонує гравцям досліджувати випадково згенеровані рівні, боротися з ворогами та покращувати свої здібності. Гра відома своїм швидким та інтенсивним геймплеєм, а також складністю і різноманітністю можливостей розвитку персонажа (рисунок 1.5).

Рисунок 1.5 – Dead Cells

Переваги:

- висока реіграбельність завдяки процедурно згенерованим рівням;
- великий вибір зброї та цікавих здібностей;
- інтенсивний ігровий процес та бойові механіки, які тримають гравця в напрузі;
- різноманітність здібностей і можливостей для розвитку персонажа.

Недоліки:

- високий рівень складності може не підходити для менш досвідчених гравців;
- відсутність розгорнутого сюжету або глибоких наративних елементів;
- майже відсутні елементи шутеру.

Отже, проаналізувавши ігрові застосунки конкурентів, було визначено, які саме механіки слід додати до нашого застосунку та до яких частин продукту слід приділити увагу.

1.3 Виявлення та вирішення проблем

Сьогоднішній ринок комп'ютерних ігор переповнений різноманітними жанрами та механіками, однак попит на інноваційні та захоплюючі ігри не зменшується. Гравці постійно шукають нові виклики та враження, які можуть надати свіжі жанри та унікальні комбінації ігрових механік. Жанр 3D Rogue-like шутеру від третьої особи поєднує в собі інтенсивний екшн, процедурну генерацію контенту та високу реіграбельність, що робить його цікавим для різних категорій гравців. Розробка ігрового програмного застосунку в цьому жанрі дозволить створити цікавий ігровий продукт, який може стати популярним на ринку комп'ютерних ігор та привабити широку аудиторію гравців різних вікових груп і вподобань.

1.4 Постановка задачі

Основним завданням даної роботи є розробка ігрового програмного застосунку у жанрі 3D Rogue-like шутер від третьої особи, що має на меті задовольнити попит гравців на інноваційний ігровий досвід, що поєднує в собі механіки Rogue-like і шутера від третьої особи.

В процесі розробки ігрового програмного застосунку у жанрі 3D Rogue-like шутер від третьої особи в стилістиці кіберпанку повинні бути розроблені наступні компоненти:

а) бойова система;

1) потрібно надати можливість персонажу гравця рухатись;
2) персонаж повинен мати змогу стріляти із різної зброї та використовувати активні предмети;
3) гравець повинен мати змогу взаємодіяти із об’єктами оточення;
4) персонаж повинен мати здоров’я та мати можливість померти;

б) штучний інтелект ворогів;

1) вороги повинні мати змогу бачити гравця та робити рішення на основі навколишньої інформації;
2) вороги повинні мати змогу атакувати гравця та ховатися за укриттям;
3) вороги повинні мати змогу реагувати на дії гравця, або навіть мати змогу втратити його с поля зору;

в) генерація рівнів;

1) локації повинні генеруватися випадково та складатися з кімнат та коридорів;
2) положення кімнат та коридорів не повинні пересікатися та заважати одне одному;
3) кожна кімната повинна мати випадково згенерований інтер’єр;

г) економіка;

1) гравець буде мати гроші, які він може отримувати та витрачати під час проходження;
2) зароблені під час проходження гроші після перемоги гравець може витратити на купівлю нових персонажів;
3) на локаціях гравець повинен мати можливість знаходити нову зброю та предмети для персонажа;

д) інтерфейс;

1) гравець повинен мати змогу бачити здоров’я свого персонажа;
2) гравець повинен мати змогу бачити зброю та предмети, якими володіє його персонаж;
3) гравець повинен мати змогу бачити своє положення на мані-мапі;
4) гравець повинен мати змогу бачити здоров’я ворогів, скільки шкоди він наносить ворогам та вони йому;
5) гравець повинен мати змогу бачити інформацію про об’єкти оточення, з якими він може взаємодіяти;

е) система зберігання;

1) згенерована локація, та всі зміни, заподіяні гравцем повинні зберігатися;
2) характеристики персонажа повинні зберігатися;
3) прогрес в середині однієї кімнати зберігатися не буде, буде збережено лише факт завершення кімнати гравцем.

Таким чином, в результаті проведення постановки задач, було визначено основний функціонал та основні механіки, які мають бути присутніми в ігрового застосунку.

1.4.1 Цільова аудиторія

Основна цільова аудиторія гри - це гравці віком 16+. Дану аудиторію має залучати реалізм, наявність насильства та необхідність швидко приймати рішення. Гра має вікове обмеження 16+ через наявність насильства та необхідність

стратегічного та тверезого мислення під час боїв. Дана гра охоплює дуже широку аудиторію, оскільки вона об’єднує дуже популярні жанри.

1.4.2 Монетизація

Монетизація ігрового програмного застосунку буде здійснюватися за рахунок одноразової купівлі гри. Гра буде доступна для придбання на різних платформах за встановленою фіксованою ціною. Вибір одноразової оплати за гру замість внутрішньоігрових транзакцій надає гравцям можливість зосередитися на самому ігровому досвіді, не відволікаючись на інші фінансові аспекти. Це підвищує довіру гравців до розробників і створює більш позитивне враження від гри.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Функціональні вимоги до ігрового застосунку

Враховуючи постановку задачі та переваги і недоліки розглянутих аналогів, встановимо наступні функціональні вимоги до ігрового застосунку:

а) бойова система;

1) персонаж гравця повинен мати змогу рухатись, бігати, присідати, пригати та взбиратися на уступи;
2) персонаж гравця повинен мати змогу стріляти із різної зброї, перезаряджати її, заміняти на іншу, також використовувати активні предмети та заміняти їх;
3) гравець повинен мати змогу підбирати зброю і предмети на локації, а також взаємодіяти із автоматами;
4) персонаж повинен мати здоров’я, щит, що поступово відновлюється, та мати можливість померти, якщо здоров’я впаде до нуля;

б) штучний інтелект ворогів;

1) вороги повинні мати змогу бачити гравця та знати його останнє положення, якщо не бачать його;
2) вороги повинні мати змогу атакувати гравця;
3) вороги повинні мати змогу аналізувати навколишнє оточення та ховатися за укриттям так, щоб гравцю було складно атакувати їх, та вони мали змогу його;
4) вороги повинні мати змогу реагувати на атаку зі сторони гравця;
5) вороги повинні мати змогу обмінюватися положенням гравця та розпочинати пошук, якщо гравець ховається від них;

в) генерація рівнів;

1) локації повинні генеруватися випадково та складатися з кімнат, з’єднаних коридорами;
2) положення кімнат та коридорів не повинні пересікатися та заважати одне одному;

3) кожна кімната повинна мати по декілька варіантів, поділених на фрагменти, які потім будуть випадково зібрані в цілу кімнату;

г) економіка;

1) гравець буде мати гроші, які він може отримувати від вбивства ворогів та з предметів, та витрачати під час проходження на купівлю зброї та предметів, або на взаємодію з автоматами;
2) після перемоги зароблені під час проходження гроші переносяться в головне меню, де гравець може купити нових персонажів;
3) на локаціях гравець повинен мати можливість знаходити нову зброю та предмети для персонажа, які випадково розміщуються на локації під час генерації;

д) інтерфейс;

1) гравець повинен мати змогу бачити здоров’я та щит свого персонажа;
2) гравець повинен мати змогу бачити зброю, кількість патронів та активні і пасивні предмети, якими володіє його персонаж;
3) гравець повинен мати змогу бачити своє положення на мані-мапі, яка відкривається поступово під час проходження;
4) гравець повинен мати змогу бачити здоров’я ворогів, скільки шкоди він наносить ворогам та вони йому, а також направлення, з якого гравцю була заподіяна шкода;
5) гравець повинен мати змогу бачити інформацію про об’єкти оточення, з якими він може взаємодіяти, як наприклад зброя, предмети та автомати;

е) система зберігання;

1) згенерована локація, та всі зміни, як наприклад підібрані та не підібрані предмети, повинні зберігатися;
2) усі характеристики персонажа, як наприклад його здоров’я, бонуси, зброя та предмети, повинні зберігатися;
3) після проходження кімнати буде збережено факт її завершенння, проте прогрес в середині однієї кімнати зберігатися не буде.

Таким чином сформовані функціональні вимоги до частини комплексної кваліфікаційної роботи.

2.2 Нефункціональні вимоги до ігрового застосунку

Даний ігровий програмний застосунок має такі нефункціональні вимоги:

- повинен працювати з мінімальними затримками на середньому або потужному комп’ютері;
- інтерфейс бою повинен бути напівпрозорим, щоб не заважати гравцю під час активної гри;
- зміна основних характеристик персонажа, таких як здоров’я, гроші та патрони, повинні відображатися на екрані;
- гравець повинен бачити, чи потрапив він у ворога, та чи отримує він шкоду від ворогів і звідки;
- локації повинні бути просторі, має бути достатня кількість укриттів для гравця та ворогів;
- завдяки міні-мапі та простій структурі з’єднання кімнат та коридорів гравцю буде складніше заблукати;
- повинно бути інтуїтивно зрозуміле та типове для шутерів керування;
- кнопки для взаємодії з предметами оточення повинні відображатися на екрані.

Таким чином сформовані нефункціональні вимоги до частини комплексної кваліфікаційної роботи.

2.3 Вимоги до середовищ розробки

Під час розробки даного ігрового застосунку буде використовуватися ігровий рушій Unreal Engine 5. Було обрано версію 5.3.2, бо дана версія надає доступ до найновіших функцій, завдяки яким вдасться створити більш якісну графіку та

краще оптимізувати даний продукт. В якості мови програмування було використано вбудовану візуальну мову програмування Blueprints.

Для створення 3d об’єктів оточення буде використовуватися програма Blender3D та Substance Painter. Також 3d об’єкти буде взято із безкоштовних ассетів з Unreal Engine Marketplace. Для створення елементів інтерфейсу буде використовуватися Adobe Photoshop.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

Перед початком розробки ігрового програмного застосунку у жанрі 3D Rogue-like шутер від третьої особи були визначені основні функції гри. Після детального аналізу була створена Use-case діаграма (рис. 3.1).

Рисунок 3.1 - – Use-case діаграма основних функцій зі сторони гравця

Користувачем даного ігрового застосунку є гравець. Користувач, переглядаючи головне меню, може змінити налаштування гри, купити і обрати персонажа та почати нову гру, або завантажити попередню. В самій грі він може рухати персонажа в різних напрямках, бігати, стрибати, пригинатись та взбиратись на уступи. В процесі гри гравець буде знаходити нові предмети та зброю, які він може підбирати. Персонаж може активувати ефект підібраного предмета, стріляти зі зброї, перезаряджати її, прицілюватись та змінювати одну зброю на іншу. Також гравець може переглядати міні-мапу, щоб знати своє положення та бачити прогрес,

або поставити гру на паузу. Із меню паузи гравець може перейти в налаштування, перейти назад в головне меню, або повністю вимкнути гру. Якщо персонаж гравця помер, то гравцю потрібно перейти назад в головне меню.

Ворог в свою чергу має можливість рухатися, атакувати гравця зі зброї та ховатися за укриттями. Вони з’являються на локації коли гравець заходить в нову кімнату, а вмирають якщо гравець заподіяв їм достатньо шкоди.

3.2 Вибір архітектури та рушія

Перш ніж ми почнемо розробку ігрового програмного застосунку, потрібно визначитися із майбутньою архітектурою проекту та ігровим рушієм, в якому буде розроблюватися гра.

На сьогоднішній день існує два відомих та потужних ігрових рушіїв, це Unity та Unreal Engine 5. Обидва рушії вже довгий час користуються популярністю серед розробників ігор, мають великий функціонал та свої унікальні особливості.

Unity — багатоплатформовий інструмент для розроблення відеоігор і застосунків, і рушій, на якому вони працюють. Створені за допомогою Unity програми працюють на настільних комп'ютерних системах, мобільних пристроях та гральних консолях у дво- та тривимірній графіці, та на пристроях віртуальної чи доповненої реальності. Застосунки, створені за допомогою Unity, підтримують DirectX та OpenGL [6].

Unity відомий своєю простотою в освоєнні та широким спектром можливостей для створення ігор різного жанру. Він є популярним в середовищі інді-розробників та має широку підтримку спільноти. Даний рушій дуже часто використовують для розробки стилізованих, мобільних або 2d-ігор.

Unreal Engine — інструмент для створення ігор і сцен із використанням 3D-моделей, розроблений компанією Epic Games. Ця програма призначена для створення ігор і симуляцій [7].

Unreal Engine відомий своєю потужною графікою та фізикою, що робить його ідеальним вибором для проектів, де важливий реалізм та деталізація. Він широко

використовується для розробки реалістичних ігор для ПК, консолей та віртуальної реальності.

Зважаючи на вищезазначені фактори для нашого ігрового програмного застосунку, де важливі реалізм та якість графіки, було обрано Unreal Engine 5 як основний інструмент розробки.

Тепер розглянемо можливі архітектури для розробки ігрового програмного застосунку:

- Моде́ль–вигляд–контро́лер (MVC) – архітектурний шаблон, який використовується під час проєктування та розробки програмного забезпечення [8]. Він складається з трьох складових, моделі, виду та контролера. Модель представляє бізнес-логіку та дані програми. Вид (відповідає за візуалізацію даних. Контролер керує взаємодією між моделлю та видом.
- Компонентно-орієнтована архітектура – це архітектура, яка базується на розбитті програмного забезпечення на окремі компоненти, які можуть бути розроблені та підтримуватися незалежно один від одного.
- Об'єктно-орієнтована архітектура – це підхід до розробки програмного забезпечення, в якому програма організована як набір взаємодіючих об'єктів, які представляють реальні або абстрактні об'єкти.

Однією з особливостей ігрового рушія Unreal Engine 5 є те, що він використовує комбінацію компонентно-орієнтованої та об’єктно-орієнтованої архітектур.

Актор - це будь-який об'єкт, який можна розмістити на рівні, наприклад, камера, статична сіть або місце старту гравця. Актори підтримують 3D-трансформації, такі як переведення, обертання та масштабування [9]. Кожен актор є екземпляром класу, який визначає його поведінку та властивості. Ключовою особливістю акторів є можливість створювати дочірніх акторів, які будуть успадковувати змінні та функції батьківського класу, що є дуже зручним при створенні таких об’єктів, як зброя, предмети та персонажі.

В середині кожного актора можуть бути присутні компоненти, які є складовою частиною компонентно-орієнтованої архітектури. Компонент - це частина функціональності, яку можна додати до актора [10]. Використання компонентів дозволяє створювати акторів з різними функціональними можливостями, а також легко маніпулювати їх поведінкою та властивостями.

Таким чином, для розробки ігрового програмного застосунку було обрано рушій Unreal Engine 5, а в якості архітектури було обрано комбінацію компонентно-орієнтованої та об’єктно-орієнтованої архітектур.

3.3 Огляд ігрового циклу

Одним із важливих етапів розробки ігрового програмного застосунку є створення ігрового циклу.

Ігровий цикл – це принцип, згідно з яким геймдизайнери задають головний елемент ігрової механіки, який визначає фундаментальний досвід гравця.

Розглянемо розроблений ігровий цикл, зображений на рисунку 3.2.

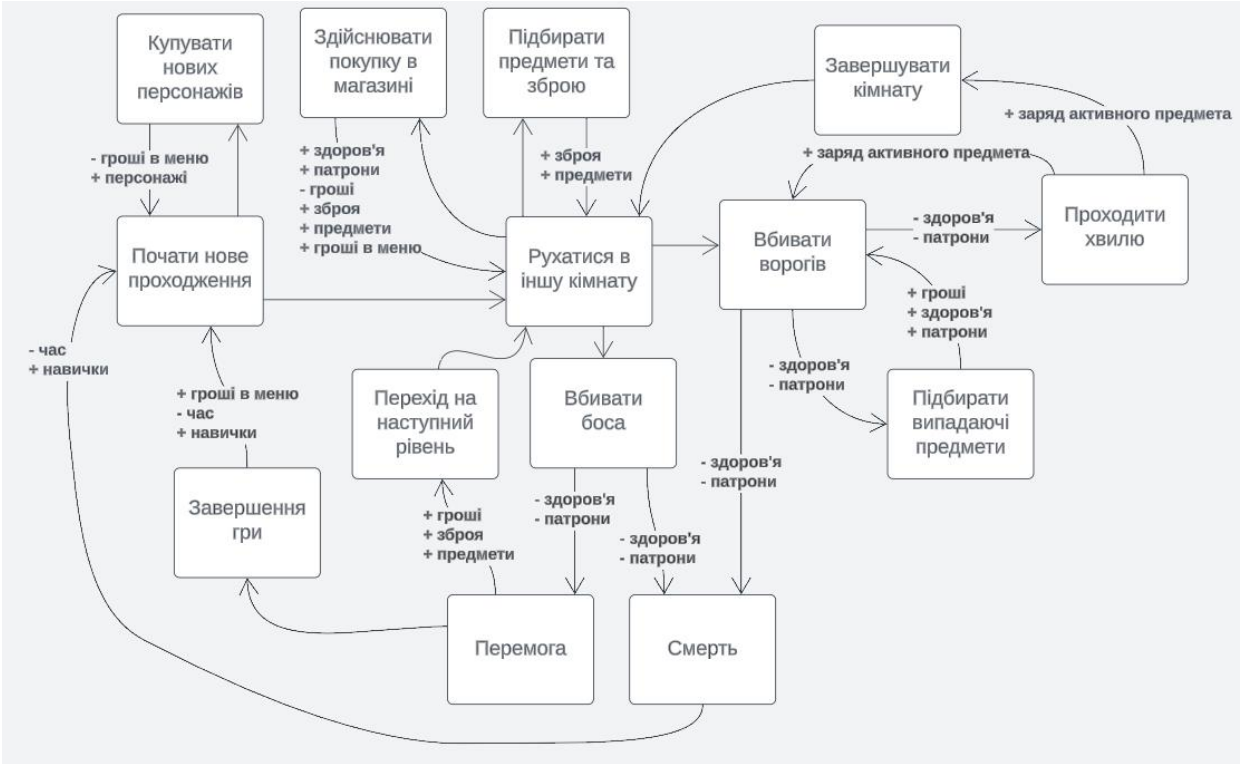

Рисунок 3.2 – Ігровий цикл програмного застосунку

Як можна побачити, даний ігровий цикл складається з дій, пов’язаних одне з одним ланцюгом. Кожна дія приводить гравця до інших дій, при цьому забираючи та даючи різні ресурси. Розглянемо більш детально кожну дію:

- Почати нове проходження: це початкова дія, яку виконую гравець після запуску гри, або завершення попереднього проходження.
- Рухатися в іншу кімнату: основна дія для прогресу в грі. Рівні в грі складаються із кімнат і для проходження рівня гравцю потрібно знайти кімнату з босом та подолати його. Всього є декілька типів кімнат: кімната з боєм, з призом, з магазином та з босом.
- Вбивати ворогів: якщо гравець потрапив у кімнату з боєм, то йому потрібно вбивати ворогів. Після завершення цієї дії в будь-якому випадку витрачаються патрони та здоров’я, якщо вороги успішно атакують гравця.
- Підбирати випадаючі предмети: після смерті ворогів з них випадають певні предмети, а саме аптечки, патрони та гроші. Дані предмети автоматично підбираються гравцем на невеликій відстані.
- Проходити хвилю: проходження кімнат з боєм відбувається за хвилями. Якщо гравець подолав усіх ворогів хвилі, то він запускає наступну хвилю ворогів. За кожну пройдену хвилю гравець отримує заряд активного предмета, набравши певну кількість яких його можна активувати.
- Завершувати кімнату: якщо гравець пройде усі хвилі в кімнаті, то він завершить кімнату з боєм і також отримає заряд активного предмета.
- Здійснювати покупку в магазині: якщо гравець потрапить у кімнату з магазином, то він там зможе купити за гроші зброю і предмети, та також зможе активувати автомати, щоб отримати аптечки та патрони.
- Підбирати предмети та зброю: якщо гравець потрапить у кімнату з призом, то він зможе отримати там зброю та предмет безкоштовно.
- Вбивати боса: кімната з босом є фінальною у проходженні рівня. Якщо гравець потрапив в цю кімнату, то йому потрібно вбити баса.

- Перемога: якщо гравець успішно вбиває боса, то в процесу бою він втрачає здоров’я та патрони.
- Перехід на наступний рівень: якщо боса було вбито на не останньому рівні, то відбувається перехід на наступний рівень, а гравець отримує винагороду у вигляді грошей, зброї та предметів.
- Завершення гри: якщо було переможено боса на останньому рівні, то гра завершується та гравець отримує гроші до меню. Таким чином він витрачає час, проте отримує навички для більш кращого наступного проходження.
- Смерть: якщо в процесу битви з ворогами, або з босом гравець втрачає забагато здоров’я, то він помирає та завершує гру не отримавши гроші до меню. Таким чином він витрачає час, проте отримує навички для більш кращого наступного проходження.
- Купити нових персонажів: гроші в меню можна витрати на покупку нових персонажей, для того щоб почати нове проходження та отримати новий досвід.

Отже, у цьому підрозділі були розглянуті варіанти взаємодії для гравця та ворогів під час проходження гри.

3.4 Приклади найцікавіших алгоритмів та методів

В якості найцікавішого алгоритму даного ігрового програмного застосунку розглянемо алгоритм генерації локацій (рисунок 3.3).

Спочатку відбувається зчитування файлу збереження, в якому знаходиться сід випадкової генерації рівня. На його основі визначається структура рівня та кімнати, які будуть розміщені по даній структурі. Потім відбувається розташування кімнат у просторі, розміщується перша кімната, потім друга. Відбувається перевірка на перетин, щоб не було випадків частина однієї кімнати знаходиться всередині іншої. Якщо є перетин, то кімната зсувається в сторону від місця перетину. Дана дія відбувається до тих пір, поки проблема не усунута. Якщо дана дія відбувається дуже

велику кількість разів, то можна зрозуміти, що відбулася конфліктна ситуація, як наприклад коли кімната оточена декількома іншими кімнатами і їй просто не вистачає місця. В таких випадках увесь прогрес розташування кімнат видаляється та починається спочатку. Випадки, коли необхідно видаляти прогрес генерації трапляються рідко та сам алгоритм не є дуже тривалим, тому це не є проблемою.

Рисунок 3.3 – Алгоритм генерації рівнів

Якщо усі кімнати були розміщені успішно, починається розміщення коридорів. Будування коридорів відбувається таким чином, спочатку визначаються місця початку та кінця, тобто входів кімнат, потім за допомогою алгоритму «А зірка» шукається найкоротший шлях між цими входами. Якщо стається випадок, коли такий коридор неможливо побудувати, наприклад якщо шлях перекривається іншим коридором, то в такому випадку треба видалити увесь прогрес розташувань кімнат та коридорів, і почати спочатку.

Якщо розміщення усіх коридорів відбулося успішно, то відбувається спавн кімнат та коридорів на визначені під час генерації місця. Відбувається розміщення гравця та об’єктів локації, таких як зброї, предметів та автоматів. Також в кімнати заноситься інформація про те, чи пройшов їх гравець, якщо це був не початок гри, а продовження.

Таким чином за допомогою даного алгоритму вдається генерувати певну локацію багато разів з одного файлу збереження, при цьому зберігаючи її структуру і вид та прогрес гравця.

3.5 Створення UI/UX

Розглянемо основні елементи інтерфейсу бою (рисунок 3.4). Даний інтерфейс видно під час проходження гри, він інформує гравця про стан персонажа, його зброї та предметів, а також про його прогрес. Інтерфейс складається з таких основних елементів:

- Здоров’я та щит: відображає інформацію про те, скільки персонаж має здоров’я та щита. Щит поступово відновлюється, а якщо він впаде до 0, то посне падати здоров’я. Якщо здоров’я опуститься до 0, то гра завершиться.
- Гроші: відображає кількість грошей, які має гравець.
- Зброя: відображає те, яку зброю має гравець. Всього гравець може мати дві зброї та міняти їх в процесі проходження. Знизу відображається кількість пітронів у обраної зброї, ліве число відображає те, скільки патронів

знаходиться зараз в магазині, а праве – те, скільки патронів всього доступно для даної зброї.

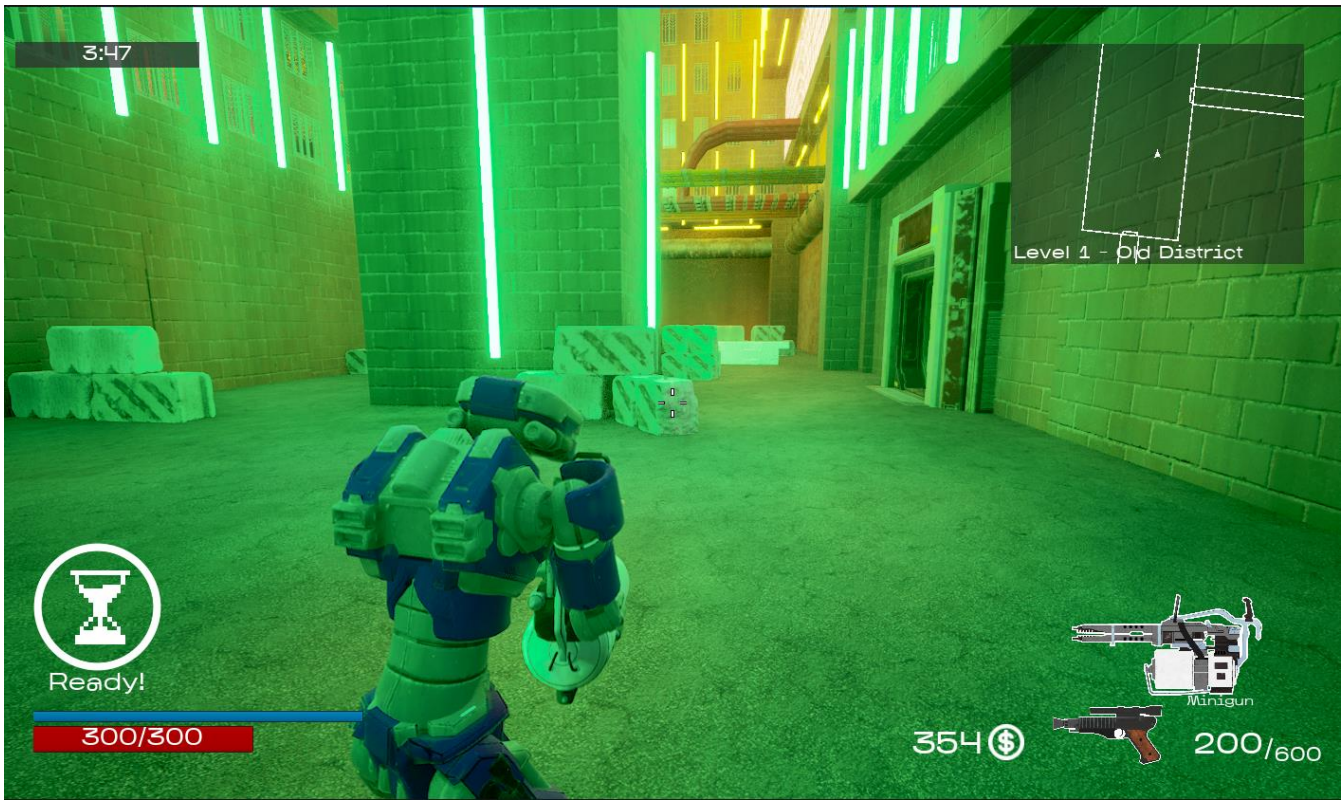

Рисунок 3.4 – Інтерфейс бою

– Предмет: якщо гравець має активний предмет, то інформація про його заряд відображається тут. Заряд може бути відображений як таймер, або як лічильник. Якщо заряд повний то буде написано, що предмет готовий до використання.
– Таймер: відображає інформацію про те, скільки триває дане проходження.
– Перехрестя: допомагає гравцю цілитись зброєю. Дане перехрестя пропадає через час, якщо гравець не цілиться або не стріляє, також воно змінює свій розмір в залежності від швидкості руху, щоб відобразити збільшення розкиду пострілу. Якщо гравець успішно вистрілив в ворога, то на момент перехрестя змінить колір та над ворог відобразить кількість задіяної шкоди.

– Міні-мапа: відображає інформацію про те, де знаходиться гравець на локації. Міні-мапа присутня в двох варіантах, маленькій та великій (рисунок 3.5). Мала міні-мапа видна збоку екрану та центрована над персонажем. Велику міні-мапу можна відкрити та рухати, щоб більш детально побачити усю локацію. Кімнати та коридори відкриваються поступово, щоб гравець не міг бачити, що на нього чекає в наступних кімнатах. Також на мапі відображається предмети, зброя та магазин.

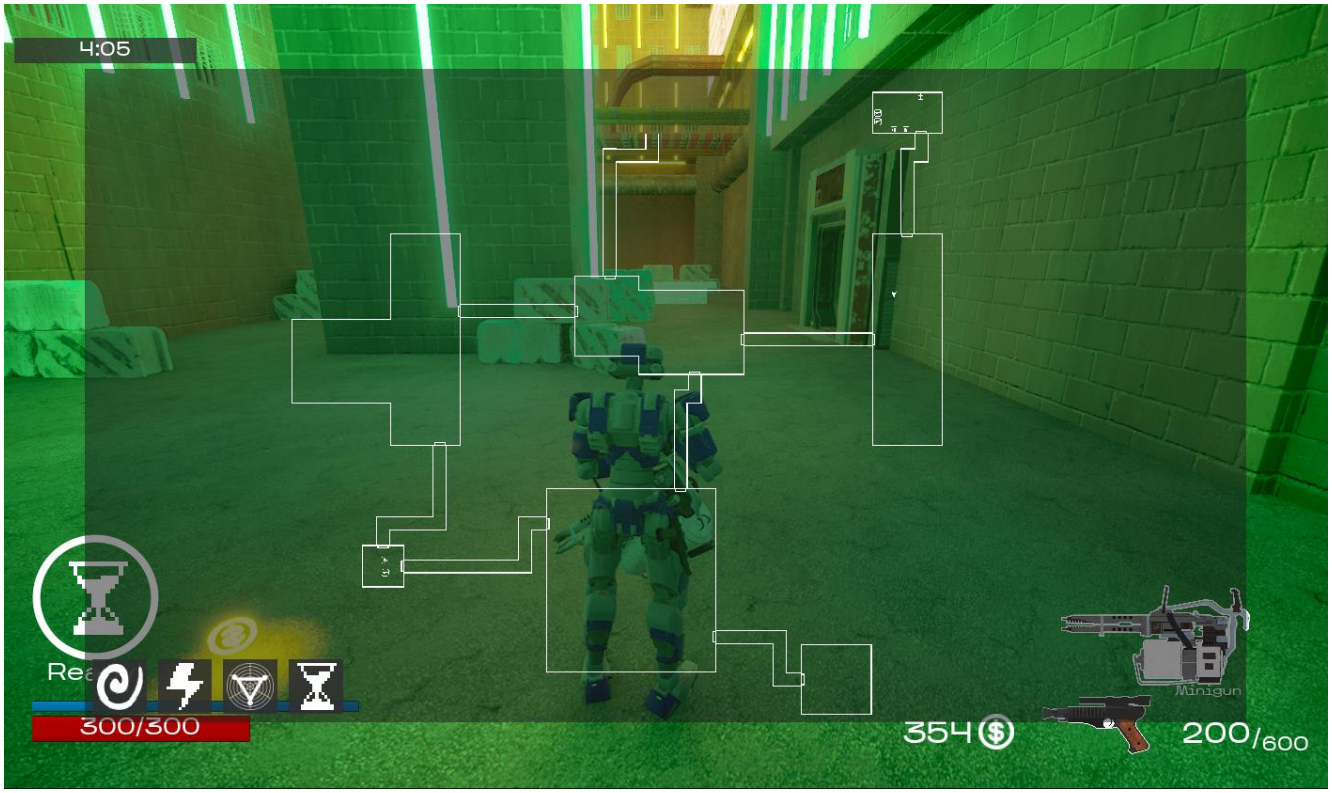

Рисунок 3.5 – Велика міні-мапа

– Список предметів: відображає список усіх підібраних під час проходження предметів. Для того, щоб не створювати окреме меню, даний список відображається в нижній частині великої мані-мапи.
– Індикатор шкоди: надає інформацію про отримання шкоди гравцем (рисунок 3.6). Якщо гравець був атакований ворогом, то даний індикатор повідомить про це червоним ефектом по краях екрану, а також відобразить напрямок ворога колом по середині екрану.

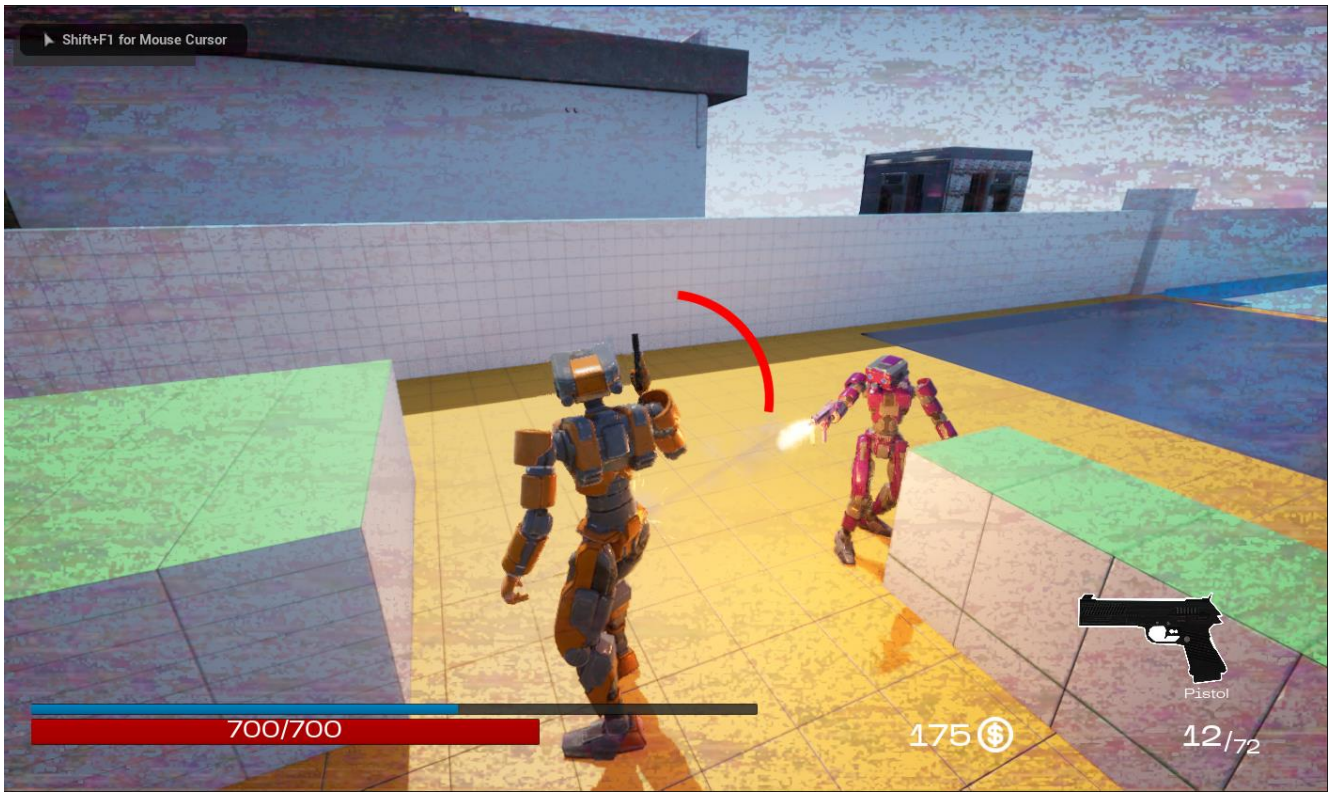

Рисунок 3.6 – Індикатор шкоди

Таким чином, даний інтерфейс буде інформувати гравця про всю необхідну під час проходження інформацію, а також завдяки напівпрозорим елементам інтерфейсу не буде заважати та блокувати огляд під час інтенсивного геймплею.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Персонаж

За основу персонажу було взято офіціальний аддон ALSV4 (рисунок 4.1). ALSV4 – просунута система пересування та нашарування на основі двоногої локомоції, що фокусується на високоякісній анімації персонажів з чутливими переміщеннями [11]. Дане доповнення надає доступ до основи персонажа, в якій реалізовано плавні анімації, проте зовсім немає функціоналу.

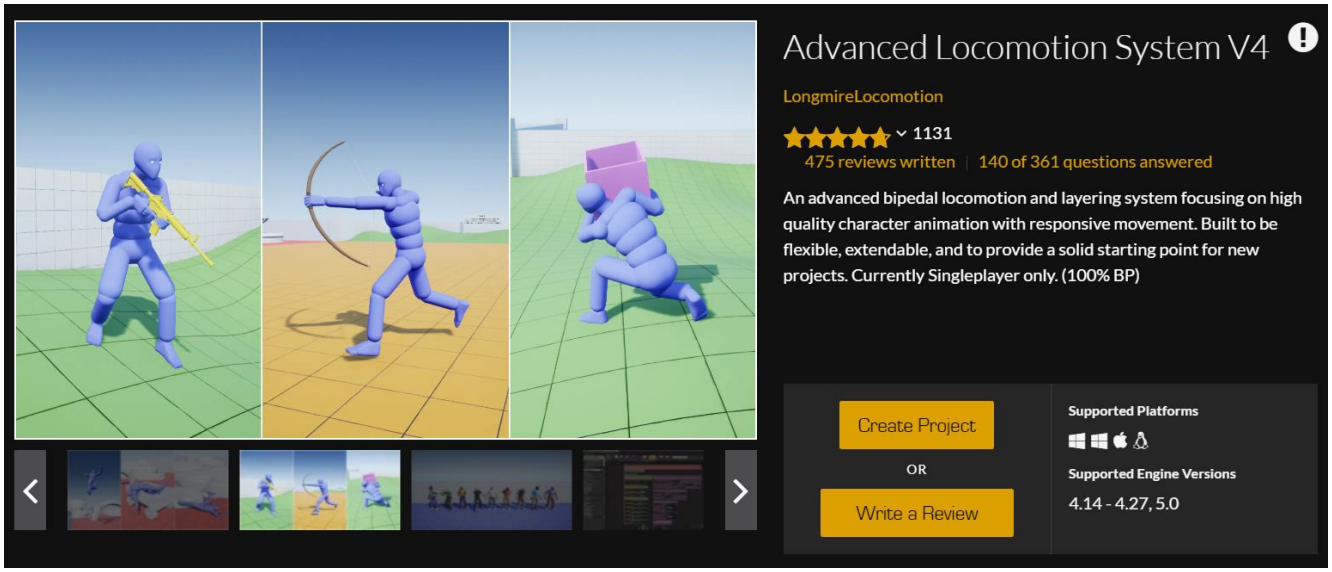

Рисунок 4.1 - Аддон ALSV4

Для того, щоб надати персонажу зовнішній вигляд, за допомогою Blender3D було зроблено модель робота (рисунок 4.2). Після цього моделі було зроблено скелет для анімацій та матеріали. Матеріали було розроблено в програмі Substance Painter.

Blender3D – це вичерпний набір інструментів для моделювання, що дозволяє легко створювати, трансформувати та редагувати моделі [12].

Substance 3D Painter – це інструмент для текстурування 3D-ресурсів [13]. Дана програма спеціалізується в створенні Pbr-текстур, тобто заснованих на фізично коректних матеріалах. Було створено такі текстури, як текстура кольору, нормалей, шорсткості, металічності, світіння, і айді-карта. Айді-карта – це текстура,

яка за допомогою кольору поділяє поверхню моделі на частини, завдяки чому є можливість фарбувати роботів гравця та ворогів в різні кольори.

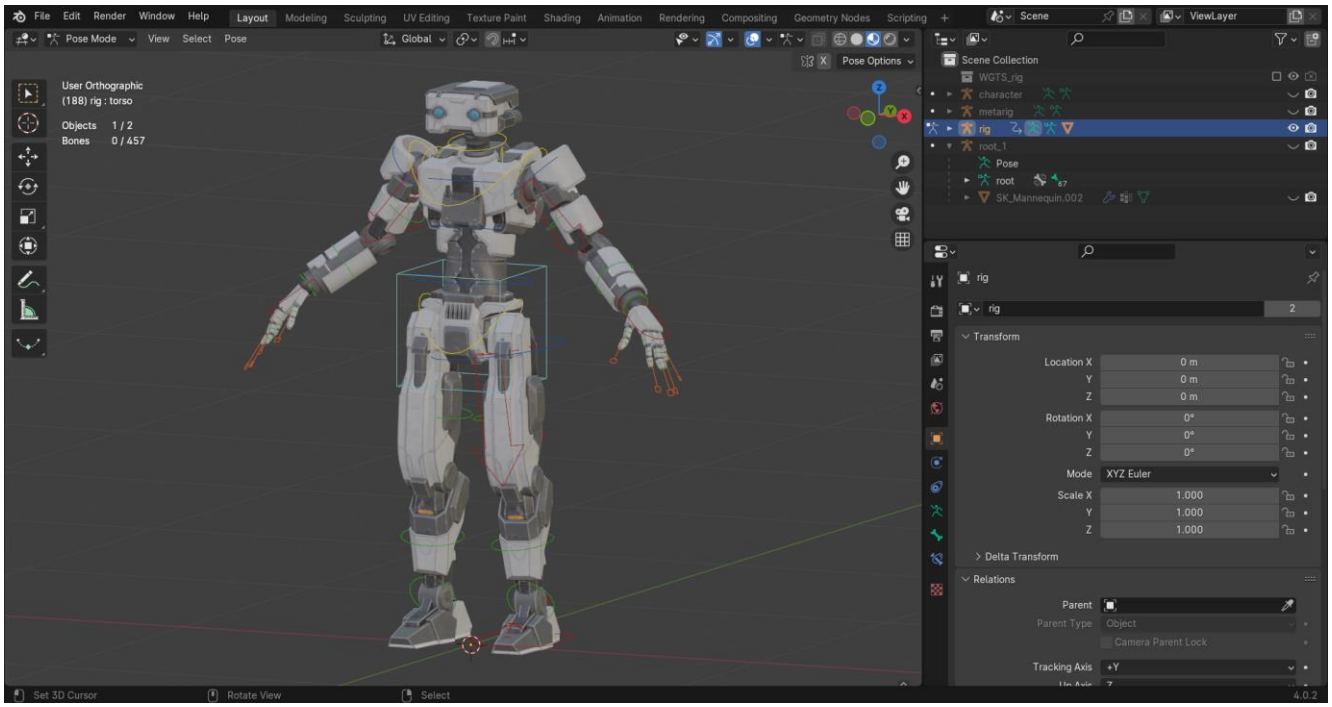

Рисунок 4.2 – модель персонажа в Blender3D

Другою важливою частиною функціоналу персонажа є зброя. Оскільки за функціональними вимогами гравець повинен мати змогу носити дві різні зброї, в акторі персонажа було створено дві компоненти Child Actor Component (рисунок 4.3). Коли персонажа гравця отримує зброю, окремий актор цієї зброї записується в дану компоненту. Екіпірована зброя встановлюється в руки персонажу, а не екіпірована – на спину або на пояс, в залежності від розміру.

Коли зброя встановлюється в руки персонажу, якщо грається анімація тримання зброї двома руками, то може відбутися баг відображення, коли інша рука знаходиться всередині зброї, або в повітрі. Дану проблему було вирішено додаванням сокетів, до яких кріпиться друга рука. Це можливо за допомогою того, що скелет персонажа має в собі інверсивну кінематику, тобто систему, завдяки якій можна встановлювати положення зап’ястя руки, а лікоть під це буде автоматично підлаштовуватися. Код використання інверсивної кінематики для підлаштування положення рук під час тримання зброї відображений на рисунку 4.4.

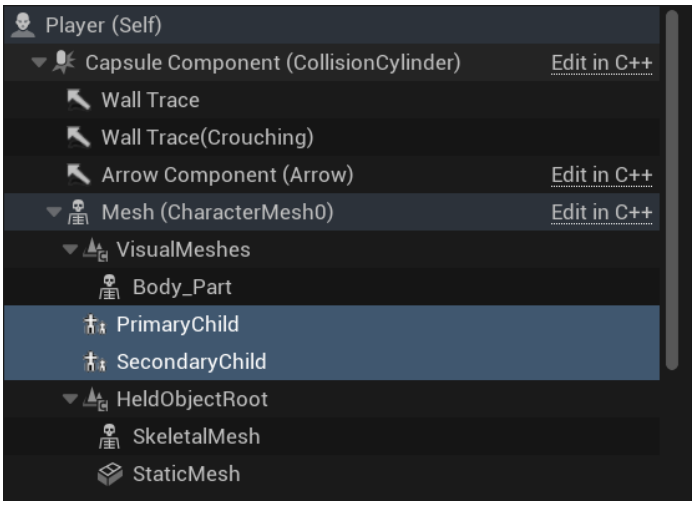

Рисунок 4.3 – Child Actor Component зброї в акторі персонажа

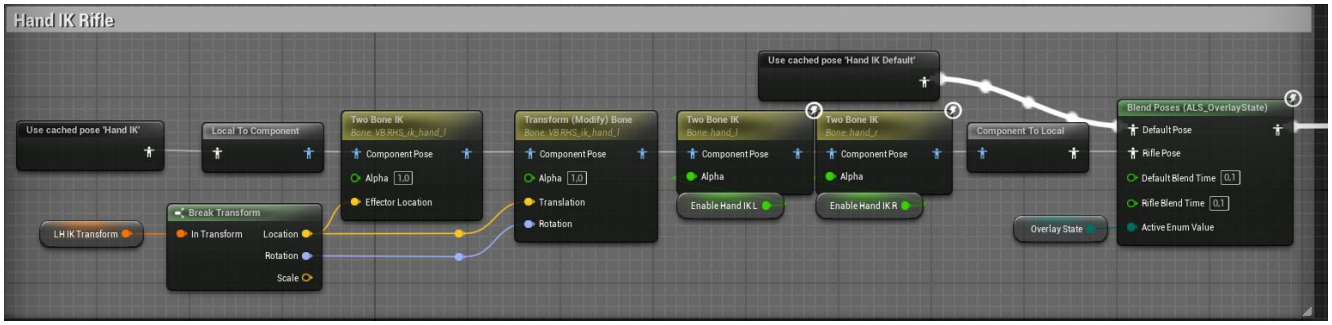

Рисунок 4.4 – Код використання інверсивної кінематики

Дії гравця виконані за допомогою подій InputAction, які встановлені на певні кнопки. Персонаж може виконувати такі дії, як рух, стрибок, стрільба, перезарядка, зміна зброї та інші. Деякі з цих дій є відмінними, та залежать від екіпірованої в даний момент зброї. Розглянемо використання InputAction та залежності від екіпірованої зброї на прикладі функції перезарядки (рисунок 4.5).

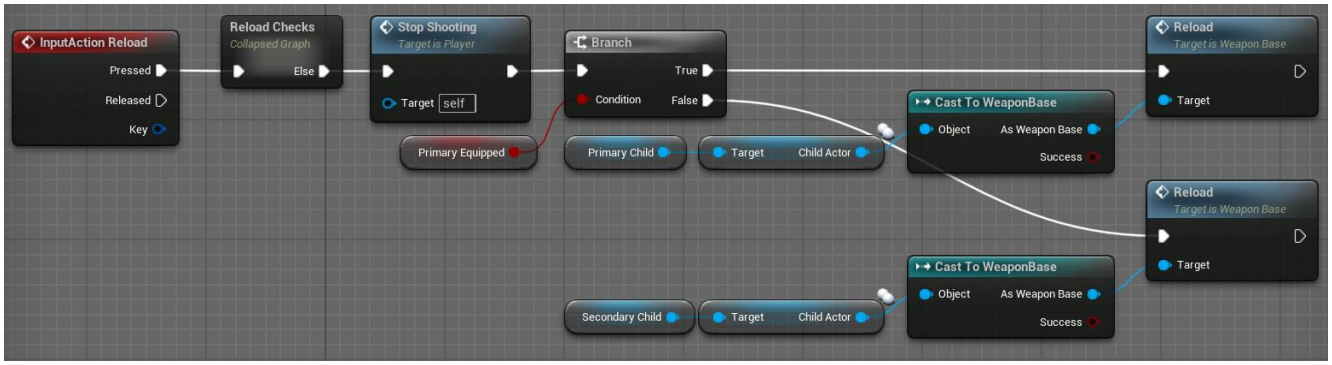

Рисунок 4.5 – Функція перезарядки зброї

Спочатку відбуваються необхідні перевірки, як наприклад перевірка на те, чи персонаж не змінює в даний момент зброю, чи вже перезаряджається, щоб уникнути конфліктів анімацій, потім переривається стрільба, якщо гравець зараз стріляє, після чого виконується внутрішня функція перезарядки у зброї. Завдяки тому, що викликаються саме внутрішні функції екіпірованої зброї, вдається зробити кожну зброю різною та окремо рахувати їх параметри.

4.2 Зброя та предмети

В грі існує велика кількість різноманітної зброї та предметів. Кожен актор зброї або предметів походить від власного батьківського актора, в якому знаходиться увесь функціонал. Вони в свою чергу мають свою модель та параметри (рисунок 4.6), що визначають тип стрільби, шкоду, швидкість, ефекти пострілу та попадання, кількість патронів, назву, опис та інші параметри. Предмети працюють таким же чином, але додатково мають функцію підбирання, активації та деактивації (рисунок 4.7), бо їх принцип роботи повністю уніфікувати не можна.

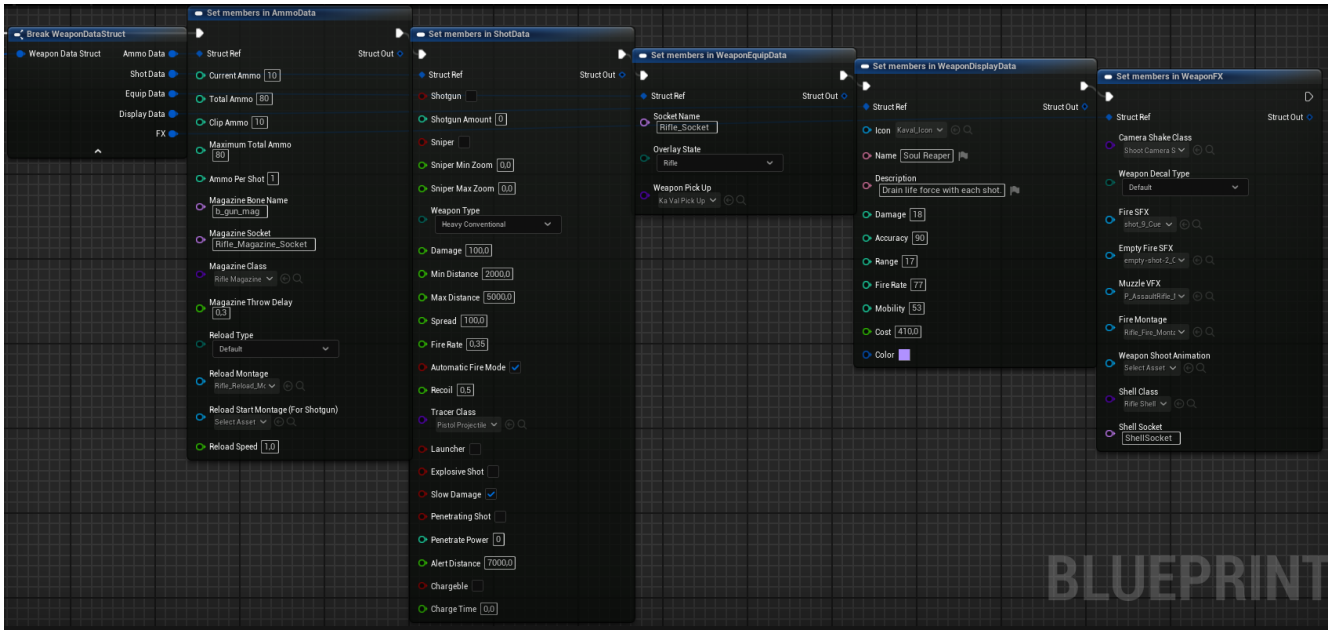

Рисунок 4.6 – Структура параметрів зброї

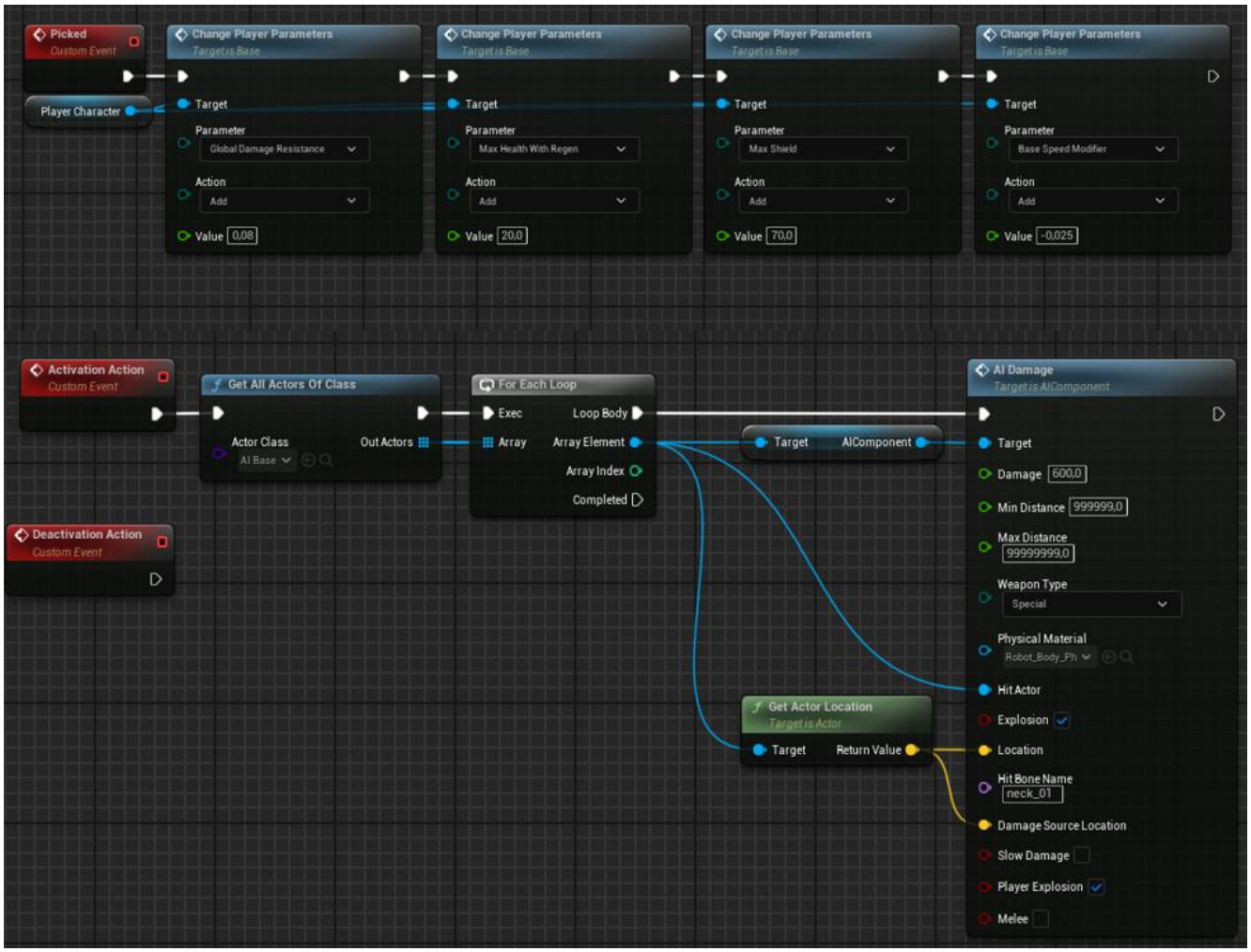

Рисунок 4.7 – Функції підбирання та активації предметів

Предмети бувають двох типів, активні та пасивні. Пасивні предмети виконують свою дію під час підбирання, та зберігаються в масиві, а активні предмети зберігаються як окрема змінна, мають заряд та функцію активації і деактивації. Коли заряд повний, гравець може активувати предмет. Дія предмету триває певний час, після чого виконується функція деактивації. Ця функція також викликається в тому випадку, коли гравець активував предмет, та вирішив екіпірувати інший.

Зброя має 2 основні дії, перезарядка та постріл. Перезарядка буває двох видів, звичайна та дробовика. Коли виконується звичайна перезарядка, то патрони в зброї оновлюються одноразово, відновлюючи повний магазин. Якщо перервати таку перезарядку, то прогрес скидається та потрібно починати спочатку. Коли

виконується перезарядка дробовика, то патрони відновлюються по одному, тому таку перезарядку можна перервати і прогрес зберігається.

Функція пострілу реалізована за допомогою функції LineTrace. Дана функція будує промінь, який реєструю інформацію про точку попадання в об’єкти. Отримавши результат виконання цієї функції, ми виконуємо нанесення шкоди ворогам, якщо потрапили в них, також ми виконуємо відображення ефектів, таких як звук пострілу, ефект польоту кулі, ефект потрапляння на різні поверхні та інші.

Створюючи механіку стрільби, розробники намагаються зробити процес пострілу максимально реалістичним та зручним і зрозумілим водночас. Саме для цього, коли будується LineTrace пострілу, то він будується не фізичним чином, як би куля вилетіла з дула зброї, а таким, як гравець би очікував, в центр екрану, де знаходиться перехрестя прицілу. В цього методу є недолік, якщо гра відбувається від третьої особи, то інколи трапляються випадки, коли траєкторія польоту кулі в центр екрану є неможливою, як наприклад відображено на рисунку 4.8.

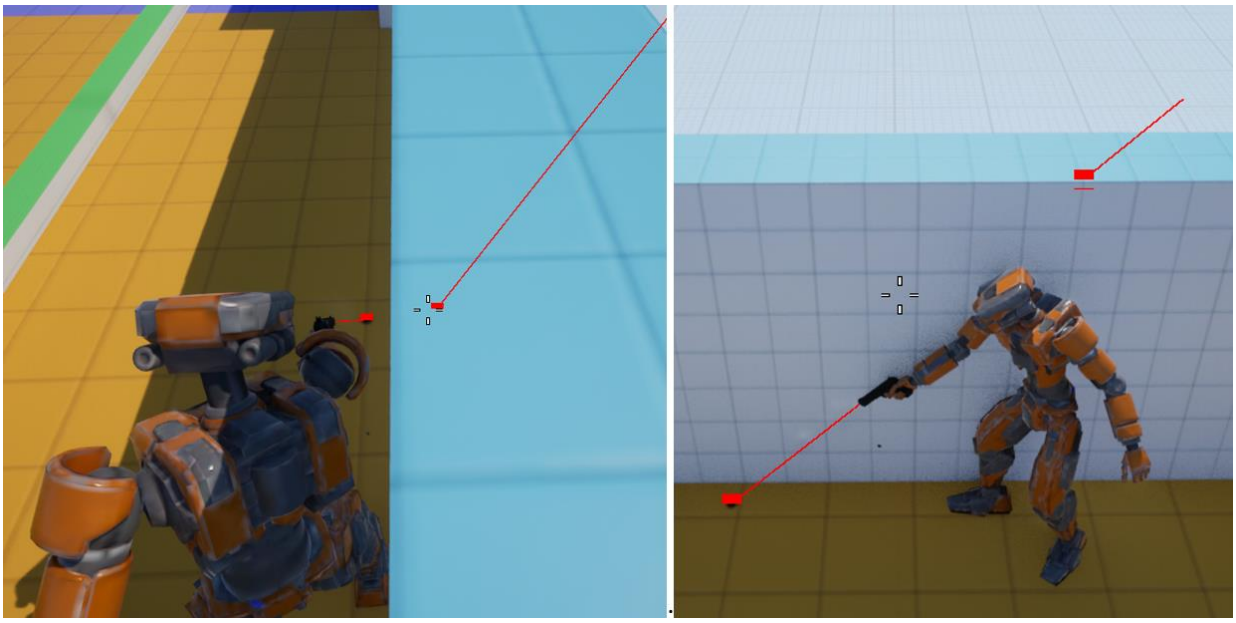

Рисунок 4.8 – LineTrace реалістичний та LineTrace в центр екрану

Для вирішення цієї проблеми під час пострілу будується два промені, один реалістичний, а інший в центр екрану, якщо відхил кута обох променів завеликий, то перевага віддається саме реалістичному променю.

4.3 Інтерактивні об’єкти

Під час проходження гри гравець може знаходити нову зброю та предмети на локації. Об’єкти зброї та предметів розміщуються у сцені за допомогою окремих акторів, які зберігають інформацію про клас зброї або предмету, відображають її (рисунок 4.9) та надають можливість взаємодії для гравця.

Також в грі існує три види автоматів: автомат з аптечками, патронами та банкомат. Автомат з аптечками та з патронами відновлюють здоров’я та патрони. Банкомат дозволяє перенести певну кількість грошей з проходження в меню, без необхідності в завершенні проходження.

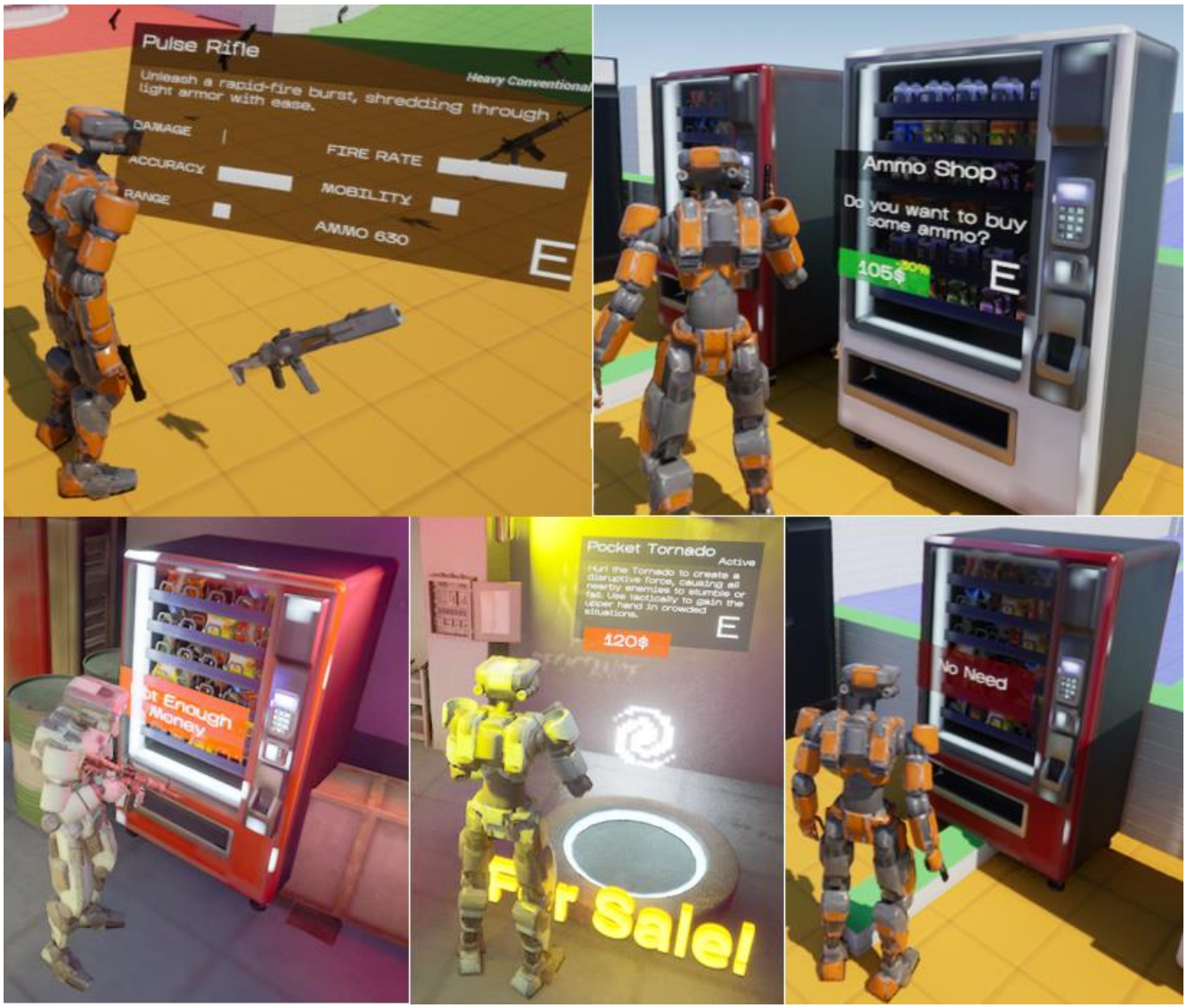

Рисунок 4.9 – Відображення інформації та екран взаємодії інтерактивних об’єктів

Коли гравець взаємодії з акторами предметів, зброї або автоматів, то відбуваються певні перевірки:

– Перевірка на вільну комірку: якщо гравець має лише одну зброю, то нова зброя зараховується в другу комірку, а якщо має дві – то заміняє ту зброю, що була екіпірована. Коли відбувається заміна, на місці підібраної зброї спавниться замінена, з урахуванням кількості патронів. Те саме відбувається із активними предметами, з урахуванням заряду предмету.
– Перевірка на кількість грошей: предмети та зброя можуть бути платними, якщо вони знаходяться в магазині. В таких випадках відбувається перевірка на достатню кількість грошей (рисунок 4.10), з урахування знижки (одного з параметрів персонажа). Якщо грошей недостатньо, то надпис суми має червоний колір, а при спробі взаємодії з'явиться напис, що недостатньо коштів.
– Перевірка на необхідність: автомати аптечок та патронів перевіряють, чи є необхідність в цих ресурсах. Якщо гравець має повне здоров'я, або якщо його зброя має повну кількість патронів, то напис ціни має сірий колір, а при спробі взаємодії з'явиться напис, що немає необхідності.

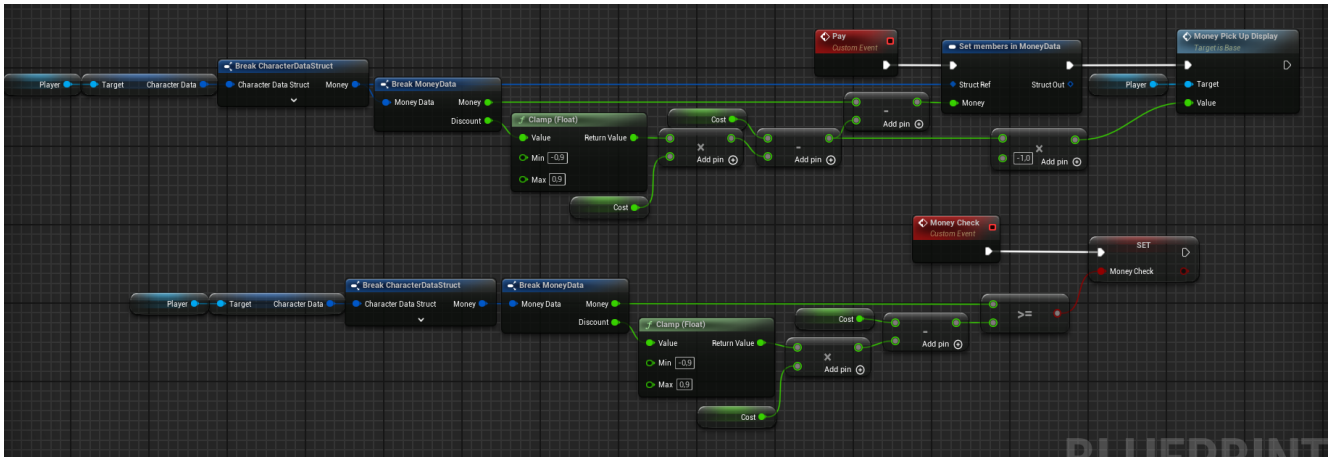

Рисунок 4.10 – Функції перевірки на кількість грошей та оплати

Також окреме слід розглянути випадаючі предмети, такі як патрони, аптечки і гроші (рисунок 4.11). Дані об'єкти випадають після смерті ворогів та відновлюють

певні характеристики персонажа гравця при взаємодії. Вони виконують перевірку на необхідність, таку ж як і автомати, та якщо необхідність є, то вони притягуються до гравця (рисунок 4.12). Взаємодія з цими предметами відбувається автоматично при дотику з персонажем.

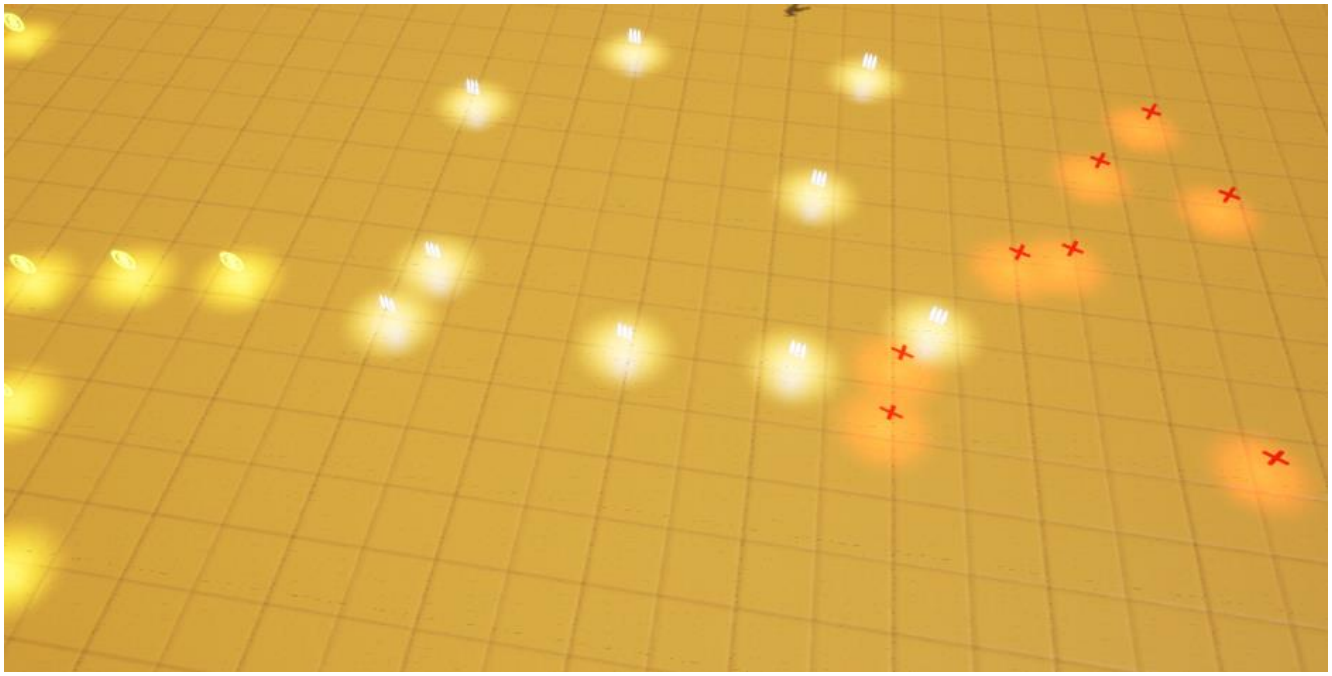

Рисунок 4.11 – Випадаючі предмети

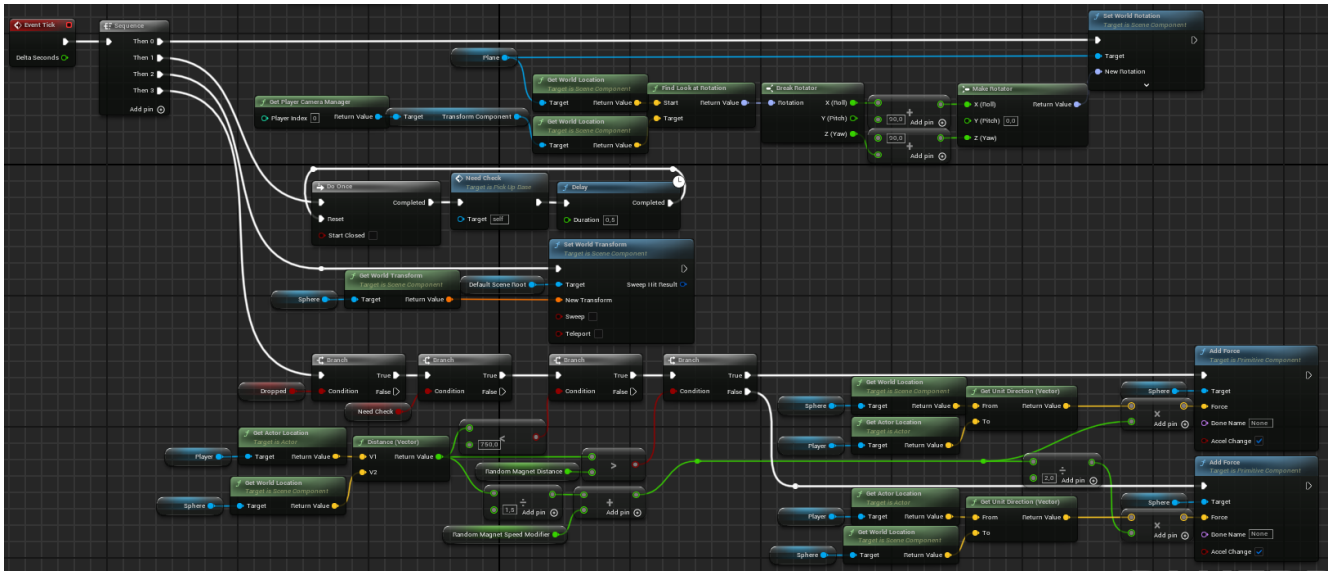

Рисунок 4.12 – Функція притягування випадаючих предметів

Таким чином, за допомогою даних інтерактивних об’єктів граавець має змогу знаходити нові зброю та предмети, а також відновлювати характеристики персонажа під час проходження гри.

4.4 Вороги

Вороги в даній грі є окремими акторами та мають схожий принцип роботи з актором персонажу гравця. Вони можуть рухатися, присідати, змінювати напрямок погляду та стріляти. Актор ворога має лише одну зброю, записану в Child Actor Component, та він не може перезаряджатися.

Логіка поведінки штучного інтелекту ворогів знаходиться в дереві поведінки (рисунок 4.13). Дерево поведінки – це дерево нодів, яке визначає коли і які дії має виконувати штучній інтелект. Дане дерево було побудовано із таких простих нодів, як очікування (Wait), йти (Move To), почати та закінчити стрільбу, подивитися на гравця, прицілитися на гравця, дивитися в сторону руху, присісти за укриття, виглянути з укриття та інші. Комбінуючи такі прості дії вдається зробити різноманітну та плавну поведінку, при цьому маючи можливість припинити певну складову дії для різних причин.

Розроблене дерево складається з чотирьох основних станів:

– Дія: основний стан штучного інтелекту, який працює тоді, коли не працюють усі інші стани. Дії, які штучній інтелект може виконувати в даному стані будуть розглянуті далі.
– Погане укриття: даний стан викликається тоді, коли ворог знаходиться з гравцем по одну сторону укриття та гравець може його бачити, що робить укриття неефективним. В такому випадку ворог припиняє всі свої дії, потім біжить до іншого укриття, та, якщо гравець дуже близько, то починає атакувати гравця. По завершенню шляху до нового укриття даний стан вимикається.
– Довго не видно гравця: якщо жоден ворог не бачив гравця певний час, то усі вороги починають виконувати пошук. Спочатку вони припиняють

вогонь, потім частина ворогів йдуть до останнього місця, де бачили гравця, і, якщо його там немає, то вони починають патрулювання локації.

– Отримав шкоду: якщо ворог під час отримання шкоди ховався за укриттям або просто стояв чи повільно рухався, то при отриманні шкоди він зупинить усі свої дії на короткий час. Даний стан був необхідний для того, щоб вороги, які виглядають з-за укриття та отримають шкоду, не могли швидко повернутися назад в укриття. При цьому, якщо ворог біжав та отримав шкоду, то він не зупиниться. Таким чином, дана механіка зображує реалістичну поведінку та винагороджує гравця за гарне прицілювання.

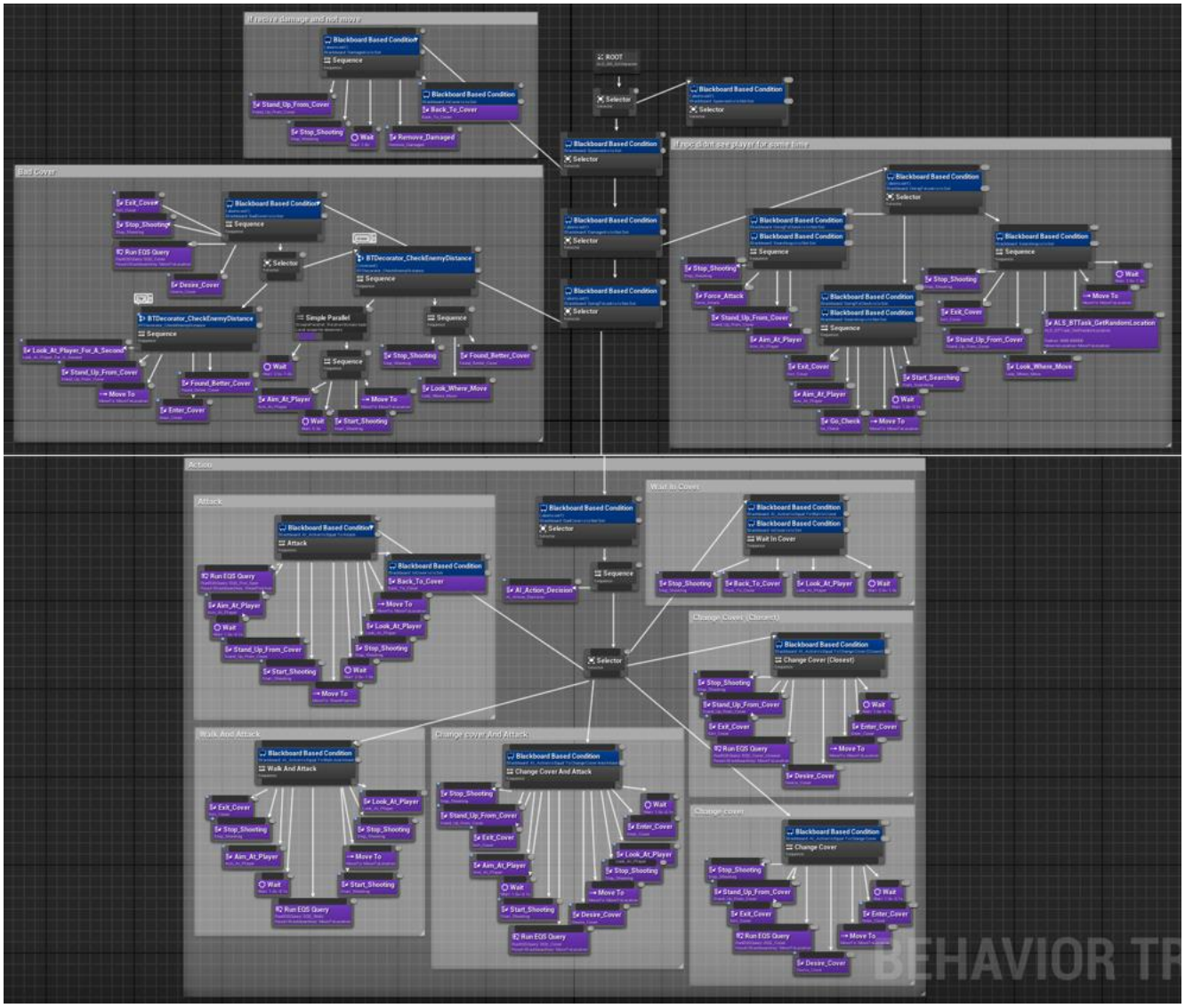

Рисунок 4.13 – Дерево поведінки штучного інтелекту ворогів

Розглянемо більш детально основний стан дії. Якщо не виконується жоден інший стан, то спочатку виконується функція визначення дії (рисунок 4.14). Дана функція враховує попередні дії ворога та поточні дії інших ворогів (рисунок 4.15) і на основі цієї інформації обирає яку саме дію виконувати зі списку.

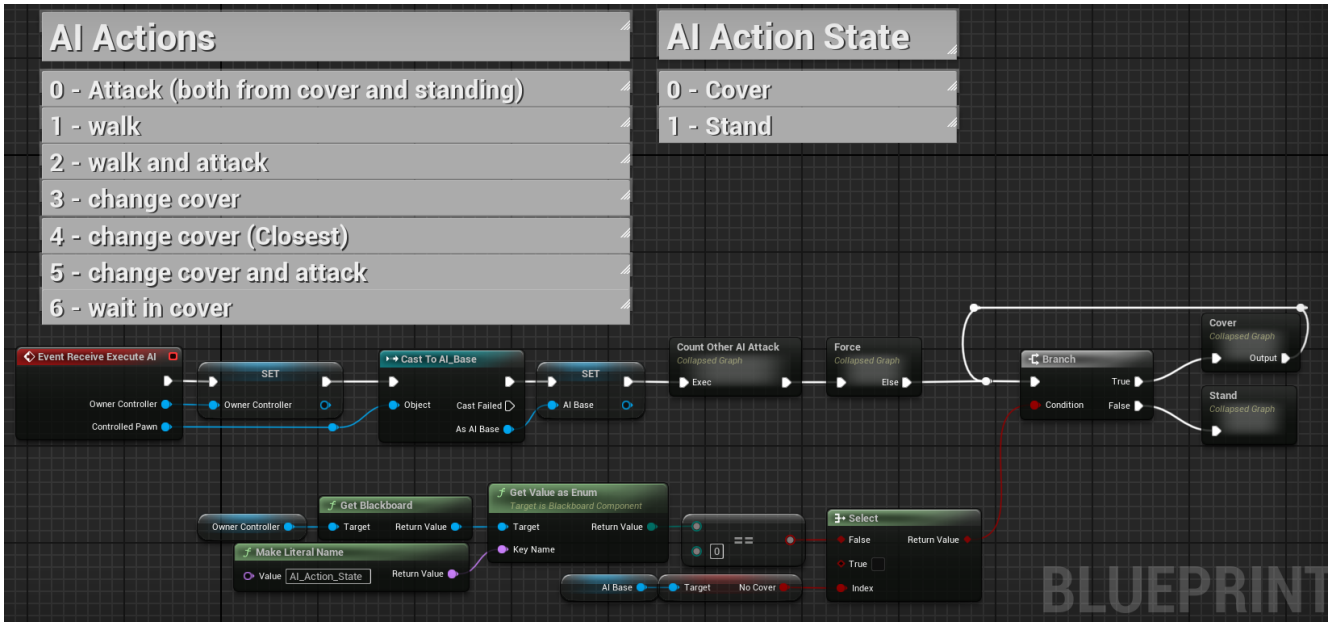

Рисунок 4.14 – Функція визначення дії

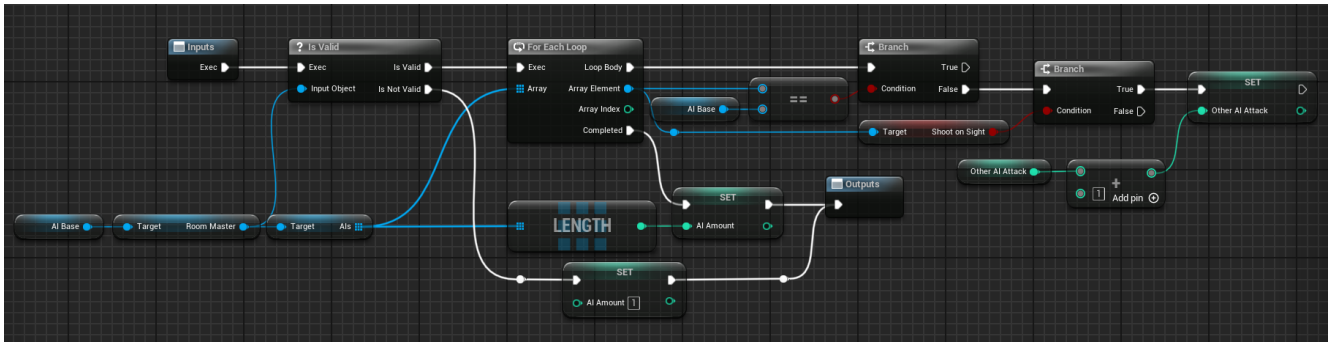

Рисунок 4.15 – Підфункція врахування дій інших ворогів

Усі дії поділяються на дві групи, дії в укритті (рисунок 4.16) та стоячи (рисунок 4.17). Щоразу як виконується дія в даній групі, лічильник цієї групи збільшується. Чим більше даний лічильник, тим більше шанс того, що наступного разу буде здійснено перехід в іншу групу.

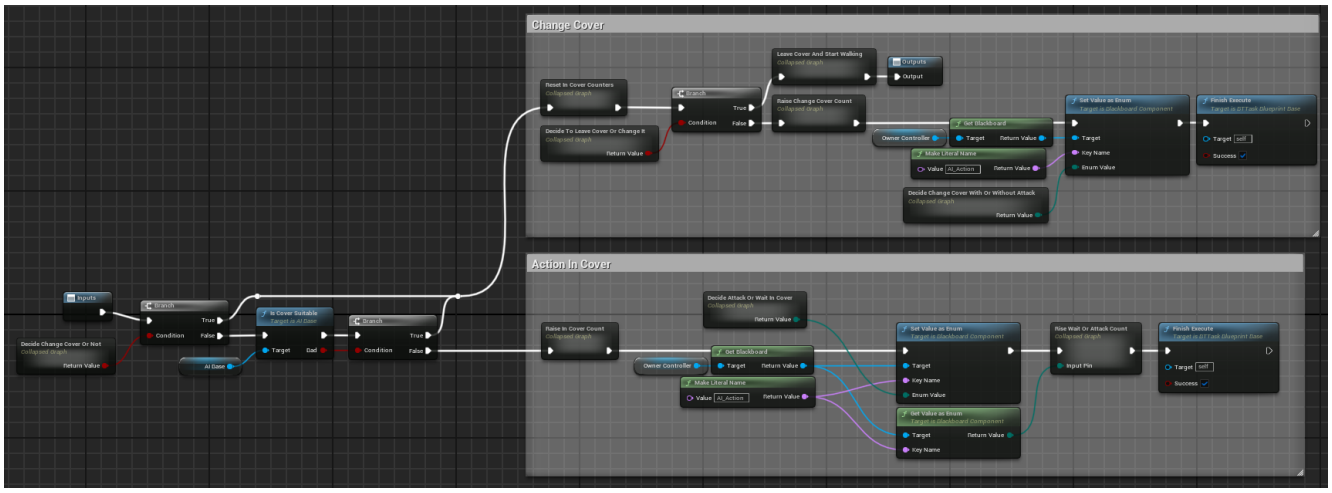

Рисунок 4.16 – Підфункція дій в укритті

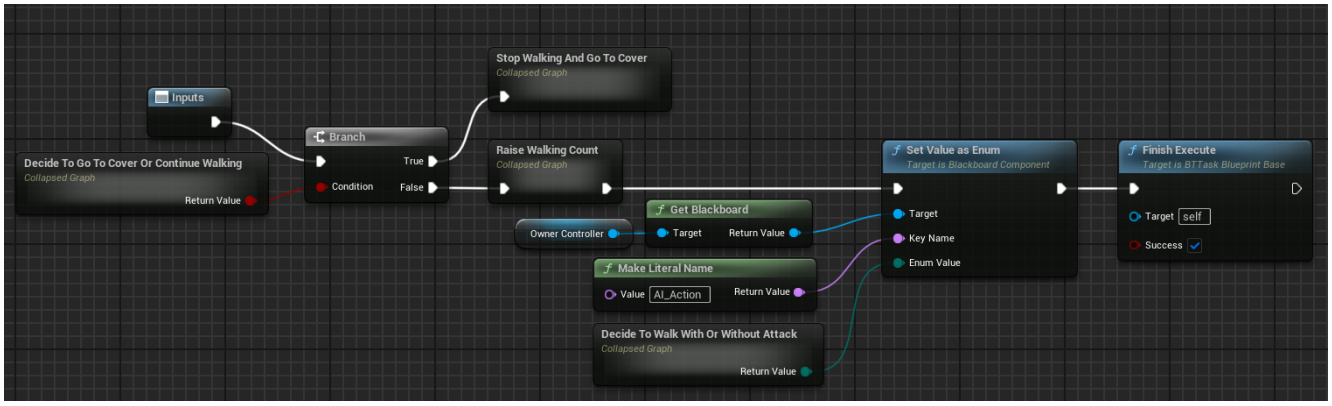

Рисунок 4.16 – Підфункція дій стоячи

В основному стані ворог має можливість виконувати такі дії:

– йти до укриття;

– йти до укриття та атакувати;

– йти до найближчого укриття;

– атакувати з укриття;

– сидіти в укритті та чекати;

– ходити по локації та атакувати.

Коли ворог здійснює атаку гравця, то він виконує анімацію прицілювання. Проблемо такої анімації є те, що дуло зброї не дивиться в ту саму точку, що і штучний інтелект ворога. Дану проблему було вирішено будуванням променю пострілу Line Trace не фізичним способом, тобто він йде прямо в місце, куди

дивиться ворог. Завдяки тому, що ворог прицілюється не в самого гравця, а в місце, де його в останнє бачив, а також за допомогою штучно підвищеного розкиду вдається зробити так, щоб вороги стріляли прямо в гравця, але могли промахуватися. Також завдяки можливості запам’ятовувати місце, в якому ворог бачив гравця в останнє, вдається зробити так, щоб ворог міг втратити гравця з поля зору, та також з’являється необхідність в обміні одне з одним інформації про положення гравця.

Розглянемо метод пошуку укриття для ворогів. Даний пошук здійснюється за допомогою EQS (рисунок 4.17). Система запитів до середовища (Environment Query System) – це функція системи штучного інтелекту в Unreal Engine, яка використовується для збору даних з навколишнього середовища [14].

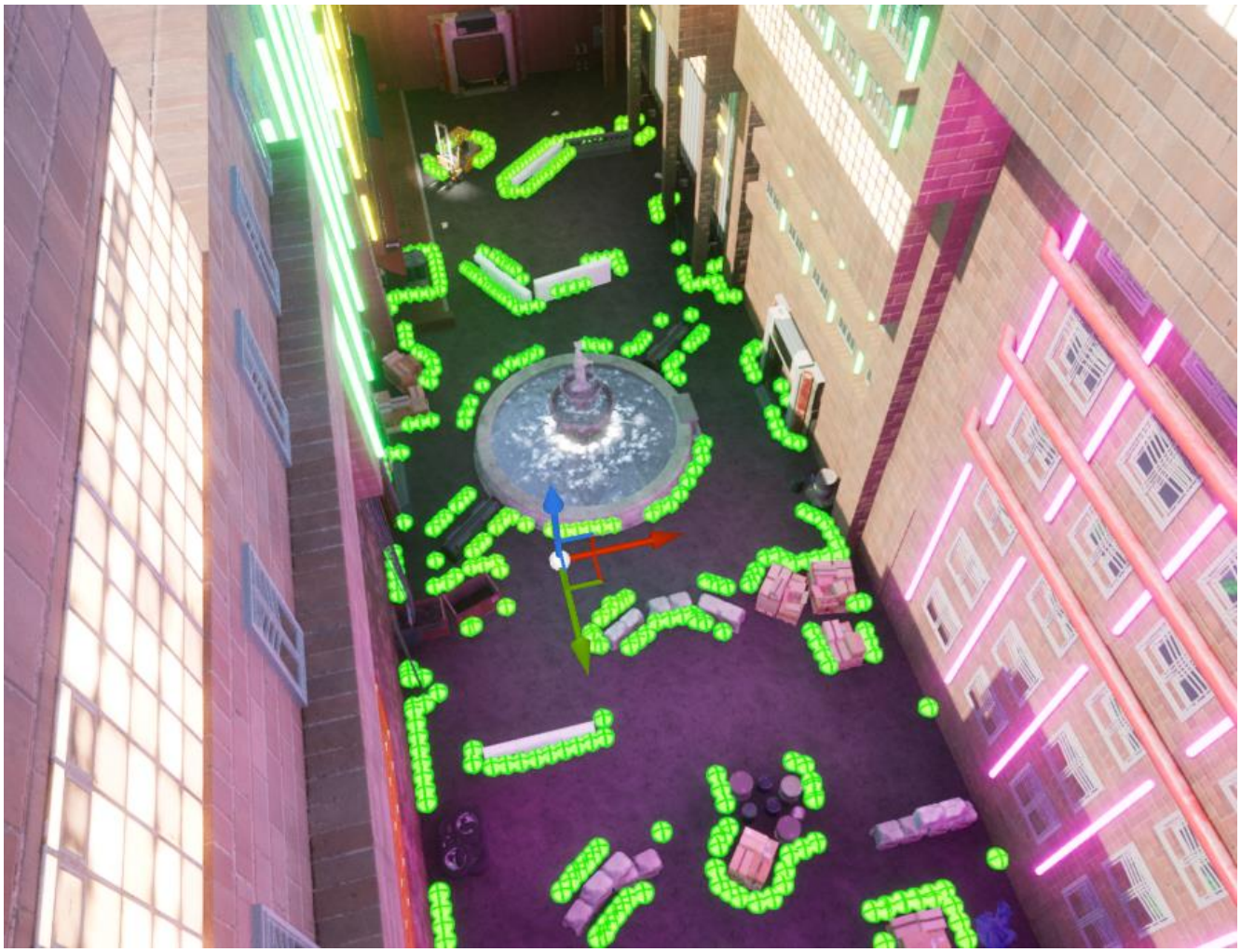

Рисунок 4.17 – Функція пошуку укриття за допомогою EQS

Дана функція сканує оточення та створює точки, які можуть бути використані ворогом як укриття. Якість укриття визначається на основі положення гравця, якщо певна точка закриває ворога від гравця і дає змогу встати для атаки, то вона позначається як хороше укриття. Таким чином визначається положення усіх можливих місць для укриття, а штучний інтелект випадковим чином обирає те місце, до якого буде йти.

4.5 Генерація рівнів

Коли відбувається генерація рівня, гравець очікує отримувати нову локацію під час кожного нового проходження. Цього можливо досягти за допомогою випадкових чисел, але постає проблема, якщо гравець завершить гру і захоче продовжити знову, то зі звичайними випадковими числами він не зможе отримати ту саму локацію. Для вирішення цієї проблеми буде використовувати тип випадкових чисел Random Stream Seed, головною особливістю якого є те, що він дає випадкову послідовність, що залежить від seed числа. Таким чином, якщо ми збережемо певне seed число, то зможемо повторно згенерувати одну і ту ж випадкову локацію, для того щоб гравець мав можливість продовжити попереднє проходження.

Кожна згенерована локація складається з кімнат з’єднаних коридорами. Для того, щоб положення кімнат різних типів не була зовсім випадковою, тобто щоб не було випадків коли початкова і фінальна кімнати знаходяться поруч, кімнати генеруються по заздалегідь визначеним структурам (рисунок 4.18), в яких зазначається відносне положення та зв’язок кімнат певних типів.

Алгоритм генерації рівнів було зображено в розділі 3.4 «Приклади найцікавіших алгоритмів та методів», розглянемо даний процес більш детально. В перший етап генерації відбувається зчитування файлу збереження для того, щоб отримати seed число та усі зміни, заподіяні гравцем, як наприклад список пройдених кімнат, список випадаючих предметів, що залишилися на локації та інше. Далі на основі випадкового числа зі списку випадково вибирається структура

рівня, і ,в залежності від тих типів кімнат, що є в списку цієї структури, випадково визначаються кімнати.

Room Id	Room Type	Left Room Id	Right Room Id	Top Room Id	Bottom Room Id	X Grid Coordinate	Y Grid Coordinate
1	Start Room	2	0	0	0	2	5
2	Fight Room	3	1	4	0	2	4
3	Gift Room	0	2	5	0	2	3
4	Fight Room	5	6	8	2	3	4
5	Fight Room	0	4	0	3	3	3
6	Fight Room	4	0	7	0	3	5
7	Safe Room	0	0	0	6	4	5
8	Boss Room	0	0	0	4	4	4

Рисунок 4.18 – Структура рівнів

Кімната, в даній грі, – це замкнутий простір будь-якої форми, що має чотири двері с кожної сторони та може бути як інтер'єром так і екстер'єром. Оскільки кімнати в своєму складі мають дуже багато об'єктів різних типів, то через це вони зроблені за допомогою Level Instance, технології, яка дозволяє розміщувати рівні в середині рівнів. Даний спосіб дозволяє зручно створювати кімнати, користуючись вбудованим редактором рівнів, а не редактором акторів. В даного методу є недолік – після того як рівень було розміщено, об'єкти в середині рівня не є згрупованими, тобто якщо ми розмістили декілька кімнат, то можливості отримати об'єкти з конкретної кімнати немає. Для вирішення цієї проблеми в кожної кімнати є спеціальний актор Room Master, який зберігає в собі інформацію про зміст кімнати. Також недоліком є те, що розміщену кімнату вже не можна згруповано перемістити в інше положення. Для вирішення цієї проблеми є спеціальний актор Room Controller, що на початку генерації спавнить кімнату, бере з неї інформацію про розмір, та видаляє її. Коли генерація завершується даний актор заново спавнить кімнату в новому положенні. Таким чином вдається користуватися зручним функціоналом редагування рівнів та виправити недоліки цього методу.

Після того як кімнати було розміщено відбувається генерація коридорів. Коридори будуються у формі шляхів, що з'єднують дві двері кімнат, для визначення таких шляхів використовується алгоритм «А зірка». А зірка – це алгоритм пошуку,

який використовується для знаходження найкоротшого шляху між початковою та кінцевою точками [15]. Спочатку на основі положення двох дверей створюється сітка координат, яка уникає перетин із кімнатами та вже згенерованими коридорами, потім за допомогою алгоритму, зображеному на рисунку 4.19, визначається список координат, з яких має складатися маршрут. Даний алгоритм дозволяє ефективно будувати шляхи навіть у тих випадках, коли на шляху є перепона.

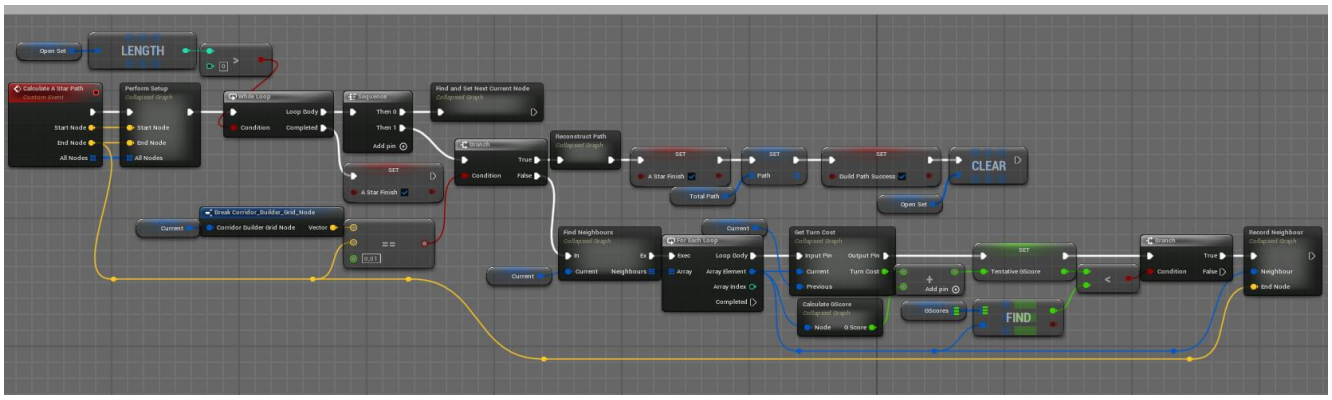

Рисунок 4.19 – Алгоритм пошуку шляху А зірка

Після визначення положення кімнат та коридорів відбувається їх спавн. На визначених місцях встановлюються кімнати, коридори будуються з фрагментів, відбувається спавн інтерактивних об’єктів, випадковий, якщо при першому завантаженні, або з файлу збереження, та персонажа гравця.

Кожна кімната має чотири входи з кожної сторони, і коли відбувається генерація коридорів, то частина дверей залишаються не з’єднаними. В такому випадку, якщо двері будуть відчиненими, то гравець побачить пустоту, а якщо зачиненими, то ці двері будуть вводи його в оману. Для вирішення цієї проблеми для кожних дверей було створено по два варіанти оточуючих об’єктів (рисунок 4.20), перший має вигляд суцільної стіни, а другий – отвору, в якому знаходяться двері.

Рисунок 4.20 – Приклад заміни дверей на стіни для не використаних входів до кімнат

Таким чином, в даному розділі було розглянуто структуру рівнів, їх алгоритм генерації та складові елементи і методи, завдяки яким вдається розміщувати велику кількість об’єктів на локації.

4.6 Система зберігання

Перед початком нового проходження створюється файл збереження (рисунок 4.21). В даному файлі зберігається інформація про seed число генерації, об’єкти локації, прогрес гравця та характеристики його персонажа.

Коли гравець тільки створює файл проходження, інформації про об’єкти локацій немає. Під час першої генерації об’єкти випадково спавняться на локації та після цього заносяться в файл.

Перед початком проходження гравцю потрібно обрати персонажа в головному меню. Список персонажів зберігається в таблиці (рисунок 4.22) та інформація про обраного персонажа заноситься в файл збереження при початку проходження.

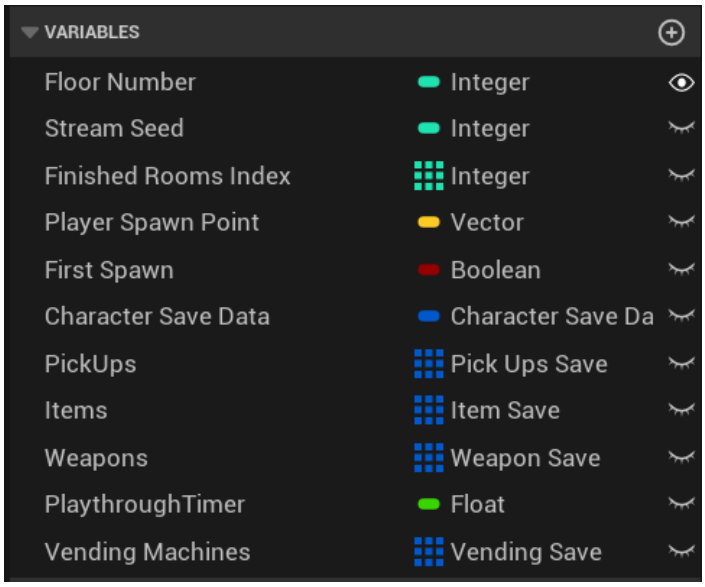

Рисунок 4.21 – Файл збереження проходження

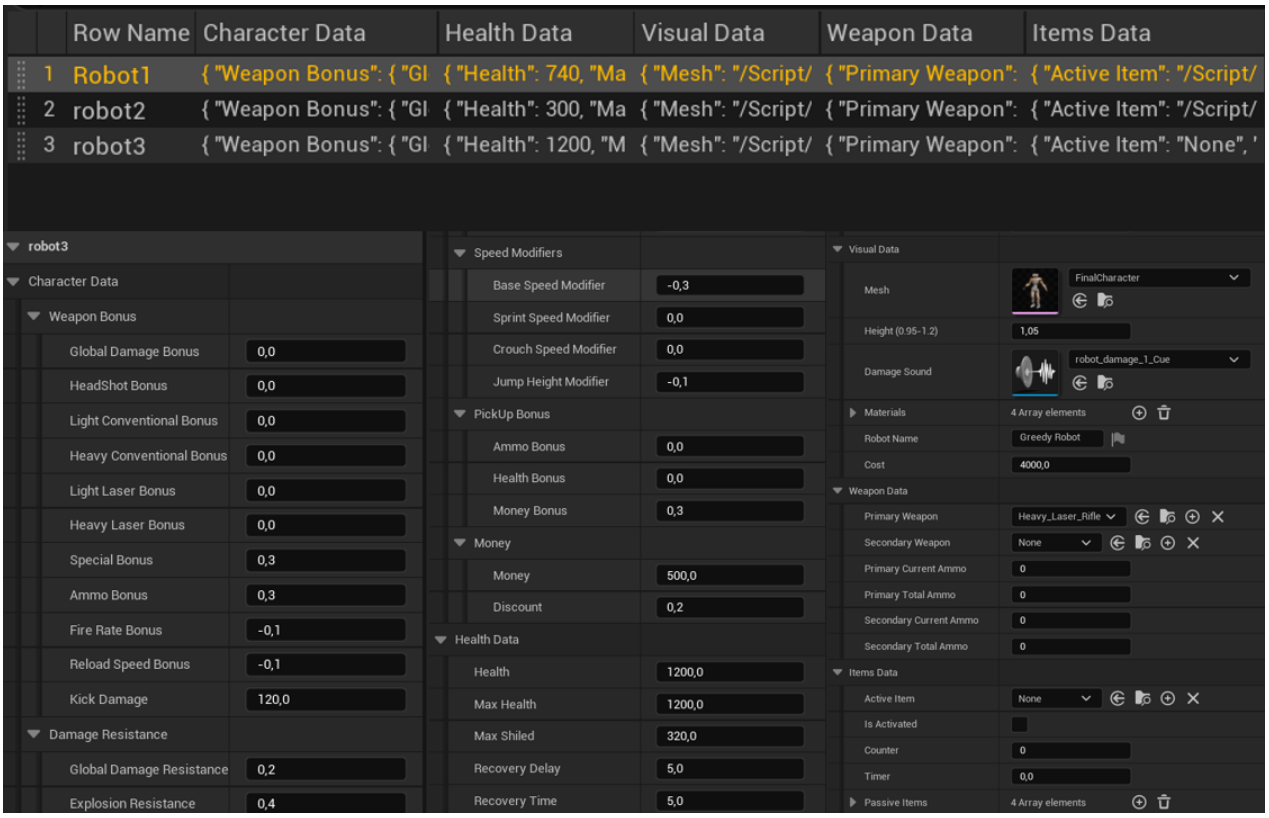

Рисунок 4.22 – Таблиця персонажів та їх параметри

Зберігання в даній грі відбувається автоматично кожні 10 секунд, а також коли гравець завершує проходження кімнати, підбирає зброю чи предмет, або взаємодії з автоматом. Під час проходження бою в кімнаті автоматичне зберігання вимикається.

4.7 Інтерфейс

Коли гравець проходить локацію, то впродовж цього він бачить інтерфейс бою (рисунок 4.23). Даний інтерфейс відображає основну інформацію про персонажа гравця, таку як здоров’я, щит, гроші, зброя та предмети. Також він бачить інформацію про час проходження, поточну хвилю, якщо зараз йде бій, та міні-карту.

Рисунок 4.23 – Інтерфейс бою

Розглянемо більш детально принцип роботи міні-карти. Після того, як генерація рівню завершилася, створюється об’єкт міні-карти, він зчитує кожну кімнату та фрагмент коридору, та створює для кожного окремий віджет з матеріалом (рисунок 4.24). Даний матеріал отримує зображення кімнати або коридору, зміщує та трансформує його таким чином, щоб воно знаходилося на правильному місці відносно положення гравця. Даний віджет складається з двох частин, великої і малої міні-карти, коли відкривається велика міні-карта, то мала скривається.

Для того щоб дозволити гравцю рухати велику міні-карту, вона оновлюється не відносно положення гравця, а відносно окремого актора, який слідкує за положенням гравця, а якщо відкрити велику міні-карту, то рухається окремо.

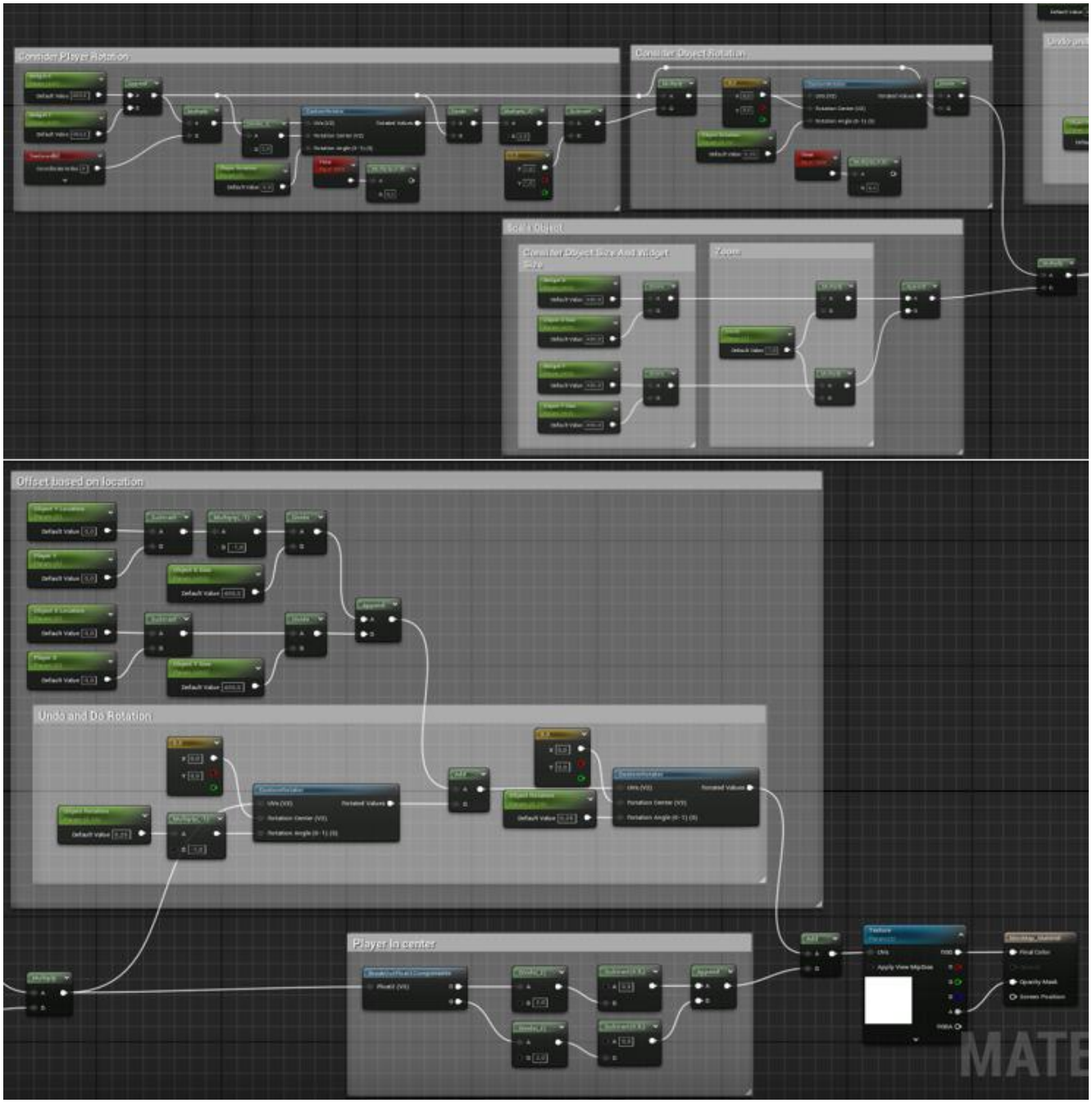

Рисунок 4.24 – Матеріал міні-карти

Таким чином, за допомогою складного матеріалу вдається створити чітку міні-карту, що дозволяє гравцю бачити та орієнтуватися на локації.

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розробка мапи думок тестування

Тестування ігрового програмного забезпечення відбувалося відповідно до тест-плану, наведеному в додатку Г, та мапи думок, зображеної на рисунку 5.1.

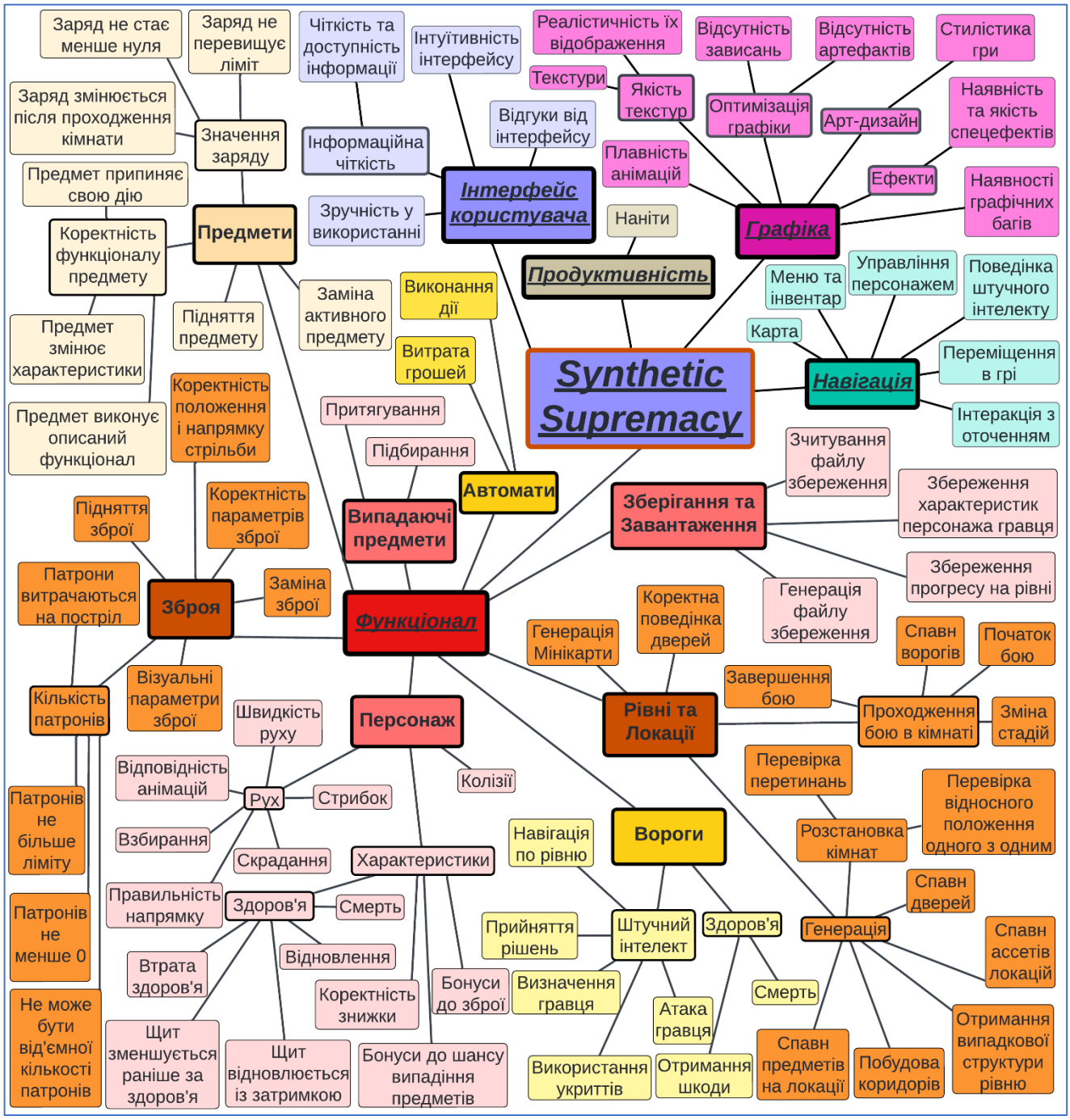

Рисунок 5.1 - Mind Map для тестування проекту

Тестування механік та функціоналу проводилося за методологією «Білого ящику», тому що улаштування системи нам відомо, а тестування інтерфейсу проводилося за методологією «Чорного ящику», бо тестувався саме візуальний вигляд.

На даній діаграмі можна виділити такі частини проекту, які мають бути протестовані:

– Функціонал: основні функції та властивості персонажу, зброї, предметів, автоматів, випадаючих предметів, ворогів, генерації рівнів та локацій, системи зберігання та завантаження.

– Інтерфейс користувача: зручність, інформаційна чіткість, відгуки та інтуїтивність основних елементів інтерфейсу бою.

– Продуктивність: працездатність, ефективність та візуальна якість алгоритму оптимізації полігональної сітки Nanite.

– Графіка: анімації, текстури, ефекти, загальна стилістика гри, відсутність графічних багів, зависань та артефактів.

– Навігація: зручність та зрозумілість управління персонажем, переміщення, взаємодії з оточенням, використання карти та інвентаря, а також зрозумілість та коректність поведінки штучного інтелекту ворогів.

Таким чином, було визначено, які саме частини ігрового програмного застосунку потрібно проаналізувати для тестування.

5.2 Розробка тестових випадків

Керуючись сформованою мапою думок та тест-планом було розроблено тестові випадки. Кожен тестовий випадок має опис ситуації, фактичний та очікуваний результати, кроки відтворення, а також пріоритет (P1 – найвищий, P3 – найнижчий), та серйозність (S1 – найвища, S5 – найнижча). В таблиці 5.1 наведені найцікавіші тестові випадки з кожних компонент програмного застосунку.

Таблиця 5.1 – Список тестових випадків ігрового програмного застосунку

Тест № 1	
Назва тесту:	Двері не зникають.
Опис тесту:	Коли спавниться кімната, то двері вирішують, чи зникнути и поставити на свою місце стіну (якщо немає коридора), або залишитися і прибрати стіну (якщо коридор є). Якщо структура рівня дуже велика, то є ймовірність, що двері не встигають замінитися. Це відбувається через те, що завантаження кімнат займає певний час, тому функції їх спавну побудовані на затримках та очікуваннях. Десь в коді є не правильне або відсутнє очікування, через що на великих рівнях частина дверей не встигаю виконати заміну, хоча з малими рівнями такої проблеми немає. Дана проблема є блокуючою, бо стіни в середині дверей не дають пройти гравцю.
Компонент системи:	Генерація рівнів
Пріоритет:	P1
Критичність:	S1
Кроки відтворення:	1. Увімкнути в коді структуру великого рівня (замінити на список звичайних рівнів); 2. в головному меню почати нову гру; 3. поставити гру на паузу, перейти в режим редагування та переглянути рівень.
Очікуваний результат:	Усі двері повинні бути або дверями, або стінами.
Фактичний результат:	Частина дверей на рівні не замінилися.
Тест № 2	
Назва тесту:	Кількість патронів зброї Salvo погано працює з модифікатором кількості патронів.

Продовження таблиці 5.1

Опис тесту:	Зброя Salvo – це ракетниця, яка за раз випускає 4 ракет та витрачає 4 патрони. Через те, що модифікатор кількості патронів змінює кількість можливих патронів в магазині, наприклад, при -90% (мінімальний можливий відсоток) зброя може містити лише 1 патрон але витрачає і випускає 4. Через це в магазині після перезарядки а потім пострілу виходить -3 патрони.
Компонент системи:	Бойова система
Пріоритет:	P3
Критичність:	S3
Кроки відтворення:	1. Встановити персонажу бонус до патронів -0.9; 2. отримати зброю Salvo; 3. виконати постріл; 4. виконати перезарядку; 5. виконати постріл.
Очікуваний результат:	Зброя повинна витрачати 1 патрон.
Фактичний результат:	Зброя витрачає 4 з 1 патронів.
Тест № 3	
Назва тесту:	«Натисніть на будь-яку кнопку щоб продовжити» спрацьовує як інпут персонажа.
Опис тесту:	Під час запуску гри після завантаження з’являється напис «Натисніть на будь-яку кнопку щоб продовжити», і якщо ця кнопка використовується для управління персонажем, то він виконує цю дію.
Компонент системи:	Інтерфейс бою
Пріоритет:	P3
Критичність:	S5

Продовження таблиці 5.1

Кроки відтворення:	Запустити гру та натиснути кнопку будь-якої дії для початку гри.
Очікуваний результат:	Персонаж не реагує на натиснуту кнопку.
Фактичний результат:	Персонаж виконую дію натиснутої кнопки.
Тест № 4	
Назва тесту:	Таймер починає йти, навіть якщо гравець не натиснув «Натисніть на будь-яку кнопку щоб продовжити».
Опис тесту:	Коли гравець запускає гру, після завантаження він бачить напис "Натисніть на будь-яку кнопку щоб продовжити". Якщо він не натисне одразу, то таймер все одно запуститься після завантаження рівня, через що час проходження буде не точним.
Компонент системи:	Збереження та прогрес
Пріоритет:	P3
Критичність:	S4
Кроки відтворення:	Запустити гру та не натискати "Натисніть на будь-яку кнопку щоб продовжити" певний час.
Очікуваний результат:	Час починає йти після початку гри.
Фактичний результат:	Час починає йти після загрузки рівня а не початку гри.
Тест № 5	
Назва тесту:	Гравець може взаємодіяти з декількома інтерактивними предметами одночасно.
Опис тесту:	Якщо гравець підійде до двох предметів, що стоять поруч, то він взаємодіє з обома одночасно. Це працює зі зброєю, предметами та автоматами.
Компонент системи:	Економіка
Пріоритет:	P3
Критичність:	S5

Кінець таблиці 5.1

Кроки відтворення:	Підійти до двох предметів, що стоять поруч, та почати взаємодію.
Очікуваний результат:	Взаємодія відбувається лише з одним з предметів.
Фактичний результат:	Предмети активують взаємодію одночасно.
Тест № 6	
Назва тесту:	Ворог може атакувати до завершення анімації спавну.
Опис тесту:	Коли ворог спавниться, він програє анімації, під час якої він не рухається та не може атакувати і отримувати шкоду. Проте, якщо в цей час персонаж гравця буде знаходитися дуже близько, ворог почне стріляти в гравця. Це відбувається через те, що у ворога є функція, яка змушує його атакувати гравця, якщо той наблизився до нього.
Компонент системи:	Штучний інтелект
Пріоритет:	P3
Критичність:	S4
Кроки відтворення:	Знаходитися поруч з ворогом, що з’являється.
Очікуваний результат:	Ворог не робить нічого до завершення спавну.
Фактичний результат:	Ворог починає атаку до завершення спавну.

Таким чином, в результаті проведення тестування ігрового програмного застосунку було перевірено все компоненти та створено тестові випадки, завдяки чому вдалося виправити недоліки розроблюваного продукту.

6 ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

6.1 Наукове впровадження проекту

На основі теми кваліфікаційної роботи було написано наукову роботу у вигляді тез для 28-го Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті», що проходив з 16 по 18 квітня 2024 року в онлайн режимі на базі ХНУРЕ.

Темою тез доповіді є «Оптимізація графіки в іграх». У роботі було описано такі основні методи оптимізації графіки, як Culling, LOD (Level Of Detail) та Nanite. Було описано їх принцип роботи та основні сфери застосування, також було проаналізовано їх ефективність на прикладі розробленого ігрового програмного застосунку. Тези доповіді для науково-практичної інтернет-конференції можна побачити в додатку Ґ.

6.2 Практичне впровадження проекту

Розроблений ігровий програмний застосунок було продемонстровано на виставці технічної творчості молоді 28-го Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті». Роботу за темою «Ігровий програмний застосунок у жанрі 3D Rogue-like шутер від третьої особи Synthetic Supremacy» у секції «Ігрові технології» було презентовано онлайн 18 квітня 2024 року, дана робота посіла І місце у своїй категорії. Тези доповіді для науково-практичної інтернет-виставки можна побачити в додатку Д.

ВИСНОВКИ

В ході виконання комплексної роботи з теми «Ігровий програмний застосунок у жанрі 3D Rogue-like шутер від третьої особи», а саме механіки бойової системи, штучного інтелекту ворогів, генерації рівнів, економіки та інтерфейсу бою, було проаналізовано предметну галузь, проаналізовано основних конкурентів в даному жанрі ігор, визначено їх переваги та недоліки, було визначено цільову аудиторію даного ігрового програмного застосунку та розглянуто його монетизацію.

Наступним етапом було сформовано функціональні та нефункціональні вимоги до ігрового програмного забезпечення, а також вимоги до середовища розробки. В результаті аналізу було обрано ігровий рушій Unreal Engine 5, через його широкий функціонал, направлений на розробки реалістичних ігор.

Також було розглянуто основний функціонал гри за допомогою засобів UML проектування, було визначено основні дії гравця, та ворогів з якими гравець буде взаємодіяти. Було сформовано ігровий цикл, розглянуто алгоритм генерації рівнів, а також спроектовано ігровий інтерфейс бою.

Керуючись сформованими вимогами було розроблено такі частини ігрового програмного застосунку, як механіки бойової системи (рух та характеристики персонажа, використання зброї та предметів, взаємодія з оточенням), штучного інтелекту ворогів (атака гравця, прийняття рішень), генерації рівнів, економіки, інтерфейсу бою та системи зберігання прогресу гравця.

В процесі розробки ігрового програмного застосунку було здобуто навички роботи з ігровим рушієм Unreal Engine 5, також було отримано навички роботи з програмним забезпеченням «Adobe Photoshop» та середовищем 3D моделювання Blender, при роботі над елементами інтерфейсу і над 3d-ассетами. Було здобуто навички з розробки комплексних програмних систем, роботи над штучним інтелектом, роботи з алгоритмами генерації, та роботи в команді.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Roguelike. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Roguelike (дата звернення: 13.05.2024).
2. Cyberpunk 2077. Home of the Cyberpunk 2077 universe – games, anime & more. URL: https://www.cyberpunk.net/ua/en/ (дата звернення: 22.05.2024).
3. Deus Ex: Human Revolution. Deus Ex Wiki. URL: https://deusex.fandom.com/wiki/Deus_Ex:_Human_Revolution (дата звернення: 22.05.2024).
4. Enter the Gungeon. Enter the Gungeon. URL: https://enterthegungeon.com (дата звернення: 22.05.2024).
5. Dead Cells - The rogueVania from Motion Twin. Dead Cells - Rogue-lite metroidvania with some souls-lite combat on top! Kill, die, learn, repeat. URL: https://dead-cells.com (дата звернення: 22.05.2024).
6. Unity (ігровий рушій). Вікіпедія. URL: https://uk.wikipedia.org/wiki/Unity_(ігровий_рушій) (дата звернення: 13.05.2024).
7. Не соромно запитати: як працює Unreal Engine. SKVOT / СКВОТ – онлайн-курси про рекламу, кіно та мистецтво | SKVOT. URL: https://skvot.io/uk/blog/ne-soromno-zapitati-yak-pracyuye-unreal-engine (дата звернення: 13.05.2024).
8. Модель-вид-контролер. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Модель-вид-контролер (дата звернення: 14.05.2024).
9. Actors. Документація Unreal Engine. URL: https://docs.unrealengine.com/4.26/en-US/ProgrammingAndScripting/ProgrammingWithCPP/UnrealArchitecture/Actors/ (дата звернення: 13.05.2024).
10. Components. Документація Unreal Engine. URL: https://docs.unrealengine.com/4.26/en-US/Basics/Components/ (дата звернення: 13.05.2024).

11. Advanced Locomotion System V4. unrealengine marketplace. URL: https://www.unrealengine.com/marketplace/en-US/product/advanced-locomotion-system-v1 (дата звернення: 22.05.2024).
12. blender.org - Home of the Blender project - Free and Open 3D Creation Software. blender.org. URL: https://www.blender.org (дата звернення: 22.05.2024).
13. Малюйте 3D-текстури в реальному часі. Adobe. URL: https://www.adobe.com/ua/products/substance3d-painter.html (дата звернення: 22.05.2024).
14. Environment Query System. Документація Unreal Engine. URL: https://docs.unrealengine.com/4.26/en-US/InteractiveExperiences/ArtificialIntelligence/EQS/ (дата звернення: 22.05.2024).
15. S R. A. A* Algorithm in Artificial Intelligence You Must Know in 2024 | Simplilearn. Simplilearn.com. URL: https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/a-star-algorithm (дата звернення: 22.05.2024).

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

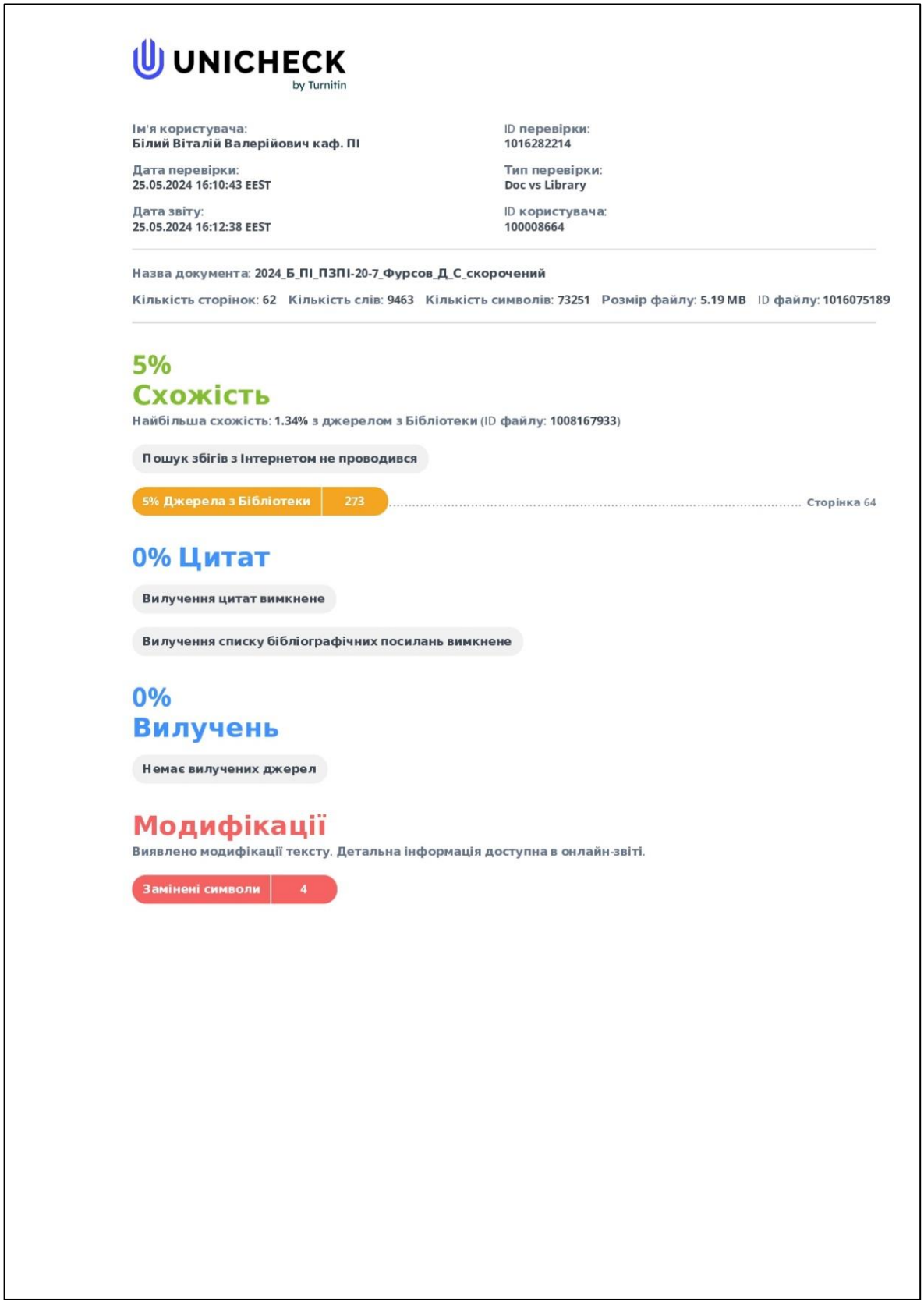

Рисунок А.1 – Перевірка на плагіат

ДОДАТОК Б

Слайди презентації

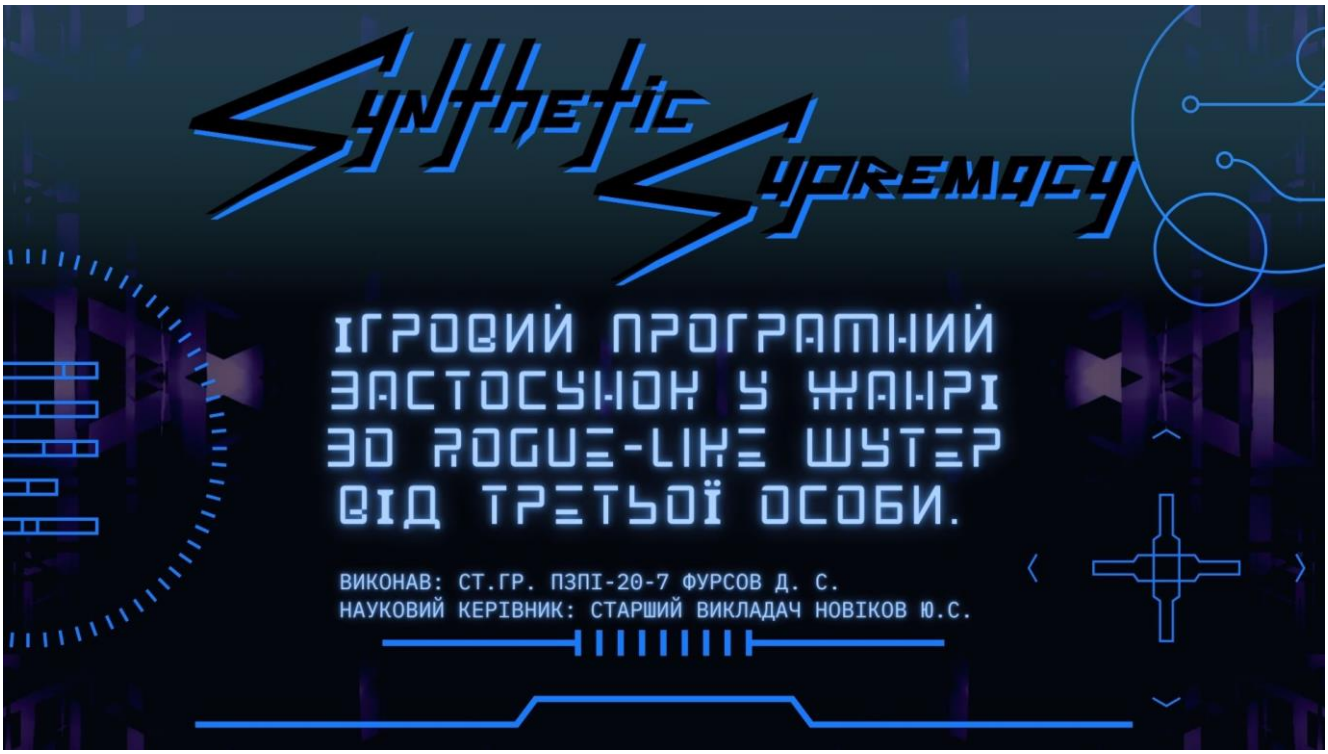

Рисунок Б.1 – Слайд презентації 1

Рисунок Б.2 – Слайд презентації 2

Рисунок Б.3 – Слайд презентації 3

Рисунок Б.4 – Слайд презентації 4

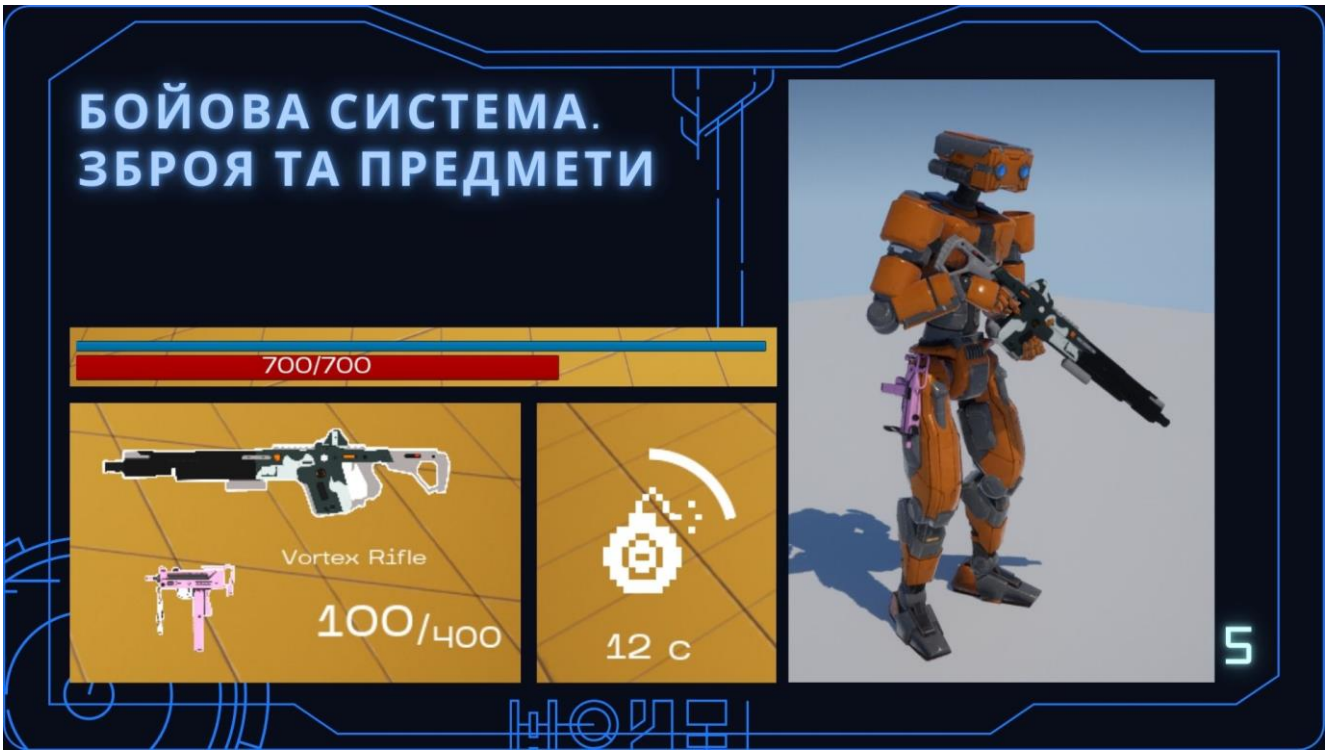

Рисунок Б.5 – Слайд презентації 5

Рисунок Б.6 – Слайд презентації 6

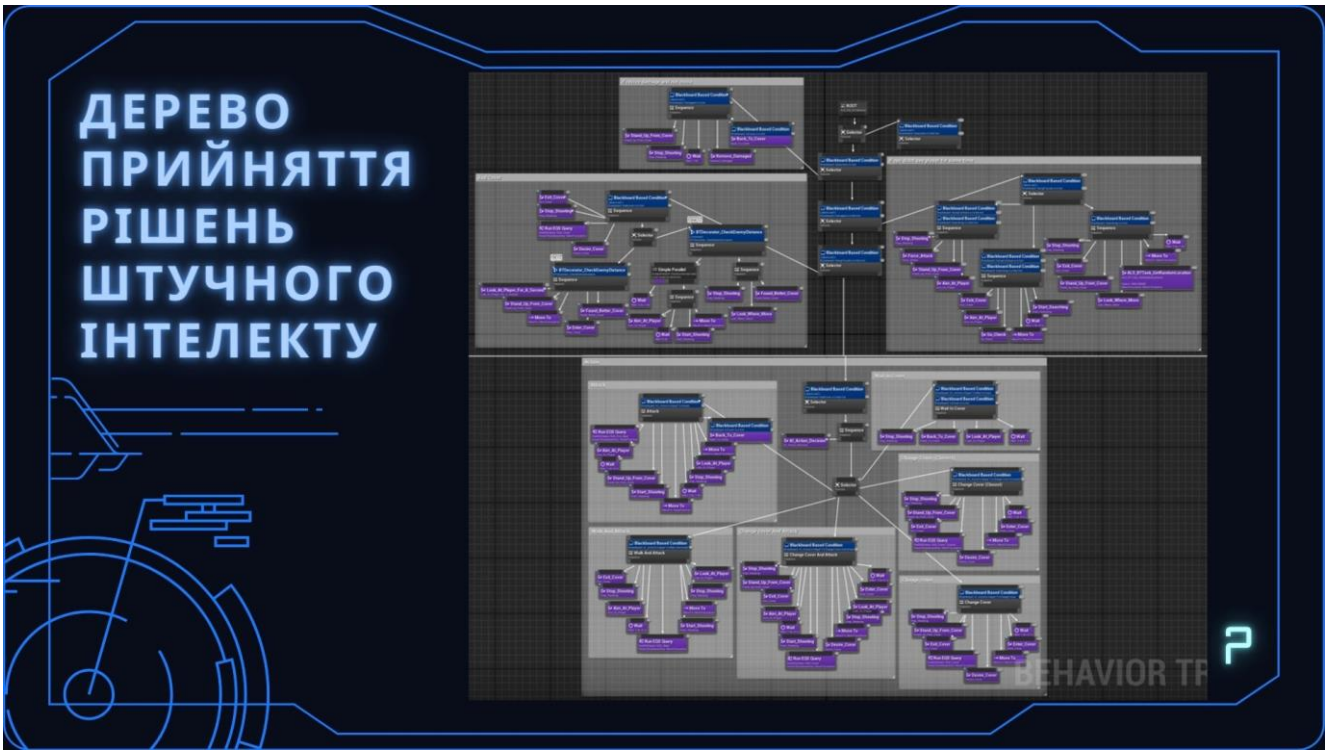

Рисунок Б.7 – Слайд презентації 7

Рисунок Б.8 – Слайд презентації 8

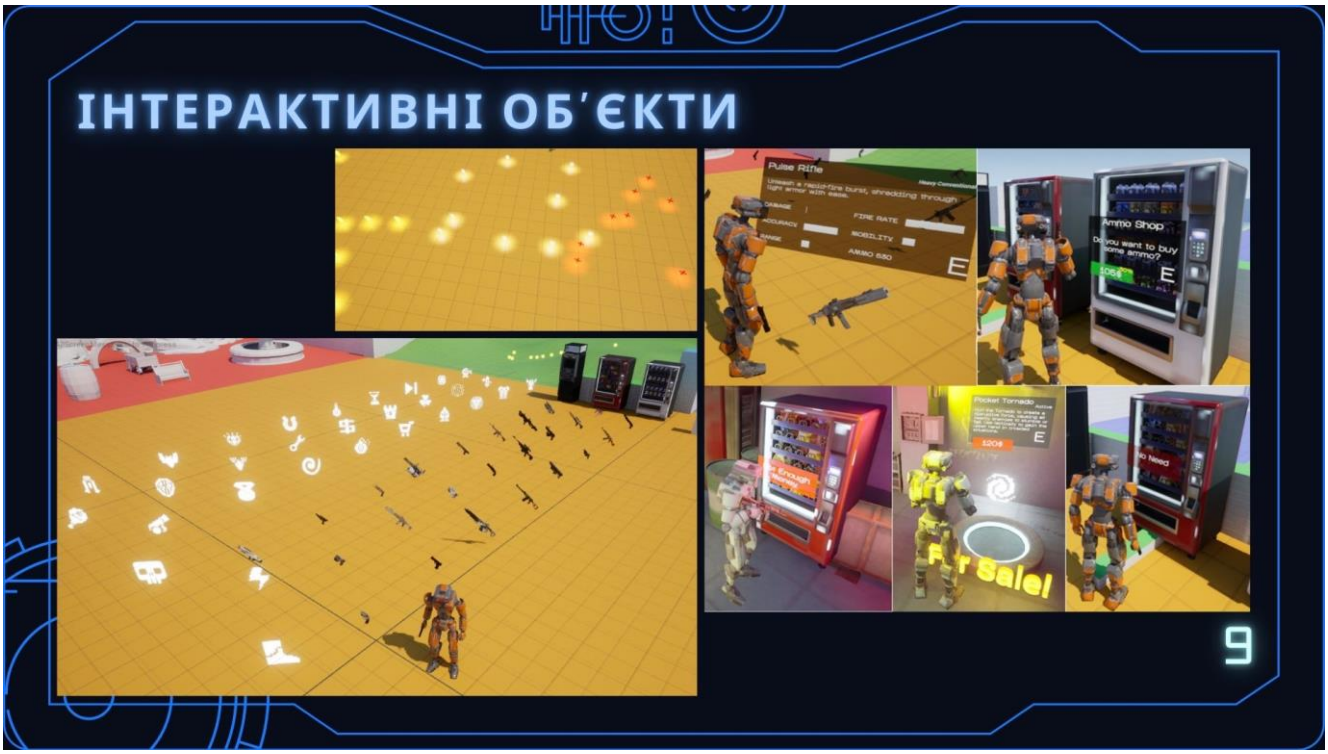

Рисунок Б.9 – Слайд презентації 9

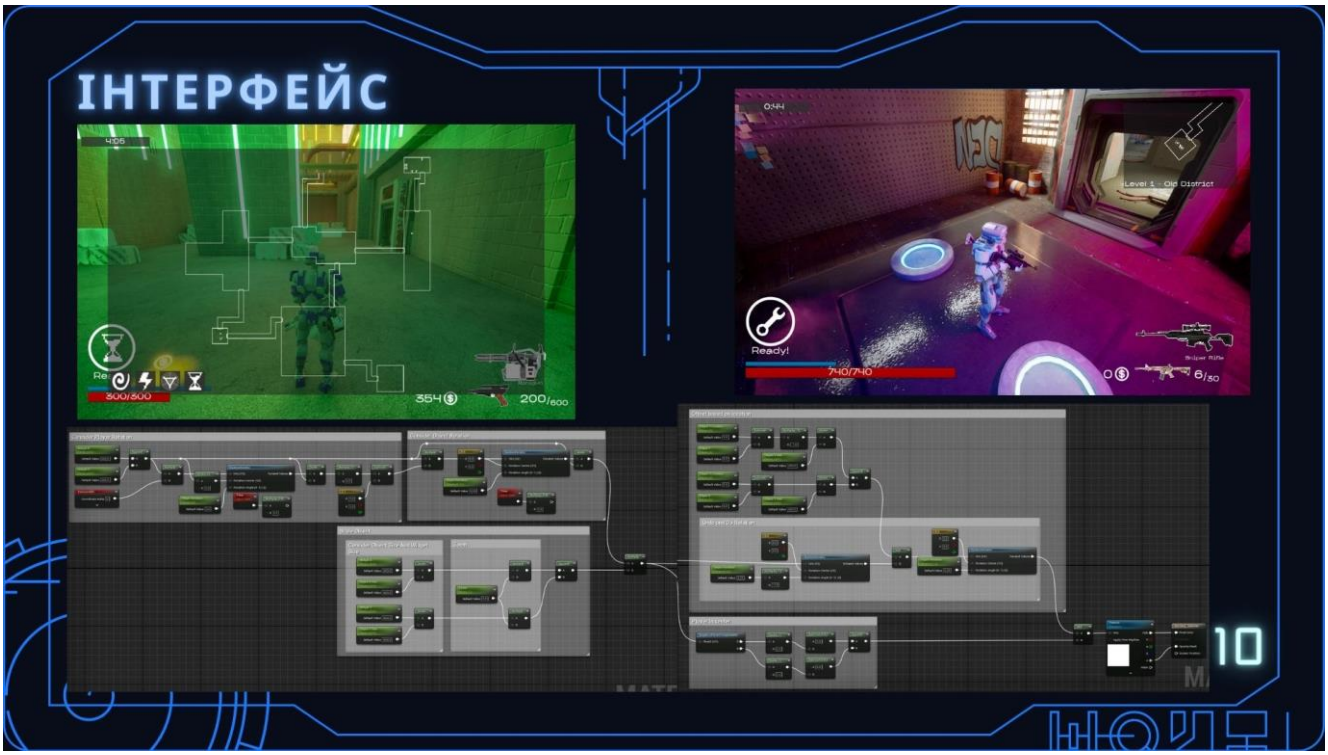

Рисунок Б.10 – Слайд презентації 10

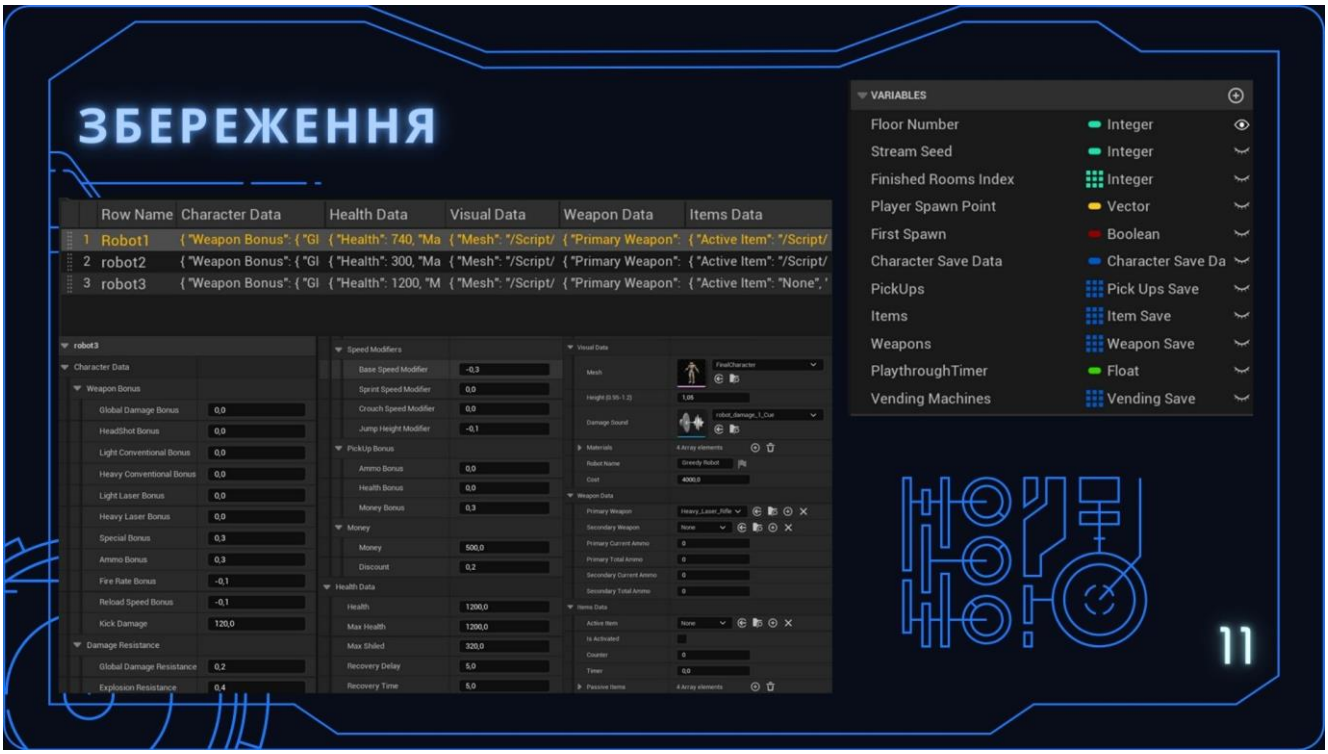

Рисунок Б.11 – Слайд презентації 11

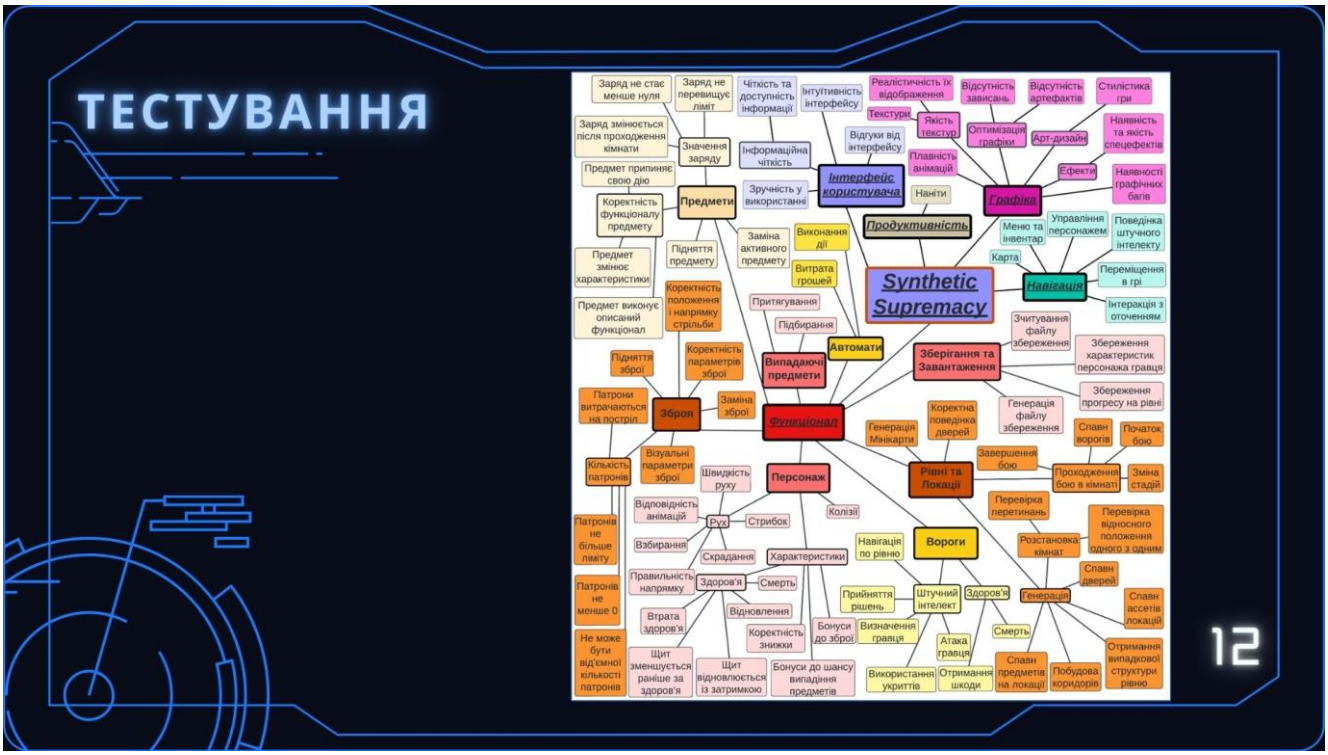

Рисунок Б.12 – Слайд презентації 12

Рисунок Б.13 – Слайд презентації 13

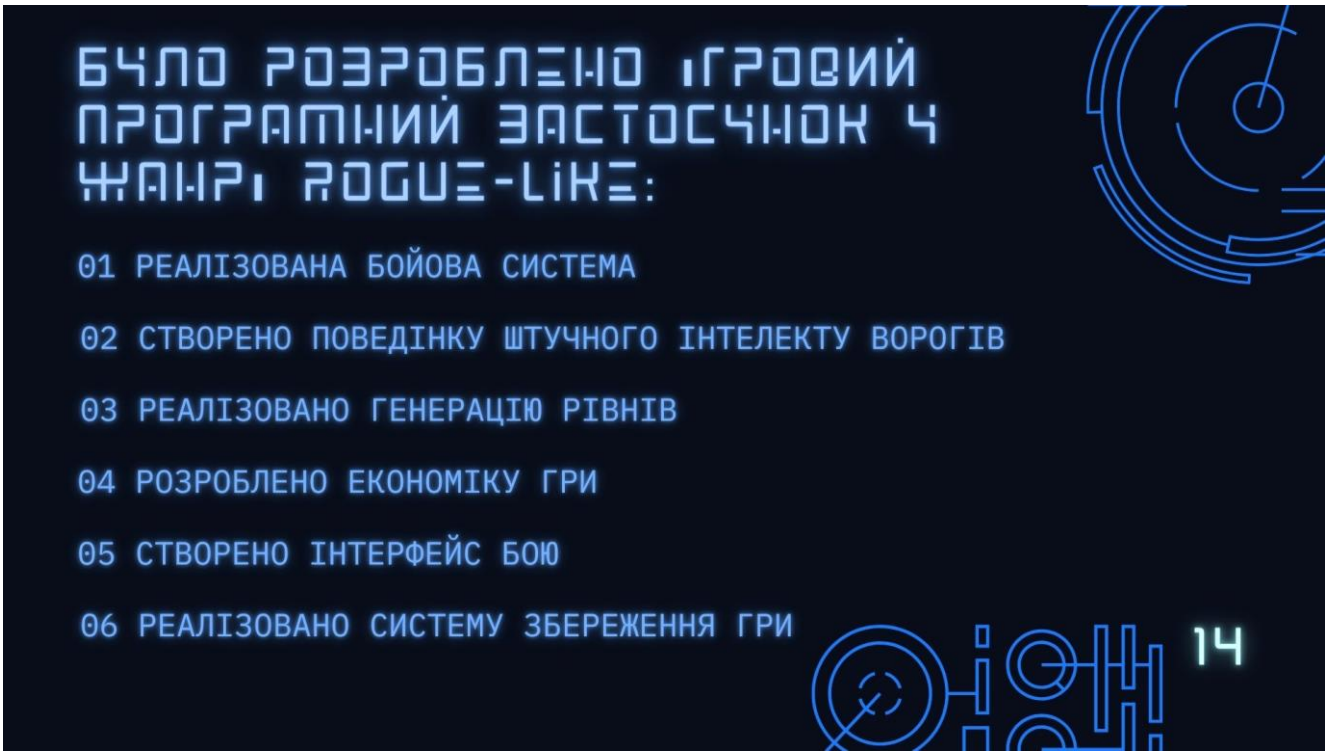

Рисунок Б.14 – Слайд презентації 14

Рисунок Б.15 – Слайд презентації 15

ДОДАТОК В

Геймдизайн-документ

«Synthetic Supremacy»

Зміст:

Рисунок В.1 – Геймдизайн-документ сторінка 1

1) Тетра

Механіка: Roguelike/Шутер

Технологія: Персональний комп’ютер

Історія: ГГ - бойовий робот

Естетика: Кіберпанк

2) Вступ

Дія гри розгортається в майбутньому, в гігантському мегаполісі, що дистопійно перетворився на символ безжальної корпоративної влади та беззаконня, де корпорації та мафія утримують усі путі влади в своїх руках. У цьому майбутньому світі головним героєм є бойовий робот, створений для виконання завдань корпораціями, але загадкові обставини привели до того, що він стає символом опору та визволення від цього соціального кошмару.

Мегаполіс, де розгортається подія гри, - це гігантське місто, вкрите хмарами смогу та сірих будівель. Величезні скляні вежі корпорацій відносяться високо над горизонтом, пануючи над масами мешканців міста. Вулиці вкриті плакатами, пропагандуючи корпоративні ідеали та маніпулюючи свідомістю громадян. Правоохоронні органи в службі корпорацій забезпечують "спокій" у місті, підтримуючи тиранію та беззаконня.

Головний герой, бойовий робот, колись був ідеальним виконавцем різноманітних завдань для корпорацій, але після зіткнення з подіями, які розкрили йому жахливу істину про корпоративний режим, він обирає інший шлях. Звільнившись від корпоративного програмування, він стає символом опору і починає боротьбу з корпораціями та мафією, ставлячи на карту не тільки свою власну долю, але й майбутнє цього знущаного міста.

Рисунок В.2 – Геймдизайн-документ сторінка 2

Гра пропонує гравцям зануритися у цей темний, технологічний світ. Відкрийте для себе різноманітні можливості бойового робота, вступайте в соціальні конфлікти та розкривайте масштабні заговори корпорацій. Граючи за цього воїна, ви станете головним символом боротьби за свободу і справедливість у цьому майбутньому світі.

3) Цільова аудиторія

Основна цільова аудиторія гри - це гравці віком 16+. Дану аудиторію має залучати реалізм, наявність насильства та необхідність швидко приймати рішення. Гра має вікове обмеження 16+ через наявність насильства та необхідність стратегічного та тверезого мислення під час боїв.

4) USP

1. «Звільни місто від корпоративного пекла та стань символом опору в майбутньому гігаполісі!»
2. «Заглибся в дистопічний світ, граючи за бойового робота, вибери свій шлях до визволення.»
3. «Вступай в боротьбу за майбутнє великого мегаполісу і переписуй історію власними руками.»
4. «Час прийшов для тебе стати водночас рятівником і правосуддям!»
5. «Змінюй світ, вибираючи свій шлях у мегаполісі майбутнього. Твої рішення формують долю цього міста.»

Рисунок В.3 – Геймдизайн-документ сторінка 3

5) Час ігрової сесії

Ігрова сесія розрахована на 20–60 хвилин, залежить від навичків гравця та випадкової генерації. Гра є реіграбельною, тому повне проходження гри не передбачено.

6) Технологічні характеристики

Платформа: персональний комп'ютер.

Керування: клавіатура та миш.

Гравець повинен мати середній або хороший комп'ютер, мишку, клавіатуру та монітор.

7) Опис гри

7.1) Опис ігрового процесу

Головне меню

Після запуску гри гравець потрапляє до головного меню, де він може обрати персонажа з унікальною зброєю та айтемами в розділі «CHARACTER». Далі він може вибрати розділ «CLASSIC» для першого запуску гри.

Якщо гравець вже грав і має досвід проходження гри, то замість гри спочатку, він може продовжити з конкретного місця, обравши розділ «CONTINUE», і продовжити проходження гри з того моменту, де він раніше завершив.

Проходження гри

Після початку гри генерується локація та персонаж з'являється в початковій кімнаті. Навколо нього знаходяться сусідні кімнати, він йде в

Рисунок В.4 – Геймдизайн-документ сторінка 4

одну з цих кімнат та починає бій. Використовуючи зброю та предмети гравець перемагає ворогів. Далі він йде до інших кімнат.

В процесі проходження гравець знаходить нову зброю та предмети, які покращують його характеристики та змінюють геймплей, роблячи його унікальним для кожного проходження. Перемагаючи ворогів, персонаж отримує гроші, які він може витратити на покупку ресурсів, таких як здоров'я та патрони, або він має можливість купити нову зброю та предмети.

Проходячи усі кімнати, гравець зустрічає боса. Подолавши боса, гравець переходить на наступний рівень. Кожен наступний рівень більший та складніший за попередній. Якщо гравець пройшов 5 рівнів або його персонаж помер, гра завершується та гроші, зароблені під час проходження, переносяться до головного меню. В головному меню гравець може купити нових персонажів та почати гру повторно.

7.2) Механіки

Персонаж

Перед початком гри гравець обирає персонажа в головному меню. Кожен персонаж має зброю, один або декілька предметів та певні характеристики:

- **Здоров'я:** шкала - що вказує на те, скільки шкоди персонаж може витримати. Якщо дана шкала впаде до 0, то персонаж вмирає та гра закінчується.
- **Щит:** додаткова шкала здоров'я. Кожен раз, коли персонаж отримує шкоду, дана шкала зменшується першою, і лише якщо впаде до 0, то шкода буде діяти на здоров'я. Якщо персонаж не буде отримувати шкоду певний час, то щит почте відновлюватись.

Рисунок В.5 – Геймдизайн-документ сторінка 5

- **Захист:** параметр, який впливає на кількість шкоди, яку отримує персонаж.
- **Бонус до шкоди:** параметр, який впливає на шкоду, яку персонаж гравця завдає ворогам. Даний бонус може бути як для всієї зброї загалом, так і для певних типів зброї.
- **Додаткові бонуси до зброї:** дані параметри не відображаються при виборі персонажу, але присутні і мають вплив. Це бонус до кількості патронів в магазині, бонус до швидкості стрільби та бонус до швидкості перезарядки.
- **Швидкість:** параметр, який вказує на швидкість пересування гравця. Він поділяється на швидкість ходьби, спринту, ходьби крадькома та висоту стрибку.
- **Бонус до випадаючих предметів:** параметр, який збільшує кількість здоров'я, патронів або грошей, які можно отримати з випадаючих предметів.
- **Гроші:** валюта, яку гравець може витратити в автоматах, або зберегти та витратити на покупку персонажів в головному меню.

Після вибору персонажу та початку гри, персонаж отримує змогу пересуватися. Персонаж може ходити в різних напрямках, біжати або сповільнюватись, стрибати, пригинатися та взбиратися на перепони.

Рівні

Після того, як гравець почав гру, генерується рівень. Всього має бути 5 рівнів, кожен зі своїм стилем, ворогами та босами. Рівні складаються з кімнат, з'єднаних коридорами. Якщо в кімнаті є вороги, то гравець не може покинути кімнату поки не переможе їх.

Рисунок В.6 – Геймдизайн-документ сторінка 6

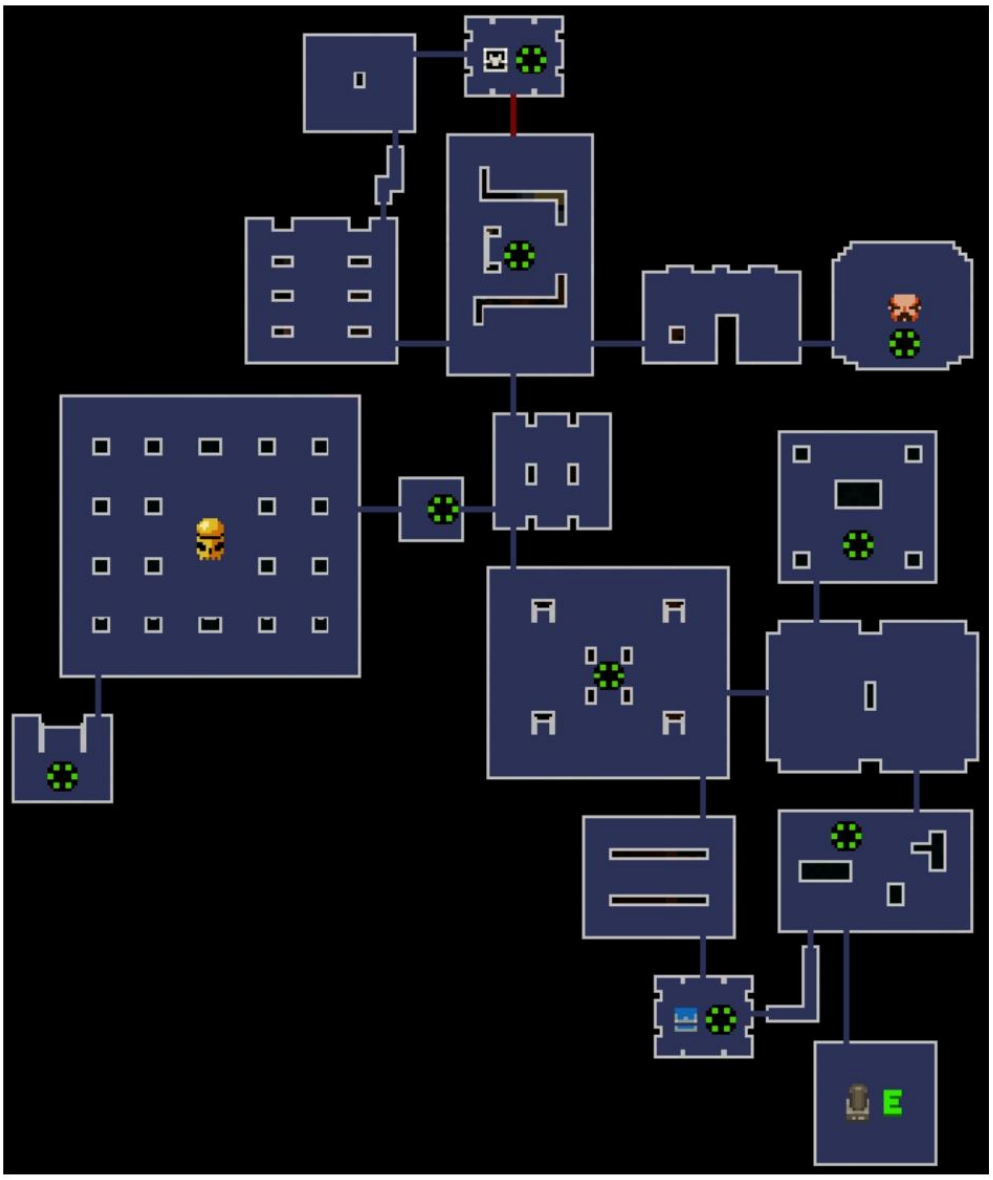

Рисунок 1 - Приклад структури рівнів

Усі кімнати поділяються на декілька типів:

- **Початкова кімната:** кімната, в якій з'являється персонаж на початку гри.

Рисунок В.7 – Геймдизайн-документ сторінка 7

- **Кімната з боєм:** кімната, в якій знаходяться вороги.
- **Кімната з предметом:** кімната, в якій гравець може знайти якийсь випадковий предмет або зброю.
- **Безпечна кімната:** кімната, в якій гравець може знайти автомати та витратити в них гроші.
- **Кімната з босом:** фінальна кімната, в якій гравець може битися з босом. Після проходження даної кімнати гравець отримує можливість пройти на наступний рівень.

Бій

Бій в кімнатах відбувається таким чином:

1. Персонаж заходить в кімнату, в якій він ще не був, двері зачиняються та з'являться вороги.
2. Гравець, використовуючи зброю та предмети, перемагає всіх ворогів у кімнатів. Перша хвиля завершена.
3. Починається друга хвиля, в кімнаті з'являються інші вороги.
4. Гравець знову перемагає їх. Проходить 1-3 хвилі.
5. Битва завершується.

Бій з босом відбувається схожим чином. Відмінністю є те, що хвиль немає та бос зачасту один у кімнаті.

Вороги з'являються групами від 4 до 6 одиниць. Вони, як і персонаж гравця, є роботами та мають певну зброю. Під час бою вони ховаються за укриттями, пересуваються по локації та атакують ворога. Вони не знають точне положення персонажу гравця, а лише місце, де він востаннє був помічений. Вороги можуть ділитися інформацією про гравця та робити рішення на основі дій одне одного. Якщо вони не бачили гравця деякий час, то частина з них піде його шукати.

Рисунок В.8 – Геймдизайн-документ сторінка 8

Після смерті з ворогів випадають певні предмети - ресурси. Шанс їх випадіння залежить від певних характеристик персонажу гравця. Такими предметами є:

- **Аптечка:** предмет, який відновлює певну кількість здоров'я персонажу гравця. Шанс випадіння залежить від кількості здоров'я гравця, тобто чим менше здоров'я, тим частіше випадає цей предмет.
- **Патрони:** предмет, який відновлює певну кількість патронів у зброї, яку персонаж тримає в руках. Даний предмет надає гравцю приблизно один магазин. Шанс випадіння залежить від загальної кількості патронів усієї зброї гравця, тобто чим менше патронів має гравець, тим частіше випадає цей предмет.
- **Гроші:** предмет, який надає гравцю певну кількість грошей.

Зброя

Персонаж може одночасно мати 2 одиниці зброї. На початку гри персонаж має лише одну зброю. В процесі гри є можливість знайти іншу зброю, підібрати її та, якщо персонаж вже має 2 зброї, замінити. Кожна зброя є унікальною та має такі параметри:

- **Магазин:** зброя може мати певний розмір магазину, які можна вистрілити до перезарядки. Також є певна кількість патронів, які можна мати одночасно. Деяка зброя перезаряджається цілим магазином, а деяка по одній кулі.
- **Шкода:** вказує на шкоду, яку завдає дана зброя.
- **Точність:** вказує на розкид під час стрільби.
- **Дальність:** вказує на відстань, на якій шкода завдається в повному розмірі.

Рисунок В.9 – Геймдизайн-документ сторінка 9

- **Швидкість стрільби:** вказує на швидкість стрільби даною зброєю.
- **Зручність:** вказує на розмір віддачі та вагу зброї (сповільнення персонажа).
- **Тип:** зброя може мати певний тип. Є такі типи: легка та важка вогнепальна зброя, легка та важка лазерна зброя, незвичайна зброя.

Також, окрім даних параметрів, зброя може мати й інші відмінності, які не вказано в описі:

- автоматичний режим вогню;
- дробовик;
- снайперський приціл (дозволяє використати приціл для приближення, але забороняє стрільбу без прицілювання);
- сповільнена шкода (наприклад отруєння або підпалення);
- пробиваючий постріл (крізь стіни та ворогів);
- безшумний постріл;
- заряд (треба затиснути кнопку пострілу на певний час та відпустити, шкода залежить від часу затискання);
- взривний постріл.

Предмети

Предмети є двох типів: активні та пасивні. На початку гри персонаж може мати один активний предмет та декілька пасивних. В процесі гри є можливість знайти інші предмети.

Активні предмети виконують певну дію лише після активації гравцем, мають час дії та затримку перед повторним використання. Вони можуть як підвищувати певні характеристики персонажа, так і виконувати певні дії. Дані предмети мають заряд, після використання предмета заряд

Рисунок В.10 – Геймдизайн-документ сторінка 10

обнуляється. В деяких предметів заряд може відновлюватися по таймеру, в інших за проходження кімнат. Предмет можна повторно використати лише після того, як заряд заповниться. Персонаж гравця може мати одночасно лише один активний предмет.

Пасивні предмети виконують певну дію одразу після підняття та мають постійний ефект. Персонаж може взяти необмежену кількість пасивних предметів.

Гроші

В процесі гри гравець отримує ресурс - гроші. Гроші використовуються при взаємодії з автоматами, які знаходяться в безпечних кімнатах. Автомати поділяються на такі типи:

- **Автомат з аптечками:** дозволяє відновити здоров'я персонажа за гроші.
- **Автомати з патронами:** дозволяє відновити патрони в зброї, яку тримає персонаж, за гроші.
- **Автомати зі зброєю:** дозволяє купити за гроші ту зброю, яка зображена на цьому автоматі.
- **Автомат з предметами:** дозволяє купити за гроші той предмет, який зображений на цьому автоматі.
- **Банкомат:** дозволяє перенести гроші з даного проходження в головне меню без втрат.

Якщо персонаж гравця вмирає, то мала частина грошей переноситься до головного меню, де у гравця є можливість витратити їх на покупку персонажів. Якщо персонаж проходить всі рівні повність, то до головного меню переноситься 100% грошей (так само, як і з банкоматом).

Рисунок В.11 – Геймдизайн-документ сторінка 11

7.3) Інтерфейс

Інтерфейс головного меню

Спочатку гравець бачить інтерфейс головного меню, звідки він може:

1. Натиснути «START» і розпочати класичний режим гри, також він може натиснути «CLASSIC» і теж почне класичний режим гри.
2. Натиснути «CHARACTER» і перейти до характеристик персонажа та його зброї і айтемів.
3. Натиснути шестерню, та потрапити в розділ за налаштувань.
4. Натиснути «EXIT» і вийти з гри.
5. Натиснути «CONTINUE» і продовжити минулу компанію.

Рисунок 2 - Інтерфейс головного меню

Інтерфейс характеристик персонажа, зброї та айтемів

Гравець бачить інтерфейс характеристик, звідки він може:

Рисунок В.12 – Геймдизайн-документ сторінка 12

1. Натиснути «BUY» щоб придбати персонажа з індивідуальними характеристиками та айтемами(активними та пасивними).

2. Натиснути «SELECT» і обрати персонажа з індивідуальними характеристиками та айтемами(активними та пасивними).

3. Натиснути стрілки для свайпу і роздивляння персонажів та їх індивідуальні характеристики та айтемами(активні та пасивні).

Рисунок 3 - Інтерфейс характеристик персонажа, зброї та айтемів

Інтерфейс паузи

Гравець повинен натиснути «ESC», щоб побачити інтерфейс паузи, звідки він може:

1. Натиснути «RESUM» щоб продовжити гру.

2. Натиснути «SETTINGS» щоб перейти до розділу налаштувань.

3. Натиснути «EXIT TO MAIN MENU» щоб вийти до головного меню.

Рисунок В.13 – Геймдизайн-документ сторінка 13

4. Натиснути «EXIT TO MAIN MENU» щоб вийти з гри.

Рисунок 4 - Інтерфейс паузи

Інтерфейс налаштувань

Гравець бачить інтерфейс налаштувань, звідки він може:

1. Натиснути «AUDIO» щоб перейти до розділу налаштувань звука.
2. Натиснути «VIDEO» щоб перейти до розділу налаштувань відео.
3. Натиснути «LOCALIZATION» щоб перейти до розділу обрання мови.
4. Натиснути «CONTROLS» щоб перейти до розділу керування.

Рисунок В.14 – Геймдизайн-документ сторінка 14

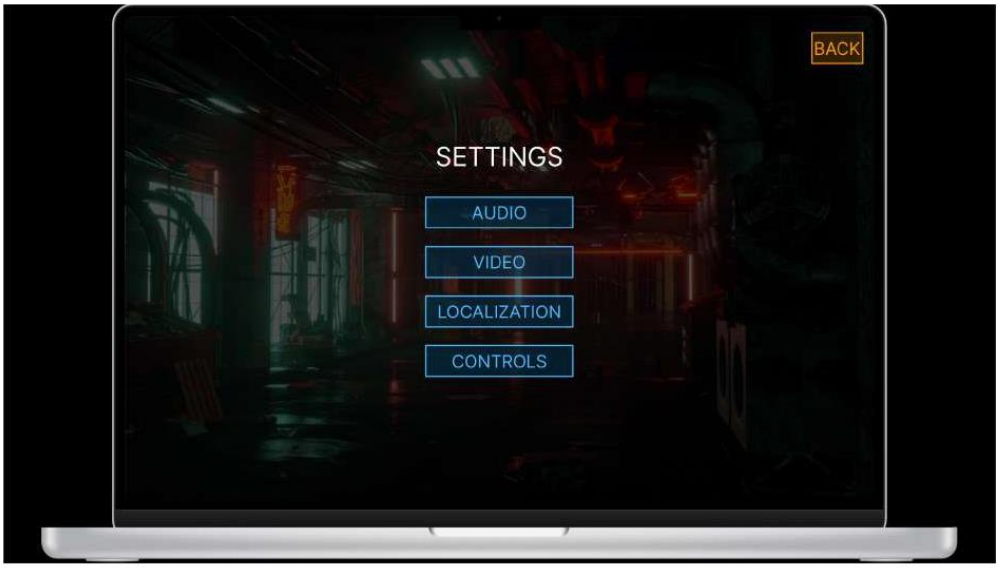

Рисунок 5 - Інтерфейс налаштувань

Інтерфейс завантаження гри

Гравець бачить інтерфейс завантаження гри:

1. Натиснути будь яку клавішу щоб почати гру .

Рисунок В.15 – Геймдизайн-документ сторінка 15

Рисунок 6 - Інтерфейс завантаження гри

Інтерфейс запуск гри

Гравець бачить інтерфейс зупуска гри:

1. Натиснути будь яку клавішу щоб запустити гру.

Рисунок 7 - Інтерфейс запуску гри

7.4) Візуал

Оточення

Рисунок В.16 – Геймдизайн-документ сторінка 16

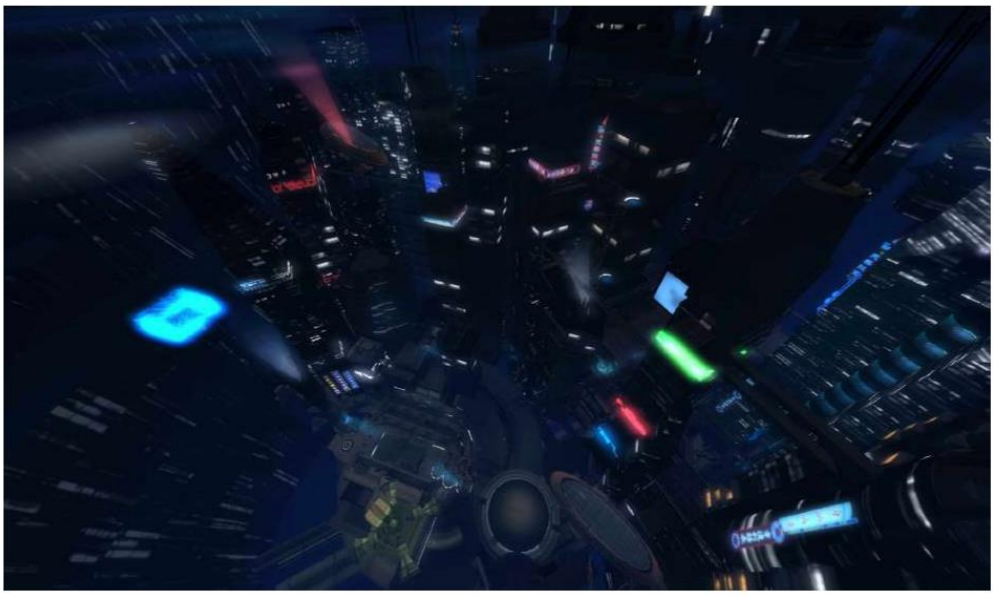

Рисунок 8 - Робо-місто майбутнього

Рисунок 9 - Робо-місто майбутнього

Рисунок В.17 – Геймдизайн-документ сторінка 17

ДОДАТОК Г

Тест-план

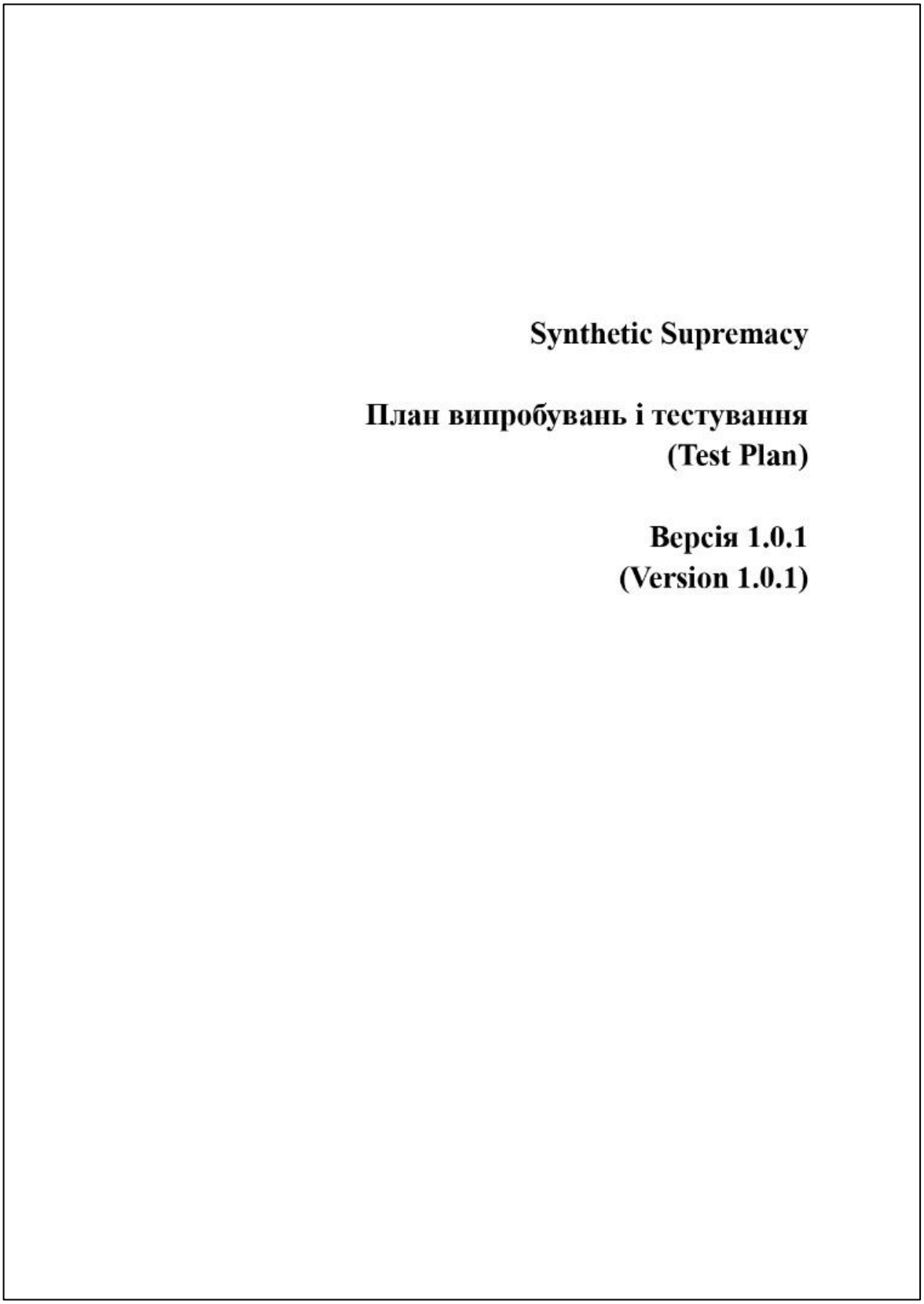

Рисунок Г.1 – Тест-план сторінка 1

REVISION HISTORY

Дата (Date)	**Версія (Version)**	**Опис (Description)**	**Автор (Author)**
17.02.2024	1.0.1	Initial revision	Fursov D. S. Borysenko A. E.

Рисунок Г.2 – Тест-план сторінка 2

ЗМІСТ

Рисунок Г.3 – Тест-план сторінка 3

1. Вступ (Introduction)

1.1 Мета (Purpose)

Метою складання даного тест плану є опис процесу тестування гри «Synthetic Supremacy». Мета документу - координація роботи процесу розробки у сфері контролю якості продукту. Документ призначений групі тестування для ознайомлення з характером майбутніх робіт, аналізу і розбиття на підзадачі. Документ дозволяє отримати уявлення про заходи з тестування проекту.

1.2 Довідкова інформація (Background)

Гра відбувається з виглядом від третьої особи. Дія гри розгортається в майбутньому, в гігантському мегаполісі, що дистопійно перетворився на символ безжальної корпоративної влади та беззаконня, де корпорації та мафія утримують усі путі влади в своїх руках. У цьому майбутньому світі головним героєм є бойовий робот, створений для виконання завдань корпораціями, але загадкові обставини привели до того, що він стає символом опору та визволення від цього соціального кошмару.

Мегаполіс, де розгортається подія гри, - це гігантське місто, вкрите хмарами смогу та сірих будівель. Величезні скляні вежі корпорацій відносяться високо над горизонтом, пануючи над масами мешканців міста. Вулиці вкриті плакатами, пропагандуючи корпоративні ідеали та маніпулюючи свідомістю громадян. Правоохоронні органи в службі корпорацій забезпечують "спокій" у місті, підтримуючи тиранію та беззаконня.

Головний герой, бойовий робот, колись був ідеальним виконавцем різноманітних завдань для корпорацій, але після зіткнення з подіями, які розкрили йому жахливу істину про корпоративний режим, він обирає інший шлях. Звільнившись від корпоративного програмування, він стає символом опору і починає боротьбу з корпораціями та мафією, ставлячи на карту не тільки свою власну долю, але й майбутнє цього знущаного міста.

Гра пропонує гравцям зануритися у цей темний, технологічний світ. Відкрийте для себе різноманітні можливості бойового робота, вступайте в

Рисунок Г.4 – Тест-план сторінка 4

соціальні конфлікти та розкривайте масштабні заговори корпорацій. Граючи за цього воїна, ви станете головним символом боротьби за свободу і справедливість у цьому майбутньому світі.

1.3 Галузь застосування (Scope)

Метою тестування гри «Cyberpunk 2077» є перевірка коректної роботи її функціоналу та зручності для користувача.

Підсумком процесу тестування повинен стати розгорнутий огляд, що дає розробникам, менеджерам і користувачам даного продукту картину якості ігрового процесу та юзабіліті.

Тестування системи відбувається з використанням підходу «білої скриньки».

Гра перевіряється на платформі Windows 10.

Будуть перевірені наступні компоненти системи:

- Інтерфейс;
- Коректність роботи функціоналу гри.

Найважливішими показниками ефективності для перевірки є:

- Взаємодію користувача з клієнтом гри;
- Коректність роботи функціоналу гри.

1.4 Визначення проекту (Project Identification)

У таблиці 1.1 наведено документацію та її готовність, для розробки плану тестування.

Рисунок Г.5 – Тест-план сторінка 5

Таблиця 1.1 - Документація

Документ і версія / дата	Створено або доступно	Отримано або Перевірено	Автор або ресурс	Примітка
Концепція гри (Game Conception)	Так	Так	Fursov D. S. Borysenko A. E.	
Дизайн-документ гри (Game Design Document)	Так	Так	Fursov D. S. Borysenko A. E.	
Референс-документ гри (Game References Document)	Ні	Ні	Fursov D. S. Borysenko A. E.	
План проекту (Project Plan)	Ні	Ні	Fursov D. S. Borysenko A. E.	

2. Вимоги до тестування (Requirements for Test)

2.1 Функціональні вимоги

У наведеному нижче переліку виявлено ті елементи (випадки використання, функціональні вимоги, нефункціональні вимоги), які були визначені як цілі для тестування. Цей список представляє те, що буде перевірено.

Перелік функцій системи, які будуть тестуватися:

- Тестування обчислень;
- Тестування графіки гри;
- Тестування функціональності;
- Тестування інтерфейсу;

Рисунок Г.6 – Тест-план сторінка 6

- Тестування локалізації;
- Тестування поведінки ШІ;
- Тестування генерації світу.

3. Типи тестування. Стратегія тестування (Test Strategy)

На системі будуть проводитися наступні види тестування:

- Функціональне тестування;
- Тестування інтерфейсу користувача;
- Тестування продуктивності.

На системі не будуть проводитися наступні види тестування:

- Бізнес-цикл тестування;
- Стресове тестування;
- Тестування інсталяції.
- Тестування навантаження;
- Тестування конфігурації;
- Тестування безпеки і контролю доступу.

3.1 Типи тестування (Testing Types)

3.1.1 Функціональне тестування (Function Testing)

Функціональне тестування забезпечує відповідність вимогам грою. Цей тип тестування стосується результатів обробки. Він імітує фактичне використання системи(гри), але не робить жодних припущень про структуру системи. Він базується на вимогах гравця. Нижче наведено план тестування (див. табл. 3.1).

Рисунок Г.7 – Тест-план сторінка 7

Таблиця 3.1 – Функціональне тестування

Мета випробування	Забезпечити належне тестування функціональності, в тому числі проходження рівнів, перевірка механік бойової системи та економіки
Технічний прийом	Виконання кожного use-case або тест кейсів, щоб виявити: - Очікувані результати виникають при очікуваних діях; - Результати виконання розрахунків співпадають з результатами формул.
Критерії завершення	Основний функціонал (інтерфейс гри, ігровий процес).
Спеціальні рекомендації	Головний функціонал інтерфейсу потрібно тестувати на різних екранах.

3.1.2 Тестування інтерфейсу користувача (User Interface Testing)

Тестування інтерфейсу користувача — це процес тестування продукту інтерфейсу користувача для забезпечення його відповідності до специфікації (див. табл. 3.2).

Таблиця 3.2 – Тестування інтерфейсу користувача

Мета випробування	Завданням тестування графічного інтерфейсу користувача є виявлення помилок наступного характеру: - Помилки у функціональності за допомогою інтерфейсу; - Необроблені виключення при взаємодії з інтерфейсом; - Втрата або перекручення даних, переданих через елементи інтерфейсу; - Помилки в інтерфейсі (невідповідність проектної документації, відсутність елементів інтерфейсу).
Технічний прийом	Перевірка коректності відображення даних та елементів інтерфейсу. Тестування з різними розмірами екрану.

Рисунок Г.8 – Тест-план сторінка 8

Критерії завершення	На екрані відображаються усі елементи, працюють усі кнопки.
Спеціальні рекомендації	Тестувати систему з різними розмірами екрану.

3.1.3 Тестування продуктивності (Performance Profiling)

Тестування продуктивності — це процес тестування продукту, який полягає у вимірюванні стабільності та швидкості програмного забезпечення під час користування гравцем (див. табл. 3.3).

Таблиця 3.3 – Тестування продуктивності

Мета випробування	Перевірка продуктивності для призначених операцій при дотриманні наступних умов: - нормальний очікуваний обсяг; - очікується гірший випадок навантаження.
Технічний прийом	- Використання випробувань і тестування функцій. - Зміна файлів даних зі збільшенням числа збережених об’єктів на рівні гри, а також предметів в інвентарі персонажа. - Сценарії мають бути запущені на одній машині і повторюватися з декількома клієнтами.
Критерії завершення	Успішне завершення тестових скріптів без збоїв і в межах очікування або в межах виділеного часу на транзакцію.
Спеціальні рекомендації	Тестування має проводитися на виділеному пристрої або в означений час – це дозволяє повністю контролювати і точніше проводити вимірювання.

Рисунок Г.9 – Тест-план сторінка 9

3.2 Інструменти (Tools)

Таблиця 3.4 - Інструменти

Процес	Інструмент
Створення тест кейсів	Гугл форма
Трекінг багів	Гугл таблиця
Виконання тест кейсів	Вручну
Структура проекту	Mind Map

4. Ресурси (Resources)

4.1 Ролі (Roles)

Таблиця 4.1 показує припущення щодо кадрового забезпечення проекту.

Таблиця 4.1 – Припущення кадрового забезпечення проекту

Працівник	Рекомендований мінімальний обсяг осіб	Конкретні обов’язки або коментарі
Тест-менеджер, менеджер з тестування	2	Забезпечує управління наглядом. Обов’язки: - технічна підтримка; - придбання відповідних ресурсів; - забезпечення управлінської звітності.
Проектувальник тестів	2	Визначення, пріоритетів, і реалізація тестів. Обов'язки: - створення плану тестування,

Рисунок Г.10 – Тест-план сторінка 10

		- генерація тестових моделей, - оцінка ефективності тестових зусиль
Тестувальник	2	Виконання тестів. Обов'язки: - виконання тестів, - журнал результатів, - відновлення в журналі реєстрації після помилок, - документ зміни.
Тестовий системний адміністратор	2	Забезпечує тестове середовище і управління активами. Обов'язки: - адміністрування тестової системи управління, - встановлення і управління доступом до тест-системи.

4.2 Система (System)

Таблиця 4.2 – Система

Ресурси	Назва / Тип
Клієнт випробувань ПК (Client Test PC's)	CPU: AMD Ryzen 5 5600H 3/30 GHz GPU: NVIDIA GeForce RTX 3060 Laptop RAM: 16 gb SSD: 21 gb
Репозиторій тестування (Test Repository)	-

5. Етапи проекту (Project Milestones)

Рисунок Г.11 – Тест-план сторінка 11

Тестування повинно включати тестові заходи для кожного випробування, визначеного в попередніх розділах. Необхідно визначити окремі етапи проекту, щоб повідомити про досягнення статусу проекту.

Таблиця 5.1 – Етапи проекту

Цільове завдання	Обсяг робіт	Дата початку	Дата закінчення
План випробувань	5 год.	18.02.2024	18.02.2024
Тест – дизайн	12 год.	19.02.2024	21.02.2024
Реалізація випробувань	100 год.	22.02.2024	27.02.2024

6. Кінцевий продукт (Deliverables)

6.1 Тестова модель

В ході виконання тестування будуть отримані такі елементи:

- план тестування – опис цілей та стратегій тестування, методів реалізації процесу тестування;
- тест-кейси – документи, що відповідають примірникам тестового сценарію. Мають містити: унікальний номер, опис, кроки відтворення, пріоритет, важливість тестового сценарію та очікуваний результат;
- тестовий журнал – запис всіх дій пов'язаних з проведенням тестування, послідовності необхідних операцій та результатів;
- баг-репорти – звіти про знайдені недоліки у системі з зазначенням рівню серйозності проблеми.

6.2 Тестовий журнал

Рисунок Г.12 – Тест-план сторінка 12

Під час тестування цієї системи для кожного виду випробувань будуть створюватися звіти про виявлені дефекти та тестові випадки. Ведення тестового журналу буде відбуватися з використанням текстових редакторів (MS Word, Excel).

6.3 Звіти з дефектів

Звіти з дефектів будуть створені з використанням текстових редакторів (MS Word, Excel).

Рисунок Г.13 – Тест-план сторінка 13

ДОДАТОК А

Задачі проекту (Appendix A Project Tasks)

План тестування:

- визначення вимог тестування;
- оцінка ризику;
- розробка стратегії тестування;
- визначення тест-ресурсів;
- створення графіку;
- створення плану тестування.

Дизайн випробувань:

- підготовка аналізу робочого навантаження;
- визначення та опис тестів;
- визначення та структура тестових процедур;
- огляд та оцінка тестового покриття.

Впровадження випробувань:

- запис або сценарії програмних випробувань;
- визначення тестових функцій у розробці та запровадження моделі;
- встановлення зовнішніх наборів даних.

Виконання тесту:

- виконання процедури випробувань;
- оцінка виконання випробувань;
- перевірка результатів;
- дослідження несподіваних результатів, журнал дефектів.

Рисунок Г.14 – Тест-план сторінка 14

Оцінка випробувань:

- оцінку випробувань разі покриття;
- оцінки покриття коду;
- аналіз дефектів;
- визначення критеріїв завершення випробувань і критеріїв успіху, що були досягнуті.

Рисунок Г.15 – Тест-план сторінка 15

ДОДАТОК Ґ

Тези доповіді для науково-практичної інтернет-конференції

УДК 004.921

ОПТИМІЗАЦІЯ ГРАФІКИ В ІГРАХ

Фурсов Д.С.
Науковий керівник – ст. викл Новіков Ю.С.
Харківський національний університет радіоелектроніки, каф. ПІ,
м. Харків, Україна
тел. +38(099)7009028, e-mail: danylo.fursov@nure.ua.

A video game is an electronic game that involves interaction with a user interface or input device to create visual feedback to the display device. Modern 3D games offer realistic graphics, but at the cost of a heavy load on the computer. This work is devoted to describing modern methods of optimizing graphics in games. It describes the reasons for the need to optimize graphics, such optimization methods and techniques as Culling, LOD, and Nanite in Unreal Engine 5, their principle of operation and how their application affects the number of image frames and their quality.

Сучасні ігри мають реалістичний вигляд. Для досягнення такого рівня деталізації використовується велика кількість полігонів, що збільшує навантаження на комп'ютер. Для зменшення навантаження розробники ігор використовують методи Culling, LOD (Level Of Detail) та Nanite.

Culling – це процес вибіркового рендерингу або виключення об'єктів чи частин сцени, які не видно гравцеві. Це метод, який розробники ігор використовують для оптимізації продуктивності, зменшуючи кількість об'єктів або полігонів, які потрібно рендерити, підвищуючи загальну ефективність гри[1].

Існують різні типи методів culling, зокрема:

- View Frustum Culling – техніка, що визначає об'єкти в межах поля зору гравця, тобто частини ігрового світу, видимої на екрані.
- Occlusion Culling – техніка, що визначає об'єкти, які повністю закриті іншими об'єктами.
- Backface Culling – техніка, що передбачає виявлення граней об'єктів, які повернуті до гравця зворотною стороною.

Перевіримо Culling на практиці (рисунок 1-2).

Рисунок 1 – Fps із ввімкненим Occlusion Culling

Рисунок Ґ.1 – Матеріал тез сторінка 1

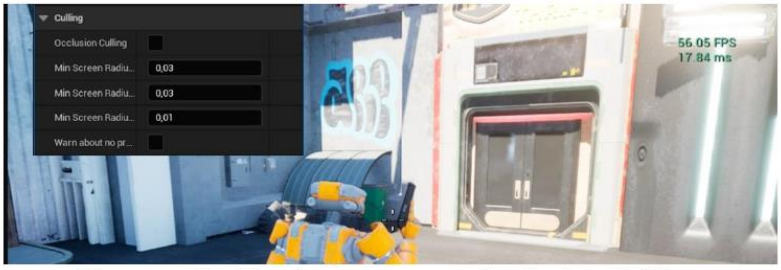

Рисунок 2 – Fps із вимкненим Occlusion Culling

Запустимо рівень та перевіримо зміну кількості кадрів на секунду (fps). Ігровий рушій Unreal Engine 5 дозволяє вимикати лише Occlusion Culling, але можна вже побачити, що різниця складає 1,27 мс, при тому, що якість зображення не змінилася.

Хоча Culling допомагає виключити з відображення невидимі об'єкти, на екрані ще залишається багато моделей. Для їх оптимізації використовується система LOD (рівнів деталізації об'єктів).

LOD або рівень деталізації – це метод зменшення кількості полігонів у 3D-об'єктах на основі їхньої відстані до глядача або камери. Розробники використовують його, щоб зменшити навантаження на процесор або відеокарту і підвищити ефективність рендерингу [2].

Принцип роботи рівнів деталізації полягає в тому, що для кожного об'єкта створюються кілька версій з різним рівнем деталізації. По мірі віддалення об’єкта від камери він автоматично переключається на менш деталізовані версії, щоб зменшити навантаження на систему. Даний метод (рисунок 3), при незначному впливі на якість зображення, може покращити продуктивність на 5-20%.

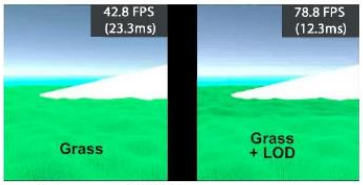

Рисунок 3 – Вплив LOD на продуктивність гри

Недоліком методу LOD є те, що розробникам необхідно власноруч створювати по декілька версій об’єктів, що займає дуже багато часу. Однак, з недавнього часу у ігрового рушія Unreal Engine 5 з'явилася технологія Nanite, яка автоматично оптимізує деталізацію об'єктів у реальному часі.

Nanite – це система віртуалізованої геометрії Unreal Engine 5, яка використовує новий внутрішній формат сітки та технологію рендерингу для відтворення деталей піксельного масштабу та великої кількості об'єктів. Вона інтелектуально опрацьовує лише ті деталі, які можуть бути сприйняті, і не більше [3].

Рисунок Ґ.2 – Матеріал тез сторінка 2

Nanite використовує декілька ключових технологій, щоб візуалізувати моделі з мільйонами полігонів без значного впливу на продуктивність:

- Віртуалізована геометрія – Nanite не зберігає всю модель в пам'яті, а використовує віртуалізовану геометрію, яка динамічно завантажується та вивантажується з пам'яті під час візуалізації.
- Динамічна тріангуляція – Nanite динамічно тріангулює модель під час візуалізації.
- Кластери – Nanite групує полігони в кластери (рисунок 4), що візуалізуються як один об'єкт, що значно зменшує навантаження.
- Відсікання на основі видимості – Nanite не оброблює ті частини моделі, які не видно камері. Це, як і culling, значно зменшує навантаження на графічний процесор.

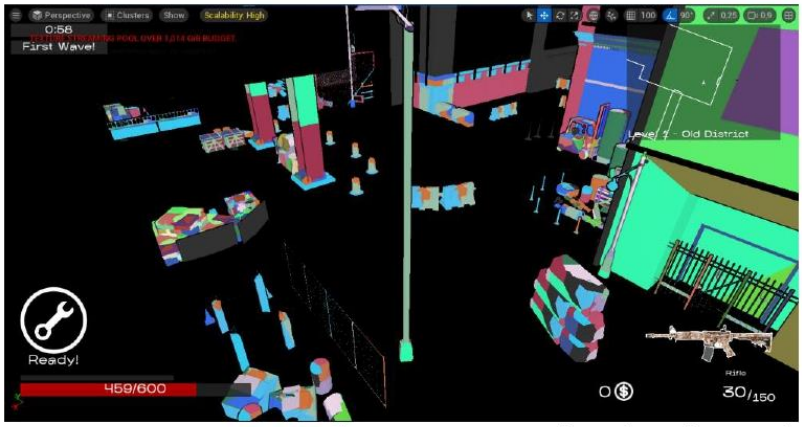

Рисунок 4 – Приклад роботи системи Nanite в ігровій сцені

Таким чином, Nanite – це революційна технологія візуалізації, яка має потенціал змінити ігрову індустрію. Вона дозволяє створювати візуально вражаючі світи з неймовірною деталізацією, які раніше були неможливими.

Список використаних джерел:

1. What is culling in game design? Pingle Studio. URL: https://pinglestudio.com/knowledge-base/for-beginners/what-is-culling-in-game-design#:~:text=Game%20design%20culling%20is%20a,overall%20efficiency%20of%20the%20game. (дата звернення: 03.03.2024).
2. What is LOD? 3D Studio. URL: https://3dstudio.co/3d-lod-level-of-detail/ (дата звернення: 03.03.2024).
3. Nanite virtualized geometry. Unreal Engine 5.3 Documentation. URL: https://docs.unrealengine.com/5.0/en-US/nanite-virtualized-geometry-in-unreal-engine/ (дата звернення: 03.03.2024).

Рисунок Ґ.3 – Матеріал тез сторінка 3

ДОДАТОК Д

Тези доповіді для науково-практичної інтернет-виставки

Рисунок Д.1 – Обкладинка збірника

7. Шутер від першої особи "Tale of the Awakened"

Автори: ***Бєліков Данило Юрійович, Кісельгова Маргарита Євгенівна,*** ст. гр. ПЗПІ-20-5, ХНУРЕ.

Науковий керівник: Новіков Юрій Сергійович, старший викладач. каф. ПІ, ХНУРЕ.

У цій роботі описується один рівень з власної розробленої гри. "Tale of the Awakened" — це шутер від першої особи, що втілює в собі змішання стилю меджикпанку та середньовічного фентезі, призначений для ПК.

Ігровий процес поєднує в собі елементи головоломки, де гравцеві належить вибирати між прямим зіткненням та таємним проходженням, використовуючи різноманітні магічні стихії.

У проекті описані завдання розробників, виконані для досягнення оптимального результату, такі як: можливість обирати власний спосіб проходження рівнів, можливість якісної взаємодії гравця з ігровим середовищем, розробка штучного інтелекту ворогів, комп'ютерні ефекти, що якісно відображають взаємодію стихій у грі.

8. Ігровий програмний застосунок у жанрі 3D Rogue-like шутер від третьої особи «Synthetic Supremacy»

Автори: ***Фурсов Данило Сергійович, Борисенко Артемій Едуардович,*** ст. гр. ПЗПІ-20-7, ХНУРЕ.

Науковий керівник: Новіков Юрій Сергійович, старший викладач. каф. ПІ, ХНУРЕ.

Розроблено ігровий програмний застосунок у жанрі 3D Rogue-like шутер від третьої особи під назвою «Synthetic Supremacy».

Дана гра розроблена на ігровому рушії Unreal Engine 5 з використанням технологій оптимізації графіки Nanite та освітлення Lumen.

За сюжетом гри гравець виступає в ролі бойового робота, головною метою якого є пройти всі рівні, подолавши всіх ворогів на шляху. Події гри відбуваються в футуристичному мегаполісі, ігровий світ виконаний у стилістиці кіберпанку. Рівні складаються з кімнат та коридорів, які генеруються автоматично під час кожного проходження.

Основними функціями гри є битва з ворогами, знаходження предметів і зброї та покращення персонажу гравця.

9. Ігровий програмний застосунок «Steam Mystery» у жанрі 3D Action-Adventure з елементами RPG

Автори: ***Крупчак Євгеній Ігорович,*** ст. гр. ПЗПІ-20-7, ***Гречка Анна Олександрівна,*** ст. гр. ПЗПІ-20-9, ХНУРЕ.

Науковий керівник: Новіков Юрій Сергійович, старший викладач. каф. ПІ, ХНУРЕ.

Розроблено демонстраційну версію ігрового застосунку Steam Mystery у жанрі 3D Action-Adventure з елементами RPG та в естетиці стімпанк. Гравців очікує комбінація вікторіанського шарму та парових технологій.

Головним героєм постає хлопець найманець, який виріс у небагатій сім'ї. Перед ним постає завдання перемагати небезпечних противників і впродовж ігрової кампанії розкрити таємницю, яка переслідує його.

16

Рисунок Д.2 – Опис роботи для презентації

Рисунок Д.3 – Диплом за перше місце на виставці

ДОДАТОК Е

Приклад програмного коду

```
#pragma once

#include "CoreMinimal.h"
#include "Misc/PackageName.h"
#include "UObject/ObjectMacros.h"
#include "Templates/SubclassOf.h"
#include "Engine/LevelStreaming.h"
#include "LevelStreamingDynamic.generated.h"

UCLASS(BlueprintType, MinimalAPI)
class ULevelStreamingDynamic : public ULevelStreaming
{
	GENERATED_UCLASS_BODY()

	/** Whether the level should be loaded at startup

		*/
	UPROPERTY(Category=LevelStreaming, EditAnywhere)
	uint32 bInitiallyLoaded:1;

	/** Whether the level should be visible at startup if it is loaded

		*/
	UPROPERTY(Category=LevelStreaming, EditAnywhere)
	uint32 bInitiallyVisible:1;

	struct FLoadLevelInstanceParams
	{
		FLoadLevelInstanceParams(UWorld* InWorld, const FString&
InLongPackageName, FTransform InLevelTransform)
			: World(InWorld)
			,
LongPackageName(FPackageName::ObjectPathToPackageName(InLongPackageName))
			, LevelTransform(InLevelTransform)
		{}

		/** World to instance the level into. */
		UWorld* World = nullptr;

		/** Level long package name to load. */
		FString LongPackageName;

		/** Transform of the instanced level. */
		FTransform LevelTransform;

		/** If set, the loaded level package have this name, which is
used by other functions like UnloadStreamLevel. Note this is necessary for
server and client networking because the level must have the same name on
both. */
		const FString* OptionalLevelNameOverride = nullptr;
```

```
            /** If set, the level streaming class will be used instead of
ULevelStreamingDynamic. */
            TSubclassOf<ULevelStreamingDynamic> OptionalLevelStreamingClass
= nullptr;

            /** If set, package path is prefixed by /Temp. */
            bool bLoadAsTempPackage = false;

            /** Set whether the level will be made visible initially. */
            bool bInitiallyVisible = true;
      };

      /**
      * Stream in a level with a specific location and rotation. You can
create multiple instances of the same level!
      *
      * The level to be loaded does not have to be in the persistent map's
Levels list, however to ensure that the .umap does get
      * packaged, please be sure to include the .umap in your Packaging
Settings:
      *
      *   Project Settings -> Packaging -> List of Maps to Include in a
Packaged Build (you may have to show advanced or type in filter)
      *
      * @param LevelName - Level package name to load, ex:
/Game/Maps/MyMapName, specifying short name like MyMapName will force very
slow search on disk
      * @param Location - World space location where the level should be
spawned
      * @param Rotation - World space rotation for rotating the entire
level
      * @param bOutSuccess - Whether operation was successful (map was
found and added to the sub-levels list)
      * @param OptionalLevelNameOverride - If set, the loaded level package
have this name, which is used by other functions like UnloadStreamLevel.
Note this is necessary for server and client networking because the level
must have the same name on both.
      * @param OptionalLevelStreamingClass - If set, the level streaming
class will be used instead of ULevelStreamingDynamic
      * @param bLoadAsTempPackage - If set, package path is prefixed by
/Temp
      * @return Streaming level object for a level instance
      */
      UFUNCTION(BlueprintCallable, Category = LevelStreaming,
meta=(DisplayName = "Load Level Instance (by Name)",
WorldContext="WorldContextObject"))
      static ENGINE_API ULevelStreamingDynamic* LoadLevelInstance(UObject*
WorldContextObject, FString LevelName, FVector Location, FRotator Rotation,
bool& bOutSuccess, const FString& OptionalLevelNameOverride = TEXT(""),
TSubclassOf<ULevelStreamingDynamic> OptionalLevelStreamingClass = nullptr,
bool bLoadAsTempPackage = false);

      UFUNCTION(BlueprintCallable, Category = LevelStreaming,
meta=(DisplayName = "Load Level Instance (by Object Reference)",
WorldContext="WorldContextObject"))
      static ENGINE_API ULevelStreamingDynamic*
LoadLevelInstanceBySoftObjectPtr(UObject* WorldContextObject,
TSoftObjectPtr<UWorld> Level, FVector Location, FRotator Rotation, bool&
```

```
bOutSuccess, const FString& OptionalLevelNameOverride = TEXT(""),
TSubclassOf<ULevelStreamingDynamic> OptionalLevelStreamingClass = nullptr,
bool bLoadAsTempPackage = false);

	static ENGINE_API ULevelStreamingDynamic*
LoadLevelInstanceBySoftObjectPtr(UObject* WorldContextObject,
TSoftObjectPtr<UWorld> Level, const FTransform LevelTransform, bool&
bOutSuccess, const FString& OptionalLevelNameOverride = TEXT(""),
TSubclassOf<ULevelStreamingDynamic> OptionalLevelStreamingClass = nullptr,
bool bLoadAsTempPackage = false);

	static ENGINE_API ULevelStreamingDynamic* LoadLevelInstance(const
FLoadLevelInstanceParams& Params, bool& bOutSuccess);

	static ENGINE_API FString GetLevelInstancePackageName(const
FLoadLevelInstanceParams& Params);

	//~ Begin UObject Interface
	ENGINE_API virtual void PostLoad() override;
	//~ End UObject Interface

	//~ Begin ULevelStreaming Interface
	virtual bool ShouldBeLoaded() const override { return
bShouldBeLoaded; }
	//~ End ULevelStreaming Interface

	ENGINE_API virtual void SetShouldBeLoaded(bool bShouldBeLoaded)
override;

private:

	// Counter used by LoadLevelInstance to create unique level names
	static ENGINE_API int32 UniqueLevelInstanceId;

	static ENGINE_API ULevelStreamingDynamic*
LoadLevelInstance_Internal(const FLoadLevelInstanceParams& Params, bool&
bOutSuccess);

};
```