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 Fantasy FPS.
Механіки головного героя, візуальні ефекти
(тема)

Виконав
студент 4 курсу, групи ПЗПІ-20-5
Бєліков Д. Ю.
(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 _____ Дудар З.В.
(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 _____Бєлікову Данилу Юрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Ігровий програмний застосунок у жанрі Fantasy FPS. Механіки головного героя, візуальні ефекти

Затверджена наказом по університету від 20 травня 2024 р. № 471Ст

2. Термін подання студентом роботи до екзаменаційної комісії 07.06.2024

3. Вихідні дані до роботи Розробити ігровий програмний застосунок в жанрі Fantasy шутер від третьої особи, а саме механіки головного героя, візуальні ефекти, 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	08.04.2024	*виконано*
2	Створення специфікації ПЗ	11.04.2024	*виконано*
3	Проектування ПЗ	15.04.202	*виконано*
4	Розробка ПЗ	15.05.2024	*виконано*
5	Тестування ПЗ	17.05.2024	*виконано*
6	Оформлення пояснювальної записки	20.05.2024	*виконано*
7	Підготовка презентації та доповіді	21.05.2024	*виконано*
8	Попередній захист	22.05.2024	*виконано*
9	Нормоконтроль, рецензування	25.05.2024	*виконано*
10	Здача роботи у електронний архів	01.06.202	*виконано*
11	Допуск до захисту у зав. кафедри	04.06.2024	*виконано*

Дата видачі завдання 8 квітня 2024 р.

Студент (ка)_____ Бєліков Д. Ю.
(підпис)

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

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 90 сторінок, 83 рисунків, 10 джерел.

ВІЗУАЛЬНІ ЕФЕКТИ, ІГРОВИЙ ПРОГРАМНИЙ ЗАСТОСУНОК, КОМП'ЮТЕРНІ ІГРИ, ТЕСТУВАННЯ, ШУТЕР, BLUEPRINT, FPS, NIAGARA SYSTEM, UNREAL ENGINE 5.

Об’єкт розробки – механіки головного героя та візуальні ефекти, які є частиною комплексної роботи з розробки ігрового програмного застосунку у жанрі Fantasy FPS.

Мета розробки – створення ігрового додатка, здатного забезпечити високий рівень залучення користувача та ефективну взаємодію з ігровим середовищем.

Метод рішення – система візуального програмування Blueprint, ігровий рушій – Unreal Engine 5. Система контролю версій Git.

У результаті розробки створено комплексні механіки для головного героя, включаючи анімації, рух, атаки та спеціальні здібності, а також реалізовані візуальні ефекти, що підсилюють атмосферу гри. Усе розроблене є частиною комплексної роботи з розробки ігрового програмного застосунку у жанрі Fantasy FPS.

BLUEPRINT, COMPUTER GAMES, FPS, GAME SOFTWARE APPLICATION, NIAGARA SYSTEM, SHOOTER, TESTING, UNREAL ENGINE 5, VISUAL EFFECTS.

The development object is the main character mechanics and visual effects, which are part of the comprehensive work on the development of a game software application in the Fantasy FPS genre.

The purpose of the development is to create a game application that can provide a high level of user involvement and effective interaction with the game environment.

The solution method is the Blueprint visual programming system, the game engine

is Unreal Engine 5. Git version control system.

As a result of the development, comprehensive mechanics for the main character were created, including animations, movement, attacks, and special abilities, as well as visual effects that enhance the game's atmosphere. Everything developed is part of the comprehensive work on the development of a game software application in the Fantasy FPS genre.

Я, Бєліков Данило Юрійович, студент гр. ПЗПІ-20-5, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Ігровий програмний застосунок у жанрі Fantasy FPS», 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 ознайомлена 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

Сучасна індустрія відеоігор розвивається надзвичайно швидкими темпами, впроваджуючи нові технології, що надають розробникам можливість створювати захоплюючий та інноваційний контент. Ця галузь постійно змінюється, зростає кількість користувачів та інвестиції в розробку ігор для різних платформ. Жанр шутерів від першої особи (FPS) займає ключову позицію в індустрії відеоігор завдяки своїй здатності забезпечувати гравцям інтенсивний та захопливий досвід. Ігри цього жанру характеризуються динамічним геймплеєм, що вимагає швидкої реакції та стратегічного мислення. Від першої особи гравці можуть максимально відчути себе частиною віртуального світу, що робить кожен бій більш реалістичним та напруженим. Саме через ці особливості жанр FPS залишається популярним серед гравців, які шукають активну взаємодію та глибоке занурення в гру.

Ця дипломна робота має на меті створити ігровий додаток, який інтегрує ігрові механіки з насиченою сюжетною лінією, надаючи гравцям унікальний рівень взаємодії з грою. Дослідження включає в себе аналіз сучасних тенденцій у відеоігровій індустрії, проектування архітектури проекту, програмування ігрових механік, а також їх тестування та оптимізацію.

Однією з основних характеристик цього проекту є розробка системи інтерактивності, яка дозволяє гравцям використовувати магічні елементи для впливу на оточення та вирішення різноманітних завдань. Це передбачає створення механізмів для застосування магічних атак, оборонних дій, а також комбінування різних елементів для досягнення конкретних цілей у грі.

Результати цієї роботи будуть актуальні для сфери цифрових розваг, зокрема для розробки відеоігор на персональні комп'ютери та ігрові консолі.

Для реалізації практичної частини проекту використовувалася система візуального програмування Blueprint та ігровий рушій Unreal Engine 5. Для контролю версій використовувалася система Git.

Кінцевий результат дипломної роботи передбачає створення не лише готового до випуску ігрового додатка, але й платформи для подальшого розвитку та оновлення, з можливістю додавання нових ігрових механік та розширення ігрового світу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Індустрія відеоігор постійно еволюціонує, впроваджуючи новітні технології, що змінюють можливості та сприйняття геймплею. Жанр шутерів від першої особи (FPS) займає особливе місце в сучасних відеоіграх, поєднуючи інтенсивні бойові дії з глибокими стратегіями та тактиками. Гравці отримують унікальний досвід, опиняючись в центрі подій, де кожен їхній крок та рішення мають вагомі наслідки. Успіх FPS-ігор багато в чому залежить від здатності інтегрувати новітні технології для створення більш реалістичного та захопливого ігрового середовища.

Одним із найяскравіших прикладів сучасних FPS-ігор є "Doom" (2016) та "Doom Eternal", які відомі своїми винятковими візуальними ефектами та динамічним геймплеєм. Історія цієї серії бере початок ще з 90-х років, коли оригінальні "Doom" та "Doom II" встановили нові стандарти для жанру. Оновлені версії продовжують традицію, пропонуючи гравцям вражаючі графічні рішення, що значно підвищують рівень занурення в гру. Візуальні ефекти в цих іграх підкреслюють інтенсивність боїв, роблячи кожен постріл і вибух максимально реалістичним та захопливим.

Що стосується ігрових механік, серія "Dishonored" є чудовим прикладом різноманітних і інноваційних рішень. Гравці мають можливість використовувати різні здібності персонажів, що дозволяє підходити до виконання завдань різними способами. Ці механіки включають в себе стелс-елементи, магічні здібності, і широкий спектр можливостей для взаємодії з оточенням. Завдяки цьому, "Dishonored" пропонує гравцям багатий ігровий досвід, що спонукає до експериментів та пошуку власних рішень.

1.2 Виявлення проблем та актуалізація рішень

Після проведення аналізу предметної галузі, можемо сказати, що існує ряд проблем, з якими зазвичай зустрічаються подібні проекти. Отже, розглянемо декілька ігрових додатків, що є конкурентними для нашого. Їх варто розглядати з погляду обраної теми у межах комплексної дипломної роботи, тому аналізуватимуться не лише ігри близькі за жанром до розроблюваного продукту (Fantasy FPS), а й інші, наприклад FPS інших піджанрів.

Першим конкурентом для розгляду буде Doom Eternal – це шутер від першої особи, розроблений компанією id Software і випущений Bethesda Softworks у березні 2020 року. Гра відзначається своїми вражаючими візуальними ефектами та динамічними бойовими механіками. Використання рушія id Tech 7 дозволяє досягти високого рівня деталізації, реалістичного освітлення та ефектів частинок, що створює захопливу та напружену атмосферу. Розглянемо ключові механіки гри. Бойові механіки – Doom Eternal відомий своїм швидким та інтенсивним геймплеєм, який вимагає від гравців постійної мобільності та використання різних видів зброї та здібностей (див. рис. 1.1).

Рисунок 1.1 – Бойові механіки в Doom Eternal

До переваг можна віднести:

- Висока швидкість та динаміка боїв, що забезпечує захопливий геймплей;
- Різноманітність зброї та здібностей, що дозволяє гравцям адаптуватися до різних ситуацій.

Основні недоліки:

- Високий темп гри може бути складним для новачків;
- Інтенсивність боїв може призводити до швидкої втоми гравців.

Візуальні ефекти – гра використовує численні ефекти частинок, освітлення та тіні, щоб створити максимально реалістичну і захопливу атмосферу (див. рис. 1.2).

Рисунок 1.2 – Візуальні ефекти в Doom Eternal

До переваг можна віднести:

- Високий рівень деталізації та реалістичність візуальних ефектів;
- Ефекти освітлення та тіні, що підсилюють атмосферу гри.

Основні недоліки:

- Високі вимоги до апаратного забезпечення;
- Надлишок візуальних ефектів може іноді перевантажувати екран, роблячи

важким сприйняття важливих деталей.

Що стосується ігрових механік, серія Dishonored пропонує багатий досвід у цьому напрямку. Dishonored 2, розроблена Arkane Studios і випущена Bethesda Softworks у листопаді 2016 року, надає гравцям можливість використовувати різноманітні здібності персонажів для досягнення своїх цілей. Гра поєднує стелс-механіки з магічними здібностями, що дозволяє гравцям підходити до завдань різними способами. Гра також вирізняється вражаючим візуальним стилем, що підсилює атмосферу та занурення. Розглянемо ключові механіки гри.

Стелс-елементи. Використання стелс-механік дозволяє гравцям тихо та ефективно проходити рівні, уникаючи прямого зіткнення з ворогами (див. рис. 1.3).

Рисунок 1.3 – Стелс-механіки в Dishonored 2

До переваг можна віднести:

- Висока варіативність геймплею завдяки можливості обирати між стелс- та бойовим підходами;
- Можливість використання оточення для прихованого проходження.

Основні недоліки:

- Складність управління стелс-здібностями може викликати труднощі у новачків;
- Деякі механіки можуть бути надто потужними, порушуючи баланс гри.

Магічні здібності. Гравці можуть використовувати різноманітні магічні здібності для вирішення завдань та боротьби з ворогами (див. рис. 1.4).

Рисунок 1.4 – Магічні здібності в Dishonored 2

До переваг можна віднести:

- Велика кількість можливостей для творчого підходу до проходження гри;
- Візуальні ефекти, що підсилюють інтенсивність та захопливість боїв.

Основні недоліки:

- Високі вимоги до управління можуть викликати труднощі у новачків;
- Надлишок можливостей може ускладнити вибір оптимальної стратегії.

Графіка та візуальні ефекти. Dishonored 2 використовує стильну графіку та деталізацію, щоб створити унікальний візуальний стиль, що підсилює атмосферу гри (див. рис. 1.5).

Рисунок 1.5 – Візуальні ефекти в Dishonored 2

До переваг можна віднести:

- Унікальний візуальний стиль, що вирізняє гру серед інших;
- Висока деталізація та якість графіки, що підсилюють атмосферу гри.

Основні недоліки:

- Деякі графічні елементи можуть бути надто навантаженими.

Наступною грою, яку розглянемо, буде Control – гра, розроблена Remedy Entertainment і випущена 505 Games у серпні 2019 року. Гра відзначається своїми інноваційними механіками та вражаючими візуальними ефектами, що створюють захопливий геймплей. Основна механіка гри включає використання телекінезу для взаємодії з оточенням та боротьби з ворогами (див. рис. 1.6). Це надає гравцям нові можливості для творчого підходу до боїв.

Рисунок 1.6 – Телекінез в Control

До переваг можна віднести:

- Інноваційний підхід до боїв завдяки використанню телекінезу;
- Можливість для взаємодії з оточенням, що додає глибини геймплею.

Основні недоліки:

- Обмежена взаємодія з оточенням: Не всі предмети в грі можуть бути підняті або використані за допомогою телекінезу. Це обмежує можливості гравця у використанні цієї здібності для вирішення всіх бойових і тактичних завдань.

Візуальні ефекти. Гра використовує численні ефекти частинок, освітлення та тіні для створення унікальної та захопливої атмосфери (див. рис. 1.7).

Окрім цього, Control демонструє високий рівень деталізації та якісну реалізацію графіки, що в поєднанні з відмінною роботою з освітленням створює особливу атмосферу, яка захоплює гравців. Особлива увага приділяється також аудіовізуальному супроводу, що підсилює ефект занурення у гру. Звукові ефекти та музика гармонійно доповнюють візуальні елементи, підкреслюючи напружені моменти та змінюючи настрій в залежності від ситуації.

Рисунок 1.7 – Візуальні ефекти в Control

До переваг можна віднести:

- Високий рівень деталізації та реалістичність візуальних ефектів;
- Ефекти освітлення та тіні, що підсилюють атмосферу гри.

Основні недоліки:

- Високі вимоги до апаратного забезпечення;

На основі аналізу ігор Doom Eternal, Dishonored 2 та Control можна зробити висновки про ключові аспекти механік головного героя та візуальних ефектів у жанрі FPS. Механіки головного героя повинні бути різноманітними та адаптивними, щоб забезпечити гравцям свободу вибору та варіативність геймплею. Візуальні ефекти повинні підсилювати атмосферу гри та забезпечувати високий рівень занурення, але не повинні перевантажувати екран, щоб уникнути перевантаження гравців.

Графіка та візуальні ефекти відіграють ключову роль у створенні атмосфери та занурення в гру. Використання сучасних технологій дозволяє досягти високої якості зображення, реалістичного освітлення та ефектів частинок, що робить ігровий процес більш захопливим та реалістичним.

Основні висновки:

- Деталізація світу та реалістичне освітлення створюють глибоке занурення в ігровий світ;
- Динамічні візуальні ефекти підсилюють інтенсивність боїв та роблять гру більш захопливою;
- Високі вимоги до апаратного забезпечення можуть обмежити доступність гри для широкої аудиторії.

Отже, аналіз конкурентних ігор, таких як Doom Eternal, Dishonored 2 та Control, надав нам важливі інсайти щодо ключових аспектів, які роблять ці проекти успішними. Вивчення їхніх механік та візуальних ефектів дозволило нам визначити, які елементи слід взяти до уваги для покращення нашого продукту.

1.3 Постановка задачі

Нижче наведено механіки та задачі, які повинні бути реалізовані в ігровому застосунку у рамках дипломного проекту. Цей розділ сфокусований на механіках головного героя та візуальних ефектах.

Механіки головного героя:

- Переміщення персонажа: Реалізація плавних та реалістичних анімацій для руху персонажа, включаючи біг, присідання, стрибки та інші види переміщення. Забезпечення природного переходу між різними станами руху.
- Бойові дії: Впровадження різних типів атак, включаючи ближній бій та стрільбу з різних видів зброї. Анімації повинні бути динамічними та відповідати різним типам зброї та ситуаціям.
- Взаємодія з оточенням: Розробка механізмів взаємодії з об'єктами на рівні, такими як підбирання предметів, відкривання дверей, активація механізмів. Це включає в себе також імплементацію фізичних реакцій об'єктів на дії гравця.

- Спеціальні здібності: Впровадження унікальних здібностей персонажа, таких як телекінез, використання магії або інших надприродних здібностей. Реалізація механік, що дозволяють гравцеві використовувати ці здібності для вирішення завдань та боротьби з ворогами.

Візуальні ефекти:

- Ефекти атак: Створення вражаючих візуальних ефектів для різних типів атак, таких як вибухи, сліди від куль, магічні спалахи. Використання системи Niagara для реалізації складних ефектів частинок.
- Освітлення: Динамічне освітлення, що реагує на дії гравця та зміну умов у грі. Включення ефектів глобального освітлення, тіней та відблисків для підсилення реалістичності.
- Анімації оточення: Впровадження анімацій для об'єктів навколишнього середовища, таких як рухливі частини механізмів, реакції рослин на дії гравця. Це підсилює занурення гравця у ігровий світ.
- Погодні ефекти: Реалізація різноманітних погодних умов, таких як дощ, сніг, туман. Використання динамічних ефектів для створення реалістичної атмосфери.

Реалізація зазначених механік та візуальних ефектів є критично важливою для досягнення високої якості геймплею та занурення гравця у гру. Ці компоненти не тільки покращують взаємодію гравця з ігровим середовищем, але й роблять ігровий процес більш захопливим та реалістичним. Впровадження цих елементів дозволить створити інноваційний та конкурентоспроможний продукт, який відповідатиме сучасним стандартам якості та очікуванням гравців.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Визначення вимог до програмної системи є одним з найбільш важливих етапів при створенні ігрового додатку, оскільки воно задає основні функціональні та нефункціональні характеристики, які повинні бути реалізовані для досягнення цілей проекту. Цей процес включає чітке окреслення того, що саме повинна робити програма, як вона буде взаємодіяти з користувачами, які технологічні рішення будуть використовуватись та які обмеження слід врахувати.

У цій частині дипломного проекту будуть визначені основні функціональні вимоги до механік головного героя та візуальних ефектів, а також забезпечення необхідної взаємодії користувача з програмною системою. Крім того, будуть зазначені ключові технічні та операційні характеристики, які повинні бути реалізовані для досягнення високої якості геймплею та користувацького досвіду.

Розглянемо детальніше функціональні вимоги для забезпечення необхідної взаємодії гравця з ігровим додатком. Вони включають такі функції:

- Атаки: Реалізація механік стрільби та завдання ушкоджень, включаючи фізичне відображення ефектів попадання різних типів снарядів, таких як магічні або стихійні.
- Відновлення здоров'я: Механізми відновлення здоров’я через споживання їжі або інших ресурсів, що забезпечують можливість відновлення гравця під час гри.
- Рух персонажа: Реалізація плавних анімацій для бігу, стрибків, прискорення та руху навпочіпки, що підвищує реалістичність ігрового процесу.
- Взаємодія з предметами: Механізми піднімання, переміщення та використання предметів на рівні.

Також будуть розглянуті вимоги до візуальних ефектів:

- Ефекти атак: Створення вражаючих візуальних ефектів для різних типів атак, таких як вибухи, сліди від куль, магічні спалахи, з використанням системи Niagara для реалізації складних ефектів частинок.

- Освітлення: Динамічне освітлення, що реагує на дії гравця та зміну умов у грі, включаючи ефекти глобального освітлення, тіней та відблисків для підсилення реалістичності.
- Анімації оточення: Впровадження анімацій для об'єктів навколишнього середовища, таких як рухливі частини механізмів або реакції рослин на дії гравця, що підсилює занурення гравця у ігровий світ.
- Погодні умови: Реалізація різноманітних погодних ефектів, таких як дощ, сніг, туман, з використанням динамічних ефектів для створення реалістичної атмосфери.

Таким чином, реалізація наведених функціональних вимог до механік головного героя та візуальних ефектів забезпечить створення якісного та захоплюючого ігрового досвіду. Важливою частиною проекту є також інтеграція динамічних ефектів та анімацій, що підсилюють візуальну привабливість гри та створюють глибоке занурення у ігровий світ. Використання сучасних технологічних рішень та ретельне опрацювання деталей гарантують високий рівень реалістичності та інтерактивності, що сприятиме утриманню інтересу гравців та позитивному сприйняттю ігрового додатку.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проектування ПЗ

У процесі розробки ігрового застосунку, була побудована загальна Use Case діаграма для всього проекту (див. рис. 3.1).

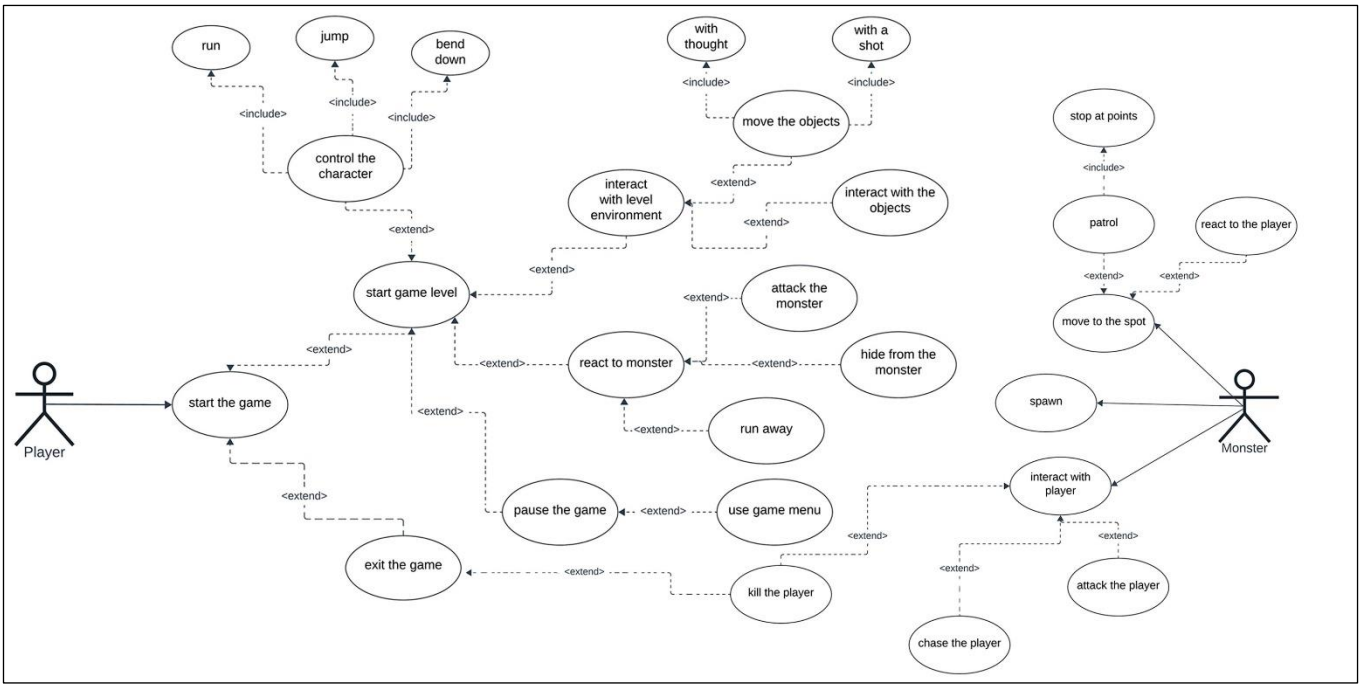


Рисунок 3.1 – Use Case діаграма ігрового застосунку

Розглянувши загальну діаграму системи, можна виділити основних акторів. "Гравець" (Player) – це ключовий актор в ігровій системі, який представляє користувача, що взаємодіє з ігровим світом. Гравець контролює головного персонажа, здійснюючи різні дії, такі як рух, використання магії, боротьба з ворогами, взаємодія з оточенням та вирішення завдань на кшталт пошуку предметів чи розгадування головоломок. Гравець також приймає стратегічні рішення, які впливають на розвиток сюжету та взаємодію з іншими персонажами гри.

Далі розглянемо основні дії гравця:

1. Start the Game (Розпочати гру): Гравець запускає нову гру або продовжує збережену.
2. Control the Character (Керування персонажем): Основний процес управління персонажем, який включає:

- Run (Бігти): Швидке переміщення персонажа по ігровому світу.
- Jump (Стрибати): Подолання перешкод або переміщення через них.
- Bend Down (Нагнутися): Ухиляння або доступ до об'єктів нижче звичайного рівня.

3. Interact with Level Environment (Взаємодія з середовищем рівня): Гравець може взаємодіяти з об'єктами або елементами ігрового середовища, що може включати:
    - Move the Objects (Рухати об'єкти): Переміщення об'єктів у рамках ігрового світу.
    - Telekinesis (Телекинез): Використання телекинезу для взаємодії зі складними або інтерактивними об'єктами.
    - Shooting (Стрільба): Взаємодія із середовищем за допомогою зброї.
4. React to Monster (Реагування на монстра): Відповідь на дії або присутність ворога, що може включати:
    - Run Away (Втекти): Відступ від зустрічі з ворогом.
    - Attack the Monster (Атакувати ворога): Атака на ворога.
    - Hide from the Monster (Сховатися від ворога): Заховатися для уникнення виявлення або атаки.
5. Pause the Game (Пауза в грі): Тимчасове призупинення ігрового процесу.
6. Use Game Menu (Використання ігрового меню): Взаємодія з меню гри для зміни налаштувань, збереження гри або виходу з неї.
7. Exit the Game (Вихід з гри): Закінчення ігрового сеансу і вихід з гри.

Ця частина описує основні дії гравця в ігровому додатку. Вона включає опис Use Case діаграми, що відображає загальні функції проекту.

3.2 Вибір архітектури та рушія

Для розробки ігрового застосунку було вирішено використовувати

найсучасніші інструменти та платформи, які відповідають технологічним вимогам проекту. Вибір відповідних платформ і інструментів є критично важливим для підвищення ефективності розробки, впровадження запланованих ігрових механік і досягнення високої якості кінцевого продукту.

Для реалізації проекту обрано Unreal Engine 5, оскільки цей ігровий рушій надає передові можливості для створення детальних і візуально вражаючих ігрових світів. Unreal Engine 5 відомий своєю підтримкою складних анімацій та освітлювальних ефектів, що дозволяє створювати реалістичні та атмосферні сцени, особливо важливі для жанру FPS.

Система візуального програмування Blueprint була обрана через її здатність забезпечити швидку та інтуїтивно зрозумілу розробку ігрових механік. Ця система дозволяє дизайнерам та розробникам легко співпрацювати, знижуючи необхідність глибоких знань у програмуванні, що оптимізує процес розробки та зменшує час, необхідний для впровадження складних ігрових функцій.

Система контролю версій Git була обрана завдяки своїй надійності та широкому визнанню в індустрії. Використання Git дозволяє ефективно управляти змінами у проекті, полегшуючи співпрацю між членами команди та забезпечуючи безпеку коду. Це є критично важливим для підтримки стабільності та масштабованості проекту, особливо в умовах багатокомпонентної розробки ігрового програмного забезпечення.

Таким чином, вибір Unreal Engine 5, Blueprint та Git дозволяє забезпечити ефективну розробку ігрового застосунку, відповідаючи високим вимогам до якості, продуктивності та стабільності проекту.

3.3 Розгляд ігрового циклу

Цей розділ присвячений детальному аналізу основних компонентів та процесів, що визначають взаємодію гравця з ігровим світом. Розглядаються динамічні

послідовності дій і реакцій, які відбуваються від моменту запуску гри до завершення ігрового сеансу. Центральне місце в аналізі займає ігровий цикл — повторюваний процес, який визначає загальний ритм і динаміку гри, впливаючи на залученість та досвід гравця. Ігровий цикл розроблюваного додатку зображений на малюнку (див. рис. 3.2).

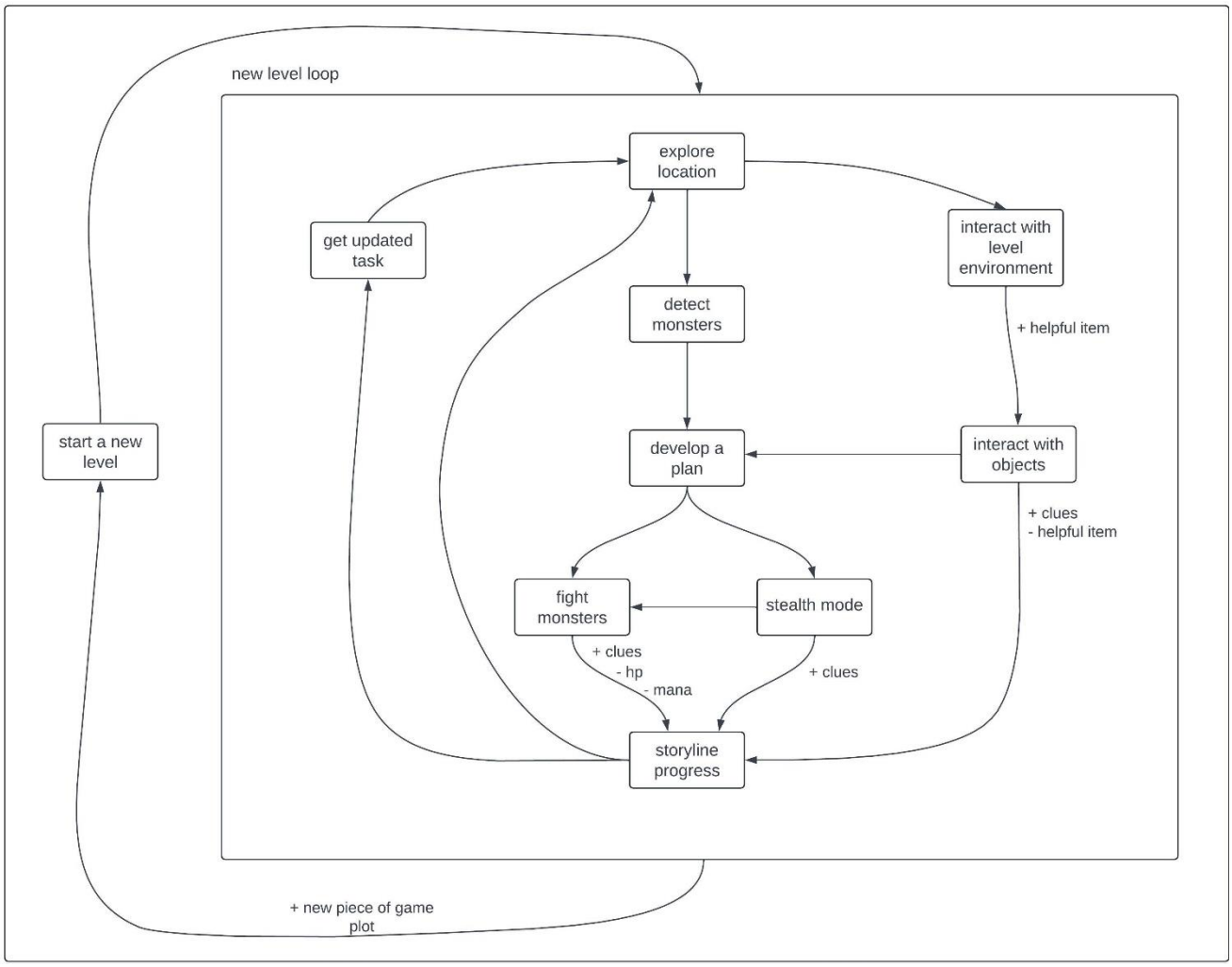


Рисунок 3.2 – Ігровий цикл програмного застосунку

Розглянувши схему ігрового циклу, можна виділити наступні етапи:

1. Запуск нового рівня:

   - Гравець починає новий рівень, що є початком нового ігрового циклу.

2. Отримання нових завдань:

- На початку рівня гравець отримує нові завдання або місії, які необхідно виконати для просування по сюжету.

3. Дослідження середовища:
   - Гравець досліджує нові території, переміщається по карті та взаємодіє з різними елементами ігрового світу.
4. Зустріч з ворогами:
   - Під час дослідження гравець може стикнутися з ворогами, яких необхідно перемогти або уникнути.
5. Планування дій:
   - Після виявлення ворогів гравець розробляє стратегію або план дій, який може включати бій або стелс-підхід.
6. Взаємодія з об'єктами:
   - Гравець взаємодіє з предметами на рівні, які можуть надати підказки або корисні предмети для подальшого просування в грі.
7. Бойові дії:
   - Якщо гравець обирає стратегію бою, він атакує ворогів, використовуючи доступні засоби і навички. Бої можуть принести нові підказки, зменшити здоров'я чи ману.
8. Стелс-підхід:
   - Як альтернативу бою, гравець може обирати стелс-підхід для уникнення ворогів, що також може надати підказки для виконання завдань.
9. Розвиток сюжету:
   - В залежності від вибору дій та результатів взаємодій на рівні, сюжет гри просувається вперед, відкриваючи нові частини історії або завдання.

Розуміння цього циклу є ключовим для оптимізації ігрової механіки та створення захоплюючих ігрових сценаріїв, які стимулюють гравців повертатися до гри знову і знову.

3.4 Розгляд найцікавіших алгоритмів

Одним із найцікавіших алгоритмів ігрового застосунку можна виділити алгоритм системи частинок Niagara для ефекту Ice Vortex (див. рис. 3.3). Цей алгоритм було обрано як найбільш повноцінний та оптимальний з частини проекту, що розробляється в рамках дипломної роботи.

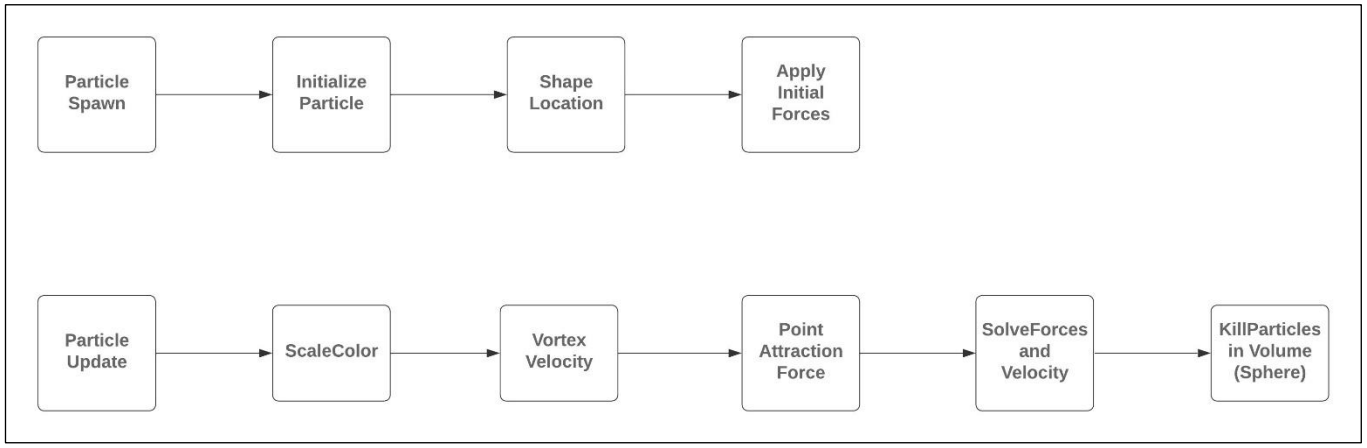


Рисунок 3.3 – Алгоритм системи частинок Niagara (Ice Vortex)

Детально розглянувши наведений алгоритм, можна виділити такі важливі його етапи:

1. ParticleSpawn (Створення частинок):
   - InitializeParticle (Ініціалізація частинки): Ініціалізація нової частинки з початковими параметрами.
   - ShapeLocation (Sphere) (Розташування в формі сфери): Установка початкової позиції частинки у формі сфери.
   - Apply Initial Forces (Застосування початкових сил): Застосування початкових сил до частинок для надання їм початкового імпульсу.
2. ParticleUpdate (Оновлення частинок):
   - ScaleColor (Зміна кольору): Плавне зменшення значення в alpha-каналі кольору частинки за допомогою кривої, щоб частинка поступово зникала.
   - VortexVelocity (Вихрова швидкість): Застосування випадкових швидкостей для хаотичного руху частинок.

- PointAttractionForce (Сила притягування до точки): Застосування сили притягування, щоб частинки поступово наближалися до центру.
- SolveForces and Velocity (Розрахунок сил і швидкостей): Обчислення нових значень сил і швидкостей частинок.
- KillParticles in Volume (Sphere) (Знищення частинок в об'ємі сфери): Видалення частинок, які наблизилися занадто близько до центру.

Розглянутий алгоритм системи частинок Niagara є найбільш цікавим з точки зору технічної реалізації тієї частини ігрового застосунку, що розробляється в рамках дипломного проекту.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Розробка механік головного героя

Створення механік головного героя є ключовим етапом у розробці ігрового застосунку, оскільки саме від цих механік залежить геймплей та загальне сприйняття гри користувачами. При розробці механік важливо дотримуватись принципів дизайну, які забезпечують збалансованість та цікавість ігрового процесу. Основні механіки головного героя:

1. Рух персонажа – забезпечення плавних та реалістичних анімацій для різних видів руху, таких як біг, стрибки, присідання та інші переміщення. Необхідно забезпечити природні переходи між різними станами, щоб гравець мав відчуття повного контролю над персонажем.
2. Бойові дії – Розробка різних типів стихійної зброї та реалізація їх взаємодії між собою, що підвищує реалістичність боїв і робить їх більш захоплюючими (див. рис. 4.1).

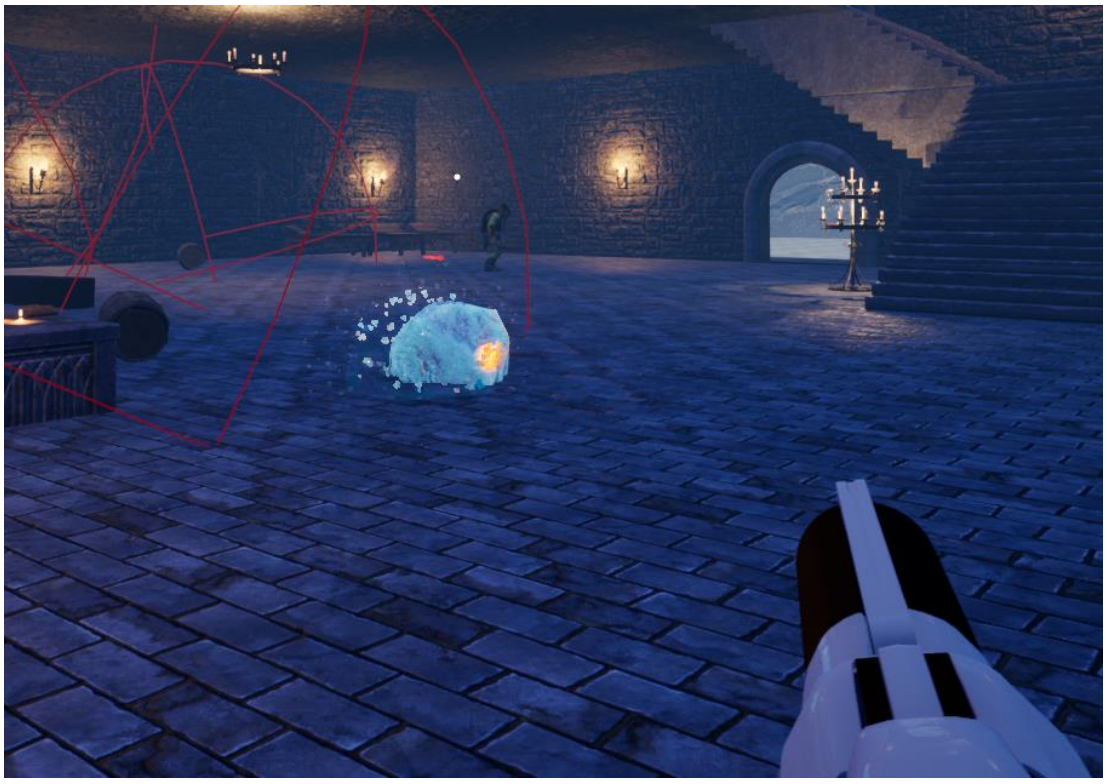


Рисунок 4.1 Приклад взаємодії вогню з льодом

3. Взаємодія з оточенням – розробка механізмів взаємодії з об'єктами на рівні, таких як підбирання предметів, відкривання дверей, активація механізмів. Важливо забезпечити фізичні реакції об'єктів на дії гравця для створення відчуття реалістичності та занурення в ігровий світ.
4. Обмеження здоров'я та мани гравця – здоров'я та мана повинні відображатись як HUD та повинні бути відновлюваним ресурсом. Здоров’я можна відновити за допомогою їжі, а ману – з часом (див. рис. 4.2).

Рисунок 4.2 – Приклад процесу відновлення здоров’я та мани

Також була створена система компонентів, що є важливим елементом у структурі ігрового застосунку, що дозволяє розширювати функціональні можливості ігрових об'єктів через додавання спеціальних модулів. Компоненти забезпечують модульність та повторне використання коду, що значно полегшує процес розробки. У рамках цього проекту були розроблені кілька ключових компонентів, які можна легко додавати до Blueprints для забезпечення додаткової функціональності "з коробки". Нижче наведені описи основних компонентів:

1. BP_Death_Component: компонент BP_Death_Component відповідає за обробку логіки смерті персонажа або об'єкта. Даний компонент автоматично

відслідковує рівень здоров’я об’єкту та викликає пов’язані зі смертю події при необхідності (див. рис. 4.3)

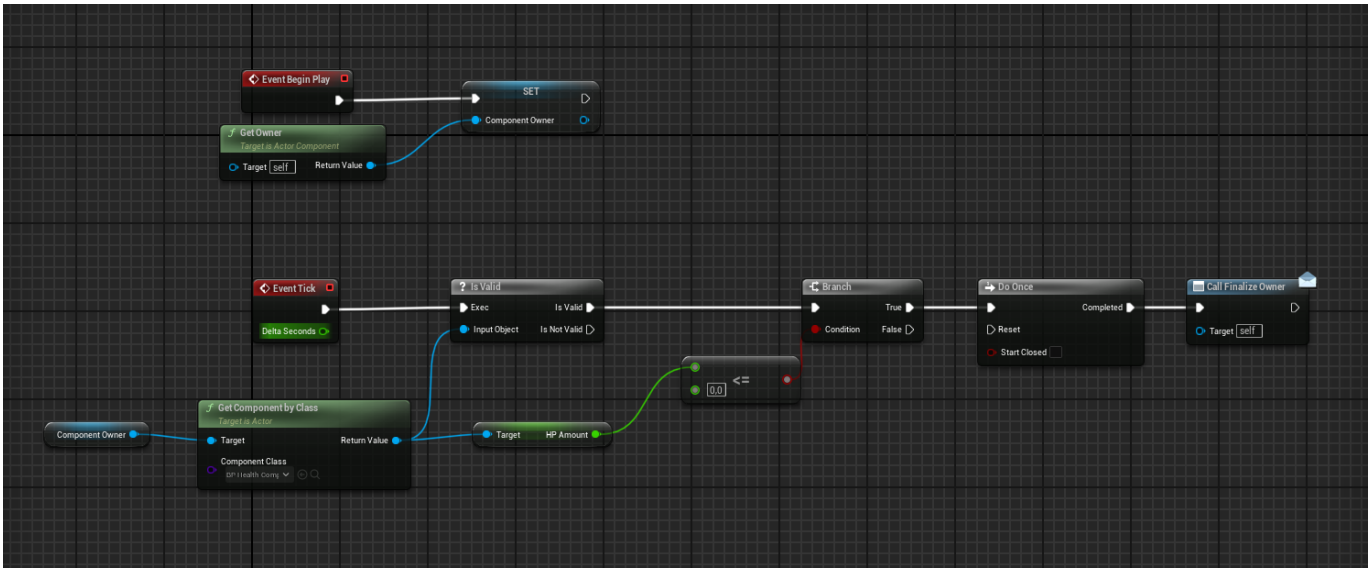


Рисунок 4.3 – Реалізація компоненту смерті.

2. BP_Health_Component: Компонент BP_Health_Component призначений для управління здоров'ям персонажа або об'єкта. Основні функції цього компонента включають:
    - Ініціалізація здоров'я: Встановлює початкові значення здоров'я та максимальний рівень здоров'я.
    - Обробка ушкоджень: Обробляє отримання ушкоджень, оновлює значення здоров'я та перевіряє, чи досягнуто критичного рівня.
    - Відновлення здоров'я: Забезпечує механізми для відновлення здоров'я через будь-які джерела.
2. BP_Flammable_Component: Компонент BP_Flammable_Component відповідає за обробку логіки займання та горіння об'єкта. З цим компонентом об’єкт чи персонаж можуть бути підпаленими. Підпалення відображається візуально за допомогою спеціальних ефектів (див. рис. 4.4) та з часом втрачають здоров’я.

Рисунок 4.4 – Приклад займаного вогнем об’єкту.

Таким чином, розробка системи компонентів дозволяє значно підвищити гнучкість та модульність ігрового застосунку. Компоненти BP_Death_Component, BP_Health_Component та BP_Flammable_Component забезпечують ключові функціональні можливості, які можуть бути легко інтегровані в будь-які ігрові об'єкти, забезпечуючи їхню взаємодію з іншими елементами гри. Це сприяє створенню більш реалістичного та захоплюючого ігрового досвіду.

4.2 Розробка візуальних ефектів

Візуальні ефекти є важливим елементом у створенні захоплюючого ігрового досвіду, надаючи грі динамічності та візуальної привабливості. У рамках цього

проекту було розроблено кілька ключових ефектів для магічних снарядів, використовуючи систему частинок Niagara в Unreal Engine. Ці ефекти включають появу та зникнення магічних снарядів, а також вибухи з UV-анімацією.

Магічні снаряди є однією з основних атак головного героя. Для створення ефекту появи снаряду було використано систему частинок Niagara. Поява снаряду супроводжується яскравими спалахами та динамічними частинками, що створюють враження енергії, що концентрується в одній точці (див. рис. 4.5).

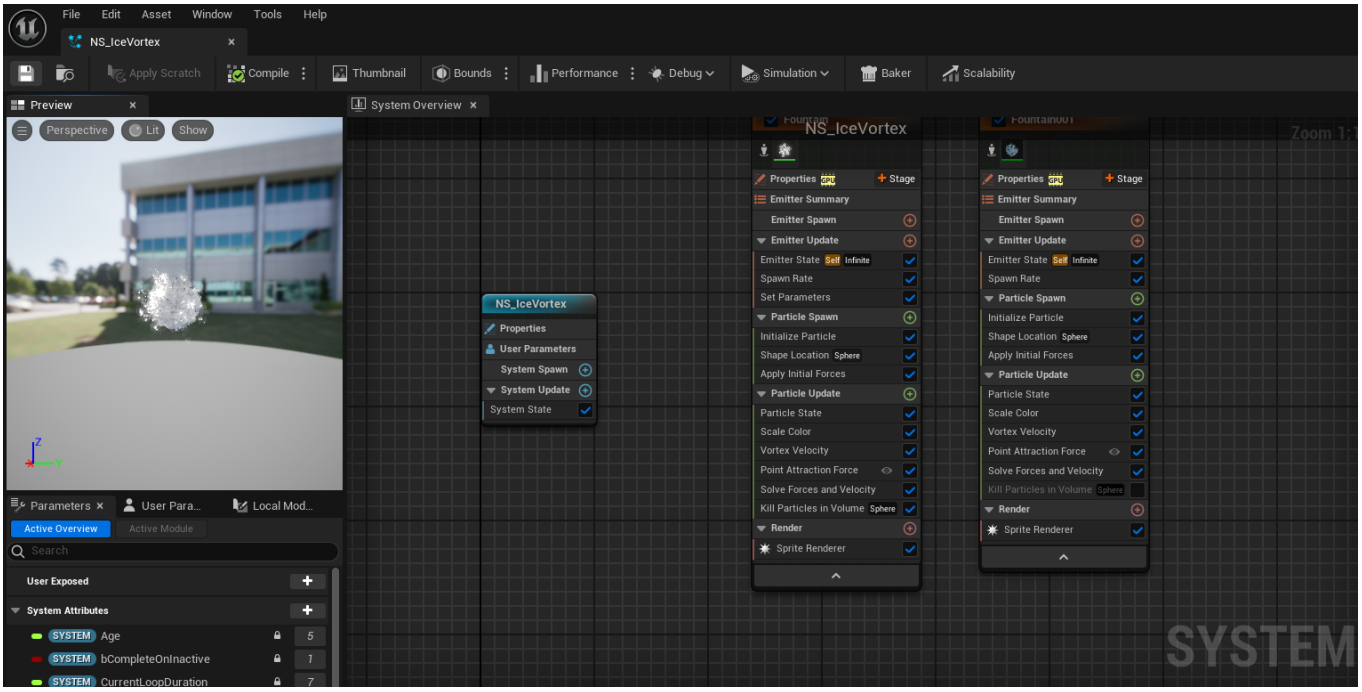


Рисунок 4.5 – Ефект льодового снаряду та його реалізація.

Для створення ефекту вибуху було розроблено складний візуальний ефект з використанням UV-анімації. Цей ефект застосовується, коли магічний снаряд стикається з перешкодою або досягає цілі. Вибух супроводжується яскравими спалахами та динамічними частинками, що створюють вражаючий візуальний ефект. Було використано текстурну анімацію для створення реалістичного ефекту вибуху. Текстура вибуху розділена на кілька кадрів, які відтворюються послідовно для створення анімації (див. рис. 4.6).

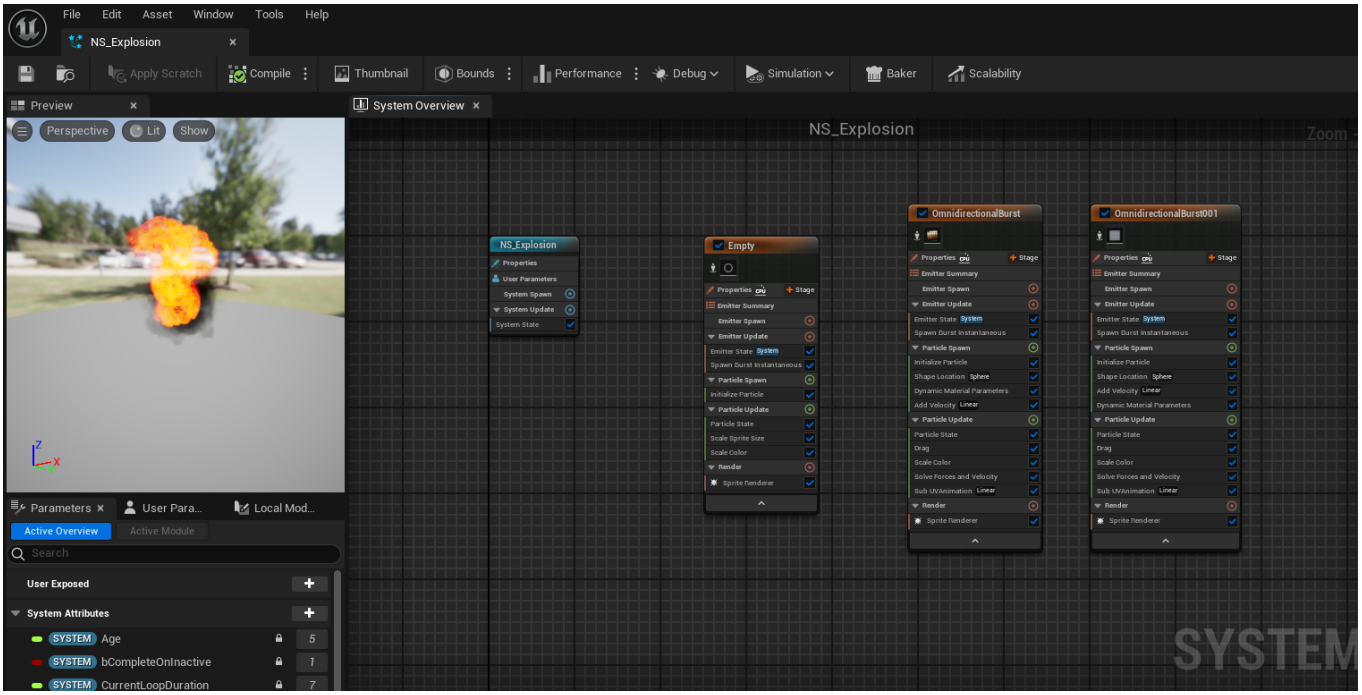


Рисунок 4.6 – приклад ефекту з текстурною анімацією

Ефект горіння був розроблений для симуляції реалістичного полум'я, що огортає об'єкт, який горить (див. рис. 4.7). Цей ефект створює враження, що об'єкт повністю охоплений вогнем, що значно підвищує рівень занурення гравця в ігровий процес. Для досягнення такого ефекту частинки обирають один із вертексів об’єкту як точку для виникнення.

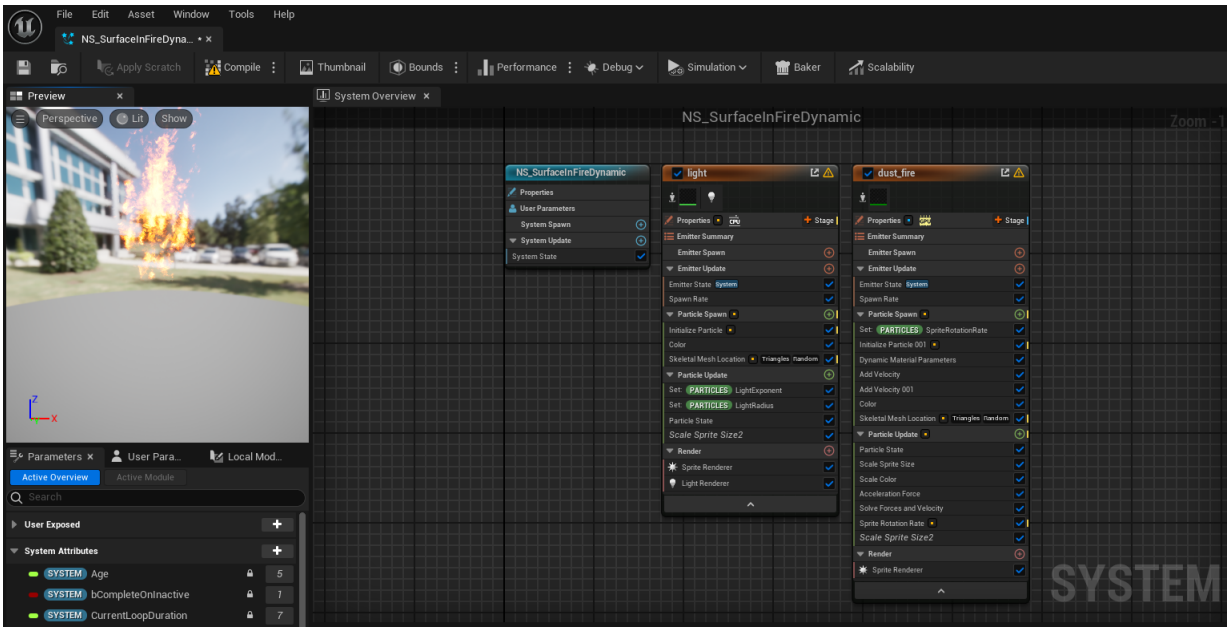


Рисунок 4.7 – Приклад ефекту вогню що охоплює об’єкт

Розробка візуальних ефектів з використанням системи частинок Niagara дозволила значно покращити візуальну привабливість гри та підвищити рівень занурення гравця. Використання ефектів для магічних снарядів, включаючи їх появу, зникнення та вибухи з UV-анімацією, а також ефекту горіння, що охоплює об'єкт, додало грі динамічності та видовищності. Завдяки використанню сучасних технологій та оптимізаційних методів[1], вдалося досягти високої якості візуальних ефектів без значного впливу на продуктивність гри

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розробка мапи думок тестування

Перелік функціональності проєкту було розроблено у вигляді карти думок (див. рис. 5.1). Тест план розроблявся згідно цієї мапи.

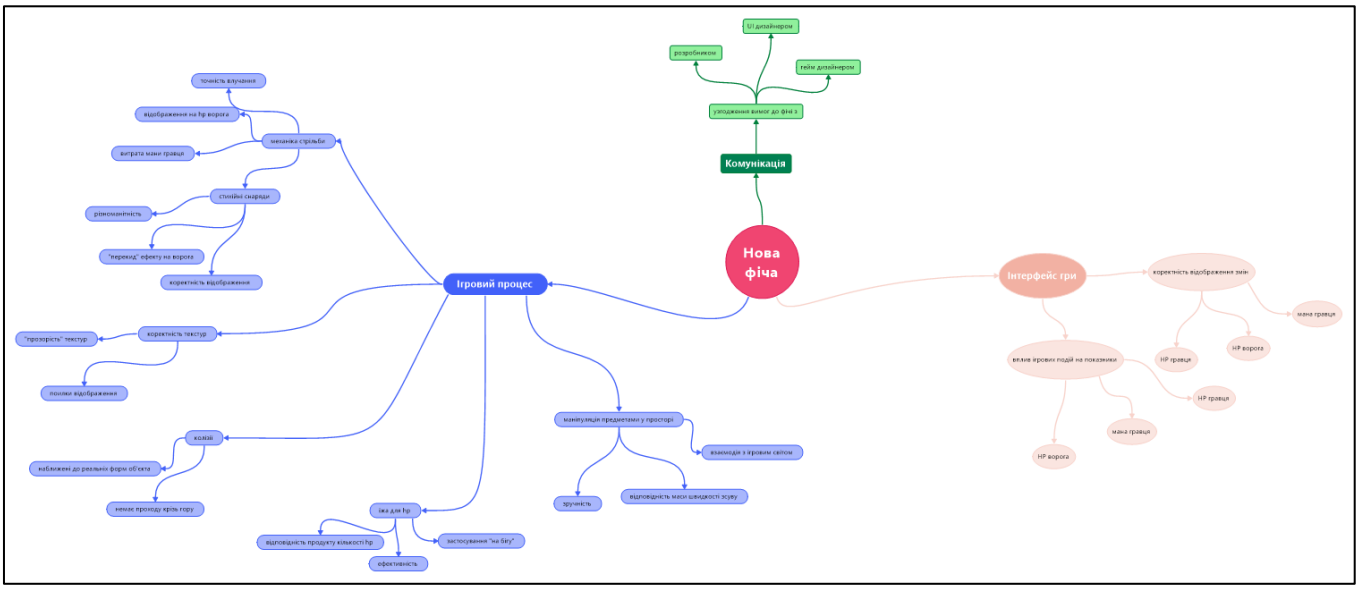


Рисунок 5.1 - Mind Map для тестування проекту

Тестування проводилось за методологією «Чорної скриньки» - спосіб тестування системи за яким тестувальники не знають деталей її реалізації.

Безпосередньо до теми цієї роботи належить тестування ігрового процесу, а саме механік головного героя. Функції які при цьому перевірялись:

- Пересування гравця – можливість рухатись на площині, стрибати, присідати, прискорюватись.
- Маніпуляція предметами на локації – можливість підняття предметів та можливість пересувати їх у просторі. Швидкість пересування залежить від ваги предмета.
- Стрільба – можливість стріляти із різних типів зброї за стихією. Кожна стихія має свої власні візуальні та фізичні ефекти.
- Взаємодія стихій – гравець повинен мати можливість комбінувати стихії для

досягнення унікальних результатів - створити наелектризовану воду, розтопити лід, тощо.

- Отримання шкоди – можливість зазнати поранень від різних джерел – удари противників, падіння з висоти, падіння тяжкого об’єкта на гравця.
- Лікування – можливість відновити здоров’я за рахунок їжі, різні типи їжі мають відновлювати різну кількість здоров’я.
- Витрата та відновлення мани – мана – ресурс необхідний для створення стихійних снарядів. Повинна витрачатись під час кожного пострілу та відновлюватись самостійно з часом.

Таким чином ігровий застосунок був розбитий на окремі системи, кожна з яких протестована.

5.2 Розробка тест-кейсів

На основі описаних раніше функціональних можливостей, було складено перелік тест-кейсів для тестування застосунку та уникненню можливості подальшої їх появи.

На таблиці 5.1 наведено приклади найбільш суттєвих тестових випадків. P – скорочення від англ. «priority» - пріоритет; S – від «severity» - тяжкість. Числа навпроти цих літер означають їх значущість. Таким чином, P4 – найнижчий пріоритет, P1 – найбільший пріоритет; S4 – не критичний баг, S1 – дуже критичний.

Таблиця 5.1 – Тестовий випадок з проникненням снарядів крізь стіни

Тест № 1	
Назва тесту:	Магічні снаряди проходять крізь деякі стіни
Опис тесту:	При низькій частоті кадрів можливе проникнення магічних снарядів крізь стіни.
Компонент системи:	Стрільба, колізії
Пріоритет:	P2
Критичність:	S3
Кроки відтворення:	Понизити частоту кадрів, стріляти у стіни
Очікуваний результат:	Снаряди не проходять крізь стіни
Фактичний результат:	Снаряди проходять крізь стіни

Тест № 2	
Назва тесту:	Персонаж не отримує шкоди від певних джерел
Опис тесту:	При низькій частоті кадрів можливий пропуск отримання шкоди гравцеві від падіння.
Компонент системи:	Компоненти головного героя
Пріоритет:	P4
Критичність:	S3
Кроки відтворення:	Понизити частоту кадрів. З деякою вірогідністю після цього можна не отримати шкоду від падіння.
Очікуваний результат:	Світло не проходить крізь текстури
Фактичний результат:	Світло проходить крізь текстури

Тест № 3	
Назва тесту:	Втрата можливості керування головним героєм
Опис тесту:	Після смерті та відродження гравець не може більше керувати головним героєм
Компонент системи:	Компоненти головного героя, UI
Пріоритет:	P1
Критичність:	S1
Кроки відтворення:	Дати померти головному герою, після чого натиснути «Start from checkpoint».
Очікуваний результат:	Можна продовжити грати з останньої точки збереження
Фактичний результат:	Гра не реагує на користувацький ввід.

Таким чином, було створено безліч тест-кейсів, на основі шаблону з прикладів наведених вище. Це дозволило вести облік багів, що знайшлись та уникнути можливість їх повторити знов.

ВИСНОВКИ

Під час розробки ігрового застосунку в рамках дипломного проекту було досягнуто значних результатів, які варто відзначити.

Реалізація механік головного героя показала відмінні результати. Персонаж був оснащений анімаціями для різних видів рухів, таких як біг, стрибки та присідання. Це дозволило створити більш реалістичний та плавний ігровий процес, що значно покращує загальний досвід гравця. Крім того, можливості взаємодії з предметами на рівнях були інтегровані таким чином, щоб вони були інтуїтивно зрозумілими та природними для гравців.

Візуальні ефекти, розроблені за допомогою системи Niagara, додали грі нових барв та динаміки. Ефект Ice Vortex став особливо вражаючим прикладом, створюючи захоплюючі сцени та додаючи глибини візуальному ряду гри. Реалістичне освітлення та ефекти погодних умов також сприяли створенню багатого та атмосферного ігрового середовища.

Значну увагу було приділено також користувацькому інтерфейсу. Його розробка і тестування допомогли створити зручний та інтуїтивний інтерфейс, який полегшує взаємодію гравця з грою. Інтерактивні елементи були оптимізовані для швидкого доступу та простого управління, що значно покращило користувацький досвід.

Вибір інструментів для розробки виявився дуже вдалим. Використання Unreal Engine 5 та системи візуального програмування Blueprint дозволило ефективно реалізувати складні ігрові механіки та візуальні ефекти. Система контролю версій Git забезпечила стабільність і керованість проекту, що є важливим аспектом для командної роботи та підтримки цілісності коду.

Загалом, технічна реалізація проекту виявилася успішною. Оптимізація системи дозволила досягти високої стабільності та продуктивності гри. Використання сучасних технологій у програмуванні та графічному дизайні сприяло створенню ігрового продукту, що відповідає сучасним стандартам якості та функціональності, надаючи користувачам унікальний ігровий досвід.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бєліков Д. Ю., МЕТОДИ ОПТИМІЗАЦІЇ ВІЗУАЛЬНИХ ЕФЕКТІВ У UNREAL ENGINE, 28 Міжнародний молодіжний форум «РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ В XXI ст.», м. Харків. 2024. С. 468-471.

2. Радіоелектроніка та молодь у XXI столітті. Т. 6 : Конференція "Інформаційні інтелектуальні системи" : матеріали 27-го Міжнар. молодіж. форуму, 10–11 трав. 2023 р. / М-во освіти і науки України, Харків. нац. унт радіоелектроніки. Харків, 2023. 471 с.

3. Circle Constraint in UE5 Niagara Tutorial URL: https://dev.epicgames.com/community/learning/tutorials/EOW/circle-constraint-in-ue5-niagara-tutorial (дата звернення: 05.05.2024).

4. A tutorial for adding Components to Actors. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/adding-components-to-an-actor-in-unreal-engine?application_version=5.0 (дата звернення: 05.05.2024).

5. Begin Play | Niagara. URL: https://dev.epicgames.com/community/learning/tutorials/j9YO/unreal-engine-begin-play-niagara (дата звернення: 25.03.2024).

6. Blueprints Quick Start Guide URL: https://docs.unrealengine.com/4.26/en-US/ProgrammingAndScripting/Blueprints/QuickStart/ (дата звернення: 11.05.2024).

7. Blueprint Overview. URL: https://docs.unrealengine.com/4.26/en-US/ProgrammingAndScripting/Blueprints/Overview/ (дата звернення: 13.05.2024).

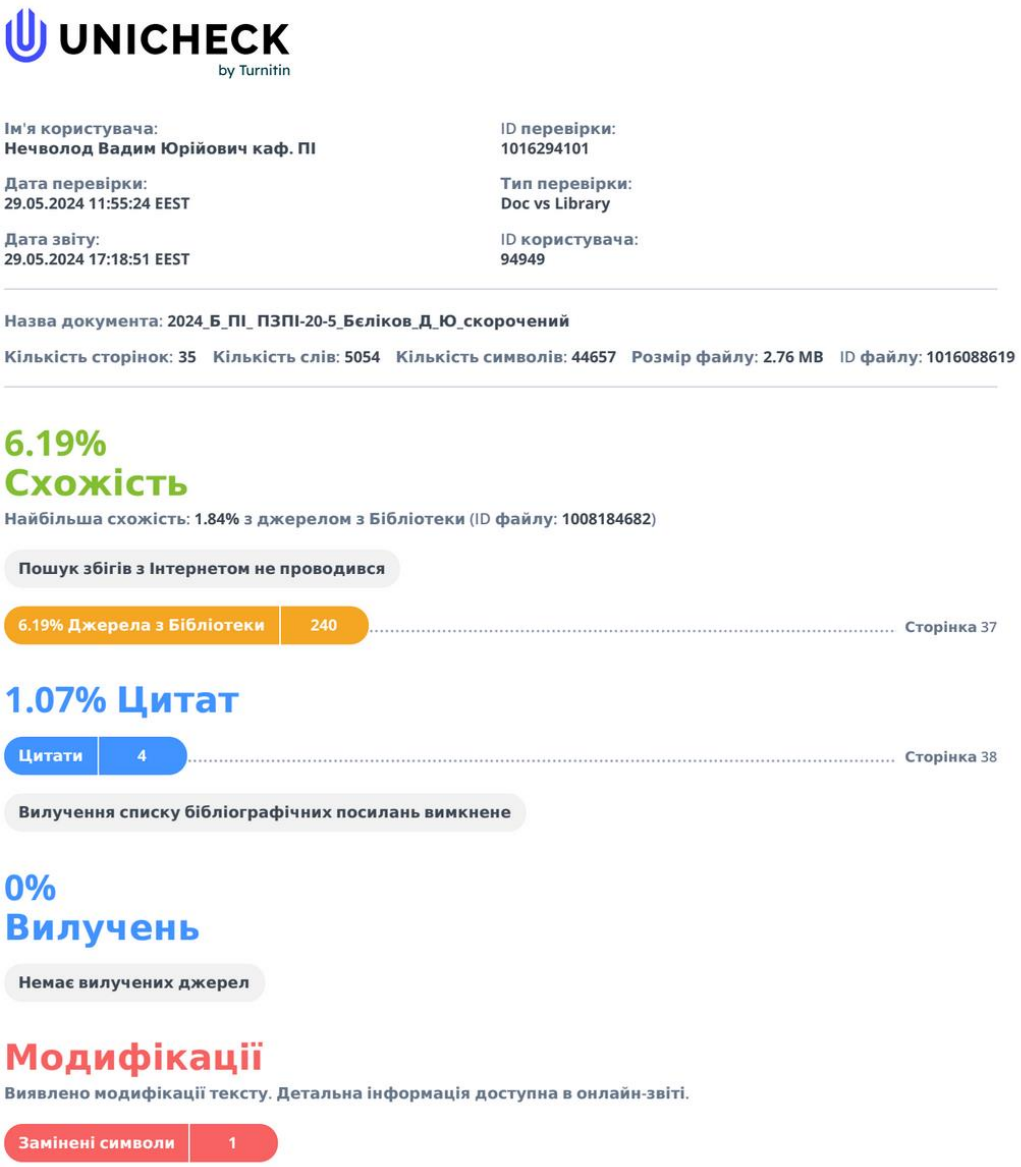
8. How to create, save, and open Level assets in UE4. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/working-with-levels-in-unreal-engine?application_version=5.0 (дата звернення: 25.03.2024).

9. Use Case Diagrams | Unified Modeling Language (UML) URL: https://www.geeksforgeeks.org/use-case-diagram/ (дата звернення: 10.05.2024).

10. Комплекс навчально-методичного забезпечення навчальної дисципліни "Основи ігрової графіки" підготовки бакалавра для студентів усіх форм навчання спеціальності 121 – Інженерія програмного забезпечення: освітня програма "Програмна інженерія" / ХНУРЕ ; розроб. Н. А. Валенда. – Харків, 2017

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



UNICHECK
by Turnitin

Ім'я користувача:
Нечволод Вадим Юрійович каф. ПІ

ID перевірки:
1016294101

Дата перевірки:
29.05.2024 11:55:24 EEST

Тип перевірки:
Doc vs Library

Дата звіту:
29.05.2024 17:18:51 EEST

ID користувача:
94949

Назва документа: 2024_Б_ПІ_ ПЗПІ-20-5_Бєліков_Д_Ю_скорочений

Кількість сторінок: 35 Кількість слів: 5054 Кількість символів: 44657 Розмір файлу: 2.76 MB ID файлу: 1016088619

6.19%
Схожість

Найбільша схожість: 1.84% з джерелом з Бібліотеки (ID файлу: 1008184682)

Пошук збігів з Інтернетом не проводився

6.19% Джерела з Бібліотеки 240 ..... Сторінка 37

1.07% Цитат

Цитати 4 ..... Сторінка 38

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 1

ДОДАТОК Б

Слайди презентації

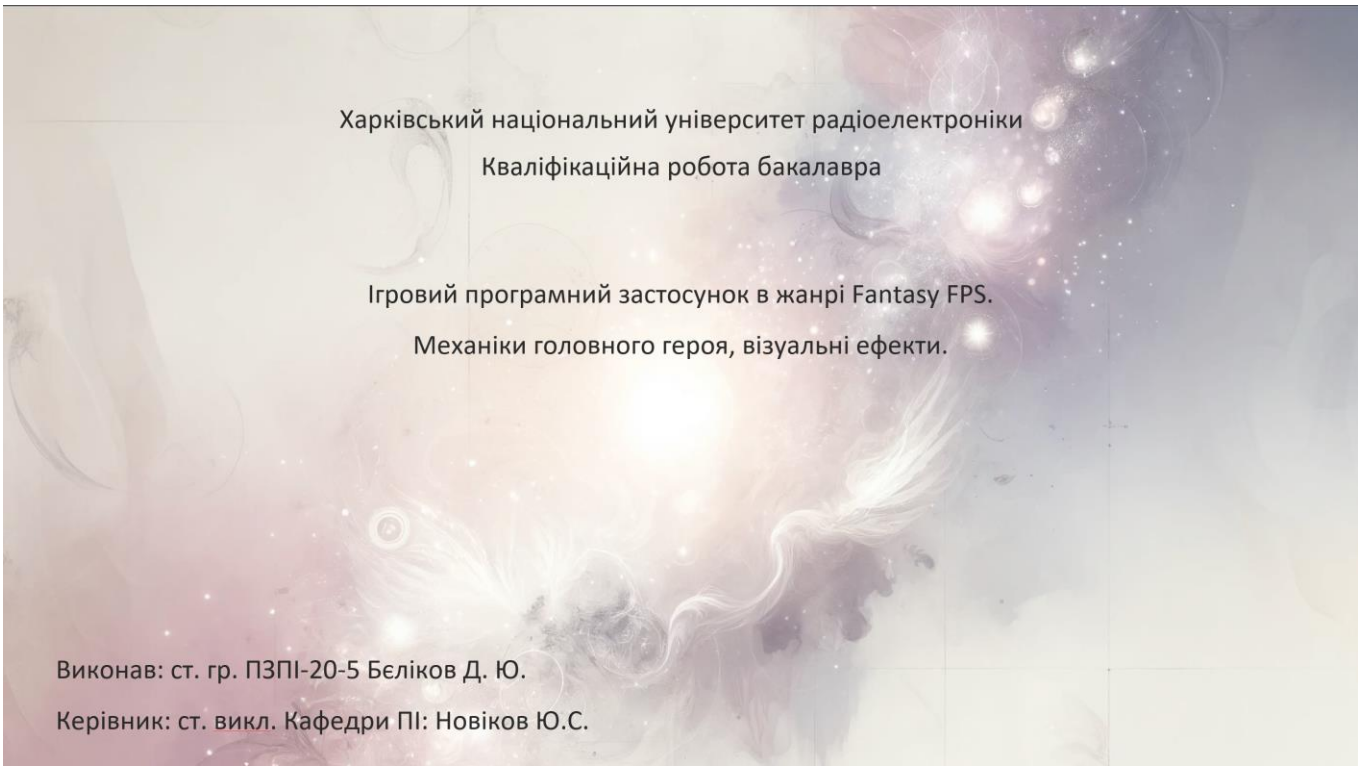


Рис. Б.1 – слайд 1

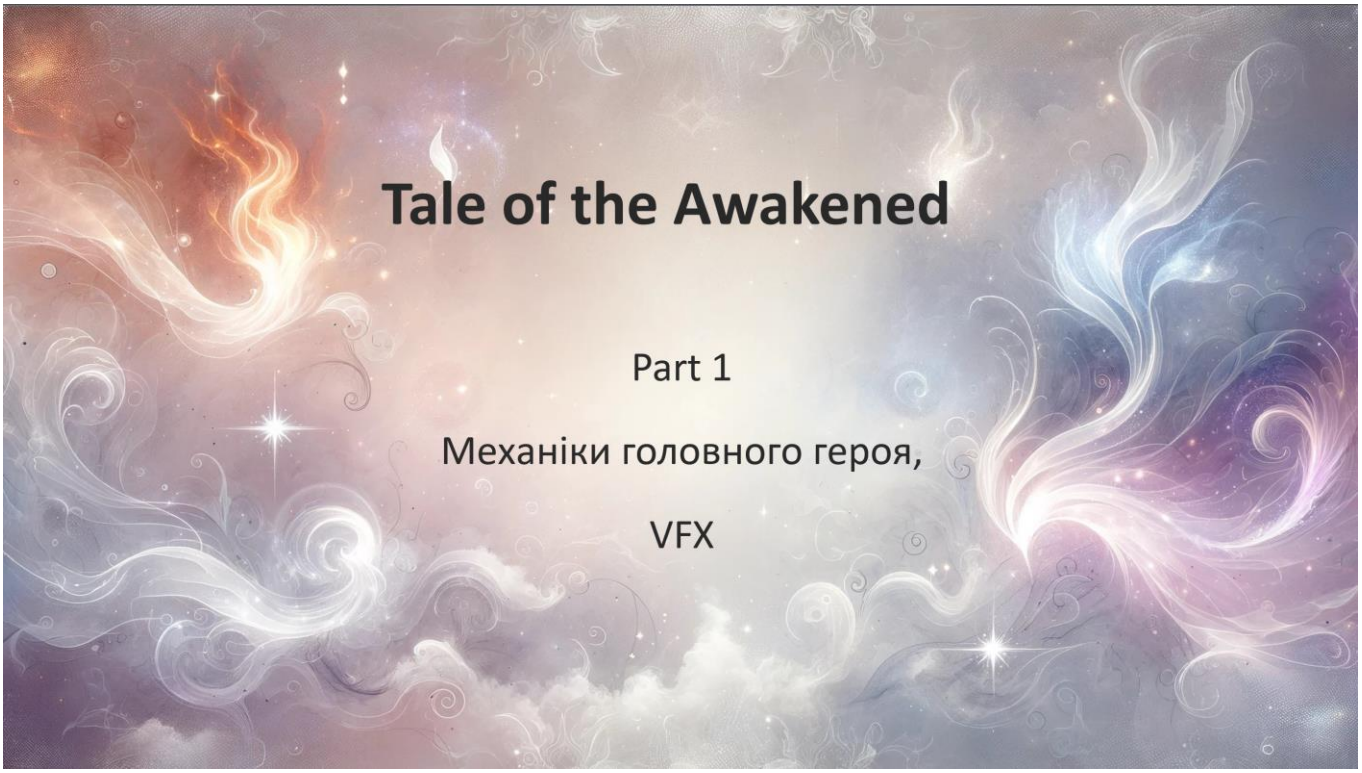


Рис. Б.2 – слайд 2

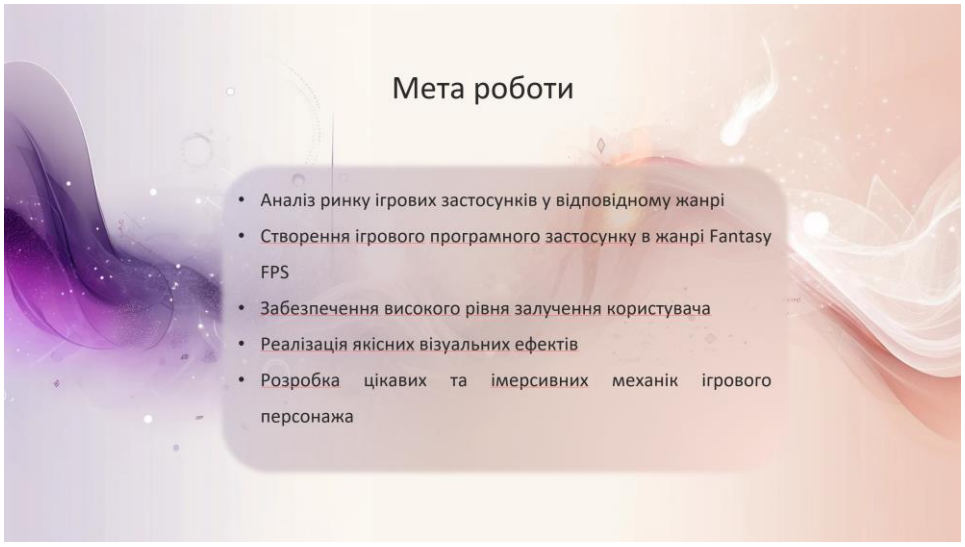


Рис. Б.3 – слайд 3

Рис. Б.4 – слайд 4

Рис. Б.5 – слайд 5

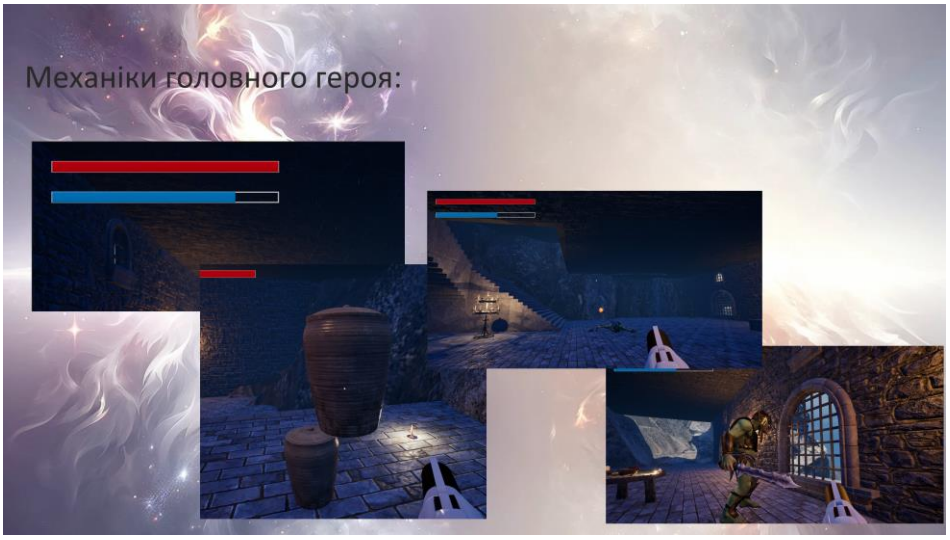


Рис. Б.6 – слайд 6

Рис. Б.7 – слайд 7

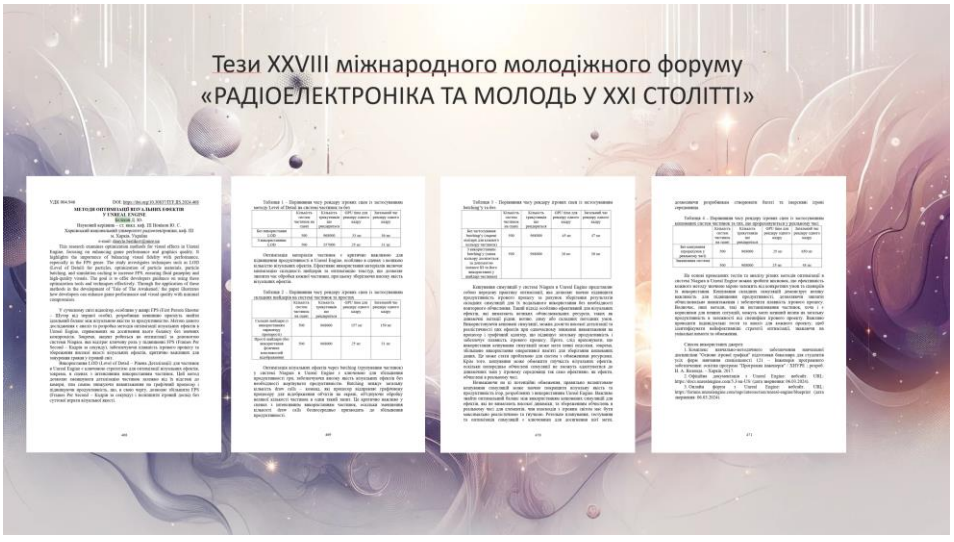


Рис. Б.8 – слайд 8

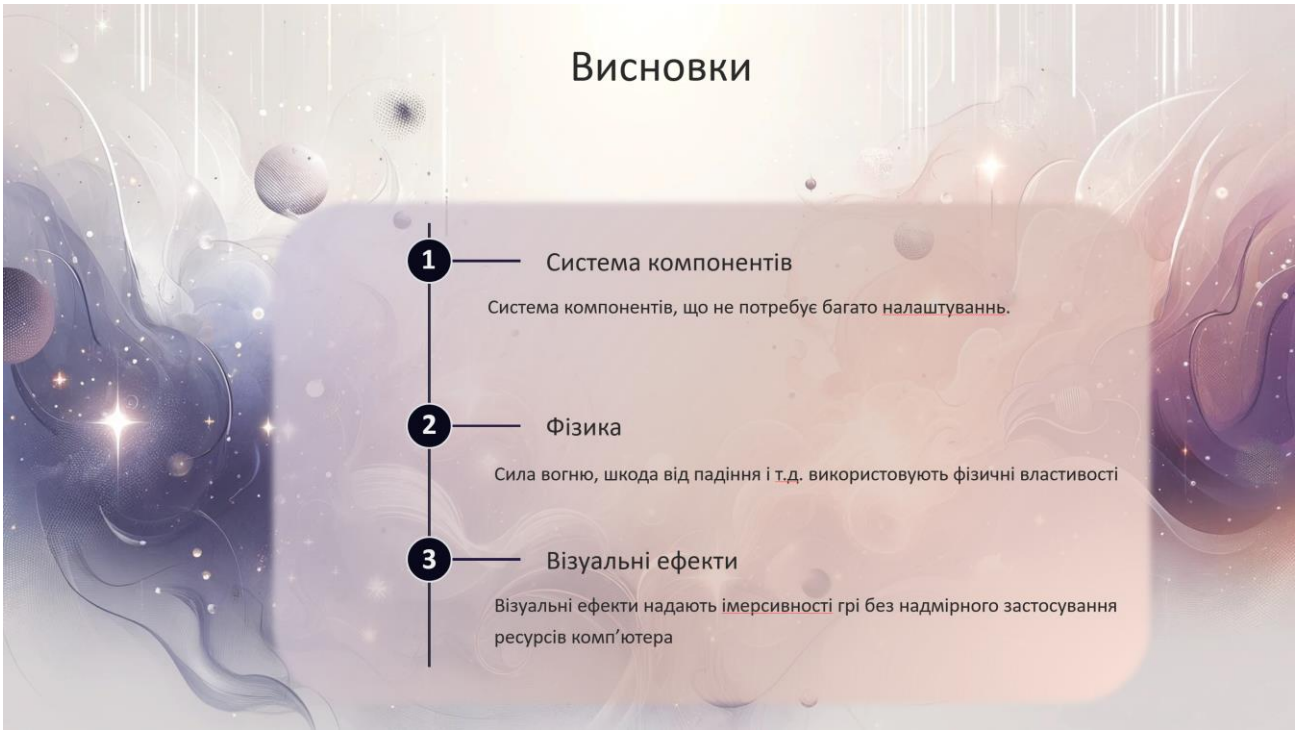


Рис. Б.9 – слайд 9

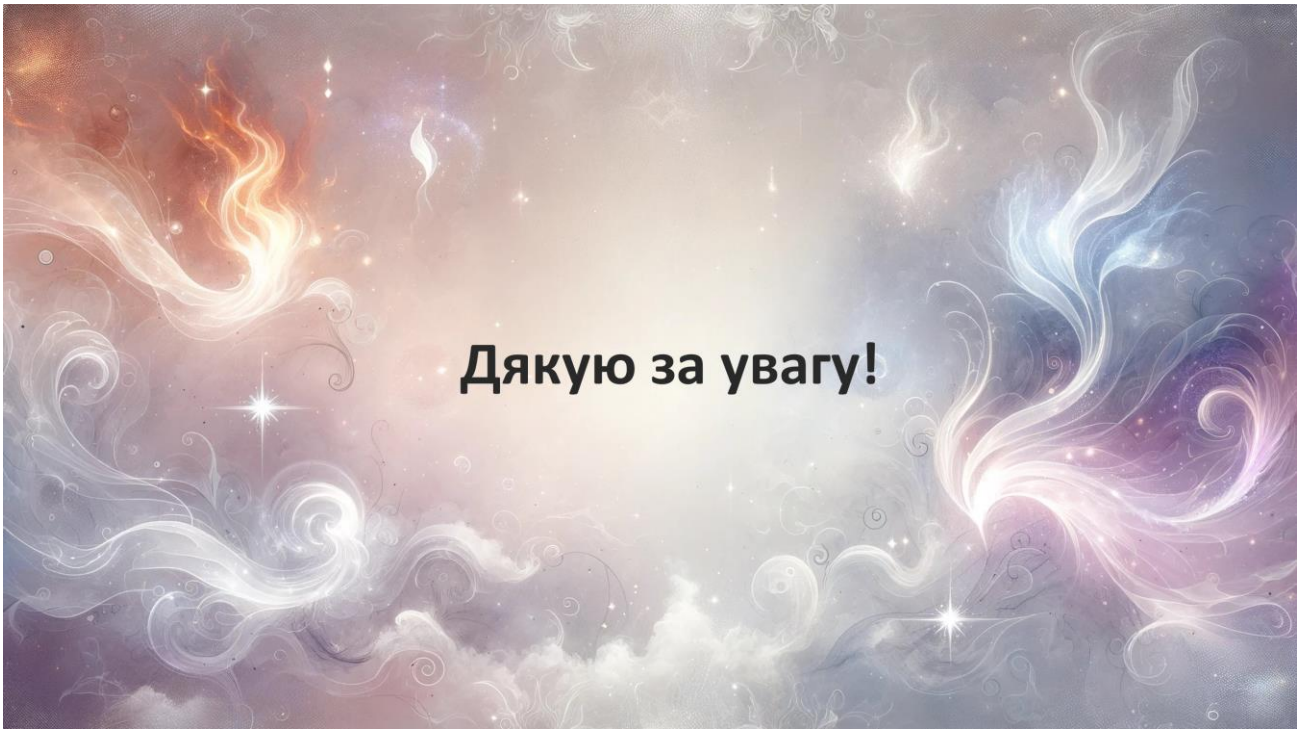


Рис. Б.10 – слайд 10

ДОДАТОК В

Геймдизайн-документ

Тетра

Технологія:	ПК, консолі
Механіка:	шутер від першої особи
Історія:	ГГ – звичайна людина у світі магії
Естетика:	зовсім трішки magicpunk та середньовічне фентезі

Рисунок В.1 – Геймдизайн-документ сторінка 1

Лор

Даний світ дуже нагадує звичні нам середньовічні фентезійні світи за виключенням магії. Зазвичай, магу для використання своїх магічних здібностей необхідно використовувати стихійник - спеціальний мінерал, що перетворює чисту ману на стихію: вогонь, лід, вода, блискавка та похідні від них. Завдяки цій магії, працює багато виробництв, а військові в свою чергу мають спеціальну "вогнепальну" зброю, що працює на основі стихійників. Попри все, магією володіє далеко не кожен (приблизно 0,2% населення). Маги є дуже поважними особами у суспільстві, адже як раз завдяки ним, їх світ не переживає постійні глобальні війни, епідемії, голод та інші біди, що спіткали наш. Якщо казати коротко, то немає критичної недостачі життєво необхідних ресурсів.

Усюди панує раса людей, але при цьому разом з ними співіснують й інші, такі як ельфи та метри (назва пов'язана з їх зростом, що зазвичай дорівнює близько одному метру). На момент гри, людству відомі два материки Фівенія (в народі Земля Творця) та Вардус (в народі Темні Землі). Фівенію населяють усі миролюбні раси (зазначені раніше люди, ельфи та метри), а Вардус - гобліни, тролі, дракони та інші потвори. Дуже давно союз людей, ельфів та метрів прогнали гоблінів з Фівенії до Вардусу та возвели незламну стіну (див. мапу), що забезпечило мир для їх нащадків, але, як виявилось, не назавжди. Багато років потому у королівській сім'ї з'явився хлопчик, що володів надпотужною магією. Завдяки ній, він міг підкорити собі майже будь-кого, від кішки до людини (чим розумніша істота - тим важче її підкорити). Ця сила хлопчика, котрого назвали Вальцем, не дала йому стати порядною людиною, він виріс в найжорстокішого принца, що знала історія. Через це, помираючи, король надав перевагу у виборі спадкоємця своєму молодшому сину. Ця ситуація розлютила Вальца, через що він вирішив відправитись у Темні Землі, де зміг підкорити собі народ гоблінів. Згодом він почав розвивати їх армію, підкоряти тролів у допомогу гоблінам, могутніх елементалей та навіть одного дракона. Згодом, через надвелику кількість водних елементалей (вони є одним цілим із усією водою у світі, вони підтримують баланс. Чим більше існує водних елементалей - тим менше води та навпаки), котрі будучі завербованими злим принцом, перестали щезати, рівень

Рисунок В.2 – Геймдизайн-документ сторінка 2

океану опустився, що дало гоблінам, котрі не ладнають з наукою і будівництвом кораблів, змогу обійти незламну стіну по новоствореним берегам. Після цього гобліни на чолі із Вальцем напали на Королівство кинджалів (батьківщина Вальца) із наміром захопити світ.

Рисунок В.3 – Геймдизайн-документ сторінка 3

Сюжет гри

Звуть нашого героя Кіт (в українській локалізації через це його всі і називають котом, а в англійській перекручують на kid (дитина)). Ще в дитинстві він помітив, що удача завжди на його стороні, тому обрав найлегший для себе метод заробітку - крадіжка та обман. Виконуючи підозріле замовлення, що само по собі не є новинкою для героя, він вперше став жертвою невдачі і розбив артефакт, котрий мав вкрасти, який містив у собі запечатану силу, що здатна наділити будь-яку істоту здібностями до магії, після чого вона перейшла до найближчої живої істоти - Кіта. Артефакт було зроблено в давнину як експеримент, котрий, як виявилось, потребує забагато жертв для створення. На створення одного подібного артефакту було витрачено більше десятка життів немагів. Злочин головного героя не залишився без покарання. Охорона конвою, яка повинна була доставити даний артефакт до придворного магу на ім'я Зерхен, тепер доставить новоствореного мага-крадія.

Після пробудження від довгого сну, Кіт одразу опиняється у камері для підсудних. Згодом, його визивають до суду перед самим королем (Леопольдом, молодшим сином старого короля). Беручи до уваги ситуацію в країні (війна, нестача магів), король прийняв єдине вірне рішення - зберегти Кіту життя, але і відпустити він його не міг. Було вирішено навчити Кіта володіти своєю новою силою і відправити його на фронт - на допомогу в один із прифронтових гарнізонів. Спокутою для Кіта повинно було стати служіння в армії до перемоги, або смерті.

Незадоволений своїм вироком, головний герой вирішує, що дезертує у будь-який зручний момент. Тут удача відвертається від нього вдруге - в першу ж ніч його перебування там, армія супротивника робить ривок та за лічені години дістається до укріплень, в одному з яких знаходиться Кіт. Побачивши смерть невинних людей, він міняє свій світогляд і вирішує боротись за тих, кого обкрадав усе своє життя до цього.

Як єдиний бойовий маг, що знаходився в тому гарнізоні, Кіт мав звітувати придворному магу (за сумісництвом, генералу армії) про все, що той бачив.

Рисунок В.4 – Геймдизайн-документ сторінка 4

Нічого з того, що він йому розказав, не здивувало Зерхена. Зваживши ситуацію, генерал вирішив відправити головного героя до Вардусу - дослідити підозрілу активність гоблінів біля одного з древніх храмів. Обрано було саме Кіта через його минуле, адже хто, як не крадій, зможе прошмигнути до бази супротивника і повернутись звідти живим?

Обстеживши руїни храму, де знаходились гобліни, Кіт знайшов документ - наказ від Вальца, в якому було сказано шукати "корону стійкого розуму". Після чергового звіту, головний герой разом із Зерхеном розуміють, що мова йде про артефакт, що згадується в одній із казок. Якщо вірити тому, що в ній написано - цей артефакт дозволяв королям у давнину не ставати жертвами нечесних радників, що могли намагатись маніпулювати ними. Стає ясно, що Вальц боїться, що саме даний артефакт може його зупинити. Чого Вальц не знав - так це того, що даний артефакт вважається навіки загубленим у пустелі елементалей - безлюдній місцевості, де вже понад тисячу років, без перерви тривають бої елементалей різних стихій. Коли один елементаль помирає - світовий баланс стихій згодом повертає його назад.

Для того щоб все ж знайти потрібний артефакт посеред пустелі, було створено спеціальний інструмент, що завдяки стихійнику, генерує воду у напрямку найближчої мани. Для того щоб знайти артефакт, спершу Кіту та команді, що була створена спеціально для даної місії, потрібно вбити всіх елементалей на території і прислухатись до найменших коливань води у приборі. Даний процес доводиться повторювати кілька разів, адже кожного разу елементалі повертаються. В результаті, команда знаходить корону, але вона виявляється розбитою.

Після даної операції було прийнято рішення - провести ще одну, в ході якої Кіт з напарниками мають визволити з полону єдиного у світі коваля-мага, що, потенційно, міг би перекувати корону. Дана місія проходить успішно, але навіть після визволення від Вальца, коваль ні на що не реагує - він просто втратив глузд після тривалого впливу магії підкорення. Єдиним шансом на лікування для нього є - зілля повного зцілення, що може надати лише могутня та нарцистична відьма

Рисунок В.5 – Геймдизайн-документ сторінка 5

- Марла. Вона є найкращим зіллєваром серед усіх королівств Фівенії, але її послуги коштують дорого, дуже дорого.

Платою за зілля повного зцілення, Марла просить дістати їй інгредієнти для зілля, що поверне їй красу та молодість. Знайти їх можна лише в Темних Землях. Робити тут нічого - Кіт вкотре відправляється до Вардусу - шукати необхідні трави та чудовиськ, рештки яких теж потрібні для зілля. Хоч і не без проблем, але головний герой знаходить усі інгредієнти та отримує натомість зілля повного зцілення.

Зілля лікує коваля. При цьому загоюються навіть старі шрами та опіки, що він постійно отримує. Придворний маг повідомляє йому про його місію - відремонтувати корону. Без зайвих слів, коваль відправляється виконувати свою роботу. Зерхен повідомляє Кіту, що саме йому доведеться боротися із Вальцем, адже головний герой, не дивлячись на недосвідченість, зарекомендував себе як одного з найсильніших бійців, що здатний вижити майже в будь-якій ситуації.

Перед початком свого, вірогідно, останнього завдання, Кіт вирішує помститися генералу, що вів армію на гарнізон котрий був знищений в самому початку історії. Потай від усіх, він споряджається на Вардус - на базу, де, за останніми даними, знаходиться цей генерал. Після успішної ліквідації, він з боєм вириває із лігва та повертається до свого королівства, де коваль як раз до того часу встиг перекувати корону. Його майстерності вистачило щоб зробити з неї браслет - більш зручний артефакт для носіння на полі бою.

Головного героя споряджають на останню битву - віддають браслет, найкращу зброю, що тільки є у королівстві, зілля мани та інше. Після - відправляють до підводного храму (котрий вже не є підводним через зниження рівня моря), лігва злодія. Кіт успішно ліквідує дракона - головного охоронця Вальца, після чого починає бій із самим ватажком гоблінів. Після запеклого бою, в ході якого Кіт міг загинути безліч разів, Вальц помирає від його руки. Після цього чари маніпуляції розумом зникають і елементалі одразу щезають, тим самим нормалізуючи рівень моря, в ході чого головний герой ледве не тоне, але

Рисунок В.6 – Геймдизайн-документ сторінка 6

в останню мить на кораблі йому на допомогу приходять друзі, яких він знайшов під час своїх небезпечних пригод.

Рисунок В.7 – Геймдизайн-документ сторінка 7

Цільова аудиторія

Основна цільова аудиторія гри – молоді люди 16-30 років, оскільки ігрові аспекти, що розглядаються в нашій грі, достатньо гарно задовольняють і відповідають їх бажанням і потребам для розваг. Гра має вікове обмеження 16+, через середній ступінь жорстокості вбивств.

Для залучення аудиторії в цій грі розраховані такі аспекти: отримання емоцій, достатній перелік різної зброї, різнобарвність бійок, іммерсивність, жорстокість.

Рисунок В.8 – Геймдизайн-документ сторінка 8

USP

- Здолайте могутнього злодія.
- Знайдіть неочікуваний шлях вирішення проблеми.
- Експериментуйте зі стихіями.
- Здолайте ворога у відкритій динамічній битві.
- Пройдіть повз ворога будучи непоміченим.
- Виконуйте додаткові завдання, що допоможуть з основними.

Рисунок В.9 – Геймдизайн-документ сторінка 9

Час ігрової сесії

Основний сюжет гри буде складатися з 7 повноцінних місій та 1 вступного завдання, під час якого буде проводитися навчання.

Сайдквести – в розробці.

Приблизне проходження однієї місії буде займати 2-3 години. Отже загальна тривалість гри складе десь 14-20 годин

Рисунок В.10 – Геймдизайн-документ сторінка 10

Технологічні характеристики

Для Windows

Мінімальні	Рекомендовані
ОС: 64-bit Windows 10	ОС: 64-bit Windows 10
Процесор: Intel Core i5-6600 (3.3Ghz), AMD Ryzen 5 1400 (3.2Ghz)	Процесор: Intel Core i7-8700 (3.2Ghz) or AMD Ryzen 5 3600 (3.6 Ghz)
Оперативна пам'ять: 8 GB	Оперативна пам'ять: 8 GB
Відеокарта: NVIDIA GeForce GTX 960 4GB, AMD Radeon RX 470 4GB	Відеокарта: NVIDIA GeForce 1080 Ti or AMD Radeon RX 5700 XT or INTEL Arc A770
Місце на диску: ? GB	Місце на диску: ? GB

Рисунок В.11 – Геймдизайн-документ сторінка 11

Опис гри

Опис ігрового процесу

Гра має лінійний сюжет, але також в ній є розгалуження у вигляді додаткових квестів. Основний сюжет складається з 7 місій, для кожної з яких розроблена окрема локація. Додаткові квести інколи мають окремі локації, але найчастіше базуються на локаціях з основного сюжету. Гра є шутером від першої особи, де гравець грає за мага, що перетворює свою магічну енергію на стихійну завдяки особливому виду зброї. Вибір спорядження/місій та сюжетні діалоги проходить у хабі – кабінеті придворного мага. Завдяки іммерсивності гри, гравець може обирати нові, несподівані, винахідливі шляхи для виконання свого завдання: від лінійного проходження до створення імпровізованих сходів із льоду, або вбивства супротивника шляхом скидання із прірви завдяки повітряній пушці.

Механіки

Бій:

- Найчастіше в битві використовується магічна «вогнепальна» зброя, що дозволяє використовувати певні стихії.
- Холодна зброя, що дозволяє виконати моментальне вбивство із стелсу, або наносити удари у відкритій битві.
- Шкала здоров'я – після отримання поранень, кількість очків здоров'я знижується і це відображається на шкалі. При досягненні 0 – наступає моментальна смерть. Поступово, лише поза битвою, очки здоров'я відновлюються.
- Шкала мани – потрібна для стрільби із магічної зброї. Різна зброя використовує різну кількість очків мани. Поступово відновлюється під час битви, але поза неї – набагато швидше.

Вплив стихій на навколишнє середовище:

- Лід: застигає, створюючи колізію, попадаючи на статичні об'єкти. Завдяки цьому гравець може створити імпровізовану барикаду, сходи і т.д. При попаданні на живу істоту – підморожує її, тим самим уповільнюючи її рухи. Час дії – 12 секунд, після чого додаткова колізія зникає, а істоти перестають отримувати штраф до рухливості. Наносить середню кількість урону.
- Вогонь: підпалює легкозаймисті предмети (спеціальні бочки з горючим, факели, свічки, тощо). Підпалює живих істот, тим самим,

Рисунок В.12 – Геймдизайн-документ сторінка 12

наносячи ефект горіння, що триває 3 секунди (під час дії цього ефекту, вони отримують урон). Наносить велику кількість урону.

- Вода: Створює водні поверхні, які можна наелектролізувати, тим самим створивши пастку. Можна використати проти живої істоти, тим самим зробивши її більш вразливою до електрики. Наносить низьку кількість урону.
- Електрика: немає впливу на статичні об'єкти окрім водяних поверхонь, які він наелектролізовує. Жива істота застигає від цієї стихії на 1 с. Наносить середню кількість урону.
- Вітер: відштовхує фізичні об'єкти та істоти, дозволяючи скинути їх з обриву, або швидко перемістити. Наносить низьку кількість урону.
- Кислота: Не має впливу на статичні об'єкти. При потраплянні на живу істоту, накладає на неї ефект роз'їдання, що триває 1 хв. Наносить низьку кількість моментального урону, але за весь період дії ефекту накопичується велика кількість урону.
- Плазма: Не має впливу на статичні об'єкти. При потраплянні на живу не створює додаткових ефектів. Наносить найбільшу кількість урону.

Стелс:

- Умова виявлення гравця: У противника є спеціальні колізії: сферична (відповідає слуху противника) та конічна (відповідає зору). Якщо гравець заходить в них, або робить шумні/підозрілі дії – у противника починає заповнюватись шкала тривоги. Якщо джерело тривоги не щезає – через 3 секунди гравця буде помічено. Також є додаткові колізії – з меншим радіусом для сфери та кутом для конуса. В цих зонах гравця помічають одразу. У режимі стелсу, колізії тривоги (жовті на малюнку) не працюють, якщо не шуміти. Колізії моментального помічання (червоні на малюнку) працюють без змін.

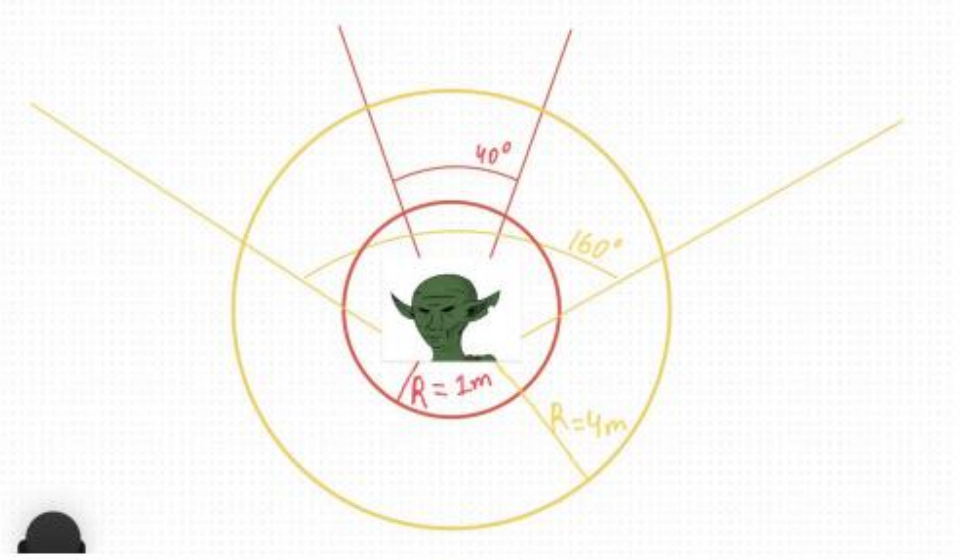


Рисунок В.13 – Геймдизайн-документ сторінка 13

- Моментальне вбивство: Якщо гравця не було помічено і противник знаходиться в полі досягнення – його можна моментально вбити холодною зброєю не здіймаючи галасу.

Іммерсивність:

- З деякими ігровими об’єктами (такими як важіль, двері та інше) можна взаємодіяти.
- Часто зустрічаються місця до яких можна потрапити лише якщо провзаємодіяти з іншими об’єктами, або створити собі шлях туди, або прибрати перешкоди.
- Гравець може підіймати неважкі об’єкти та переміщувати їх (як у Скайрім).
- Деякі об’єкти можна підібрати (ніяк не відображається у hud), що дозволяє взаємодіяти з новими об’єктами. Наприклад, таким чином можна підібрати недостатню деталь для механізму, або ключ від закритої двері.
- Ігрові персонажі можуть отримувати урон від прискорення (гальмування). Тобто, якщо їх моментальне прискорення завелике, що в рамках гри може бути досягнуто лише за рахунок падіння, удару об стіну.

Рисунок В.14 – Геймдизайн-документ сторінка 14

Інтерфейси

Екран запуску гри

Екран загрузки рівня

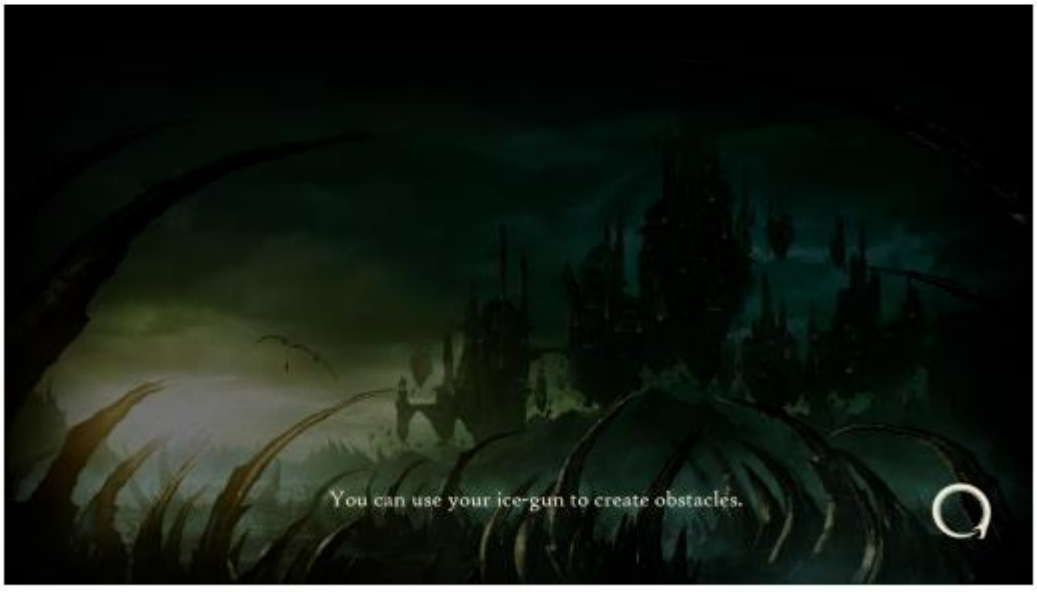


Рисунок В.15 – Геймдизайн-документ сторінка 15

Геймплей під час битви

Геймплей поза битвою

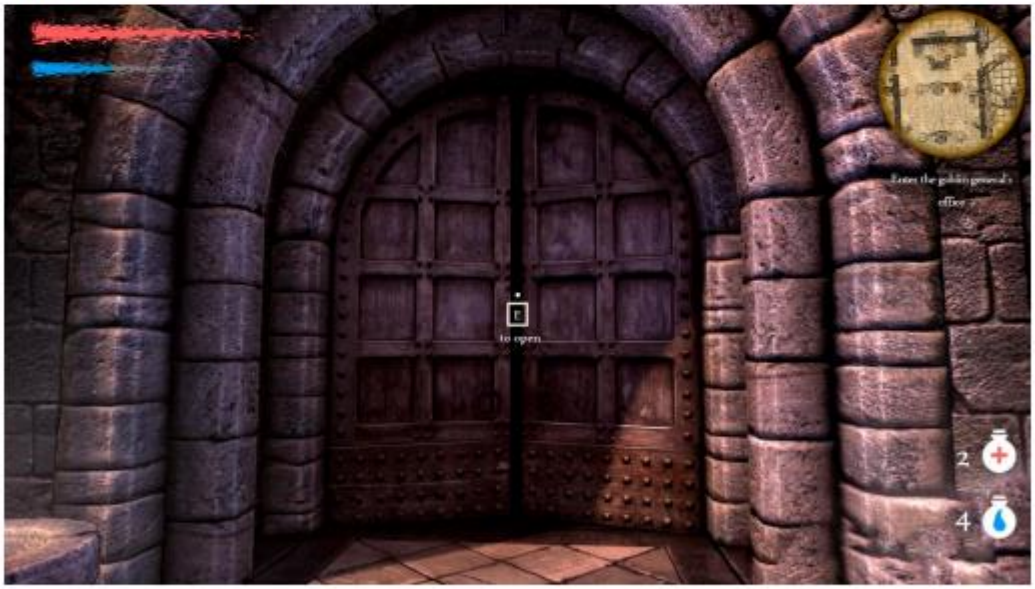


Рисунок В.16 – Геймдизайн-документ сторінка 16

Хаб (місце для діалогів з персонажами, вибору рівня, екіпіровки, тощо)

Інтерфейс інвентаря

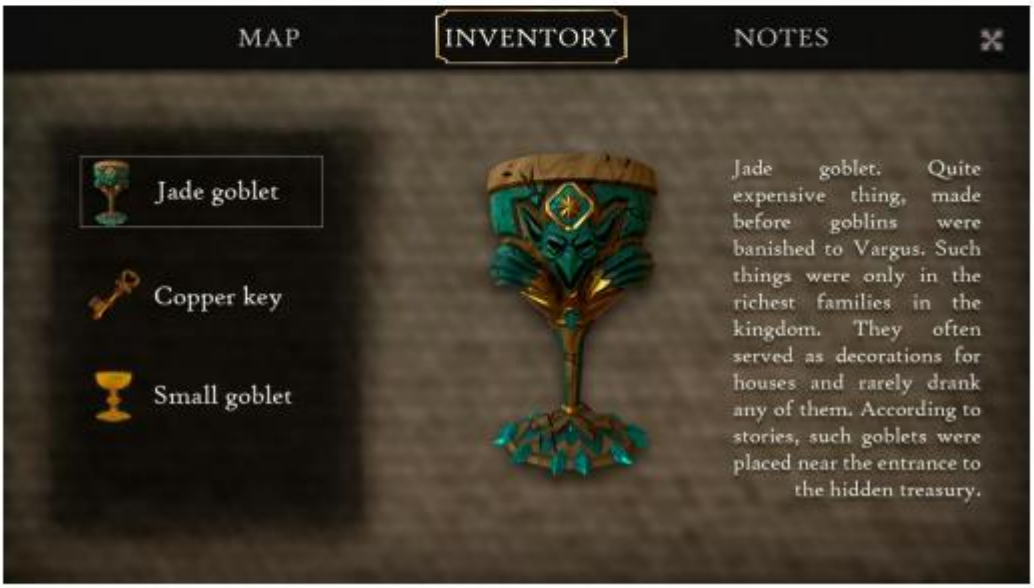


Рисунок В.17 – Геймдизайн-документ сторінка 17

Інтерфейс з мапою

Інтерфейс із записками

Рисунок В.18 – Геймдизайн-документ сторінка 18

Візуал

ГГ

https://docs.google.com/presentation/d/1p0ElywFLtYaHxAAQaiij2w4imadesnKYySpUzfF2rIU/edit?usp=sharing

Локації

https://docs.google.com/presentation/d/1zzaVCnDaXkSvSHeo7MkjOxBQWm6YxWE7OahR9zywB6Y/edit?usp=sharing

Другорядні персонажі

https://docs.google.com/presentation/d/1AwujuGxxOABpTjDZgNPuL0yl4EWIFYFMmDVcQIyuKNo/edit?usp=sharing

Рисунок В.19 – Геймдизайн-документ сторінка 19

ДОДАТОК Г

Тест-план

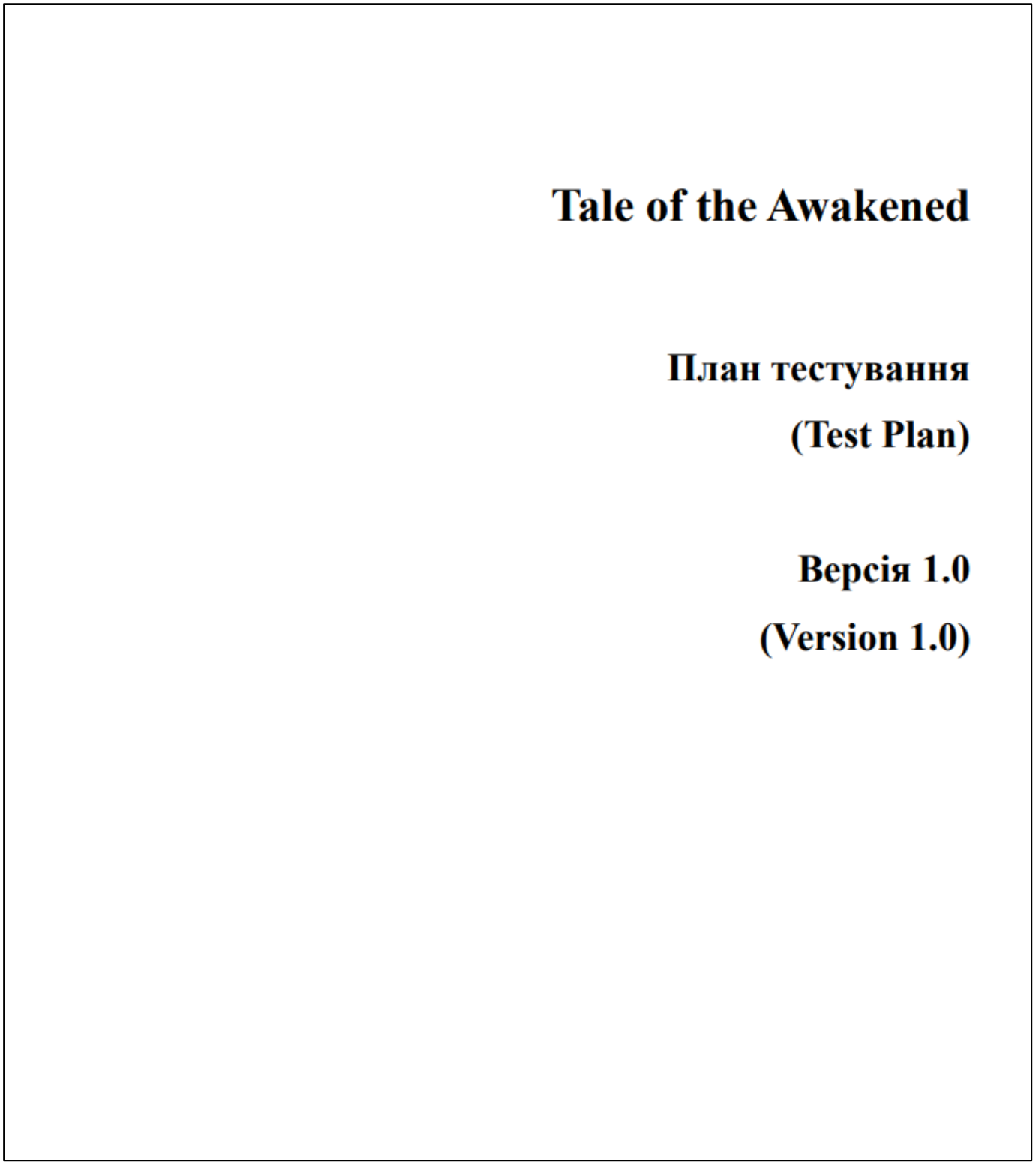


Рисунок Г.1 – Тест-план сторінка 1

Зміст

Рисунок Г.2 – Тест-план сторінка 3

1. Вступ (Introduction)

1.1.Мета (Purpose)

Метою даного плану тестування є детальний опис процедур і підходів до тестування гри "Tale of the Awakened". Документ має на меті забезпечити високу якість продукту через систематичне виявлення та виправлення помилок, покращення геймплею та інтерфейсу. Цей документ призначений для координації діяльності команди тестувальників, а також для інформування розробників про методи та результати тестування.

1.2.Довідкова інформація (Background)

Головний персонаж гри - Кіт, звичайна людина, що несподівано для себе відкриває здатність до магії після розбиття давнього артефакту. Незважаючи на своє походження, він знаходить у собі силу стати могутнім магом, що володіє різноманітними стихіями. Його минуле - це історія про звичайне життя у світі, де магія є привілеєм небагатьох, але одна подія радикально змінює його долю.

Гра має лінійний сюжет, але також в ній є розгалуження у вигляді додаткових квестів. Основний сюжет складається з 7 місій, для кожної з яких розроблена окрема локація. Додаткові квести інколи мають окремі локації, але найчастіше базуються на локаціях з основного сюжету.

- Гра пропонує широкий арсенал зброї Найчастіше в битві використовується магічна «вогнепальна» зброя, що дозволяє використовувати певні стихії.
- Холодна зброя, що дозволяє виконати моментальне вбивство із стелсу, або наносити удари у відкритій битві.

Рисунок Г.3 – Тест-план сторінка 4

Шкала здоров'я – після отримання поранень, кількість очків здоров'я знижується і це відображається на шкалі. При досягненні 0 – наступає моментальна смерть. Поступово, лише поза битвою, очки здоров'я відновлюються.

Шкала мани – потрібна для стрільби із магічної зброї. Різна зброя використовує різну кількість очків мани. Поступово відновлюється під час битви, але поза неї – набагато швидше.

Система урону в "Tale of the Awakened" базується на стихійному впливі та фізичних атаках. Вороги та гравець можуть отримувати урон від різних стихій, що впливають на них по-різному: вогонь може підпалити та завдати тривалого урону; лід - заморозити та сповільнити; блискавка - паралізувати ціль. Ефективність урону залежить від вразливостей ворогів до певних стихій, що додає стратегічний елемент у виборі зброї та магії для бою.

1.3. Галузь застосування (Scope)

Метою тестування гри "Tale of the Awakened" є перевірка коректної роботи її функціоналу та зручності для користувача. Результатом процесу тестування повинен стати розгорнутий огляд, що надасть розробникам, менеджерам і користувачам даного продукту картину якості ігрового процесу та юзабіліті. Для аналізу системи використовується методика "чорної скриньки", (мануальне, функціональне та нефункціональне тестування). Гра перевіряється на платформі Windows 11.

Ризики і непередбачувані обставини, що можуть зустрітися у проекті: недостатня база знань для розробки проекту.

1.4. Визначення проекту (Project Identification)

Рисунок Г.4 – Тест-план сторінка 5

6

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

Таблиця 1.1 – Документація

Документ і версія / дата	Створено або доступно	Отримано або перевірено	Автор або ресурс	Примітка
Специфікація вимог	Так	Так	Бізнес-аналітик	
Функціональна специфікація	Так	Так	Бізнес-аналітик	
План проекту	Так	Так	Project manager	
Специфікація дизайну	Ні	Ні	UX дизайнер	
Прототип	Так	Так	Команда розробників	
Керівництво користувача	Так	Так	Усі розробники та бізнес- аналітик , тестер	
Бізнес модель	Так	Так	Бізнес-аналітик	
Модель даних	Так	Так	Команда розробників	
Бізнес-функції	Так	Ні	Команда розробників	
Оцінка ризиків	Так	Так	QA	

Рисунок Г.5 – Тест-план сторінка 6

2. Вимоги до тестування (Requirements for Test)

2.1. Функціональні вимоги

Перелік функцій системи, які будуть тестуватися:

- Інтерфейс меню:
  - а) анімації переходів між елементами інтерфейсу;
  - б) переходи за допомогою клавіатури поміж елементами інтерфейсу;
  - в) вікна з описом предметів в інвентарі;
  - г) перевірка меню нотатків та записок;
  - д) перевірка інтерфейсу налаштувань гри;
  - е) перевірка інтерфейсу закінчення гри (сметрі).
- Інтерфейс гри:
  - а) перевірка відображення значень стану гравця, а саме: здоров'я, кількість мани.
- Ігровий процес:
  - а) перевірка механік стрілянини, а саме: завдання шкоди, фізичне відображення ефектів попадання стихійних снарядів;
  - б) перевірка коректності завантаження текстур;
  - в) перевірка поведінки мешів персонажів
  - г) перевірка механіки поповнення здоров'я через їжу;
  - г) проходження завдань;
  - д) перевірка механіки керування предметами;
  - е) перевірка коректності розподілення маси в предметах;
  - є) перевірка нанесення урону гравцю самим гравцем (падіння з вікна);
  - і) перевірка режиму стелсу;
  - ї) перевірка зони сприйняття ворогів (зір, слух);
  - й) перевірка іммерсивності рівня;

Рисунок Г.6 – Тест-план сторінка 7

к) перевірка взаємодії з оточуючим середовищем;

л) перевірка колізій на локації;

м) перевірка відображення попадання стихійних снарядів в певні предмети на локації.

- Штучний інтелект ворогів:

а) перевірка патрулювання ворогів;

б) перевірка системи пошуку гравця ворогом;

в) перевірка коректності поведінки ворогів в різних випадках.

Рисунок Г.7 – Тест-план сторінка 8

3. Типи тестування. Стратегія тестування (Test Strategy)

У процесі оцінки системи будуть застосовані наступні методики тестування:

- Функціональне Тестування: Зосереджене на перевірці функцій гри, щоб забезпечити їх правильну роботу згідно з вимогами.
- Тестування Інтерфейсу Користувача: Спрямоване на оцінку інтуїтивності, зручності та візуальної привабливості інтерфейсу гри.
- Тестування Безпеки та Контролю Доступу: Має на меті виявити потенційні вразливості у захисті даних гри та систем контролю доступу.
- Тестування Відмовостійкості та Відновлення: Перевіряє здатність гри протистояти помилкам та відновлювати роботу після збоїв.
- Тестування Конфігурації: Оцінює сумісність гри з різними системними налаштуваннями та обладнанням.
- Тестування Інсталяції: Переконується, що процес встановлення гри проходить безперешкодно на підтримуваних платформах.

Стратегія Тестування:

На першому етапі проводиться комплексне системне тестування, яке базується на аналізі взаємодії користувача з інтерфейсом гри від моменту запуску. Метою цього етапу є ідентифікація потенційних недоліків у дизайні інтерфейсу та внесення пропозицій щодо його оптимізації для покращення користувацького досвіду.

На другому етапі, в процесі функціонального тестування, планується виявлення помилок, які впливають на ігровий процес, шляхом тестування нетипових ситуацій та перевірки основних ігрових сценаріїв.

3.1. Типи тестування (Testing Types)

Функціональне тестування (Function Testing)

Рисунок Г.8 – Тест-план сторінка 9

Функціональне тестування перевіряє, наскільки гра відповідає заданим вимогам. Цей метод зосереджений на перевірці результатів обробки даних, імітуючи реальне користування системою без врахування її внутрішньої структури. Він орієнтований на очікування та потреби користувачів. Детальний план тестування представлено у таблиці 3.1.

Таблиця 3.1 – Функціональне тестування

Мета випробування	Забезпечення відповідності функціональності гри заданим вимогам. Метою є переконатися, що всі функції працюють коректно, відповідають визначеним специфікаціям та задовольняють потреби користувачів. (використовується підхід «чорний ящик»).
Технічний прийом	Використання заздалегідь визначених сценаріїв та тестових випадків, що відтворюють різноманітні умови використання програми користувачами, без залучення інформації про внутрішню структуру програми. Основна увага приділяється тестуванню кожної функції гри на відповідність очікуваному результату.
Критерії завершення	Всі функції програми протестовано і підтверджено їхню правильну роботу в усіх тестових сценаріях. Відсутність помилок, що впливають на основні операції користувачів, і забезпечення, що всі виявлені дефекти були виправлені та перевірені.
Спеціальні рекомендації	Головний функціонал системи потрібно тестувати на різних ОС, різних екранах, на різному «залізі».

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).

Рисунок Г.9 – Тест-план сторінка 10

Таблиця 3.2 – Тестування інтерфейсу користувача

Мета випробування	Перевірка на інтуїтивність, зручність використання і відповідність інтерфейсу користувача до встановлених вимог, а також ідентифікація можливих незручностей у взаємодії користувача з програмою.
Технічний прийом	Використовувати тестові сценарії, що наслідують реальну поведінку користувачів при роботі з інтерфейсом, та застосування евристичного аналізу для оцінювання зручності користування інтерфейсом..
Критерії завершення	Усі плановані тестові сценарії виконані без виявлення критичних помилок, забезпечена відповідність очікувань користувачів до фактичної роботи інтерфейсу, отримано позитивні відгуки від тестувальників щодо інтуїтивності та зручності використання..
Спеціальні рекомендації	Запрошувати до тестування різноманітні групи користувачів для охоплення широкого спектру можливих взаємодій з інтерфейсом. Використовувати інструменти для відстеження руху очей, щоб аналізувати, наскільки ефективно розміщені елементи інтерфейсу з точки зору користувача.

Тестування Відмовостійкості та відновлення (Failover and Recovery Testing)

Цей вид тестування гарантує працездатність гри або збереження даних. Відмово стійка система має:

- точки відновлення;
- локалізацію пошкоджень у пошкодженому компоненті;
- обмеження розповсюдження несправності;

Рисунок Г.10 – Тест-план сторінка 11

- доступність режимів реверсії.

Таблиця 3.3 - Тестування відмовостійкості та відновлення

Мета випробування	Забезпечення стабільності та надійності гри, здатності системи ефективно реагувати на помилки та відновлювати свою роботу після збоїв або втручань ззовні.
Технічний прийом	Симуляція різноманітних збоїв та відмов, включаючи раптове вимкнення живлення, втрату мережевого з'єднання або збої програмного забезпечення. Моніторинг системи на предмет її здатності автоматично відновлювати функціональність без втрати даних користувача.
Критерії завершення	Усі симульовані сценарії збоїв та відмов пройдені, при цьому система продемонструвала здатність до швидкого відновлення роботи з мінімальними втратами даних. Записано час відновлення для різних типів збоїв.
Спеціальні рекомендації	Розробити комплексний план відновлення після збоїв, що включає автоматичне створення резервних копій даних гри та їх швидке відновлення. Проводити регулярні тренування команди технічної підтримки для ефективного реагування на критичні збої.

Тестування інсталяції (Installation Testing)

Інсталяційне тестування зосереджено на перевірці процесів успішного встановлення, налаштування, а також оновлення або деінсталяції гри. У сучасному ігровому середовищі загальноприйнятою практикою є розгортання ігор через інсталяційні програми на ігрових платформах, які також вимагають ретельного тестування.

Рисунок Г.11 – Тест-план сторінка 12

13

Таблиця 3.4 - Тестування інсталяції

Мета випробування	Забезпечити, що гра може бути успішно встановлена, налаштована, оновлена або видалена з системи користувача без помилок та перешкод.
Технічний прийом	Виконання серії тестів інсталяції через стандартні інсталяційні процедури, які включають використання інсталяторів на різних ігрових платформах, перевірка налаштувань за замовчуванням та користувацьких налаштувань, а також тестування процесів оновлення та видалення програми.
Критерії завершення	Усі процеси інсталяції, налаштування, оновлення та видалення пройдені без виявлення критичних помилок або збоїв. Програма успішно інсталюється та функціонує на підтримуваних операційних системах згідно з документацією.
Спеціальні рекомендації	Проведення тестів на різноманітних конфігураціях апаратного забезпечення та версіях операційних систем, щоб охопити можливі варіанти використання кінцевими користувачами. Врахування особливостей інсталяційних процедур на різних ігрових платформах для виявлення специфічних для платформи помилок.

Інструменти (Tools)

Таблиця 3.5 – Інструменти

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

Рисунок Г.12 – Тест-план сторінка 13

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	1	Забезпечує управління наглядом. Обов'язки: - технічна підтримка; - придбання відповідних ресурсів; - забезпечення управлінської звітності.
Проектувальник тестів	1	Визначення, пріоритетів, і реалізація тестів. Обов'язки: - створення плану тестування; - генерація тестових моделей; - оцінка ефективності тестових зусиль.
Тестувальник	1	Виконання тестів. Обов'язки: - виконання тестів; - журнал результатів; - відновлення в журналі реєстрації після помилок.
Тестовий системний адміністратор	1	Забезпечує тестове середовище і управління активами. Обов'язки: - адміністрування тестової системи управління; - встановлення і управління доступом до тест-системи.

4.2. Система (System)

Рисунок Г.13 – Тест-план сторінка 14

Таблиця 4.2 – Система

Мінімальні	Рекомендовані
ОС: 64-bit Windows 10	ОС: 64-bit Windows 10
Процесор: Intel Core i5-6600 (3.3Ghz), AMD Ryzen 5 1400 (3.2Ghz)	Процесор: Intel Core i7-8700 (3.2Ghz) or AMD Ryzen 5 3600 (3.6 Ghz)
Оперативна пам'ять: 8 GB	Оперативна пам'ять: 8 GB
Відеокарта: NVIDIA GeForce GTX 960 4GB, AMD Radeon RX 470 4GB	Відеокарта: NVIDIA GeForce 1080 Ti or AMD Radeon RX 5700 XT or INTEL Arc A770
Місце на диску: 30 GB	Місце на диску: 30 GB

Рисунок Г.14 – Тест-план сторінка 15

5. Етапи проекту (Project Milestones)

Тестування має охоплювати тестові процедури для кожного виду випробувань, описаних у попередніх секціях. Важливо встановити конкретні фази проекту для забезпечення звітності про прогрес стану проекту.

Таблиця 5.1 – Етапи проекту

Цільове завдання	Обсяг робіт	Дата початку	Дата закінчення
План випробувань	5 год	25.02.2024	25.02.2024
Тест – дизайн	10 год	26.02.2024	01.02.2024
Реалізація випробувань	50 год	01.03.2024	21.03.2024

Рисунок Г.15 – Тест-план сторінка 16

6. Кінцевий продукт (Deliverables)

6.1. Тестова модель

Таблиця 6.1 – Етапи проекту

Test Plan	Повний документ планування, який містить підхід, ресурси, графік.
Test Cases	Група передумов виконання, очікуваних виконання умов та результатів. Розроблюється для тестового сценарію.
Requirements Traceability Matrix	Матриця відповідності вимог використовується QA-інженерами для валідації покриття вимог щодо продукту тестами. Мета документу полягає в тому, щоб з'ясувати: – які вимоги покриті тестами, а які ні; – надлишковість тестів.
Test Data	Дані, які існують до виконання тесту. Використовуються для виконання Test Cases.
Defect Report	Технічний документ, який містить в собі повний опис бага, що включає інформацію, як про сам баг, так і про умовах виникнення даного бага.
Test Summary Report	Документ високого рівня, який підсумовує проведені тестові дії.

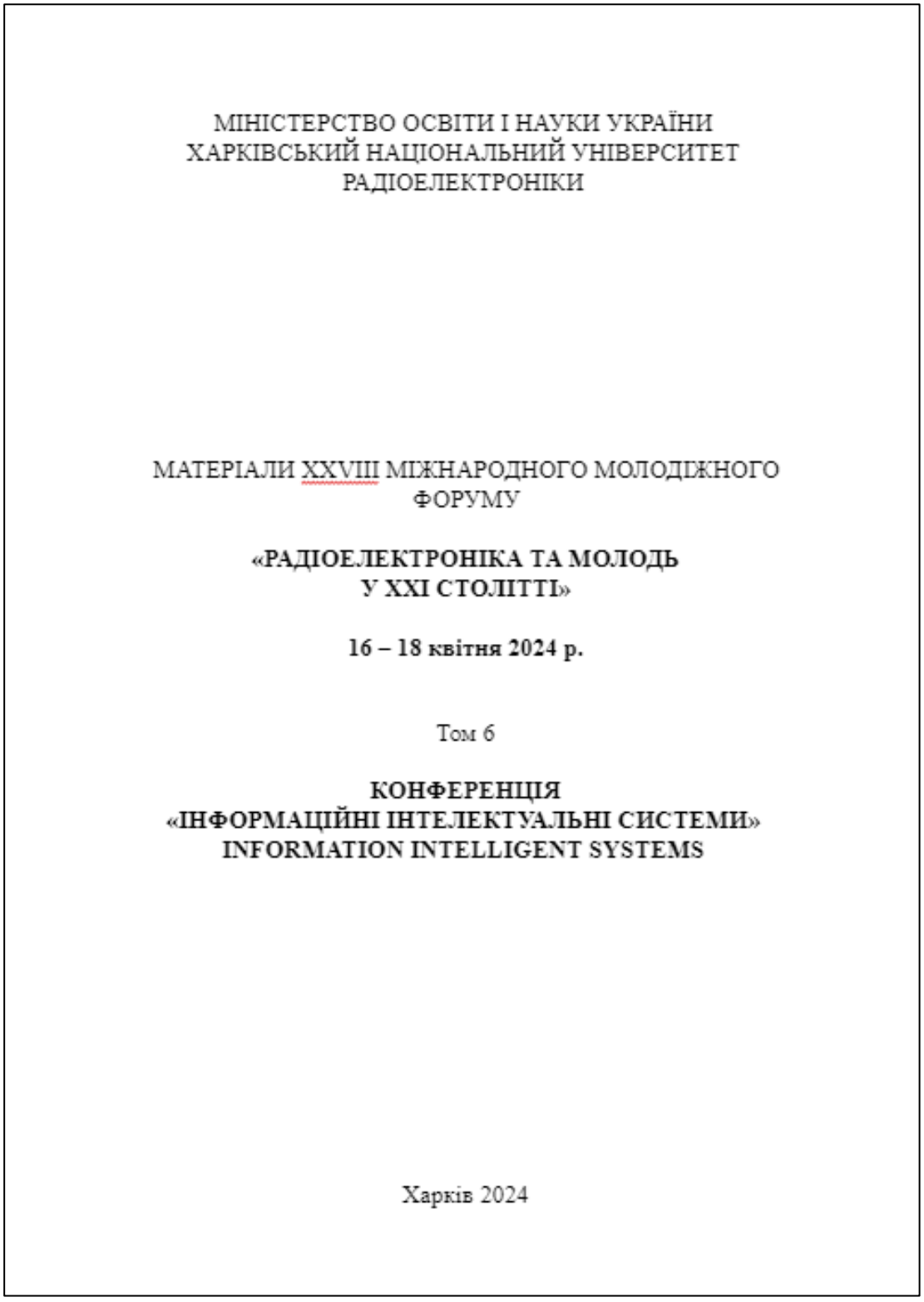
6.2. Звіти з дефектів (Defect Reports)

Bug-report

Рисунок Г.16 – Тест-план сторінка 17

ДОДАТОК Ґ

Тези доповіді для науково-практичної інтернет-конференції



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
РАДІОЕЛЕКТРОНІКИ

МАТЕРІАЛИ XXVIII МІЖНАРОДНОГО МОЛОДІЖНОГО ФОРУМУ

«РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ У XXI СТОЛІТТІ»

16 – 18 квітня 2024 р.

Том 6

КОНФЕРЕНЦІЯ
«ІНФОРМАЦІЙНІ ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ»
INFORMATION INTELLIGENT SYSTEMS

Харків 2024

Рисунок Ґ.1 – Обкладинка збірника

АЛФАВІТНИЙ ПОКАЖЧИК

953

Рисунок Ґ.2 –Алфавітний покажчик збірника

УДК 004.946 DOI: https://doi.org/10.30837/IYF.IIS.2024.468

МЕТОДИ ОПТИМІЗАЦІЇ ВІЗУАЛЬНИХ ЕФЕКТІВ У UNREAL ENGINE

Бєліков Д. Ю.
Науковий керівник – ст. викл. каф. ПІ Новіков Ю. С.
Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна
e-mail: danylo.bielikov@nure.ua

This research examines optimization methods for visual effects in Unreal Engine, focusing on enhancing game performance and graphics quality. It highlights the importance of balancing visual fidelity with performance, especially in the FPS genre. The study investigates techniques such as LOD (Level of Detail) for particles, optimization of particle materials, particle batching, and simulation caching to increase FPS, ensuring fluid gameplay and high-quality visuals. The goal is to offer developers guidance on using these optimization tools and techniques effectively. Through the application of these methods in the development of 'Tale of The Awakened,' the paper illustrates how developers can enhance game performance and visual quality with minimal compromises.

У сучасному світі відеоігор, особливо у жанрі FPS (First Person Shooter – Шутер від першої особи), розробники невпинно прагнуть знайти ідеальний баланс між візуальною якістю та продуктивністю. Метою даного дослідження є аналіз та розробка методів оптимізації візуальних ефектів в Unreal Engine, спрямованих на досягнення цього балансу без значних компромісів. Зокрема, акцент робиться на оптимізації за допомогою системи Niagara, яка відіграє ключову роль у підвищенні FPS (Frames Per Second – Кадрів за секунду), забезпечуючи плавність ігрового процесу та збереження високої якості візуальних ефектів, критично важливих для занурення гравця у ігровий світ.

Використання LOD (Level of Detail – Рівень Деталізації) для частинок в Unreal Engine є ключовою стратегією для оптимізації візуальних ефектів, зокрема, в сценах з інтенсивним використанням частинок. Цей метод дозволяє зменшувати деталізацію частинок залежно від їх відстані до камери, тим самим знижуючи навантаження на графічний процесор і підвищуючи продуктивність, що, в свою чергу, дозволяє збільшити FPS (Frames Per Second – Кадрів за секунду) і поліпшити ігровий досвід без суттєвої втрати візуальної якості.

Рисунок Ґ.3 – Матеріал тез сторінка 1

Таблиця 1 – Порівняння часу рендеру ігрових сцен із застосуванням методу Level of Detail на системі частинок та без

	Кількість систем частинок на сцені	Кількість трикутників що рендериться	GPU time для рендеру одного кадру	Загальний час рендеру одного кадру
Без використання LOD	500	968000	33 мс	36 мс
З використанням LOD	500	357000	25 мс	31 мс

Оптимізація матеріалів частинок є критично важливою для підвищення продуктивності в Unreal Engine, особливо в сценах з великою кількістю візуальних ефектів. Ефективне використання матеріалів включає мінімізацію складності шейдерів та оптимізацію текстур, що дозволяє знизити час обробки кожної частинки, при цьому зберігаючи високу якість візуальних ефектів.

Таблиця 2 – Порівняння часу рендеру ігрових сцен із застосуванням складних шейдерів на системі частинок та простих

	Кількість систем частинок на сцені	Кількість трикутників що рендериться	GPU time для рендеру одного кадру	Загальний час рендеру одного кадру
Складні шейдери (з використанням параметру прозорості)	500	968000	157 мс	159 мс
Прості шейдери (без використання фізичних властивостей відображення)	500	968000	25 мс	31 мс

Оптимізація візуальних ефектів через batching (групування частинок) у системі Niagara в Unreal Engine є ключовою для збільшення продуктивності гри, забезпечуючи високу якість візуальних ефектів без необхідності жертвувати продуктивністю. Batching знижує загальну кількість draw calls – команд, які процесор відправляє графічному процесору для відображення об'єктів на екрані, об'єднуючи обробку великої кількості частинок в один такий запит. Це критично важливо у сценах з інтенсивним використанням частинок, оскільки зменшення кількості draw calls безпосередньо призводить до збільшення продуктивності.

Рисунок Ґ.3 – Матеріал тез сторінка 2

Таблиця 3 – Порівняння часу рендеру ігрових сцен із застосуванням batching'у та без

	Кількість систем частинок на сцені	Кількість трикутників що рендериться	GPU time для рендеру одного кадру	Загальний час рендеру одного кадру
Без застосування batching'у (окремі емітери для кожного кольору частинок)	500	968000	45 мс	47 мс
З використанням batching'у (зміна кольору досягається за допомогою instance ID та його використанні у шейдері частинки)	500	968000	26 мс	30 мс

Кешування симуляцій у системі Niagara в Unreal Engine представляє собою передову практику оптимізації, яка дозволяє значно підвищити продуктивність ігрового процесу за рахунок зберігання результатів складних симуляцій для їх подальшого використання без необхідності повторного обчислення. Такий підхід особливо ефективний для візуальних ефектів, які вимагають великих обчислювальних ресурсів, таких як динамічні імітації рідин, вогню, диму або складних погодних умов. Використовуючи кешовані симуляції, можна досягти високої деталізації та реалістичності цих ефектів при одночасному зниженні навантаження на процесор і графічний адаптер, що підвищує загальну продуктивність і забезпечує плавність ігрового процесу. Проте, слід враховувати, що використання кешування симуляцій може мати певні недоліки, зокрема, збільшене використання оперативної пам'яті для зберігання кешованих даних. Це може стати проблемою для систем з обмеженими ресурсами. Крім того, кешування може обмежити гнучкість візуальних ефектів, оскільки попередньо обчислені симуляції не зможуть адаптуватися до динамічних змін у ігровому середовищі так само ефективно, як ефекти, обчислені в реальному часі.

Незважаючи на ці потенційні обмеження, правильно налаштоване кешування симуляцій може значно покращити візуальну якість та продуктивність ігор, розроблених з використанням Unreal Engine. Важливо знайти оптимальний баланс між використанням кешованих симуляцій для ефектів, які не вимагають високої динаміки, та збереженням обчислень в реальному часі для елементів, чия взаємодія з ігровим світом має бути максимально реалістичною та гнучкою. Ретельне планування, тестування та оптимізація симуляцій є ключовими для досягнення цієї мети,

Рисунок Ґ.3 – Матеріал тез сторінка 3

дозволяючи розробникам створювати багаті та імерсивні ігрові середовища.

Таблиця 4 – Порівняння часу рендеру ігрових сцен із застосуванням кешованих систем частинок та тих, що прораховуються у реальному часі

	Кількість систем частинок на сцені	Кількість трикутників що рендериться	GPU time для рендеру одного кадру	Загальний час рендеру одного кадру
Без кешування (прорахунок у реальному часі)	500	968000	25 мс	650 мс
Закешована система	500	968000	25 мс	48 мс

На основі проведених тестів та аналізу різних методів оптимізації в системі Niagara в Unreal Engine можна зробити висновок, що ефективність кожного методу значною мірою залежить від конкретних умов та сценаріїв їх використання. Кешування складних симуляцій демонструє велику важливість для підвищення продуктивності, дозволяючи знизити обчислювальне навантаження і забезпечити плавність ігрового процесу. Водночас, інші методи, такі як інстанціювання частинок, хоча і є корисними для певних ситуацій, можуть мати менший вплив на загальну продуктивність в залежності від специфіки ігрового проекту. Важливо проводити індивідуальні тести та аналіз для кожного проекту, щоб ідентифікувати найефективніші стратегії оптимізації, зважаючи на унікальні вимоги та обмеження.

Список використаних джерел:

1. Комплекс навчально-методичного забезпечення навчальної дисципліни "Основи ігрової графіки" підготовки бакалавра для студентів усіх форм навчання спеціальності 121 – Інженерія програмного забезпечення: освітня програма "Програмна інженерія" / ХНУРЕ ; розроб. Н. А. Валенда. – Харків, 2017.

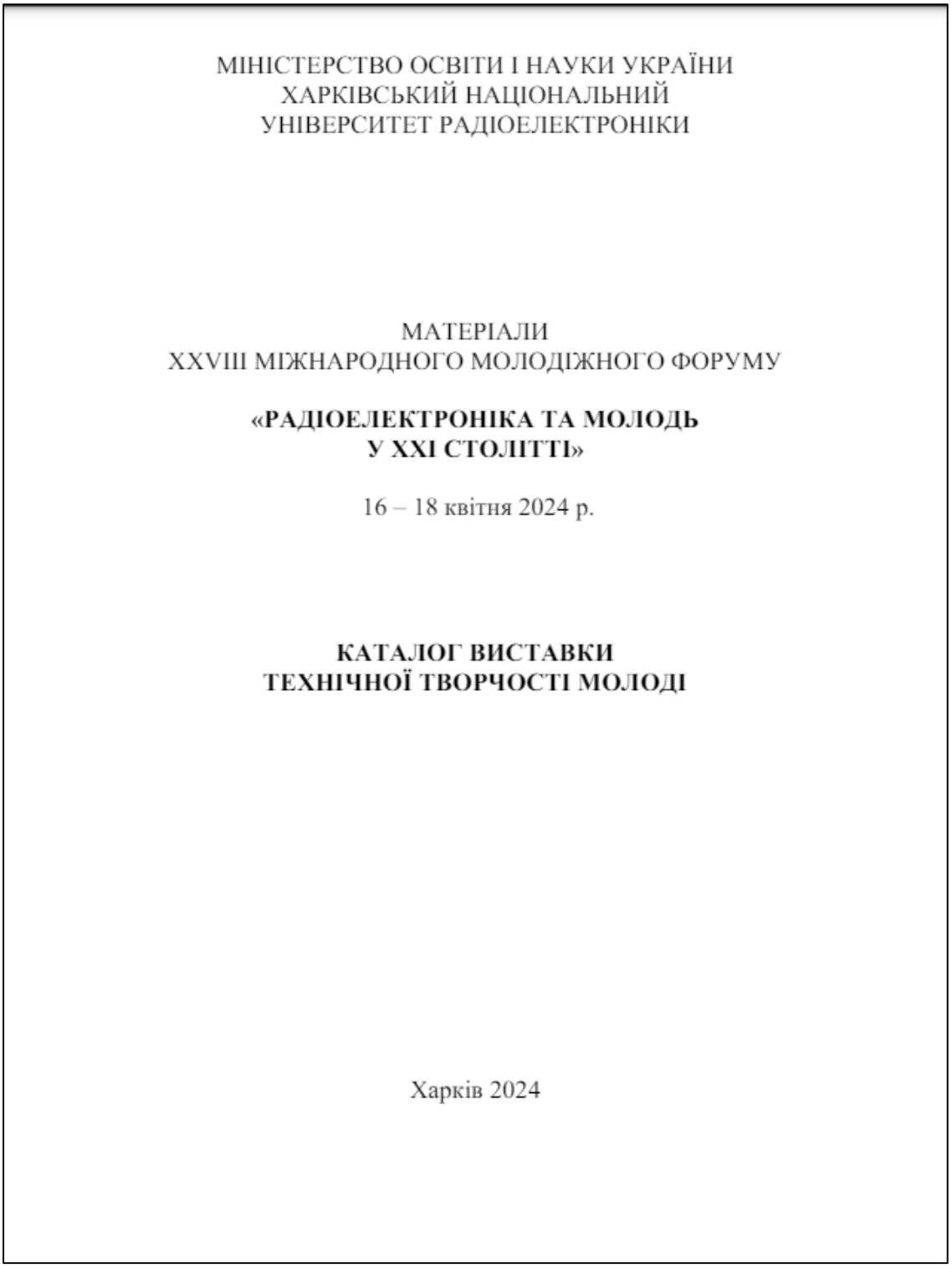
2. Офіційна документація з Unreal Engine: вебсайт. URL: https://docs.unrealengine.com/5.3/en-US/ (дата звернення: 06.03.2024).

3. Онлайн форум з Unreal Engine: вебсайт. URL: https://forums.unrealengine.com/tags/intersection/unreal-engine/blueprint (дата звернення: 06.03.2024).

Рисунок Ґ.4 – матеріал тез сторінка 4

ДОДАТОК Д

Тези доповіді для науково-практичної інтернет-виставки



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

МАТЕРІАЛИ
XXVIII МІЖНАРОДНОГО МОЛОДІЖНОГО ФОРУМУ

**«РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ
У XXI СТОЛІТТІ»**

16 – 18 квітня 2024 р.

**КАТАЛОГ ВИСТАВКИ
ТЕХНІЧНОЇ ТВОРЧОСТІ МОЛОДІ**

Харків 2024

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.

Рисунок Д.2 – Опис роботи для презентації

ДОДАТОК Е

Приклад програмного коду

Блупринт BP_FirstPersonCharacter

```
/** The character you control in the game, includes firing logic */
UCLASS(Blueprintable, BlueprintType)
class ABP_FirstPersonCharacter : public ACharacter
{
GENERATED_BODY()
public:
/** Please add a function description */
UFUNCTION(BlueprintPure)
void GetInteractivityTracer(FVector& LineStart, FVector& LineEnd);

/** Please add a function description */
UFUNCTION(BlueprintPure)
void GetDraggingLocation(FVector& Location);
public:
/** Please add a variable description */
UPROPERTY(BlueprintReadOnly, VisibleAnywhere, Category="Default")
TObjectPtr<UBP_ManaComponent_C> ManaComponent;

/** Please add a variable description */
UPROPERTY(BlueprintReadOnly, VisibleAnywhere, Category="Default")
TObjectPtr<UBP_DeathComponent_C> DeathComponent;

/** Please add a variable description */
UPROPERTY(BlueprintReadOnly, VisibleAnywhere, Category="Default")
TObjectPtr<UBP_HealthComponent_C> HealthComponent;

/** Please add a variable description */
UPROPERTY(BlueprintReadOnly, VisibleAnywhere, Category="Default")
TObjectPtr<USkeletalMeshComponent> FirstPersonMesh;

/** Please add a variable description */
UPROPERTY(BlueprintReadOnly, VisibleAnywhere, Category="Default")
TObjectPtr<UCameraComponent> FirstPersonCamera;

/** Please add a variable description */
UPROPERTY(BlueprintReadOnly, VisibleAnywhere, Category="BP_FirstPersonCharacter")
TObjectPtr<UTimelineComponent> DeathScreenOpacity;

/** Please add a variable description */
UPROPERTY(BlueprintReadOnly, VisibleAnywhere, Category="BP_FirstPersonCharacter")
TObjectPtr<UTimelineComponent> Start Crouching;
```

```
/** Please add a variable description */
UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Default")
bool bHasRifle;

/** Please add a variable description */
UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Default")
TObjectPtr<UBP_Weapon_Component_C> Weapon;

/** Please add a variable description */
UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Default")
bool AbleToCrouch;

/** Please add a variable description */
UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Default")
bool AbleToSprint;

/** Please add a variable description */
UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Default")
TObjectPtr<UWBP_HealthBar_C> HealthBar;

/** Please add a variable description */
UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Default")
double InteractionDistance;

/** Please add a variable description */
UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Default")
TObjectPtr<ABP_Food_C> AvailableFood;

/** Please add a variable description */
UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Default")
TObjectPtr<UWBP_HUD_C> HUD;

/** Please add a variable description */
UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Default")
TObjectPtr<UPrimitiveComponent> DraggingObject;

/** Please add a variable description */
UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Default")
double DraggingDistance;

/** Please add a variable description */
UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Default")
TObjectPtr<UWBP_ManaBar_C> ManaBar;
};
```

Блупринт BP_WeaponComponent

```
/** Please add a class description */
```

```
	UCLASS(Blueprintable, BlueprintType)
	class UBP_Weapon_Component : public USkeletalMeshComponent
	{
		GENERATED_BODY()
	public:
		/** Please add a function description */
		UFUNCTION(BlueprintPure, Category="Default")
		FTransform GetProjectileTransform(FVector& OutputPin);
	public:
		/** Please add a variable description */
		UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Default")
		TObjectPtr<ABP_FirstPersonCharacter_C> First PersonCharacter;

		/** Please add a variable description */
		UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Default",
	meta=(MultiLine="true"))
		FVector ProjectileOffset;

		/** Please add a variable description */
		UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Default")
		TObjectPtr<UClass> ProjectileClass;
};
```