МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ Чорноморський національний університет імені Петра Могили Факультет комп'ютерних наук Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інженерії програмного забезпечення, канд. техн. наук, доцент
_____ Є. О. Давиденко
«___»_____ 2024р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Розробка RPG-гри на рушії Unreal Engine 5

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ – 408. 22010801

Здобувач _____Д. Ю. Александров
«___»_____ 2024р.

Керівник д-р. техн. наук, професор _____А. В. Швед
«___»_____ 2024р.

_____А. О. Алексєєва
Консультант канд. техн. наук, доцент «___»_____ 2024р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ Чорноморський національний університет імені Петра Могили Факультет комп'ютерних наук Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри інженерії програмного забезпечення, доцент, канд. тех. наук.
_____Є. О. Давиденко
«_____» _____20_____р.

ЗАВДАННЯ на виконання кваліфікаційної роботи бакалавра

Видано здобувачу групи 408 факультету комп'ютерних наук

_____Александрову Денису Юрійовичу_____
(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи
____« Розробка RPG-гри на рушії Unreal Engine 5»____ Затверджена наказом по ЧНУ від «22»__грудня_____2023р. № ____269_____

2. Строк представлення кваліфікаційної роботи «___» _____20 _р.

3. Очікуваний результат роботи та початкові дані якщо такі потрібні:

Ігровий застосунок в жанрі RPG на основі рушія Unreal Engine 5

4. Перелік питань, що підлягають розробці:

__Дослідження основних принципів та особливостей жанру Action RPG; аналіз конкурентів на ринку ігор в цьому жанрі;вивчення можливостей та обмежень рушія Unreal Engine; специфікація вимог до ігрового застосунку; проєктування графічного дизайну; створення механік гри; розробка логіки гри; моделювання, кодування, тестування та налагодження ПЗ

5. Перелік графічних матеріалів:

Презентація

6. Завдання до спеціальної частини:

Аналіз охорони праці на робочих місцях фахівців з інформаційних технологій

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Канд. Техн. наук, доцент Алєксєєва Анна Олександрівна	Кафедра екології	Спеціальна частина з охорони праці

Керівник роботи д-р. техн. наук, професор Швед А.В.

(посада, прізвище, ім'я, по батькові)

_____

(підпис)

Завдання прийнято до виконання

Александров Денис Юрійович

(прізвище, ім'я, по батькові студента)

_____

(підпис)

Дата видачі завдання «____»_____20____р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи Тема:« Розробка RPG-гри на рушії Unreal Engine 5»

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРБ	06.11.2023	12.11.2023	виконано
2.	Огляд літератури за темою роботи	13.11.2023	19.11.2023	виконано
3.	Аналіз предметної галузі	20.11.2023	03.12.2023	виконано
4.	Розробка проєктних рішень	10.01.2024	19.01.2024	виконано
5.	Моделювання та конструювання ПЗ	29.01.2024	11.02.2024	виконано
6.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	12.02.2024	25.04.2024	виконано
7.	Розробка спеціальної частини з охорони праці	26.04.2024	29.04.2024	виконано
8.	Відгук керівника КРБ	10.05.2024	20.05.2024	виконано
9.	Оформлення КРБ та презентації	22.05.2024	03.06.2024	виконано
10.	Попередній захист	03.06.2024	05.06.2024	виконано
11.	Рецензування	12.06.2024	17.06.2024	виконано
12.	Захист кваліфікаційної роботи			

Розробив здобувач _____Александров Д. Ю._____ _____
(прізвище, ім'я, по батькові) *(підпис)*
«__» _____ 20__ р.

Керівник роботи __д-р. техн. наук, професор Швед А.В. _____
(посада, прізвище, ім'я, по батькові) *(підпис)*
«__» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Розробка RPG-гри на рушії Unreal Engine 5»

Студент 408 гр.: Александров Денис Юрійович

Керівник: д-р. техн. наук, професор Швед А.В.

Об’єктом кваліфікаційної роботи є процес розробки ігрового застосунку в жанрі RPG.

Предметом кваліфікаційної роботи є інструментарій рушія Unreal Engine 5 для розробки гри у жанрі RPG.

Метою кваліфікаційної роботи є розробка ігрового застосунку на платформі Unreal Engine 5, що відповідає інтересам гравців та прихильників комп’ютерних ролевих ігор за рахунок розробки унікального гемплею та сюжету гри.

Для досягнення мети необхідно вирішити такі завдання:

1) проведення аналізу існуючих ігор в жанрі RPG;
2) дослідження функціональних можливостей рушія Unreal Engine 5 та його інструментарію для розробки ігрового застосунку в жанрі RPG;
3) розробка концепту ігрового світу та персонажів на основі жанру RPG;
4) розробка геймплею, включаючи бої, локації та інші ігрові елементи;
5) розробка графічних, анімаційних та інших ефектів;
6) проведення тестування.

Робота спрямована на розробку високоякісного ігрового застосунку в жанрі RPG на основі рушія Unreal Engine.

Графіка вищого рівня, різноманітність геймплейних можливостей та використання передових технологій у розробці ігрових застосунків.

У вступі обґрунтовується актуальність обраної теми, визначається предмет та об’єкт роботи, надається короткий огляд завдань проєкту.

Перший розділ присвячений аналізу жанру RPG, вивченню існуючих аналогів та етапам розробки.

У другому розділі проводиться аналіз вимог до системи та процесу її створення, а також вивчається історія та основні поняття UML з подальшим створенням діаграм.

Третій розділ презентує використаний стек технологій для розробки ігрових застосунків та описує виконану роботу з моделювання.

У четвертому розділі детально розглядається процес створення інтерфейсу користувача, реалізація керування персонажем та інших компонентів гри.

У висновках виконується оцінка виконаної роботи та отриманих результатів.

У спеціальній частині з охорони праці та безпеки в надзвичайних ситуаціях йдеться про техніку безпеки при роботі в офісних приміщеннях із комп’ютерним обладнанням.

КРБ викладена на 67 сторінок, вона містить 4 розділи, 36 ілюстрацій, 1 таблицю, 16 джерел в переліку посилань.

Ключові слова: ігровий застосунок, RPG, Unreal Engine, графічна складова, геймплей, тестування, ігрові механіки.

ABSTRACT

to the qualification work of the bachelor

"Development of an RPG game using Unreal Engine 5"

Student 408 gr.: Aleksandrov Denys Yuriyovych

Head: dr. technical of Sciences, professor Shved A.V.

The object of the qualification work is the process of developing a game application in the RPG genre.

The subject of the qualification work is the toolkit of the Unreal Engine 5 engine for developing a game in the RPG genre.

The purpose of the qualification work is to develop a game application on the Unreal Engine 5 platform that meets the interests of players and fans of computer role-playing games by developing a unique gameplay and game plot.

To achieve this goal, it is necessary to solve the following tasks:

1) analysis of existing games in the RPG genre;

2) study of the functionality of Unreal Engine 5 and its tools for developing a game application in the RPG genre;

3) development of the concept of the game world and characters based on the RPG genre;

4) gameplay development, including battles, locations and other game elements;

5) development of graphics, soundtrack and other audiovisual effects;

6) conducting testing of the developed prototype.

The work is aimed at developing a high-quality game application in the RPG genre based on the Unreal Engine.

The graphic component of the game is of high quality, various gameplay options are available, and modern game application development technologies are used.

The introduction justifies the relevance of the chosen topic, defines the subject and object of the work, and provides a brief overview of the project tasks.

The first chapter is devoted to the analysis of the RPG genre, the study of existing analogues and the stages of development.

In the second chapter, an analysis of the requirements for the system and the process of its creation is carried out, as well as the history and basic concepts of UML are studied, followed by the creation of diagrams.

The third section presents the used technology stack for the development of game applications and describes the performed modeling work.

The fourth chapter examines in detail the process of creating a user interface, the implementation of character management and other game components.

In the conclusions, an analysis of the work carried out and the results obtained is carried out.

The special part on occupational health and safety in emergency situations deals with safety techniques when working in office premises with computer equipment.

Bachelor's qualifying work is laid out on 67 pages, it contains 4 sections, 36 illustrations, 1 table, 16 sources in the list of references.

Keywords: game application, RPG, Unreal Engine, graphic component, gameplay, testing, game mechanics.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

NPC – «Персонаж, керований комп'ютером»

RPG – Role Play Game

UE5 – Unreal Engine 5

UI – User Interface

UML – Unified Modeling Language

3D – Тривимірна модель

ВСТУП

Рольові ігри (RPG) завжди вражали своєю здатністю перенести гравців у фантастичні світи, де кожне рішення може визначати подальший хід подій. З плином часу, з технічним прогресом та зростанням амбіцій розробників, RPG ігри перейшли на новий рівень завдяки передовим технологіям.

Unreal Engine 5, як передовий рушій для розробки відеоігор, грає важливу роль у трансформації світу геймдеву. Відзначаючись потужними інструментами, він відкриває нові горизонти для створення ігор, в яких реалістичність графіки та глибина геймплею стають ключовими факторами. Unreal Engine 5 ставить перед розробниками завдання не лише технічної вдосконаленості, але й творчого застосування передових можливостей для втілення інноваційних ідей у галузі RPG ігор.

У контексті цього дослідження та розробки RPG проєктів, Unreal Engine 5 виступає не просто як інструмент, але як каталізатор для створення вражаючих ігор. Розробники отримують можливість ефективно використовувати передові технології для втілення ідей у фантастичні віртуальні світи, де кожен детально пророблений елемент ігрового середовища має вплив на геймплей та вражає гравців.

Робота націлена на подальше розуміння взаємодії між Unreal Engine 5 і розробкою RPG ігор. Досліджуємо ключові етапи створення гри, де технічні можливості рушія впливають на вибір геймдевелоперів та сприяють формуванню нових тенденцій у світі відеоігор. Це дозволить нам зазирнути у майбутнє геймдеву, де інновації та технічний прогрес об'єднуються для створення захопливих та передових RPG ігор.

Основна увага у цій роботі буде зосереджена на оптимізації геймплею та поліпшенні візуальної атмосфери гри.

Unreal Engine - потужний інструмент для розробки відеоігор з вражаючим графічним движком та розгалуженими можливостями, що дозволяють створювати реалістичні та захоплюючі ігрові витвори.

Об’єктом кваліфікаційної роботи є процес розробки ігрового застосунку в жанрі RPG.

Предметом кваліфікаційної роботи є інструментарій рушія Unreal Engine 5 для розробки гри у жанрі RPG.

Метою кваліфікаційної роботи є розробка ігрового застосунку на платформі Unreal Engine 5, що відповідає інтересам гравців та прихильників комп’ютерних ролевих ігор за рахунок розробки унікального гемплею та сюжету гри.

Для досягнення мети необхідно вирішити такі завдання:

1) проведення аналізу існуючих ігор в жанрі RPG;

2) дослідження функціональних можливостей рушія Unreal Engine 5 та його інструментарію для розробки ігрового застосунку в жанрі RPG;

3) розробка концепту ігрового світу на основі жанру RPG;

4) створення геймплею, що включає в себе бої, локації та інші ключові ігрові елементи;

5) розробка графічних, анімаційних та інших ефектів;

6) проведення тестування.

Результат цієї роботи полягає в глибокому розумінні та ефективному використанні Unreal Engine 5 для розробки рольової гри (RPG). Дослідження охоплює ключові аспекти роботи з рушієм, включаючи створення деталізованих графічних образів, реалізацію динамічного освітлення та розгалужені можливості для інтерактивності.

1 АНАЛІЗ ІГРОВИХ ЗАСТОСУНКІВ В ЖАНРІ RPG

1.1 Розгляд характерних рис ігрового жанру RPG

Рольова відеогра, або RPG, це жанр відеоігор, де основна частина гри полягає у керуванні персонажем чи групою персонажів. Гравці досліджують вигаданий світ гри, виконують різноманітні завдання, відомі як "квести", та розвивають своїх персонажів відповідно до сюжету гри.

Рольові відеоігри виникли в середині 1970-х років і швидко стали популярним жанром. Перші зразки, такі як pedit5 (рис 1.1) та Dungeon, були створені на основі настільної гри Dungeons & Dragons. Спочатку ці ігри складалися переважно з текстового опису подорожей у підземеллях, збору скарбів та боротьби з чудовиськами [1].

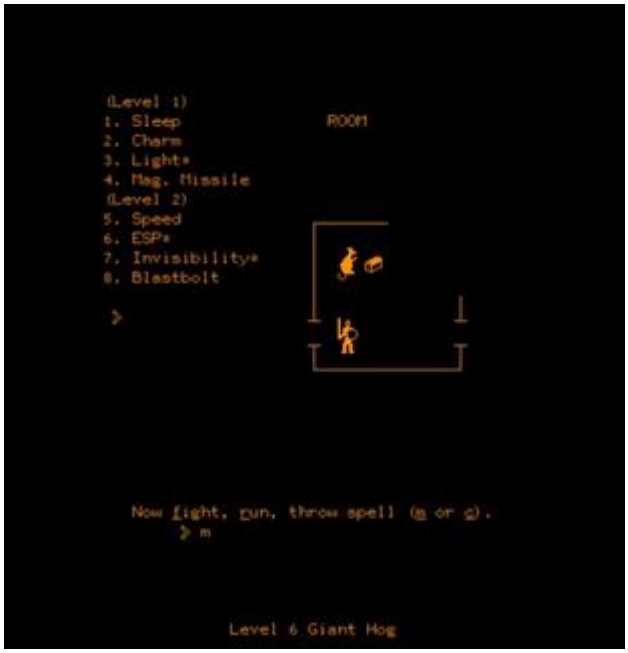


Рисунок 1.1 – Приклад ігрового процесу гри Pedit5

У 1980-х роках з'явилися культові серії, які визначили подальший розвиток жанру, такі як Rogue, Ultima і Wizardry (рис 1.2). Ці ігри вже використовували графіку, натомість текстового опису, що дозволило гравцям насолоджуватися виглядом світу з першої особи або зверху. Також в цей

період з'явилися перші RPG для ігрових приставок, наприклад, Dragonstomper та The Black Onyx.

Рисунок 1.2 – Приклад ігрового процесу гри Wizardry

У 1983 році вийшла Ultima III (рис 1.3), яка ввела нові стандарти для комп'ютерних рольових ігор, такі як псевдотривимірна графіка та окремі екрани для боїв і подорожей. У той же період з'явилася і серія Might and Magic, яка поєднувала фентезійну тематику з науковою фантастикою та різними класами персонажів.

Рисунок 1.3 – Приклад ігрового процесу гри Ultima III

У 1987 році виходить Final Fantasy (рис 1.4), яка стала класикою консольних японських рольових ігор. Ця серія відома різноманітністю бойових систем та незалежністю кожної гри у серії.

Рисунок 1.4 – Приклад ігрового процесу гри Final Fantasy

У 2000-х роках RPG-ігри стали кросплатформенними, що означає їх доступність на різних платформах. Було видано численні продовження вже відомих ігор, таких як Diablo III, The Elder Scrolls V: Skyrim і інші. Також спостерігалося змішання рольових ігор з іншими жанрами, а також поширення інді-ігор, включаючи ті, які наслідують класичні RPG [2].

У цей період також з'явилися антиутопічні проєкти, де сюжет був орієнтований на реалізм, наприклад, Divinity II: Ego Draconis, Tyranny та інші. Ці ігри відзначалися більш морально складними сюжетами та важливими рішеннями для гравця.

1.2 Аналіз особливостей ігрового застосунка в жанрі RPG

Розробка ігрового застосунку в жанрі RPG (рольова гра) також є цікавим і складним процесом. Тут важливо не лише створення захоплюючого геймплею, а й розробка глибокої сюжетної лінії, системи прогресу героя та

взаємодії зі світом гри. Ось кілька ключових етапів у процесі розробки ігрового застосунку в жанрі RPG:

1) **Концепція і Проєктування**: Визначення основної ідеї гри, її світу, персонажів, системи бойових механік, системи прогресу тощо. Проєктування гри полягає в створенні документації, що описує всі аспекти гри;

2) **Створення Світу**: Розробка вигляду та атмосфери гри, створення світу з усіма його місцями, персонажами, історіями тощо. Це включає концепт-арт, дизайн рівнів, архітектуру світу;

3) **Розробка Бойової Системи**: Створення механік бою, включаючи бойові навички, системи атаки та захисту, магічні здібності тощо;

4) **Створення Персонажів і Прогресу Героя**: Розробка головного героя та інших персонажів, системи прокачки, вдосконалення навичок і отримання нового обладнання;

5) **Діалогова Система і Сюжет**: Розробка сюжетної лінії, діалогів з персонажами, взаємодії зі світом гри, прийняття рішень, що впливають на хід подій;

6) **Графіка і Звук**: Розробка графічного дизайну, анімацій персонажів, музики та звукового супроводу, які створюють атмосферу гри;

7) **Тестування і Балансування**: Перевірка гри на помилки, баланс геймплею, адаптація до різних пристроїв і платформ;

8) **Оптимізація і Полірування**: Покращення продуктивності гри, усунення помилок, додавання додаткових функцій;

9) **Випуск і Підтримка**: Реліз гри на обраній платформі і подальша робота над виправленням помилок, відповідь на запити гравців, випуск оновлень тощо.

Ці функціональні особливості роблять процес розробки ігрового застосунку в жанрі RPG відмінним від інших жанрів і вимагають специфічного підходу до їх реалізації.

1.3 Аналіз існуючих аналогів в жанрі RPG

Dota 2 - це комп'ютерна гра в жанрі "многокористувацький онлайн боївка або MOBA" (Massive Online Battle Arena). Гра була розроблена компанією Valve Corporation і є продовженням популярної гри "Defense of the Ancients" (Dota), яка спочатку була модифікацією для гри Warcraft III [3].

У грі Dota 2 гравці формують команди з п'яти осіб і вибирають персонажів, відомих як "герої", кожен з яких має унікальні навички і роль в грі. Мета гри - знищити "Анкс", базу противника, розташовану на протилежних кінцях карти.

Гравці отримують золото та досвід, розвиваючи свого героя, вибираючи стратегічні предмети та взаємодіючи з іншими гравцями. Dota 2 відома своєю складною геймплейною механікою та активними турнірами, де команди з усього світу змагаються за великі призові фонди.

Розробник: Valve Corporation

Архітектура: Client-server

Мова реалізації: C++та Lua

Рисунок 1.5 – Ігровий процес Dota 2

Перелік функцій/характеристик:

1) Герої та Навички: Велика кількість героїв з унікальними навичками та ролями в грі;

2) Мапа: Ігровий світ гри, що складається з трьох ланей та джунглів, з точками для атак та оборони;

3) Геймплей: Знищення Анкса противника, збільшення рівня героя, отримання золота та придбання предметів;

4) Турніри та Рейтинг: Масштабні глобальні турніри, система рейтингу гравців та матчмейкінг;

5) Командний Гравець проти Комп'ютера: Можливість грати проти штучного інтелекту або з друзями у режимі команди;

6) Події та Оновлення: Регулярні оновлення гри, нові герої, ігрові режими та косметичні зміни;

7) Косметичні Предмети: Великий вибір косметичних предметів для персоналізації героїв та предметів;

8) Експорт та Трансляції: Підтримка трансляцій та можливість експорту гри для гравців та глядачів;

9) Мережевий Режим: Гра в режимі онлайн проти інших гравців по всьому світу.

Переваги:

1) Глибокий Геймплей: Складна стратегія, великий вибір героїв та навичок роблять гру цікавою і вимагаючою;

2) Безкоштовна Гра: Dota 2 є безкоштовною для гравців, що дозволяє всім спробувати гру без витрат;

3) Активна Спільнота та Турніри: Жива глобальна спільнота гравців та численні турніри з великими призовими фондами;

4) Постійні Оновлення: Регулярні оновлення вносять нових героїв, ігрові режими та косметичні зміни, що тримають гру свіжою;

5) Компетитивність: Dota 2 - це есенція компетитивних ігор, з великою кількістю гравців та професійних команд;

6) Красива Графіка: Гра має вражаючу графіку та вражаючий дизайн персонажів та карт;

7) Широкий Вибір Героїв та Стратегій: Різноманіття героїв та можливість вибору різних стратегій забезпечують глибину геймплею;

8) Підтримка eSport: Dota 2 є однією з найпопулярніших героїчних многокористувацьких онлайн боїв у світі eSport, з численними турнірами та подіями.

Недоліки:

1) Висока Вхідна Планка: Гра вимагає великого часового та інтелектуального зусилля для того, щоб зрозуміти стратегії, навички героїв та особливості геймплею;

2) Токсична Спільнота: В ігровому середовищі можна зустріти негативні висловлювання, надмірну конкуренцію та нешанобливу поведінку гравців;

3) Велика Кількість Героїв: Для новачків велика кількість героїв та їхніх навичок може бути плутаною та складною;

4) Залежність від Команди: Гра має великий акцент на командній грі, що може призвести до негативного досвіду для гравців, які не можуть забезпечити ефективну комунікацію зі своєю командою;

5) Довгі Партії: Одна гра може тривати від 30 хвилин до годин або більше, що може бути не зручним для гравців, які не мають багато часу для ігор;

6) Висока Системні Вимоги: Для гри можуть знадобитися потужні обчислювальні ресурси, що може бути обмежливим для деяких користувачів з менш потужними комп'ютерами;

7) Баланс Героїв: На час виходу нових героїв або змін можуть виникати проблеми з балансом, що впливає на геймплей.

Джерело інформації: https://www.dota2.com/home?l=ukrainian

League of Legends (LoL) - це відома масова многокористувацька онлайн-гра в жанрі MOBA (Massive Online Battle Arena), розроблена компанією Riot Games. Гра вийшла в 2009 році і стала однією з найпопулярніших у світі MOBA-ігор [4].

У League of Legends гравці утворюють команди з п'яти осіб і керують чемпіонами (героями), кожен з яких має свої унікальні навички та роль в грі. Мета полягає в тому, щоб знищити базу противника, використовуючи стратегічні елементи, командну гру та індивідуальні навички гравців.

League of Legends відрізняється великою кількістю чемпіонів, регулярними оновленнями, турнірами і частими змінами балансу для підтримки живого та висококонкурентного ігрового середовища. Гра доступна безкоштовно для гравців, але також має систему мікротранзакцій для придбання косметичних предметів та іншого контенту.

Розробник: Riot Games

Архітектура: Client-server

Мова реалізації: C++та Lua

Рисунок 1.6 – Ігровий процес League of Legends

Перелік функцій/характеристик:

1) Чемпіони: Великий вибір унікальних чемпіонів з різними навичками і ролями в грі;

2) Карта: Гравці грають на карті "Summoner's Rift", яка має ланцюги, ліс та бази противників;

3) Мета гри: Знищити базу противника, використовуючи командну гру та стратегічні рішення;

4) Система Рівнів та Досвіду: Гравці отримують рівні, збільшуючи свої можливості під час гри;

5) Магазин та Предмети: Можливість купувати предмети для покращення характеристик чемпіона;

6) Турніри та Конкурентний Режим: Ліги, рангові системи та регулярні турніри для конкурентного гравців;

7) Спільнота та Соціальні Функції: Системи чату, друзів, кланів та інші соціальні можливості в грі;

8) Косметичні Предмети: Можливість придбання косметичних предметів для персоналізації чемпіонів та їх вигляду;

9) Події та Оновлення: Регулярні оновлення гри з новими чемпіонами, балансом, режимами та подіями;

10) Експорт та eSport: Активна е-спортивна сцена з численними турнірами та лігами.

Переваги:

1) Безкоштовна Гра: LoL є безкоштовною для гравців, що дозволяє більшій кількості людей спробувати гру без витрат;

2) Різноманіття Чемпіонів: Величезний вибір унікальних чемпіонів різних стилів гри, що надає гравцям багато варіантів;

3) Активна Спільнота: Велика та активна спільнота гравців, яка сприяє соціальній взаємодії та обміну досвідом;

4) Конкурентний Геймплей: Завдяки конкурентному режиму, гравці можуть випробувати свої сили у високорівневих матчах та турнірах;

5) Е-Спорт Та Турніри: Лиги та турніри LoL входять до числа найбільших та найбільш популярних в світі електронного спорту;

6) Регулярні Оновлення: Розробники регулярно випускають оновлення, які додають нових чемпіонів, режими та виправляють баланс;

7) Придбання Косметичних Предметів: Система мікротранзакцій дозволяє гравцям придбавати косметичні предмети та персоналізувати свої чемпіонів.

Недоліки:

1) Токсична Спільнота: Явище негативної поведінки гравців, таке як грубість та образи, може впливати на гармонію гри та враження від гри;

2) Схильність до Швидкого Розвитку: Для новачків важко вловити всі аспекти гри через велику кількість чемпіонів, навичок та стратегій;

3) Залежність від Команди: Для успіху часто потрібна гарна комунікація та співпраця з іншими гравцями, що може бути викликом для одиночних гравців;

4) Довгі Партії: Час тривалості одного матчу може бути великим, що не завжди зручно для гравців, які мають обмежений час для гри;

5) Високі Системні Вимоги: Деякі обчислювальні ресурси можуть бути потрібні для комфортної гри на високих налаштуваннях, що може бути обмеженням для деяких гравців;

6) Баланс Гри: Іноді зміни в балансі чемпіонів можуть викликати невдоволення у гравців, особливо серед тих, хто пристосував свою гру під конкретних чемпіонів.

Джерело інформації: https://www.leagueoflegends.com/en-gb/

Heroes of the Storm (HotS) - це масова многокористувацька онлайн-гра в жанрі MOBA (Massive Online Battle Arena), розроблена компанією Blizzard

Entertainment. Гра була випущена у 2015 році і представляє собою унікальний підхід до жанру MOBA [5].

Основна відмінність HotS від інших MOBA полягає в тому, що вона об'єднує персонажі та локації з різних ігор Blizzard, таких як Warcraft, StarCraft, Diablo та Overwatch. Гравці формують команди з п'яти осіб і вибирають героїв (чемпіонів), кожен з яких має унікальні навички та взаємодіє з ігровим світом, щоб здобути перевагу над противниками.

До основних рис HotS варто відзначити красиву графіку, коротший тривалість матчів порівняно з іншими MOBA, і акцент на командній грі та стратегії. Гра також має ряд режимів гри, включаючи звичайні матчі, ранговий режим, та спеціальні події.

Розробник: Blizzard Entertainment

Архітектура: Client-server

Мова реалізації: C++

Рисунок 1.7 – Ігровий процес Heroes of the Storm

Перелік функцій/характеристик:

1) Унікальні Герої: Вибір героїв з різних світів Blizzard, кожен з унікальними навичками та стилями гри;

2) Карти та Локації: Різні ігрові карти з унікальними завданнями та механікою гри, що впливає на стратегію команд;

3) Командна Гра: Гравці формують команди по п'ять осіб і співпрацюють для досягнення спільних цілей;

4) Короткі Матчі: Матчі в HotS зазвичай тривають коротший час порівняно з іншими MOBA, що підходить для більш казуальних гравців;

5) Завдання та Об'єкти: У кожному матчі гравці зустрічаються з унікальними завданнями та об'єктами, які впливають на гру;

6) Моди гри: Включаючи режими Quick Match, Ranked, Brawl та інші;

7) Косметичні Предмети: Можливість придбати косметичні предмети для персоналізації героїв;

8) Е-Спорт та Турніри: Підтримка електронного спорту з регулярними турнірами та чемпіонатами;

9) Спільнота та Соціальні Функції: Чат, система друзів, кланів, що підтримують взаємодію гравців.

Переваги:

1) Унікальний Геройський Склад: Гра використовує персонажі та локації з ігор Blizzard, що надає унікальний та цікавий характер героям і світам гри;

2) Короткі Матчі: Матчі в HotS можуть бути коротшею, ніж в інших MOBA, що робить гру доступною для тих, хто має обмежений час для гри;

3) Командний Підхід: Гра акцентує на командній грі, що робить HotS менше залежним від індивідуальної вправності та більше від командного спілкування та стратегії;

4) Різноманітні Карти: Різні ігрові карти з унікальними завданнями сприяють великому різноманіттю стратегій та підходів до гри;

5) Завдання та Об'єкти: Цільові завдання та об'єкти впливають на гру, додаючи стратегічні аспекти та змінюючи динаміку матчу;

6) Гнучкість в Виборі Героїв: Гравці можуть вибирати героїв, що відповідають їхньому стилю гри, а не тільки роль чи стратегію;

7) Графіка та Дизайн: Як із численних ігор Blizzard, HotS має вражаючий дизайн персонажів та деталізовану графіку;

8) Багато Видів Розваг: З різними режимами гри, такими як Quick Match, Ranked, Brawl, гравці можуть знайти формат, який їм подобається.

Недоліки:

1) Менше Глибини Системи Індивідуального Росту: У порівнянні з іншими MOBA, HotS може виглядати менш зрілим з точки зору індивідуального росту гравця, так як відсутні рівні чемпіонів та індивідуальна економіка;

2) Закриття Розробників на Початку 2021 Року: У січні 2021 року компанія Blizzard оголосила про закриття основної частини команди, яка працювала над HotS, і змінила фокус на інші проєкти, що може вплинути на підтримку та оновлення гри;

3) Менша Гравецька База: У порівнянні з іншими популярними MOBA, гравців HotS може бути менше, що може вплинути на час очікування у черзі та рівень конкурентної гри;

4) Неактивна Розвиток: Після оголошення про зміни в розробці у січні 2021 року, HotS більше не отримує регулярні оновлення та контент, що може вплинути на інтерес гравців та довготривалість гри;

5) Недостаток Кроссплатформенності: Гра не підтримує кроссплатформенність між різними платформами, що може бути недоліком для гравців, які хочуть грати разом із друзями на різних пристроях.

Джерело інформації: https://heroesofthestorm.blizzard.com/en-us/

1.4 Специфікації вимог до ігрового застосунку

Призначення системи: створення можливості для гравців брати участь у захоплюючих боях і відчувати існування в унікальному віртуальному світі.

Сфера застосування: сфера розваг, розвиток ігрової індустрії.

Характеристики користувачів: користувачі віком від 14 років, повинні мати ПК чи ноутбук.

Загальна структура і склад системи:

Функції системи:

1) Механіка керування персонажем, включаючи рух, атаки та інші дії;

2) Бойова система з різноманітними механіками, включаючи магію, бойові прийоми та спеціальні ефекти;

3) Система прогресу та розвитку персонажа, яка дозволяє гравцю підвищувати рівень свого персонажа;

4) Механіка збору досвіду, що дозволяє персонажу розвиватися під час гри;

5) Система розвитку магії, яка дозволяє гравцеві вдосконалювати свої заклинання та магічні здібності;

6) Механіка поведінки для противників, що визначає їхні дії та стратегії в бою;

7) Система звуків та музики, яка створює атмосферу гри та підсилює емоційний досвід гравця;

8) Механіка розподілу очок навичок та характеристик, яка дозволяє гравцю впливати на розвиток свого персонажа.

9) Система здоров'я, яка автоматично відновлюється під час гри або за допомогою певних засобів;

10) Механіка збереження гри, яка дозволяє гравцеві зберігати свій прогрес та продовжувати гру з попереднього місця зупинки.

Вимоги до технічного забезпечення:

- операційна система: macOS Sonoma або вище;
- процесор: Intel Core i7 2,6 GHz 6-ядерний;
- відеокарта: AMD Radeon Pro 5300M 4GB;
- оперативна пам'ять: не менше 16 ГБ;
- місце на жорсткому диску: не менше 100 ГБ.

Архітектура програмної системи: однокористувацька.

Системне програмне забезпечення:

Для розробки ігрового застосунку на Unreal Engine 5 необхідно мати комп'ютер з операційною системою macOS або вище. Крім того, вам потрібно буде завантажити та встановити оновлення для самого Unreal Engine 5, щоб мати доступ до всіх його функцій та можливостей.

Мова і технологія розробки ПЗ: C++ або система Blueprints.

Інтерфейс користувача:

Потрібно розробити інтерфейс, який буде легким у використанні та зрозумілим для користувачів. Його головна мета - забезпечити зручне керування персонажем, взаємодію з ігровим світом і доступ до налаштувань гри. Це означає створення чіткого та лаконічного дизайну, який дозволяє гравцям швидко зорієнтуватися та ефективно взаємодіяти з інтерфейсом, не відволікаючись від головного геймплею.

Висновки до розділу 1

Жанр рольових ігор (RPG) є одним з найпопулярніших серед геймерів у всьому світі. Він визначається глибиною наративу та можливістю гравця впливати на розвиток подій у віртуальному світі. Основні характеристики цього жанру включають систему розвитку персонажів, наявність сюжету, великий відкритий світ або лінійний сценарій, бойову систему та взаємодію з НПС іншими гравцями.

Під час аналізу особливостей ігрового застосунка в жанрі RPG було виявлено, що вони включають в себе глибоку наративну структуру, можливість кастомізації персонажа, великий світ для дослідження, систему бойових механік, інтерактивність та відчуття присутності у вигаданому світі.

Аналіз існуючих аналогів в жанрі RPG показав, що на ринку існує багато ігор з різними особливостями та підходами. Серії, такі як " Wizardry", "Final Fantasy", "Ultima III", "Pedit5", "Divinity II" і багато інших, відзначаються своїми унікальними характеристиками та приваблюють різний шар гравців.

Специфікації вимог до ігрового застосунку включають в себе не лише технічні аспекти, але й вимоги стосовно глибини наративу, геймплейних можливостей та візуального виконання. Графічна якість, стабільність роботи, та оптимізація для різних платформ є невід'ємною частиною вимог. Крім цього, привабливий сюжет, можливість кастомізації персонажів, і глибина взаємодії з віртуальним світом також вважаються важливими аспектами для задоволення гравців. Підтримка мультиплеєра та регулярні оновлення з метою покращення геймплею додають до ігрового досвіду інтерактивності та тривалості. Також важливою є можливість ефективного управління грою та навігація по ігровому світу. Загалом, успішний RPG ігровий застосунок повинен поєднувати ці аспекти, щоб забезпечити задоволення та захоплення для широкого кола гравців.

Отже, RPG ігри продовжують залишатися популярним жанром у галузі відеоігор, пропонуючи гравцям широкий спектр варіантів для іммерсивного досвіду у віртуальних світах.

2 МОДЕЛЮВАННЯ АРХІТЕКТУРИ ІГРОВОГО ЗАСТОСУНКУ

2.1 Загальні принципи моделювання ПЗ

UML (Unified Modeling Language) - це стандартна мова моделювання, яка використовується в об'єктно-орієнтованому програмуванні. Вона є ключовою складовою уніфікованого процесу розробки програмного забезпечення. UML забезпечує графічні засоби для створення абстрактних моделей системи, відомих як UML-моделі, та використовується для визначення, візуалізації, проєктування та документування програмних систем. Хоча UML не є мовою програмування, вона дозволяє генерувати код з UML-моделей для виконання [6].

Перша версія UML (1.0) була випущена 13 січня 1997 року консорціумом UML Partners за ініціативою Object Management Group (OMG), організації, що встановлює стандарти в галузі об'єктних технологій та баз даних. У вересні 1997 року версія 1.1 UML була представлена на голосуванні в OMG. UML одразу ж здобув підтримку від провідних компаній індустрії IT, таких як Microsoft, IBM, Hewlett-Packard, Oracle, DEC, Sybase, Logic Works та інші.

Мова UML може бути застосована на всіх етапах життєвого циклу аналізу бізнес-систем та розробки прикладних програм. Завдяки різноманітним видам діаграм, які підтримуються UML, та широкому набору можливостей для відображення різних аспектів системи, UML стає універсальним засобом для опису як програмних, так і ділових систем.

Використання діаграм UML дає змогу легко перевести модель системи у програмний код. Однак основною причиною використання мови UML є забезпечення ефективного спілкування між розробниками.

UML була спеціально створена для оптимізації процесу розробки програмних систем, що призводить до підвищення ефективності розробки та поліпшення якості кінцевого продукту. Інструменти автоматичної генерації

коду забезпечують перетворення моделей UML у вихідний код, що додатково прискорює процес розробки.

UML успішно використовується у багатьох програмних проєктах і забезпечує спрощення розробки документації та покращення супроводжуваності проєкту.

Уніфікована мова моделювання (UML) має кілька версій, кожна з яких включає в себе різні вдосконалення та розширення. Ось деякі з найбільш відомих версій UML та їх зміни [7]:

1) **UML 1.x**: Перша версія UML вийшла у 1997 році. Вона встановила базові концепції та елементи UML, такі як класи, об'єкти, відносини, діаграми варіантів використання та інші. У цьому випуску були представлені діаграми класів, прецедентів, послідовностей, станів, активностей, компонентів та розгортання;

2) **UML 2.0**: Випущена у 2005 році, ця версія UML внесла значні зміни та розширила можливості мови. У випуску 2.0 було додано нові діаграми, такі як діаграма проміжного рівня, діаграма прецедентів, діаграма композитних структур, діаграма об'єктів та інші. Крім того, введено поняття профілю UML, яке дозволяє розширювати мову за допомогою власних нотацій та елементів;

3) **UML 2.x**: Після випуску UML 2.0 були представлені декілька версій, таких як UML 2.1, 2.2, 2.3 та 2.4, які внесли дрібні зміни та виправлення помилок у стандарт. Однак, основні концепції та можливості залишилися незмінними;

4) **UML 2.5**: Ця версія була випущена у 2015 році та включала значні зміни у структурі та нотації UML. UML 2.5 внесла уточнення до ряду діаграм, включила нові можливості, такі як покращена підтримка робочих профілів та удосконалення визначень стандарту.

2.2 Діаграма варіантів використання

Діаграма варіантів використання (або діаграма прецедентів) - це інструмент моделювання, який використовується для візуалізації функціональності системи з точки зору її користувачів. Вона описує всі можливі способи взаємодії користувачів або зовнішніх сутностей з системою та ілюструє, як ці взаємодії відбуваються через прецеденти або варіанти використання [9].

У діаграмі варіантів використання виокремлюються наступні основні елементи:

1) **Актори**: Це сутності або ролі, які взаємодіють з системою. Актори можуть бути реальними користувачами, зовнішніми системами або іншими програмними модулями;

2) **Прецеденти (варіанти використання)**: Це конкретні дії або функції, які виконуються системою. Вони описують, що робить система від імені користувачів або зовнішніх сутностей;

3) **Зв'язки між акторами та прецедентами**: Вони показують, як актори взаємодіють з прецедентами, тобто які дії можуть виконувати актори та які прецеденти вони викликають;

4) **Опис прецедентів (варіантів використання)**: Кожен прецедент може бути детально описаний з точки зору його специфікації, включаючи вхідні дані, дії, які виконуються системою, і вихідні результати. Це допомагає уточнити очікувану поведінку системи в різних сценаріях;

5) **Альтернативні та виключні сценарії**: Поміж кожним прецедентом можуть існувати альтернативні шляхи або виключні сценарії, які відбуваються, коли певні умови виконуються або не виконуються. Це включає в себе помилки, виняткові ситуації та інші випадки, які можуть виникати під час взаємодії з системою;

6) **Діаграма послідовності**: Додатковою інформацією може бути діаграма послідовності, яка показує послідовність повідомлень між акторами

та об'єктами системи в конкретному сценарії використання. Це допомагає краще зрозуміти взаємодію між сутностями в рамках кожного прецеденту;

7) **Умови виконання**: Для кожного прецеденту можуть бути визначені умови виконання, які вказують, коли та за яких обставин цей прецедент може бути викликаний. Це допомагає обмежити доступ до функціональності системи залежно від контексту.

Діаграма варіантів використання допомагає розуміти, як система повинна вести себе в різних ситуаціях та як вона повинна відповідати на дії користувачів або зовнішніх систем. Вона є потужним інструментом для аналізу вимог до системи, проєктування її функціональності та визначення тестових випадків.

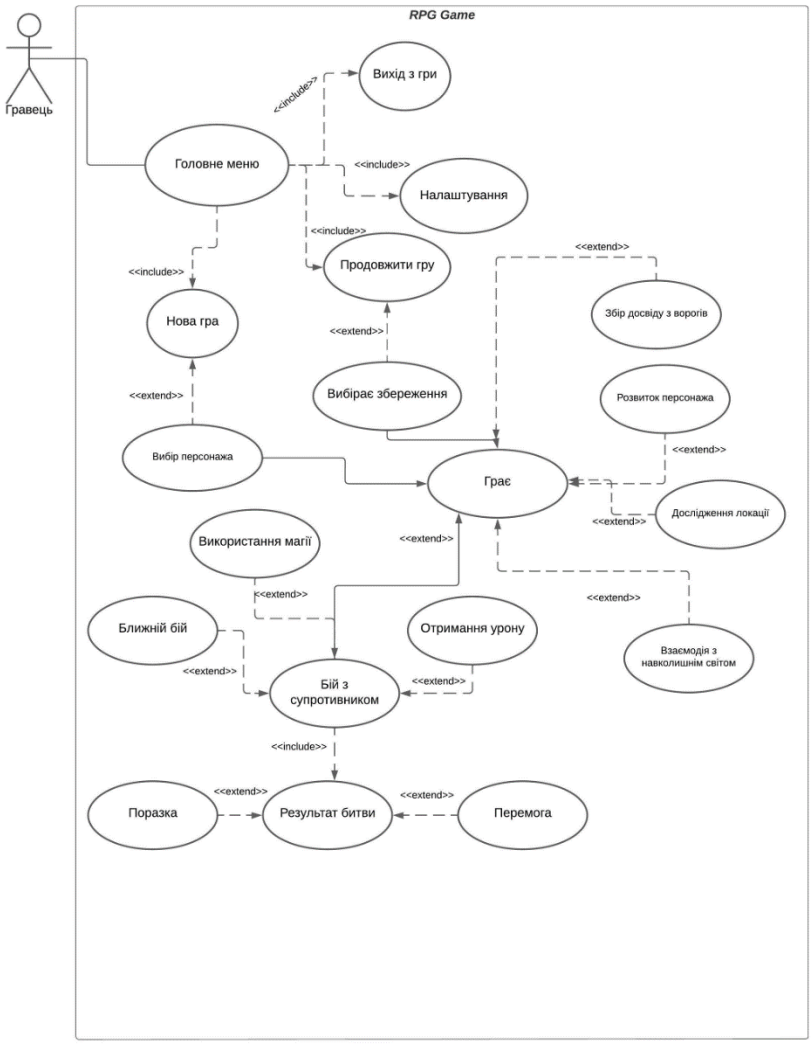


Рисунок 2.2 – Діаграма варіантів використання

Короткий usercase:

Користувач відкриває ігровий додаток на своєму пристрої і потрапляє на головне меню гри. З цього меню він може обрати різні опції, такі як створення нової гри або продовження збереженої. Обравши нову гру, він може вибрати персонажа та розпочати саму гру. Гравець з’являється на початковій локації, готовий взаємодіяти з ігровим світом.

Поверхневий usercase:

Головний сценарій (успішний)

Гравець запускає ігровий додаток на своєму пристрої і обирає опцію "Продовжити гру" у головному меню. Після цього він обирає свій збережений профіль, де зберігаються всі його досягнення та прогрес у грі. Під час гри гравець зустрічається з різноманітними завданнями, включаючи боротьбу з ворогами та вивчення світу гри. Він активно намагається збирати досвід, щоб покращувати вміння свого персонажа та розвивати його характеристики. Гравець продовжує досліджувати світ гри, прагнучи досягти максимального рівня успіху та прогресу.

Альтернативні сценарії:

1) Гравець вибирає опцію "Нова гра" і обирає рівень складності перед тим, як розпочати гру з початку. Також він може зберегти свій прогрес у новий слот профілю, щоб мати можливість продовжити гру з поточного місця. Після цього у нього буде два профіля з різним прогресом гри;

2) Гравець відкриває налаштування, щоб змінити графічні параметри, звукові ефекти, чутливість контролера та інші налаштування за його власними вподобаннями;

3) Після завершення гри гравець обирає опцію "Вийти з гри", щоб зберегти свій прогрес і вийти з ігри;

4) Гравець переглядає опцію "Credits", щоб переглянути список розробників та інших працівників, які брали участь у створенні гри;

5) Після збереження гравець може обрати опцію "Продовжити гру", щоб знову насолоджуватися ігровим світом та продовжити гру з місця, де він останній раз завершив;

6) У разі виникнення проблем гравець може спробувати перезавантажити гру або звернутися за допомогою до технічної підтримки.

Повний usercase

Use Case Name: Гравець розпочинає нову гру в жанрі RPG

Scope: System

Level: RPG_Map

Primary Actor: Гравець

Stakeholders and interests:

- Гравець бажає грати в нову гру в жанрі RPG, де він може зануритися в захоплюючий світ, взаємодіяти з цікавими персонажами та виконувати різноманітні завдання;
- Розробники бажають, щоб гравець зміг успішно почати гру і насолоджуватись її геймплеєм, надаючи йому зручний інтерфейс, доступний навчальний посібник та належну підтримку під час гри. Вони також можуть враховувати побажання гравця щодо геймплею та розширювати гру, щоб задовольнити його потреби в розвагах.

Preconditions:

– Користувач має доступ до комп'ютера, що дає йому можливість використовувати різноманітні програми та додатки, включаючи ігрові;

– На пристрої користувача встановлено застосунок, побудований на основі рушія Unreal Engine, що забезпечує високу якість графіки та реалістичний геймплей;

– Гравець має системні можливості, рекомендовані для гри, що дозволяють йому насолоджуватись ігровим досвідом без перешкод та затримок.

Success Guarantee:

– Гравець успішно запускає нову гру в жанрі Action RPG на своєму пристрої, використовуючи застосунок, побудований на основі рушія Unreal Engine;

– Гравець, будучи знайомим з основними елементами гри, з легкістю розпізнає інтерфейс та механіку гри, що дозволяє йому продовжити геймплей без перешкод. Він може насолоджуватись ігровим досвідом та взаємодіяти з ігровим світом без додаткових викликів.

Main Success Scenario:

– Гравець запускає застосунок на своєму пристрої;

– В головному меню гри гравець обирає опцію "Нова гра";

– Система пропонує гравцеві вибрати персонажа для гри;

– Гравець обирає бажаного персонажа з доступних варіантів;

– Після вибору персонажа система генерує світ для гравця;

– Гравець потрапляє в ігровий світ і отримує контроль над головним героєм;

– Починаючи зі стартової локації, гравець оглядається та знайомиться з інтерфейсом гри;

– Гравець розпочинає дослідження ігрового світу та боротьбу з ворогами;

– Під час боїв з ворогами гравець отримує досвід;

– Гравець має можливість зберегти гру та продовжити геймплей пізніше, якщо це необхідно.

Extensions:

1а. Гравець відмовляється вибирати персонажа і повертається до головного меню гри.

2а. Гравець не може обрати бажаного персонажа:

– Система пропонує гравцеві на вибір 3 персонажів з різними навичками;

– Гравець обирає одного з персонажів;

– Гравець повертається до кроку 5 головного сценарію.

За. Гравець не успішно розпочинає гру через технічні проблеми:

– Система виводить повідомлення про проблему та пропонує спробувати ще раз;

– Гравець повторює спробу ще раз;

– Якщо проблема не вирішена, гравець перезавантажує застосунок чи звертається до технічної підтримки.

Special Requirements:

– Гра повинна містити різних персонажів, які відповідають різним вподобанням гравців, щоб кожен гравець міг знайти персонажа, який відповідає його стилю гри та індивідуальним уподобанням;

– Гра повинна мати систему збереження прогресу гравця, щоб гравець міг зберігати свій прогрес і продовжувати гру з того місця, де він останній раз закінчив, без втрати досягнутого прогресу;

– Гра повинна мати інтуїтивний інтерфейс, який допоможе гравцеві ознайомитись з основними елементами гри легко і швидко. Це допоможе новачкам швидко впоратися з грою і насолоджуватися її геймплеєм, а досвідчені гравці також оцінять зручність та простоту управління.

Technology and Data Variations List:

Для створення гри рекомендується обрати Unreal Engine 5 для розробки.

Frequency of Occurrence:

Частота виконання кожного випадку використання (use case) при взаємодії з системою залежить від кількості гравців, які користуються нею.

Miscellaneous:

Можна додати мультиплеєрний режим для гри з друзями та систему квестів.

2.3 Діаграми взаємодії

Діаграма взаємодії в контексті UML (Unified Modeling Language) є одним із типів структурних діаграм, які використовуються для моделювання взаємодії між об'єктами або компонентами системи. Ця діаграма фокусується на відображенні взаємодії між об'єктами або ролей у системі в конкретних сценаріях або випадках використання [8].

Діаграма взаємодії використовується для:

1) **Візуалізації взаємодії**: Діаграми взаємодії дозволяють представити взаємодію між об'єктами або ролями у системі у формі послідовності повідомлень або дій. Це дозволяє розробникам краще зрозуміти, як взаємодіють різні частини системи;

2) **Уточнення вимог**: Діаграми взаємодії допомагають уточнити вимоги до системи, описуючи конкретні сценарії взаємодії між об'єктами або компонентами. Це дозволяє виявити потреби системи та встановити, як різні частини системи мають взаємодіяти між собою;

3) **Тестування та валідація**: Діаграми взаємодії можуть бути використані для розробки тестових сценаріїв, які перевіряють правильність взаємодії між компонентами системи. Це допомагає підтвердити, що система працює вірно та відповідає вимогам;

4) **Документації**: Діаграми взаємодії можуть бути використані для створення документації проєкту, яка пояснює, як система працює та які інтеракції відбуваються між її частинами.

Узагальнюючи, діаграма взаємодії є корисним інструментом для аналізу, проєктування та документування системи, який допомагає зрозуміти та уточнити взаємодію між її компонентами.

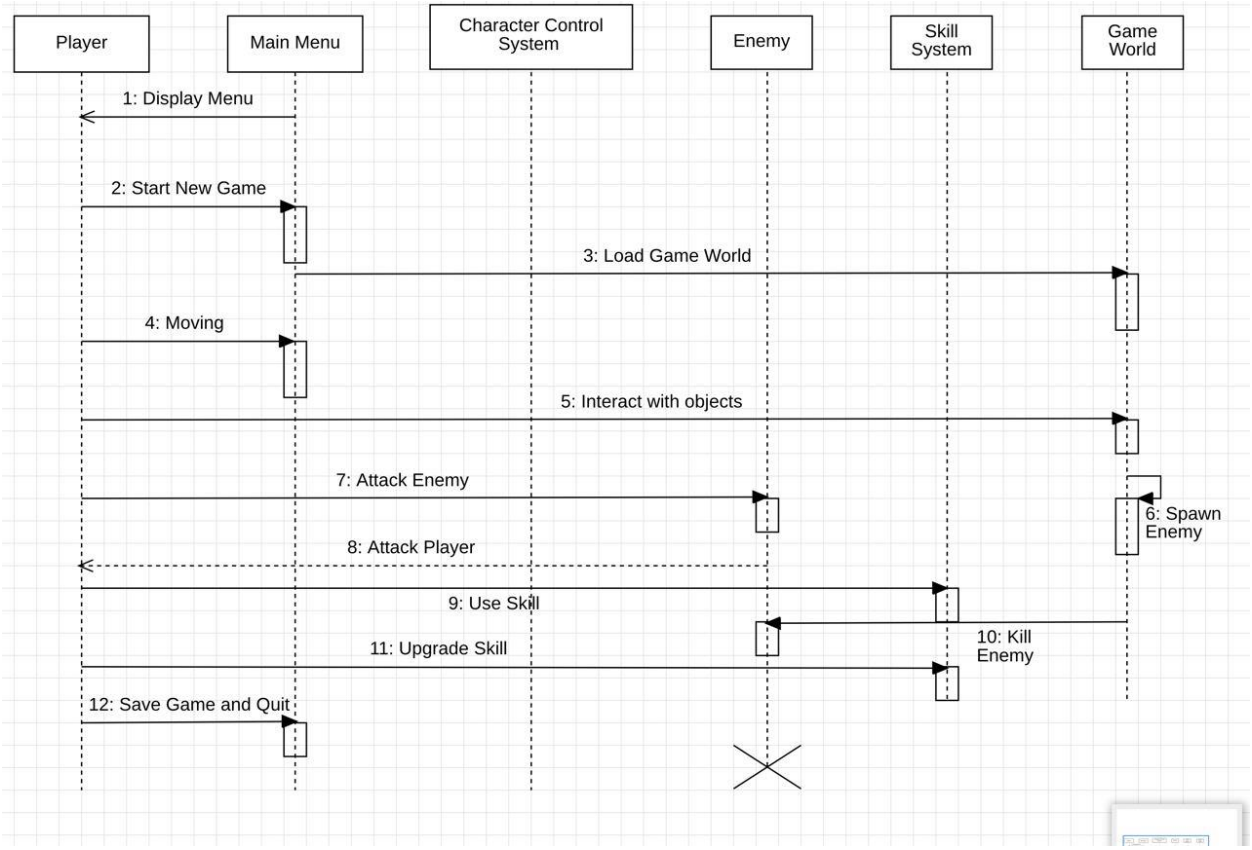


Рисунок 2.1 – Sequence діаграма «Gameplay»

Від «Main Menu» до «Player» - Reply Message: Display Menu, гравець запустив застосунок. Відкрилось головне меню.

Від «Player» до «Main Menu» - Message: Start New Game, гравець обирає почати нову гру.

Від «Main menu» до «Game World» - Message: Load Game World завантаження ігрового світу.

Від «Player» до «Hero Control System» - Message: Moving, керує головним героєм.

Від «Player» до «Game World» - Message:Interact with objects, взаємодія з світом.

Від «Game World» до «Game World» - Self Message:Spawn Enemy, розміщення ворогів.

Від «Player» до «Enemy» - Message:Attack Enemy, бій з супротивником.

Від «Enemy» до «Player» - Reply Message:Attack Enemy, бій з гравцем.

Від «Player» до «Skill System» - Message:Use Skill, використати вміння.

Від «Game World» до «Enemy» - Message:Kill Enemy, прибирання Enemy зі світу після його смерті.

Від «Player» до «Skill System» - Message: Upgrade Skill, покращити вміння.

Від «Player» до «Main Menu» - Message: Save Game and Quit, гравець виходить з гри та зберігає поточний прогрес.

2.4 Діаграма діяльності

Діаграма діяльності - це вид діаграми в рамках моделювання систем, який використовується для візуалізації послідовності дій або операцій у системі або процесі. Вона зображує послідовність взаємодій між об'єктами або ролями в системі від початку до кінця конкретного процесу або сценарію [10].

Ця діаграма корисна для:

1) **Опису процесів:** Дозволяє візуалізувати послідовність кроків або дій, які потрібно виконати в рамках конкретного процесу або сценарію.

2) **Аналізу процесів:** Дозволяє аналізувати та вдосконалювати поточні процеси, виявляти можливі проблеми та оптимізувати робочі процедури.

3) **Документації:** Служить засобом документування процесів, проєктування або опису функціональності системи.

4) **Взаємодії з користувачем:** Використовується для моделювання взаємодії користувача з системою, відображення послідовності дій, які користувач виконує для досягнення певної мети.

Загалом, діаграми діяльності допомагають у сприйнятті, аналізі та проєктуванні процесів або взаємодій у системах різного типу.

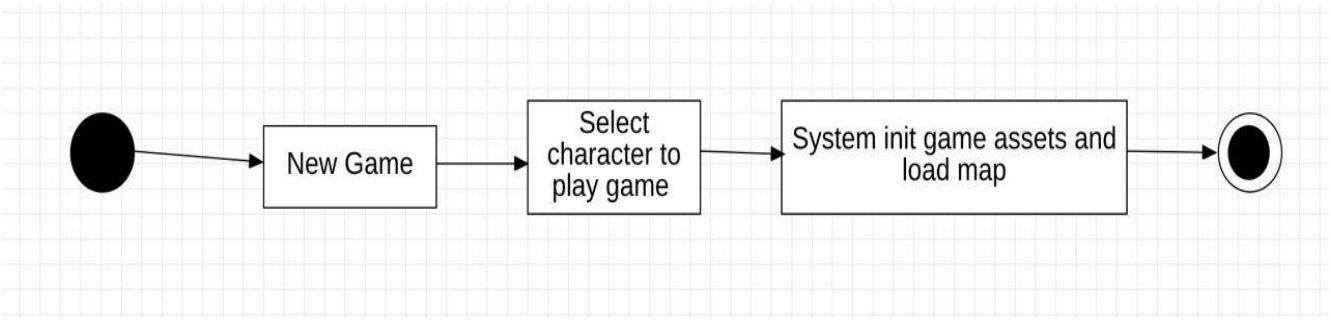


Рисунок 2.3 – Діаграма діяльності «Start a New Game»

Пояснення:

1) Користувач обирає опцію "Почати нову гру" у головному меню;

2) Система відображає користувачеві список доступних персонажів для вибору;

3) Користувач обирає одного з персонажів зі списку;

4) Система ініціалізує ігрові ресурси, такі як обраного персонажа гравця та ворогів;

5) Система завантажує ігрову карту для нової сесії гри.

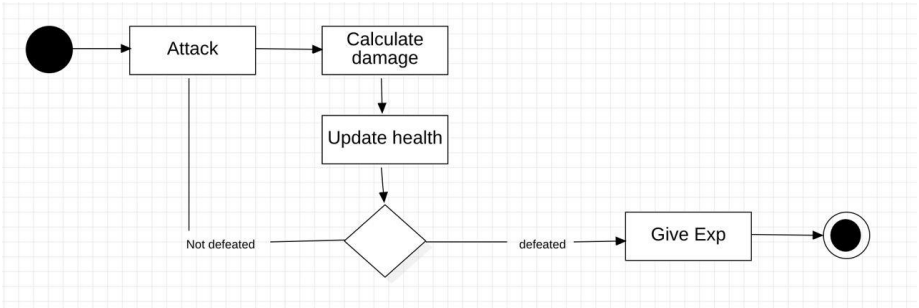


Рисунок 2.4 – Діаграма діяльності «Attack an Enemy»

Пояснення:

1) Користувач активує опцію "Атакувати" під час бою.

2) Система проводить розрахунок завданої шкоди, що персонаж гравця наносить ворогу, враховуючи різні фактори.

3) Здоров'я ворога оновлюється системою в залежності від отриманої шкоди.

4) Система перевіряє, чи був ворог знищений.

5) У разі перемоги, система надає гравцю очки досвіду.

2.5 Діаграма станів

Діаграма станів - це вид діаграми, який використовується для візуалізації станів об'єкта або системи та переходів між цими станами в залежності від зовнішніх подій або внутрішніх умов[11].

Цей вид діаграми корисний для:

1) Моделювання поведінки системи: Діаграми станів дозволяють моделювати різні стани, у яких може перебувати система, та умови, за яких система переходить від одного стану до іншого.

2) Уточнення вимог: Вони допомагають уточнити вимоги до системи, визначаючи, як система повинна реагувати на певні події та зміни станів.

3) Аналіз та проєктування програмного забезпечення: Використовуються для аналізу та проєктування різноманітних систем, включаючи програмне забезпечення, системи управління та багато інших.

4) Розробки структури програмного забезпечення: Діаграми станів допомагають розробникам розуміти та визначати, як система повинна вести себе в різних ситуаціях та які події можуть призвести до зміни її стану.

Загалом, діаграми станів використовуються для моделювання та аналізу поведінки системи в різних сценаріях та умовах.

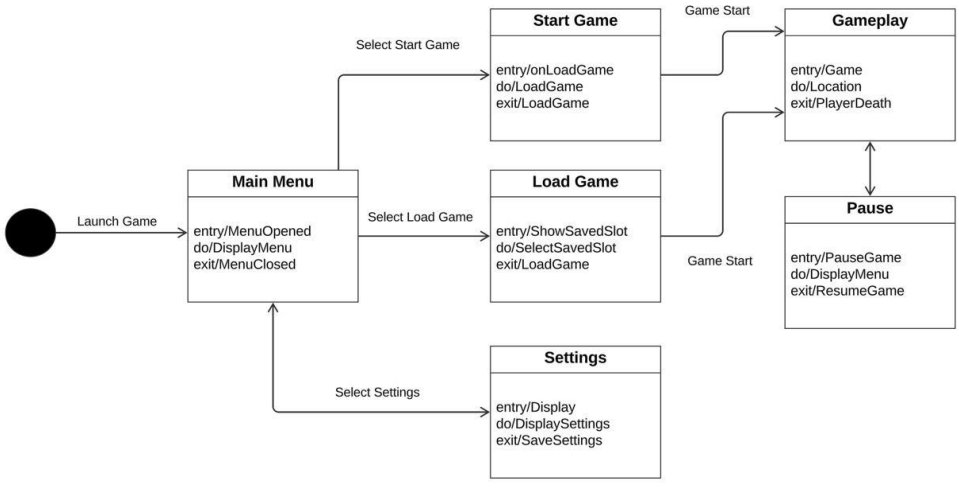


Рисунок 2.5 – Діаграма стану в цілому

У цій діаграмі (рис. 2.5) присутні різні стани, включаючи стан головного меню, в якому перебуває користувач після запуску гри. Головне меню, у свою чергу, може мати кілька станів, що змінюються в залежності від вибраних користувачем опцій.

Стани "Start/Load Game" ініціюють сам процес гри, де гравець може активувати такий стан, як "Pause".

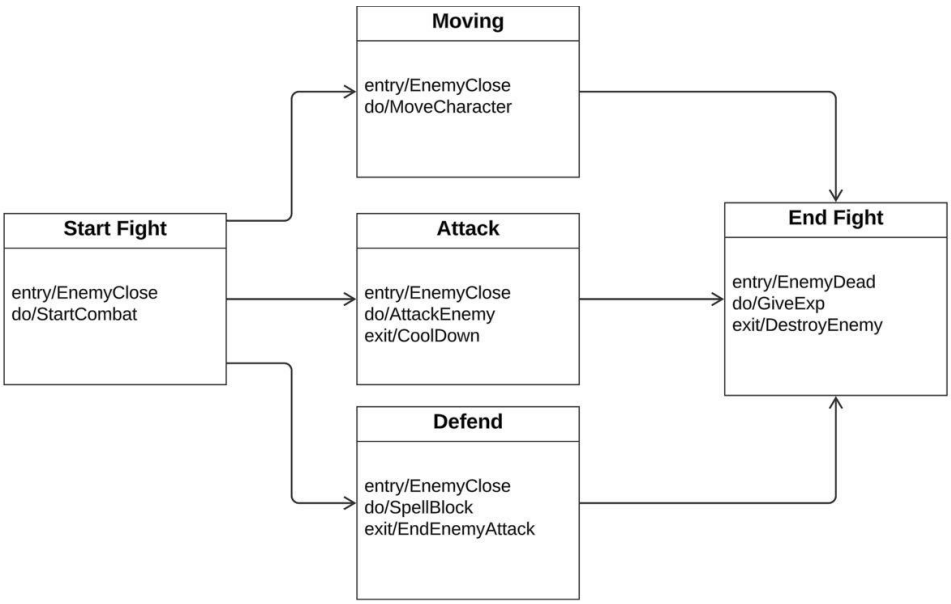


Рисунок 2.6 – Діаграма стану «Бій з супротивником»

У цій діаграмі (рис. 2.6) показані різні стани бою з ворогом, які спрацьовують тоді, коли ворог перебуває дуже близько до героя. Завершенням стану бою з ворогом є смерть ворога.

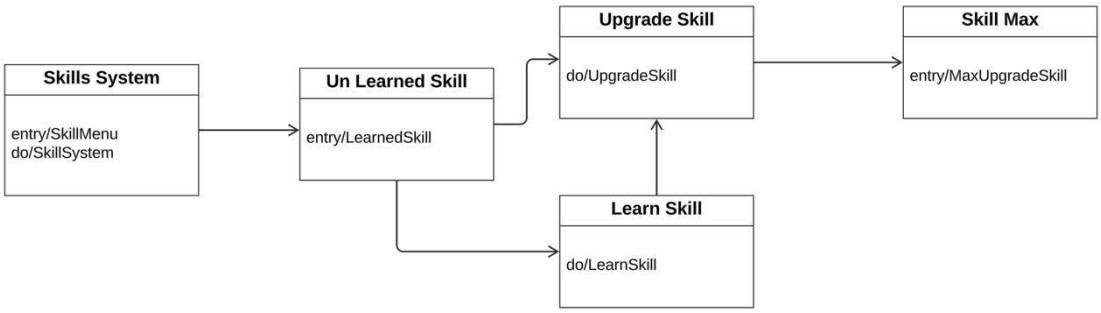


Рисунок 2.7 – Діаграма стану «Система навичок»

У цій діаграмі (рис. 2.7) показані різні стани системи навичок (Skill System):

1) Стан "Skills System": це початковий стан, в якому гравець може переглянути всі доступні навички та їх характеристики;

2) Стан "Unlearned Skill": це стан, в якому гравець може обрати навичку, яку він хоче вивчити, але поки що не має можливості це зробити;

3) Стан "Learn Skill": це стан, в якому гравець може вивчити нову навичку;

4) Стан "Upgrade Skill": це стан, в якому гравець може покращити свої наявні навички;

5) Стан "Skill Max": це стан, в якому гравець досягнув максимального рівня вивчення навички, отримавши додаткові бонуси та можливості в грі.

Висновки до розділу 2

У другому розділі розглянуто основні підходи та методи моделювання програмного забезпечення, які є фундаментом для подальшого розгляду конкретних типів діаграм UML. Цей розділ покликаний визначити основні принципи, на яких ґрунтується моделювання ПЗ, такі як модульність, інкапсуляція та абстракція, що сприяють поліпшенню структури та якості програмних продуктів.

Діаграма взаємодії, розглянута у другому розділі, є потужним інструментом для візуалізації та аналізу взаємодії між об'єктами або компонентами системи. Ця діаграма дозволяє моделювати взаємодію шляхом відображення послідовностей або кооперативних дій, що сприяє кращому розумінню та аналізу функціональних вимог системи.

Діаграма варіантів використання, яка розглядалася в третьому розділі, дозволяє детально описати різні сценарії використання системи, визначаючи потенційних користувачів та їх взаємодію з системою. Це допомагає виявити функціональні вимоги та спростити їх реалізацію.

Діаграма діяльності та діаграма станів, які були розглянуті у четвертому і п'ятому розділах відповідно, дозволяють моделювати послідовності дій та поведінку об'єктів або системи відповідно. Вони є важливими інструментами для визначення функціональності та станів системи.

Загалом, розділи про діаграми UML надають засоби для моделювання різних аспектів системи, що сприяє кращому розумінню, аналізу та розробці програмного забезпечення.

Також розглядаються сценарії використання у різних формах - коротка, поверхнева та повна.

3 ВИБІР ТЕХНОЛОГІЙ РОЗРОБКИ, ІМПОРТ ТА АНІМАЦІЯ ОБ'ЄКТІВ

3.1 Розгляд технологій програмної реалізації ігрового застосунку

Вибір правильного ігрового рушія є одним із найважливіших рішень, яке стоїть перед розробником на початкових етапах створення гри. Ігровий рушій визначає не лише технічні можливості гри, але й впливає на процес розробки, вимоги до апаратного забезпечення, а також на кінцевий результат у вигляді графіки, продуктивності та загального геймплею.

На ринку існує кілька популярних ігрових рушіїв, кожен з яких має свої сильні та слабкі сторони. Unreal Engine 5 [12], Unity [13], Godot [14], CryEngine [15] та GameMaker Studio [16] — всі вони широко використовуються в індустрії ігор, проте кожен з них краще підходить для певних типів проєктів та команд. Наприклад, Unreal Engine 5 славиться своєю фотореалістичною графікою та потужними інструментами для розробки ігор ААА-класу, в той час як Unity відомий своєю простотою використання та широким спектром застосування.

Таблиця 3.1 – Порівняльна таблиця Unreal Engine 5 та його аналогів

Ігровий рушій	Переваги	Недоліки
Unreal Engine 5	**- Фотореалістична графіка**: Завдяки технологіям Lumen (глобальне освітлення в реальному часі) та Nanite (віртуалізована геометрія мікрополігонів) забезпечується надзвичайно висока якість зображення. **- Потужні інструменти для розробки**: Включає в себе розширені інструменти для створення ігор ААА-класу, такі як Blueprints (візуальне програмування), потужний	**- Високі системні вимоги**: Потребує потужного апаратного забезпечення для ефективної роботи. **- Крута крива навчання**: Може бути складним для новачків через велику кількість функцій та складність інструментів. **- Великі розміри файлів проєктів**: Проєкти можуть займати багато місця на диску.

Продовження таблиці 3.1

	редактор матеріалів та Cascade (система частинок). - **Велика спільнота**: Величезна кількість ресурсів, форумів, уроків та підтримки від спільноти. - **Безкоштовний доступ**: Безкоштовний для використання з роялті лише при досягненні певного комерційного успіху.	
Unity	- **Простий у використанні**: Підходить для початківців, має інтуїтивний інтерфейс та зручні інструменти для розробки як 2D, так і 3D ігор. - **Широке застосування**: Підтримка різних жанрів ігор, включаючи VR/AR, мобільні та десктопні ігри. - **Великий магазин активів**: Unity Asset Store пропонує безліч готових до використання активів, скриптів та плагінів. - **Кросплатформенність**: Можливість експорту на більшість платформ, включаючи Windows, macOS, iOS, Android, консолі та інші.	- **Менш потужний для високоякісної графіки**: Графічні можливості не настільки розвинуті, як у Unreal Engine. - **Платні версії**: Деякі розширені функції доступні лише у платних версіях. - **Додаткові плагіни**: Може потребувати додаткових плагінів для реалізації певних функціональностей.
CryEngine	- **Високоякісна графіка**: Пропонує чудову якість графіки та реалістичну фізику.	- **Крута крива навчання**: Складний для новачків через велику кількість функцій.

Кінець таблиці 3.1

	- **Потужний інструментарій**: Добре підходить для створення фотореалістичних ігор. - **Безкоштовний доступ**: Можливість безкоштовного використання з опцією роялті.	- **Обмежені навчальні матеріали**: Менше ресурсів та уроків у порівнянні з іншими рушіями. - **Менша спільнота**: Менша кількість розробників та активностей у спільноті.
GameMaker Studio	- **Простий у використанні**: Ідеальний для розробки 2D-ігор з інтуїтивним візуальним редактором. - **Швидке створення прототипів**: Можливість швидкого створення і тестування ігрових ідей. - **Підтримка експорту на різні платформи**: Можливість експорту на більшість платформ, включаючи мобільні та десктопні.	- **Обмежена підтримка 3D**: Основний акцент на 2D-іграх, менш потужний для 3D-графіки. - **Платна версія**: Деякі розширені функції доступні лише у платній версії. - **Менша гнучкість**: Менш гнучкий у порівнянні з іншими рушіями для складних проєктів.

Niagara System — це потужна і гнучка система візуальних ефектів (VFX), інтегрована в Unreal Engine 4 і пізніші версії, включаючи Unreal Engine 5. Вона дозволяє розробникам створювати складні та високоякісні ефекти, такі як дим, вогонь, вода, вибухи, магічні заклинання та інші динамічні елементи, які роблять ігровий світ більш реалістичним та захоплюючим [17].

Ключові Особливості Niagara System:

1) **Візуальне Програмування**: Niagara System використовує візуальне програмування, що дозволяє створювати та налаштовувати ефекти без необхідності писати код. Це робить процес створення ефектів більш інтуїтивним та доступним для дизайнерів, які можуть працювати безпосередньо з інтерфейсом рушія.

2) **Модульна Архітектура**: Система побудована на модульній архітектурі, що дозволяє легко комбінувати різні модулі для створення

складних ефектів. Кожен модуль відповідає за певний аспект ефекту, наприклад, рух частинок, їх зовнішній вигляд, поведінку та взаємодію з іншими елементами.

3) **Підтримка GPU**: Niagara System підтримує обчислення на графічному процесорі (GPU), що дозволяє створювати більш складні та продуктивні ефекти. Це особливо важливо для сучасних ігор, де велика кількість динамічних ефектів може значно навантажувати систему.

4) **Гнучкість та Налаштовуваність**: Система дозволяє тонко налаштовувати кожен аспект ефектів, включаючи розмір, колір, форму, поведінку частинок та їх взаємодію з іншими об'єктами. Це забезпечує розробникам повний контроль над створенням візуальних ефектів.

Використання Niagara System:

Niagara System широко використовується в різних аспектах розробки ігор та інтерактивних додатків:

1) **Ігри**: В ігровій індустрії Niagara System використовується для створення реалістичних та захоплюючих візуальних ефектів, які додають глибині та атмосферності ігровим світам. Це включає візуалізацію погодних умов, вибухів, магічних заклинань, спецефектів у боях та багато іншого.

2) **Віртуальна реальність (VR)**: У проєктах VR Niagara System допомагає створювати реалістичні та інтуїтивно зрозумілі ефекти, що значно підвищує занурення гравців у віртуальні світи.

3) **Кіновиробництво**: Niagara System також знаходить застосування в кіновиробництві та анімації, де використовується для створення спецефектів у фільмах та анімаційних роликах.

4) **Симуляції та Візуалізації**: В інженерії, архітектурі та наукових симуляціях Niagara System використовується для візуалізації різних процесів, таких як потік рідин, вітрові симуляції та інші фізичні явища.

Завдяки своїй гнучкості та потужності, Niagara System стала невід'ємним інструментом для розробників, які прагнуть створювати високоякісні та реалістичні візуальні ефекти в своїх проєктах.

3.2 Моделювання ігрового світу

Створення рівня в Unreal Engine 5 (UE5) є важливим етапом розробки ігрових проєктів, що дозволяє втілити ідеї в реальність за допомогою потужних інструментів і можливостей рушія.

Після налаштування проєкту в Unreal Engine 5 можна переходити до створення нового рівня. Для цього відкрий верхнє меню, знайди пункт File і вибери його. У випадаючому меню знайди опцію New Level і натисни на неї. З'явиться вікно з кількома доступними шаблонами для створення нового рівня. Обираємо Open World, та натискаємо створити (рис 3.1).

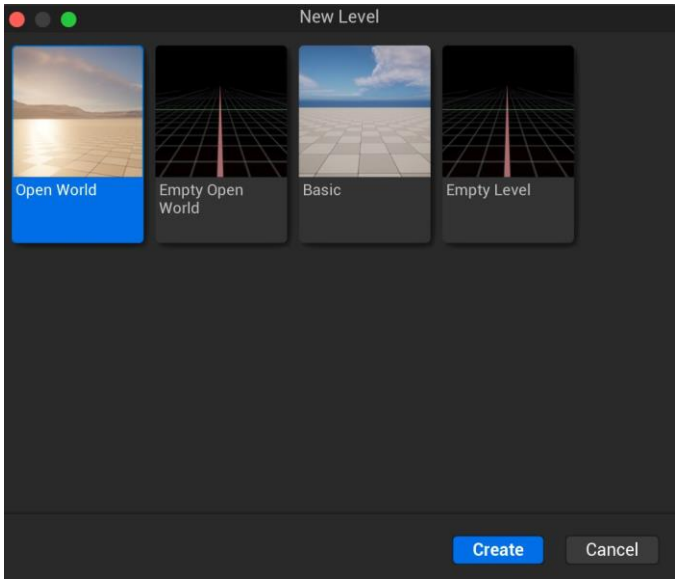


Рисунок 3.1 – Вікно створення нового рівня

Після створення нового рівня, переходимо до створення ландшафту за допомогою інструментів ландшафту. Для цього у верхньому меню обераємо вкладку Modes і знаходимо розділ Landscape. У панелі налаштувань встановлюємо параметри ландшафту, такі як розмір та роздільна здатність, і натискаємо Create. Далі використовуємо інструменти Smooth, Flatten, Noise задля формування ландшафта (рис 3.2).

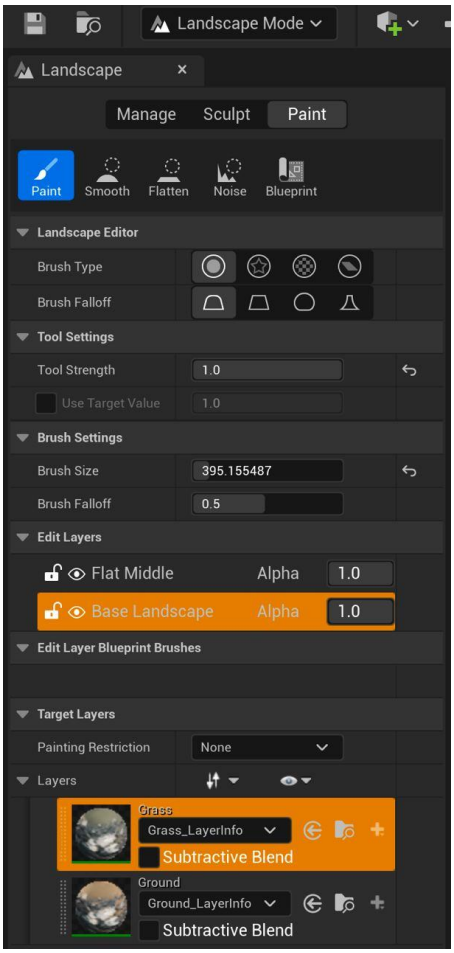


Рисунок 3.2 – Вікно Landscape

Після створення та налаштування ландшафту в Unreal Engine 5 настає час додати текстури для землі, трави та рослин за допомогою готових асетів. Спочатку відкриваємо Content Browser, що розташований у нижній частині вікна Unreal Engine 5, де зберігаються всі асети, включаючи текстури. Якщо потрібних текстур ще немає, їх можна імпортувати до проєкту. Для цього клацаємо правою кнопкою миші в Content Browser, обираємо "Import to /<YourFolder>" і вибираємо потрібні файли текстур для землі, трави та рослин.

Створюємо новий матеріал для кожного типу текстури. Клацаємо правою кнопкою миші в Content Browser, обираємо "Create" і далі "Material". Називаємо матеріали відповідно до їхньої функції, наприклад, "M_Ground", "M_Grass" та "M_Plants". Відкриваємо створений матеріал подвійним клацанням. У вікні редактора матеріалів перетягуємо відповідну текстуру з Content Browser у вікно редактора матеріалів та підключаємо текстуру до

базового кольору (Base Color) матеріалу. Для більш реалістичного вигляду можна також підключити нормальну карту (Normal Map) та карту відбивання (Roughness Map), якщо вони є.

Далі повертаємося до ландшафту у Viewport та обираємо інструмент Paint у вкладці Landscape. У розділі Target Layers додаємо новий шар для кожного матеріалу, натискаючи кнопку "+" і обираючи "Weight-Blended Layer (normal)". Називаємо кожен шар відповідно до матеріалу, наприклад, "GroundLayer" або "GrassLayer", і перетягуємо створені матеріали до відповідних шарів у Target Layers. Тепер можна розфарбовувати ландшафт, вибираючи потрібний шар у Target Layers і використовуючи пензлик у Viewport для нанесення текстури на ландшафт. Розмір і інтенсивність пензлика можна налаштувати для більш точного нанесення.

Для додавання рослинності обираємо інструмент Foliage у верхньому меню. У панелі Foliage додаємо асети рослин, перетягуючи їх з Content Browser у вкладку Foliage Types, та налаштовуємо параметри розподілу рослинності, такі як щільність і масштаб (рис 3.3). Використовуємо пензлик для нанесення рослинності на ландшафт, що дозволяє додати дерев, кущів та іншої рослинності, надаючи світові більш реалістичний вигляд.

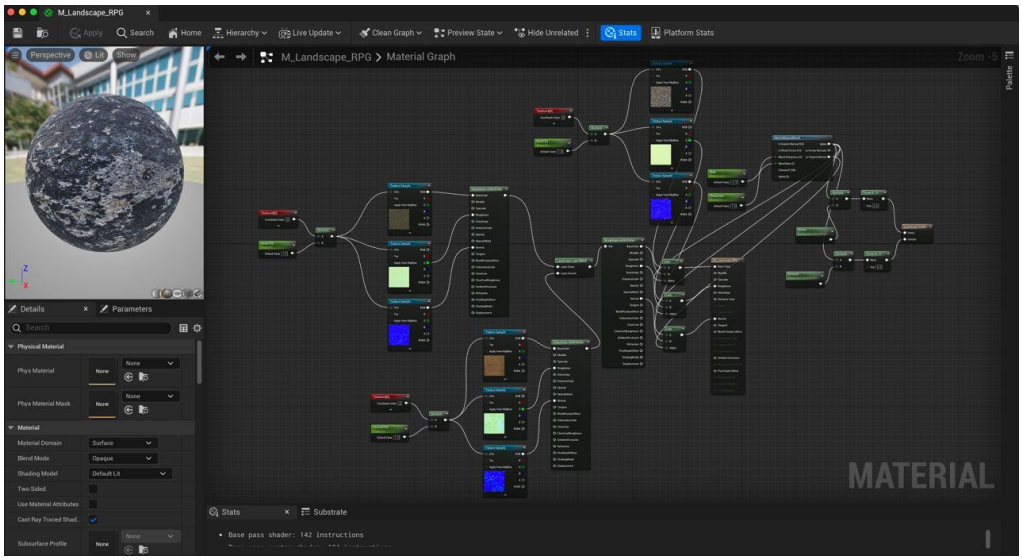


Рисунок 3.3 – Вікно створеної текстури світу

Після вимкнення зеленого фону, переходимо до наступного етапу - розміщення дерев та іншої рослинності для створення більш реалістичної сцени.

Натхненні природою, використовуємо інструмент "Foliage" для додавання дерев на ландшафт. Перетягуючи асети з деревами з Content Browser на сцену, розміщуємо їх у відповідних місцях, створюючи натуральні лісові області.

Рисунок 3.4 – Результат розміщення дерев та накладання текстр на ігровий світ

Далі, щоб збагатити нашу сцену різноманітною рослинністю, додаємо інші види рослин, такі як кущі, трава, та квіти. Використовуючи той же інструмент "Foliage", розміщуємо ці рослини на ландшафті, створюючи природний, збалансований вигляд сцени (рис 3.4).

3.3 Імпорт та анімація головного героя

Спочатку відкриваємо Epic Games Launcher і переходимо до магазину Unreal Engine. В пошуковому рядку вводимо "Paragon: Aurora". Знаходимо потрібний асет і завантажуємо його, натискаючи на кнопку "Free" або "Add to Cart", якщо асет безкоштовний, і оформлюємо покупку.

Після завершення покупки відкриваємо вкладку "Library" в Epic Games Launcher. Знаходимо Paragon: Aurora у списку придбаних асетів. Натискаємо

кнопку "Add to Project" і обираємо проєкт зі списку. Натискаємо "Add to Project" ще раз, щоб розпочати імпортування асету до проєкту.

Коли імпортування завершиться, відкриваємо проєкт в Unreal Engine 5. Відкриваємо Content Browser і знаходимо папку з імпортованим асетом Paragon: Aurora. Перетягуємо модель Aurora на сцену і налаштовуємо її позицію, обертання та масштаб за допомогою інструментів редагування.

Далі приступаємо до створення анімацій для Paragon: Aurora, таких як анімації смерті, бігу та спокою.

Відкриваємо Unreal Engine 5 і знаходимо імпортований асет Paragon: Aurora у Content Browser. Перетягуємо модель Aurora на сцену, якщо вона ще не додана.

Переходимо до вкладки "Animation" у Content Browser і створюємо нову анімацію. Клацаємо правою кнопкою миші на модель Aurora і вибираємо "Create" > "Animation" > "Animation Sequence". Називаємо анімацію відповідно до її типу, наприклад, "Aurora_Death".

Для створення анімації смерті, відкриваємо "Aurora_Death" (рис 3.5) у редакторі анімацій. Додаємо ключові кадри (keyframes) у потрібні моменти, щоб визначити початкову і кінцеву позиції Aurora під час анімації смерті. Налаштовуємо проміжні кадри, щоб створити плавний перехід від початкової до кінцевої позиції. Використовуємо інструменти редагування для коректного налаштування рухів моделі.

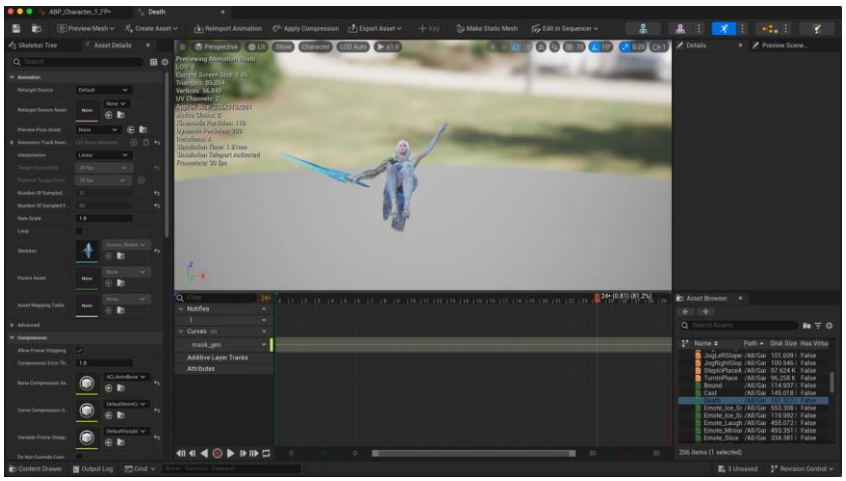


Рисунок 3.5 – Анімація смерті

Далі створюємо анімацію бігу. Повторюємо попередні кроки, створюючи нову анімацію під назвою "Aurora_Run" (рис 3.6). У редакторі анімацій додаємо ключові кадри, щоб визначити рухи Aurora під час бігу. Налаштовуємо рухи рук, ніг і тіла, щоб створити реалістичну анімацію бігу.

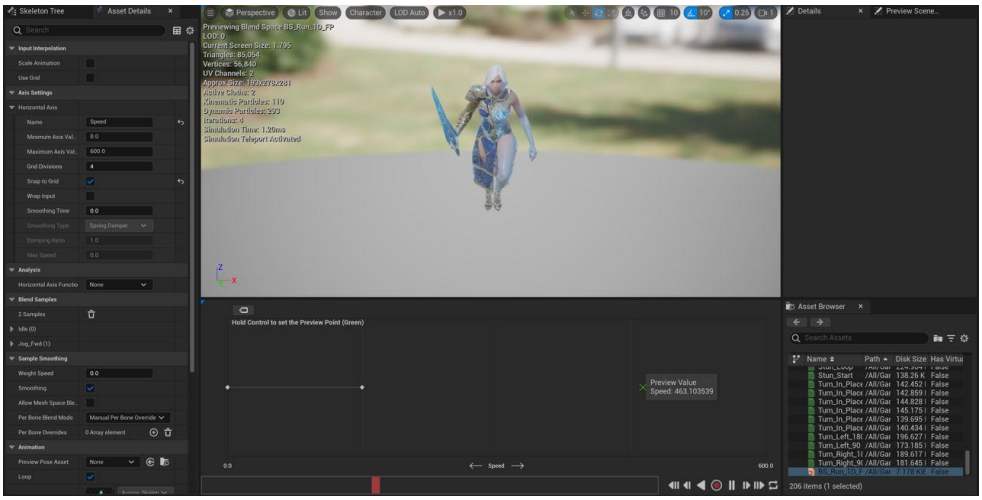


Рисунок 3.6 – Анімація бігу

Наступною створюємо анімацію спокою. Створюємо нову анімацію під назвою "Aurora_Idle" (рис 3.7). У редакторі анімацій додаємо ключові кадри для визначення позицій Aurora під час спокою. Налаштовуємо дрібні рухи, такі як дихання або легке погойдування, щоб надати анімації більш природний вигляд.

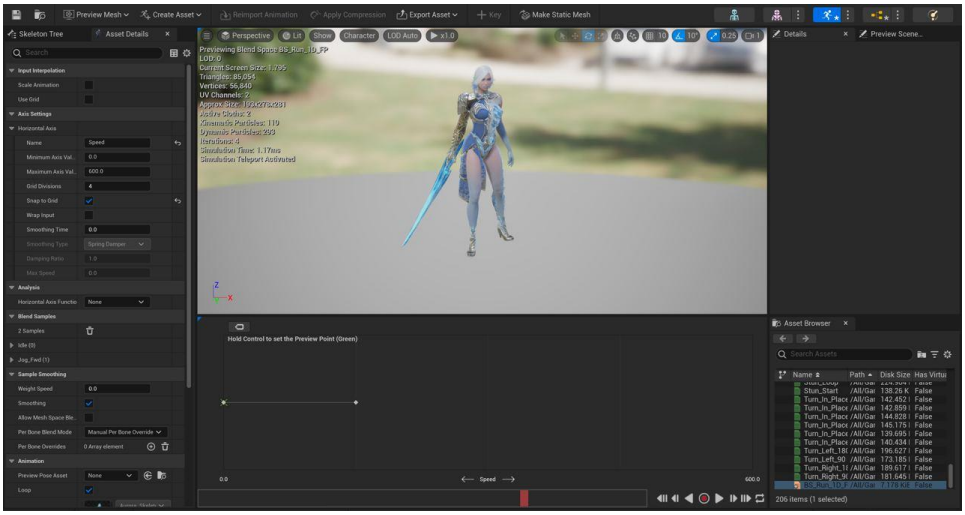


Рисунок 3.7 – Анімація спокою

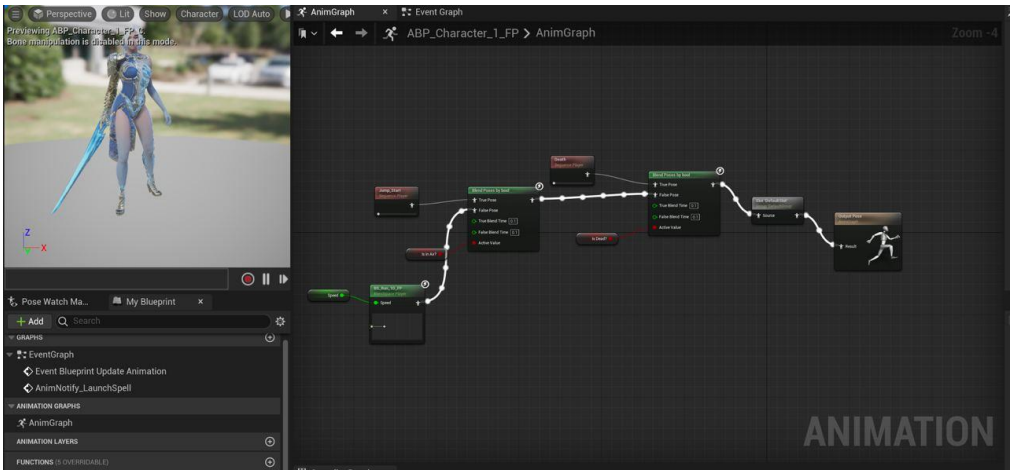


Рисунок 3.8 – Загальна схема всіх анімацій героя

Після створення всіх анімацій, зберігаємо їх і переходимо до налаштування анімаційного блендера (Animation Blueprint). Це дозволить плавно перемикатися між різними анімаціями в залежності від стану персонажа. Створюємо новий Animation Blueprint для Aurora (рис 3.8) і налаштовуємо його відповідно до потреб проєкту.

3.4 Створення та анімація здібностей

Створення та анімація ефектів Fireball, Lightning і Fireburst за допомогою Niagara System в Unreal Engine 5 включає кілька важливих кроків. Спочатку відкриваємо Content Browser, клацаємо правою кнопкою миші і вибираємо "FX", а потім "Niagara System". У вікні, що з'явиться, обираємо "New System from Template" і натискаємо "Next". Вибираємо шаблон, який найбільш підходить для створення ефекту Fireball, наприклад, "Directional Burst", і натискаємо "Finish".

Після створення Niagara System відкриваємо його для налаштування. Додаємо емітер для частинок, які будуть складати вогняну кулю. Налаштовуємо параметри емітера, такі як форма, розмір, швидкість і колір частинок, щоб створити ефект Fireball. Використовуємо модуляцію розміру, кольору та швидкості для створення реалістичного вигляду вогняної кулі (рис 3.9).

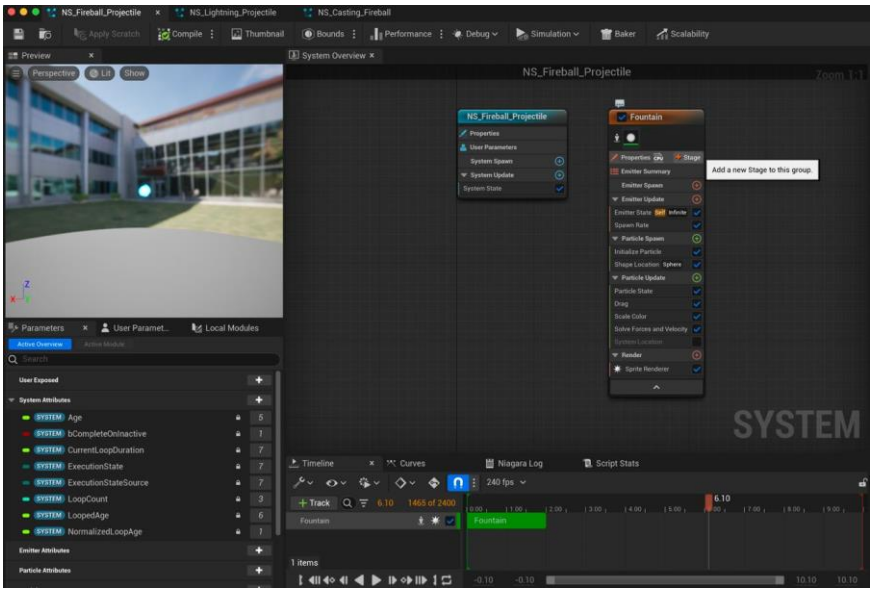


Рисунок 3.9 – Створений та анімований Fireball

Для створення ефекту Lightning, повторюємо процес, починаючи з створення нової системи Niagara. Обираємо відповідний шаблон, такий як "Beam", для створення блискавки. Налаштовуємо емітер так, щоб він генерував частинки у вигляді лінії, додаючи ефект розгалужень та змінної яскравості для досягнення реалістичного ефекту блискавки (рис 3.10).

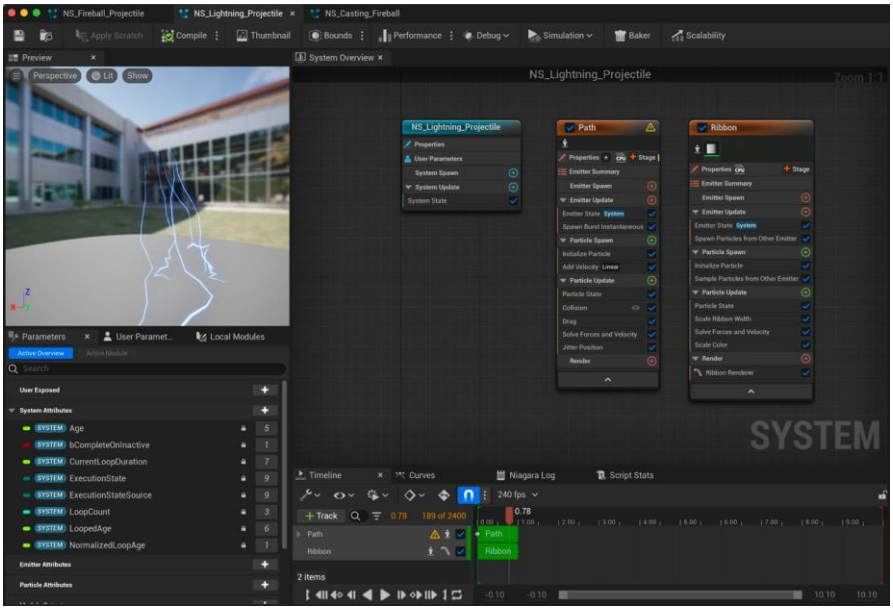


Рисунок 3.10 – Створений та анімований Lightning

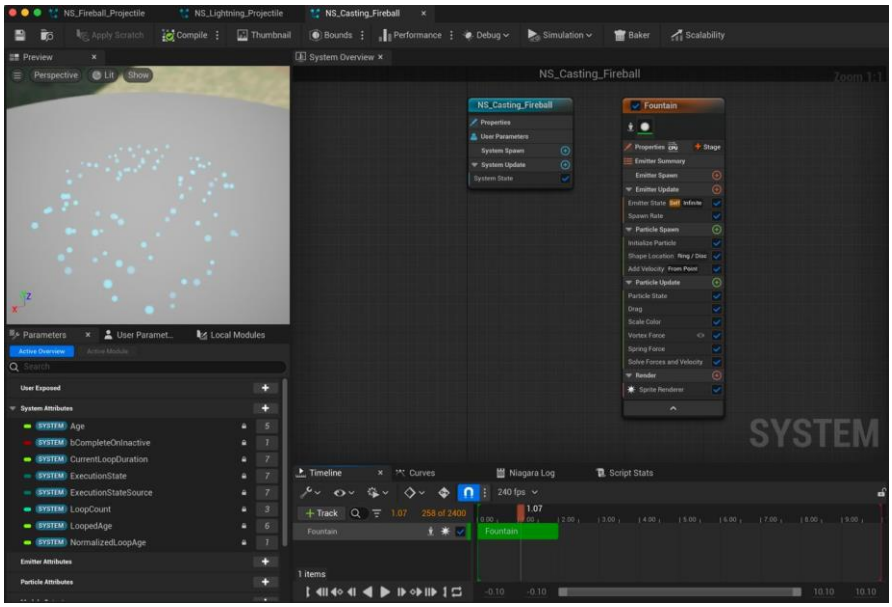


Рисунок 3.11 – Створений та анімований Fireburst

Для Fireburst створюємо нову систему Niagara та обираємо шаблон, який підходить для вибуху. Налаштовуємо емітер, щоб створити великий і швидкий спалах частинок, налаштовуємо їхній розмір, швидкість і колір для імітації вибуху вогню. Використовуємо модуляцію параметрів, щоб додати динаміки і реалістичності ефекту (рис 3.11).

Висновки до розділу 3

У процесі розробки гри було розглянуто різні технології, зокрема ігрові рушії, інструменти для моделювання та анімації, а також технології для створення візуальних ефектів. Завдяки цьому вдалося визначити найбільш підходящі інструменти для проєкту, зокрема, Unreal Engine 5, який надає потужні можливості для створення високоякісних ігор.

Наступним етапом було моделювання ігрового світу. Використовуючи Unreal Engine 5, було створено детальний та реалістичний ландшафт за допомогою інструментів Landscape. Після цього застосовано готові асети для землі, трави та рослин, що дозволило створити привабливе ігрове середовище, яке стане основою для подальшої розробки.

Другий етап полягав у імпорті та анімації головного героя. Після завантаження моделі Paragon: Aurora, було імпортовано її до проєкту та розроблено набір анімацій для різних станів персонажа (смерть, біг, спокій). Розробка та впровадження цих анімаційних ефектів додало динаміки та реалістичності головному герою, зробивши його більш живим та інтерактивним.

Завершальним етапом було створення та анімація здібностей персонажа за допомогою Niagara System. Створено ефекти Fireball, Lightning і Fireburst, налаштувавши параметри емітерів для досягнення реалістичності та динамічності цих здібностей. Ці візуальні ефекти значно збагатили геймплей, додавши видовищності та унікальності персонажу.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ІГРОВОГО ЗАСТОСУНКУ

4.1 Створення меню гри

Для розробки меню проєкту необхідно виконати наступні дії :

1) відкрити проєкт Unreal Engine 5;

2) у Content Browser обираємо "User Interface" і створюємо новий "Widget Blueprint" (в проєкті має назву "MainMenu");

3) додаємо три кнопки у Widget Blueprint і додаємо три кнопки: "Start New Game", "Load Game" і "Quit Game". Компонент "Button" палітри (Palette) Widget Blueprint Editor додаємо в область конструктора (Designer) та розміщуємо у відповідності до дизайну проєкту (вертикально або горизонтально).

Після розміщення кнопок, налаштовуємо події для кожної з них. Вибираємо кнопку "Start New Game", переходимо до вкладки "Graph" і додаємо подію "OnClicked". В цій події створюємо логіку для запуску нової гри, завантажуючи відповідний рівень. Повторюємо ці кроки для кнопки "Load Game", налаштовуючи її для завантаження збереженої гри. Для кнопки "Quit Game" додаємо подію "OnClicked", яка викликає функцію для виходу з гри.

Зберігаємо Widget Blueprint. Відкриваємо рівень, на якому має відображатися меню. В Blueprint цього рівня додаємо логіку для створення та відображення Widget Blueprint при запуску гри. Це забезпечить появу головного меню при запуску гри. Таким чином, створюємо меню гри з трьома кнопками в Unreal Engine 5, яке дозволяє почати нову гру, завантажити збережену гру або вийти з гри (рис 4.1).

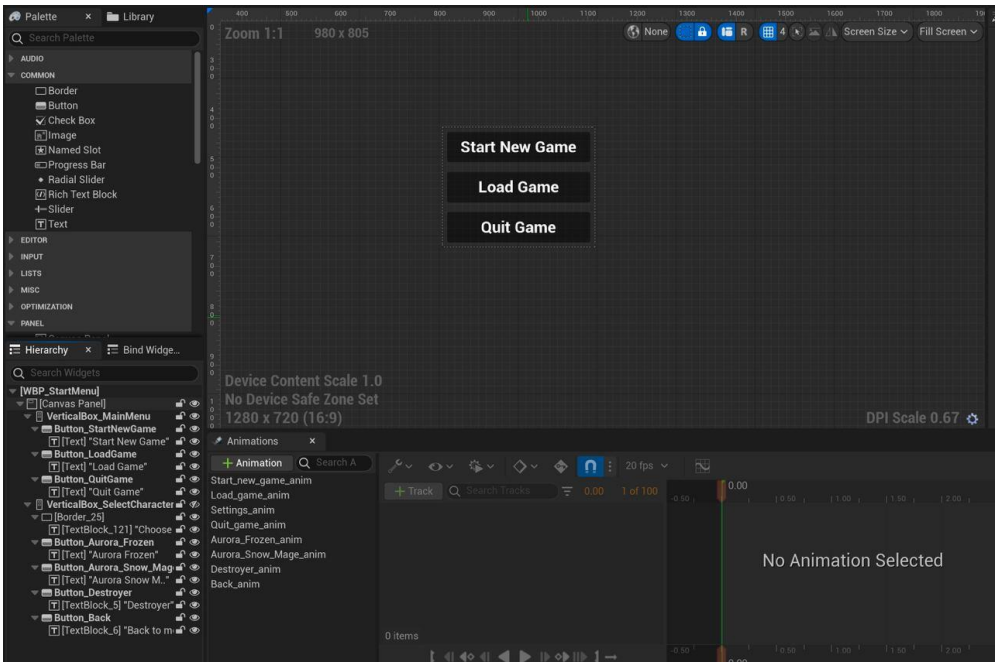


Рисунок 4.1 – Головне меню гри

Для створення меню вибору героя в Unreal Engine 5 з чотирма кнопками - "Aurora Frozen", "Aurora Snow Mage", "Destroyer" і "Back to Menu" - слід виконати кілька кроків. Ось як це зробити:

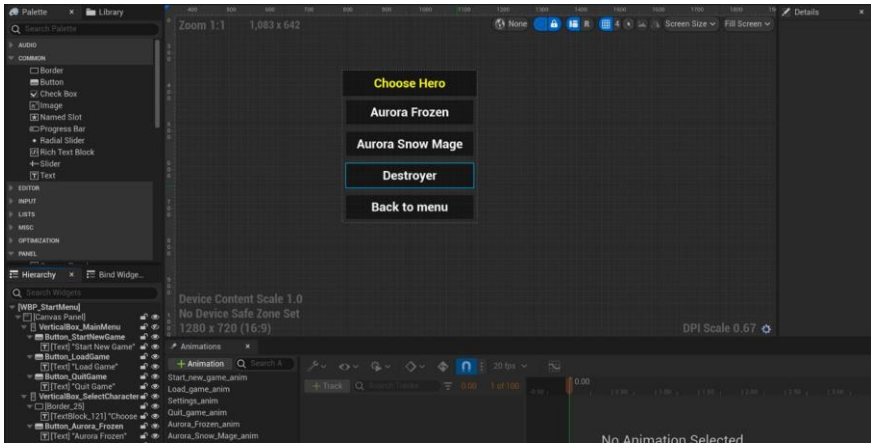
1) У Content Browser клацаємо правою кнопкою миші, вибираємо "User Interface" і створюємо новий "Widget Blueprint". Називаємо його "CharacterSelectMenu".

2) Відкриваємо створений Widget Blueprint і додаємо чотири кнопки. У Widget Blueprint Editor знаходимо компонент "Button" у палітрі (Palette) і перетягуємо його в область конструктора (Designer). Розміщуємо кнопки вертикально або горизонтально, залежно від вашого дизайну. Додаємо текст до кожної кнопки, щоб вони відображали "Aurora Frozen", "Aurora Snow Mage", "Destroyer" і "Back to Menu".

3) Налаштовуємо події для кожної кнопки. Вибираємо кнопку "Aurora Frozen", переходимо до вкладки "Graph" і додаємо подію "OnClicked". В цій події створюємо логіку для вибору героя Aurora Frozen. Це може включати встановлення змінної, яка визначає вибраного героя, або завантаження відповідного рівня чи сцени. Повторюємо ці кроки для кнопок

"Aurora Snow Mage" і "Destroyer", налаштовуючи їх для вибору відповідних героїв.

4) Для кнопки "Back to Menu" додаємо подію "OnClicked", яка повертає гравця до головного меню. Це може бути виконано шляхом завантаження головного меню або закриття поточного меню і відкриття головного меню.



5) Рисунок 4.2 – Меню вибору героя

Зберігаємо Widget Blueprint. Відкриваємо рівень, на якому має відображатися меню вибору героя. В Blueprint цього рівня додаємо логіку для створення та відображення Widget Blueprint при запуску гри. Це забезпечить появу меню вибору героя при запуску гри або при виборі відповідної опції з головного меню (рис 4.2).

4.2 Реалізація управління головного героя

Налаштовуємо управління головним героєм за допомогою Blueprint Class в Unreal Engine 5. Спочатку налаштовуємо ввід у налаштуваннях проєкту. Відкриваємо Unreal Engine 5 і переходимо до вашого проєкту. В меню Edit обираємо Project Settings, а потім у розділі Engine знаходимо Input. Тут додаємо нові осі вводу (Axis Mappings) і події (Action Mappings) для управління персонажем, наприклад, для руху вперед/назад, вліво/вправо, та атаки.

Далі відкриваємо Content Browser, клацаємо правою кнопкою миші і вибираємо "Blueprint Class". Створюємо новий клас на основі "Character" і називаємо його, наприклад, "MainCharacter". Відкриваємо створений Blueprint для редагування. В графіку подій (Event Graph) додаємо події для осей MoveForward і MoveRight, які налаштували раніше. Для руху вперед/назад використовуємо "Add Movement Input", де "Axis Value" буде визначати напрямок і силу руху. Аналогічно додаємо подію для осі MoveRight, щоб реалізувати рух вліво та вправо (рис 4.3).

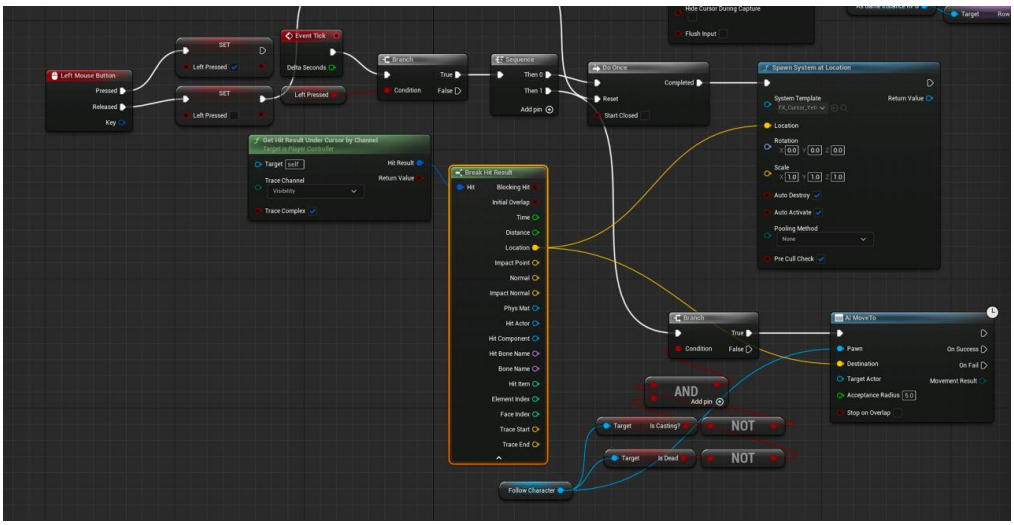


Рисунок 4.3 – Реалізація управління персонажем

Для реалізації атаки створюємо нову подію, наприклад, "Attack", і додаємо до неї логіку атаки (рис 4.4). Це може бути виклик анімації атаки або завдання шкоди ворогу.

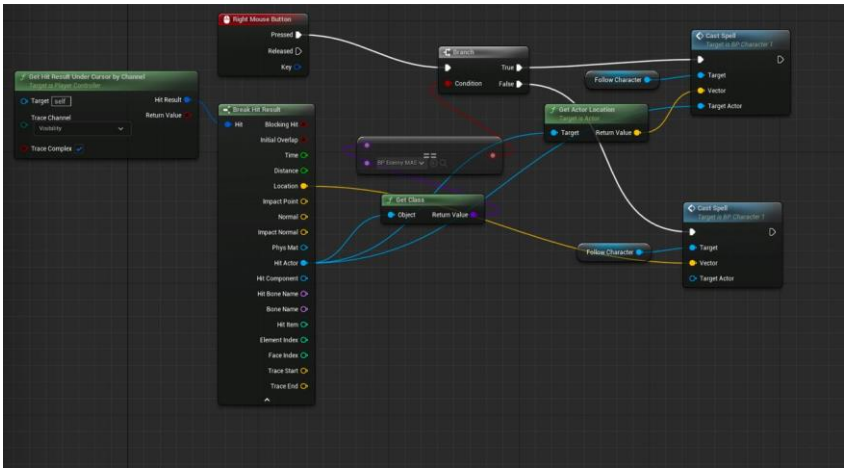


Рисунок 4.4 – Реалізація атаки

Налаштовуємо камеру для вашого персонажа. Додаємо компонент "Camera" до вашого Blueprint і розміщуємо його так, щоб він надавав оптимальний огляд вашого персонажа під час гри. Налаштовуємо прив'язку камери до руху персонажа для плавного слідування за ним.

Зберігаємо Blueprint. Тепер персонаж повинен відповідати на введення з клавіатури або контролера, дозволяючи рухатися вперед, назад, вліво, вправо, та атакувати. Переходимо до нашого ігрового рівня і переконуємося, що персонаж використовує новостворений Blueprint Class. Запускаємо гру і тестуємо управління, щоб переконатися, що все працює правильно і без помилок.

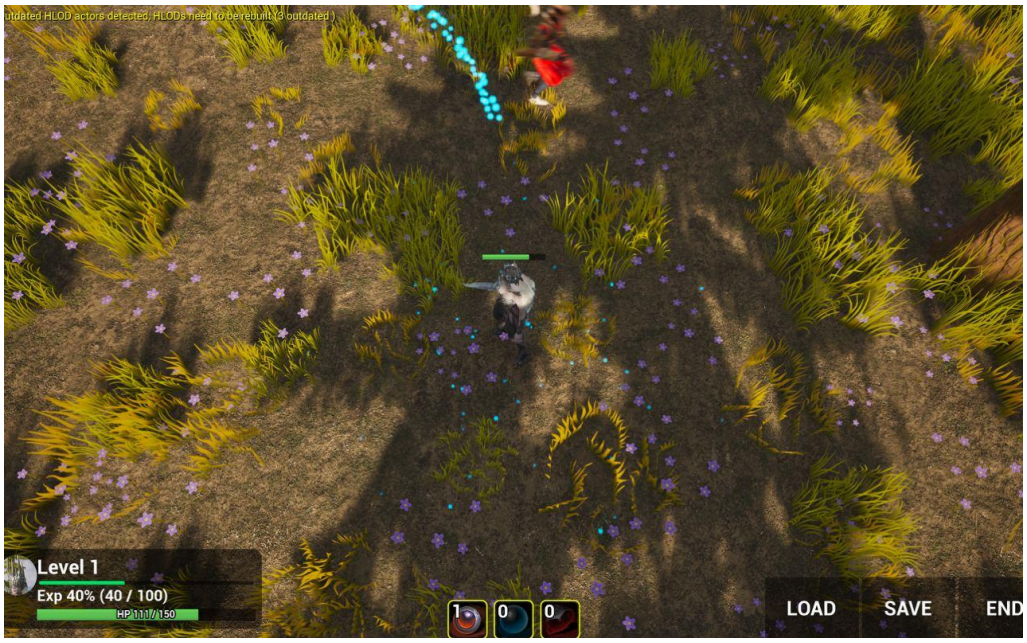


Рисунок 4.5 – Загальний результат

Таким чином, налаштовуємо управління головним героєм за допомогою Blueprint Class в Unreal Engine 5, забезпечуючи йому всі необхідні дії для інтерактивного ігрового процесу (рис 4.5).

4.3 Реалізація додаткових елементів головного героя

Для створення функціоналу запуску заклять (Launch Spell) у Blueprint персонажа (Character_BP), спочатку відкриваємо Character_BP у Blueprint Editor. Переходимо до графіка подій (Event Graph) і створюємо нову подію для

запуску заклять. Для цього можемо використати існуючий ввід (наприклад, клавішу або кнопку миші) або додати новий ввід у налаштуваннях проєкту.

Налаштовуємо ввід для заклять. Відкриваємо Edit > Project Settings і переходимо до розділу Input. Додаємо нову подію (Action Mapping) для заклять, назвавши її, наприклад, "LaunchSpell". Призначаємо клавішу або кнопку миші для цієї події.

Повертаємося до Character_BP. У графіку подій додаємо нову подію "LaunchSpell" з вкладки "Input". Ця подія буде викликатися при натисканні призначеної клавіші або кнопки.

Далі створюємо логіку для запуску заклять. Залежно від типу заклять, які хочемо реалізувати, створюємо різні ефекти або об'єкти, що представляють ці закляття. Наприклад, для створення вогняної кулі (Fireball) можемо використовувати Niagara System або інший відповідний компонент.

Додаємо відповідні компоненти або ефекти до нашого Character_BP. Це може бути компонент Niagara System, який створює візуальний ефект закляття. Також можемо створити новий Blueprint для закляття, який буде містити логіку його поведінки (наприклад, рух до цілі, завдання шкоди тощо).

Повертаємося до графіка подій Character_BP і створюємо функцію для запуску заклять. Називаємо її, наприклад, "LaunchSpellFunction". У цій функції додаємо логіку для створення та запуску заклять. Використовуємо "Spawn Actor from Class" для створення об'єкта закляття і "Add Movement Input" або інші відповідні функції для керування його рухом.

Підключаємо подію "LaunchSpell" до функції "LaunchSpellFunction", щоб запускати цю функцію при натисканні відповідної клавіші або кнопки.

Налаштовуємо параметри заклять, такі як швидкість, напрямок, ефекти, щоб забезпечити бажану поведінку закляття. Тестуємо нашого персонажа в грі, щоб переконатися, що функціонал запуску заклять працює правильно і без помилок (рис 4.6).

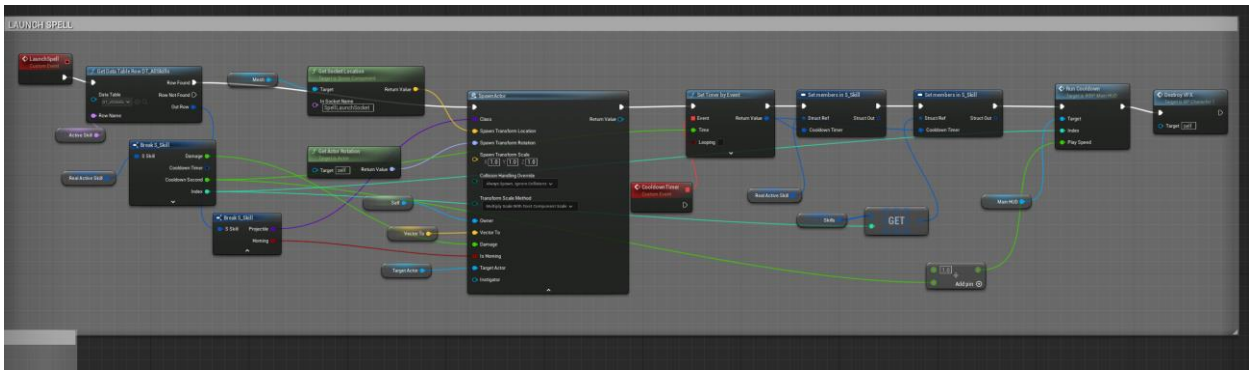


Рисунок 4.6 – Реалізація навичок

Створюємо функцію, яка контролює смерть головного героя та відродження на найближчій точці спауну. Відкриваємо Character_BP у Blueprint Editor. Створюємо нову подію, яка викликається при смерті персонажа, наприклад, коли здоров'я досягає нуля. Додаємо нову функцію у Character_BP і називаємо її "OnDead". В цій функції додаємо логіку для обробки смерті персонажа.

Створюємо виклик анімації смерті у функції "OnDead" за допомогою "Play Animation" або аналогічної функції. Після завершення анімації смерті налаштовуємо відродження персонажа. Для цього знаходимо найближчу точку спауну і переміщуємо персонажа до неї.

Далі переходимо до створення точок спауну. У Content Browser створюємо новий Blueprint Class на основі "Actor" і називаємо його "SpawnPoint". Розміщуємо кілька екземплярів цього спаун-пойнта на рівні в місцях, де хочемо, щоб персонаж міг відродитися.

Повертаємося до функції "OnDead" в Character_BP і додаємо логіку для пошуку найближчої точки спауну. Використовуємо "Get All Actors of Class" для отримання всіх спаун-пойнтів на рівні, а потім визначаємо найближчу до поточної позиції персонажа. Після цього використовуємо "Set Actor Location" для переміщення персонажа до координат найближчого спаун-пойнта (рис 4.7).

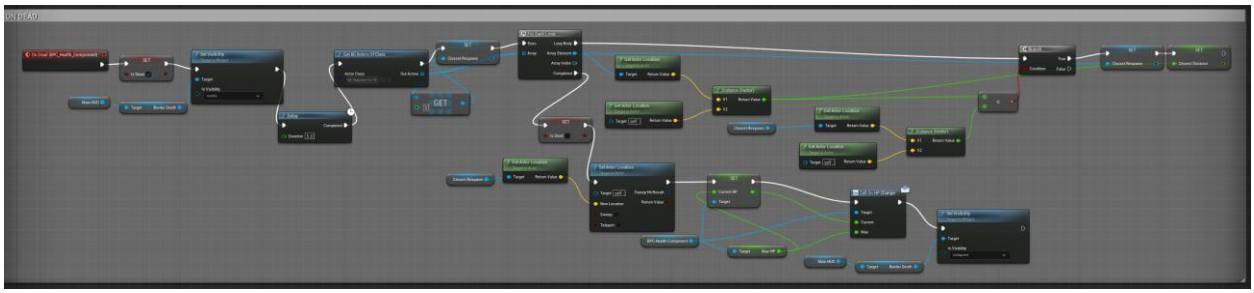


Рисунок 4.7 – Реалізація смерті головного героя

Створюємо функцію OnHpChange, яка контролює зміну здоров'я головного героя в залежності від нанесеного урону. Відкриваємо Character_BP у Blueprint Editor. Додаємо нову змінну типу float для здоров'я персонажа і називаємо її "Health". Встановлюємо початкове значення для цієї змінної.

Нову функцію у Character_BP і називаємо її "OnHpChange". В цій функції додаємо логіку для обробки зміни здоров'я персонажа. В параметри цієї функції додаємо змінну типу float, яка буде представляти кількість нанесеного урону, "Damage".

В тілі функції "OnHpChange" віднімаємо значення урону від поточного значення змінної "Health". Після цього перевіряємо, чи здоров'я не досягло нуля або менше. Якщо здоров'я персонажа менше або дорівнює нулю, викликаємо функцію "OnDead" для обробки смерті персонажа.

Додаємо події або виклики "OnHpChange" у тих місцях, де персонаж отримує урон, наприклад, в зіткненнях з ворогами або пастками. Це дозволить автоматично змінювати здоров'я персонажа при отриманні урону і викликати відповідну реакцію на смерть (рис 4.8).

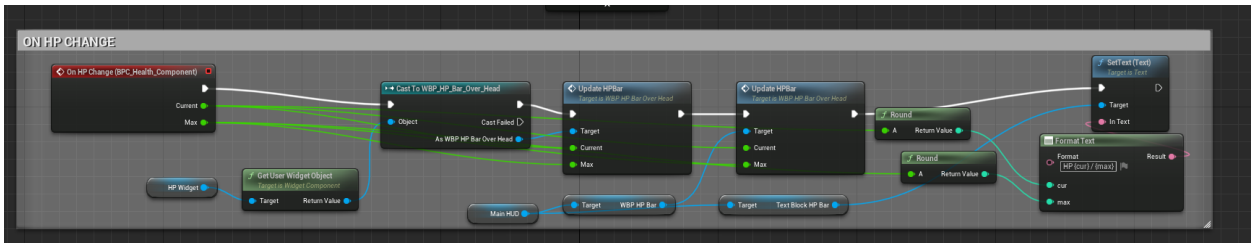


Рисунок 4.8 – Реалізація зімни здоров'я від урону противника

Створюємо функцію SetSkills, яка контролює вибір скіла та його прокачку в Character_BP. Відкриваємо Character_BP у Blueprint Editor.

Додаємо нову змінну типу масив, яка буде зберігати інформацію про доступні скіли, і називаємо її "Skills". Кожен елемент цього масиву може бути структурою, що містить інформацію про назву скіла, рівень прокачки, максимальний рівень і ефект.

Нову функцію у Character_BP і називаємо її "SetSkills". В параметри цієї функції додаємо змінні, які будуть представляти вибір скіла та його прокачку. Наприклад, додаємо змінну типу string для назви скіла, яку називаємо "SkillName", і змінну типу int для рівня прокачки, яку називаємо "SkillLevel".

В тілі функції "SetSkills" додаємо логіку для знаходження обраного скіла у масиві "Skills". Перебираємо масив "Skills" за допомогою циклу і порівнюємо назву кожного скіла з значенням змінної "SkillName". Коли знайдено відповідний скіл, перевіряємо, чи не перевищує новий рівень прокачки значення максимального рівня для цього скіла. Якщо все в порядку, оновлюємо рівень прокачки цього скіла.

Додаємо логіку для відображення нових можливостей або ефектів, пов'язаних з прокачаним скілом. Це може включати зміну анімацій, збільшення шкоди, зміну візуальних ефектів або інші покращення. Використовуємо відповідні функції або події для застосування цих змін у грі.

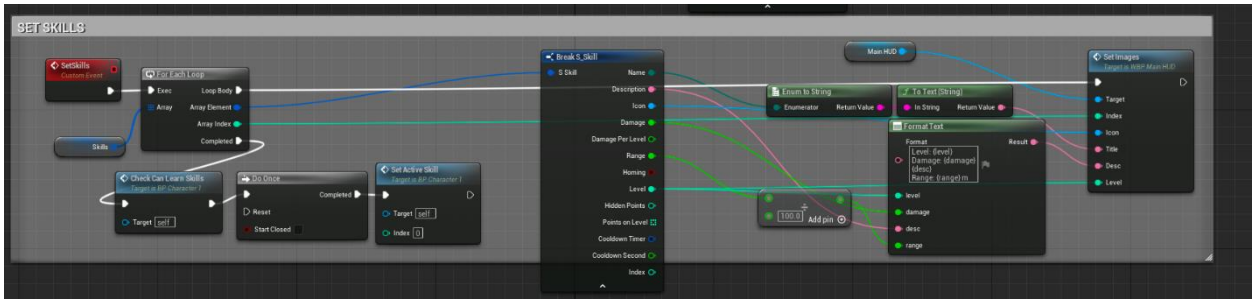


Рисунок 4.9 – Реалізація вибору та прокачки скілів

Додаємо події або виклики "SetSkills" у тих місцях, де гравець може вибирати та прокачувати скіли, наприклад, у меню прокачки персонажа або при отриманні нового рівня (рис 4.9).

4.4 Реалізація поведінки ворогів

Для того, щоб додати ворога скелета з анімаціями у Enemy_Master Blueprint, спочатку імпортуємо моделі та анімації для скелета. Відкриваємо Unreal Engine 5, завантажуємо необхідні файли з 3D моделлю та відповідними анімаціями, такими як ходьба, атака, смерть, та імпортуємо їх у Content Browser.

Далі створюємо анімаційний Blueprint для скелета. У Content Browser клацаємо правою кнопкою миші на скелетну модель і вибираємо "Create" > "Animation Blueprint", називаючи його "Skeleton_AnimBP". У цьому Animation Blueprint налаштовуємо State Machine, додаючи необхідні стани для різних анімацій, таких як Idle, Walk, Attack, і Death.

Наступним кроком створюємо Blueprint ворога. В Content Browser клацаємо правою кнопкою миші і вибираємо "Blueprint Class", обираємо "Character" і називаємо створений Blueprint "Enemy_Skeleton". Відкриваємо цей Blueprint для редагування, додаємо "Skeletal Mesh" компонент, який міститиме модель скелета. У властивостях "Skeletal Mesh" компонента вибираємо імпортовану модель скелета.

Після цього додаємо Animation Blueprint до скелетної сітки, вибираючи "Skeleton_AnimBP" у властивостях "Animation" компонента. Налаштовуємо поведінку ворога, додаючи логіку для пересування та атак у графіку подій (Event Graph) у "Enemy_Skeleton" Blueprint. Використовуємо анімаційні події та функції для відтворення відповідних анімацій у відповідь на події, такі як наближення до гравця або нанесення удару.

Нарешті, інтегруємо ворога скелета в рівень, розмістивши його на карті. Переконуємося, що він належним чином реагує на гравця, атакує та відтворює відповідні анімації. Таким чином, створюємо ворога скелета з анімаціями у Enemy_Master Blueprint, забезпечуючи інтерактивний ігровий процес з ворогами (рис 4.10).

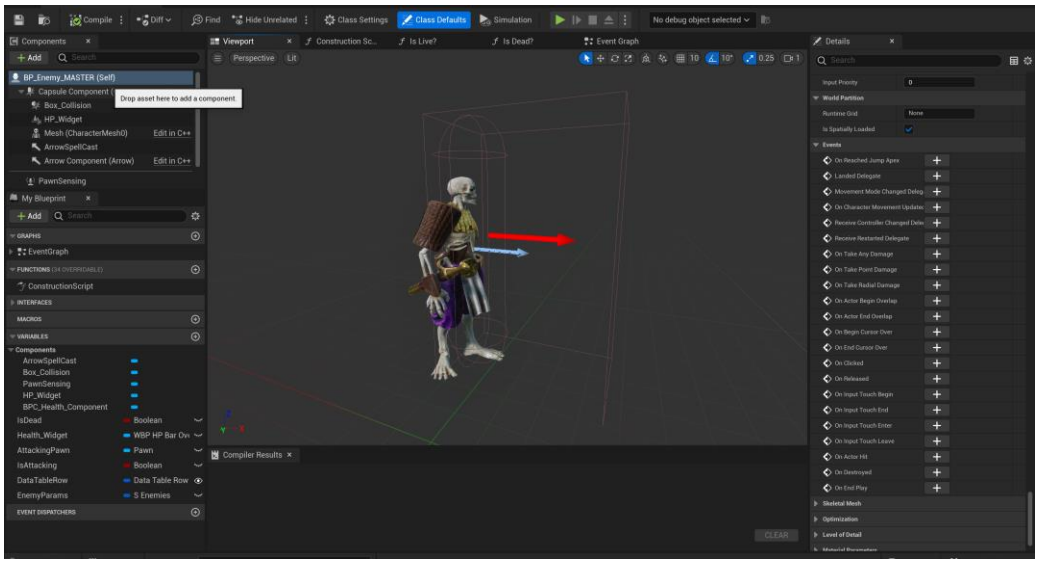


Рисунок 4.10 – Імпортований ворог зі всіма анімаціями

У Event Graph реалізуємо атаку ворога скелета з ближньої та дистанційної відстані.

Для атаки ближнього бою, визначаємо подію атаки при контакті гравця з радіусом атаки ворога. При спрацюванні цієї події, запускаємо анімацію атаки ворога та проводимо перевірку на стан гравця та ворога перед нанесенням шкоди.

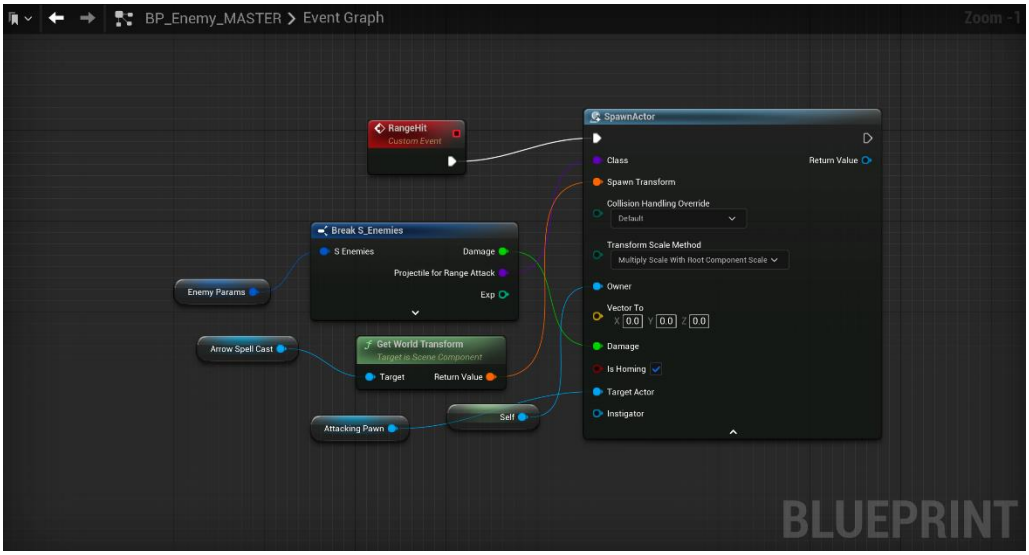


Рисунок 4.11 – Реалізація ближньої та атаки на відстані

Для атаки за допомогою магії, створюємо окрему функцію атаки магією. Визначаємо подію атаки магією при відповідній відстані від гравця. При

спрацюванні цієї події, запускаємо анімацію атаки магією ворога та створюємо магічний ефект, який наносить шкоду гравцеві.

В обох випадках проводимо перевірку на життя та смерть гравця та ворога перед нанесенням шкоди, забезпечуючи правильну логіку бойового процесу у грі (рис 4.11).

Висновки до розділу 4

У четвертому розділі було зроблено важливі кроки для створення цілісного і цікавого ігрового досвіду. Починаючи зі створення основного меню гри з рядом опцій для користувача, надали гравцеві зручний спосіб керувати грою. Реалізація управління головним героєм за допомогою Blueprint class дозволила створити інтуїтивно зрозумілий і ефективний спосіб взаємодії з персонажем, що допомагає поглибити іммерсію гравця в грі.

Додавання додаткових елементів головного героя, таких як навички та їх прокачка, робить геймплей більш різноманітним і дає гравцеві більше можливостей для розвитку персонажа. Це збільшує глибину і складність гри, роблячи її більш захоплюючою.

Успішна реалізація поведінки ворогів додає викликів для гравця і створює додаткові шари стратегії в грі. Вороги здатні реагувати на дії гравця та взаємодіяти з ним, що робить кожну зустріч у грі унікальною та цікавою.

Загалом вищезгадані фактори сприяють створенню ігрового середовища, яке пропонує глибокий і цікавий геймплей, роблячи гру захоплюючою та відмінною для гравців будь-якого рівня досвіду.

ВИСНОВКИ

У рамках виконання кваліфікаційної роботи бакалавра був проведений аналіз та огляд жанру RPG. Виділено його основні характеристики, такі як можливість поглибленого занурення гравців у фантастичні світи, боротьба з ворогами, розвиток персонажів та дослідження відкритих світів.

Unreal Engine 5 відіграє ключову роль у трансформації розробки RPG ігор, надаючи розробникам потужні інструменти для створення реалістичної графіки та глибокого геймплею. Цей рушій стимулює творчість і дозволяє втілювати інноваційні ідеї, відкриваючи нові горизонти для фантастичних віртуальних світів.

Для досягнення поставленої мети було вирішено наступні завдання:

1) проведено аналіз існуючих ігор в жанрі RPG, виявлено їх переваги та недоліки;

2) сформульовано специфікацію вимог до розроблюваного ігрового застосунку;

3) досліджено функціональні можливості рушія Unreal Engine 5 та його інструментарію для розробки ігрового застосунку в жанрі RPG;

4) розроблено концепт ігрового світу на основі жанру RPG;

5) розроблено унікальний геймплей, що включає бої, локації та інші ключові ігрові елементи;

6) розроблено графічні, анімаційні та інших ефекти;

7) проведено тестування програмного забезпечення ігрового застосунку.

Результатом роботи є глибоке розуміння та ефективне використання Unreal Engine 5 для розробки RPG ігор. Дослідження охоплює ключові аспекти роботи з рушієм, включаючи створення деталізованих графічних образів, реалізацію динамічного освітлення та розгалужені можливості для інтерактивності.

Ця робота підкреслює важливість технічного прогресу та інновацій у створенні передових RPG ігор. Основний акцент зроблено на оптимізації геймплею та поліпшенні візуальної атмосфери, що дозволяє створювати вражаючі ігрові світи. Unreal Engine 5 виступає не лише як інструмент, а як каталізатор для нових тенденцій у світі відеоігор.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. RPG і CRPG – що це таке, суть, види, жанри і приклади РПГ в ігровій індустрії. URL: https://termin.in.ua/rpg-i-crpg/ (дата звернення: 02.05.2024);

2. Кращі старі RPG. URL: https://gameshare.com.ua/17-30-luchshix-staryx-rpg-igr-na-pk---1997-2010-g.html (дата звернення: 02.05.2024);

3. Warcraft 3: Reforged. URL: https://warcraft3.blizzard.com/en-us/ (дата звернення: 02.05.2024);

4. League of Legends. URL: https://leagueoflegends.fandom.com/uk/wiki/League_of_Legends (дата звернення: 02.05.2024);

5. Heroes of the Storm. URL: https://itc.ua/articles/predvaritelnyiy-obzor-heroes-of-the-storm/ (дата звернення: 02.05.2024);

6. Using State Diagrams (UML) to Write Better User Stories. URL: https://www.devonblog.com/user-story/using-state-diagrams-uml-to-write-better-user-stories/ (дата звернення: 02.05.2024);

7. UML versions and UML modelling tools/software. URL: https://eboobae.hashnode.dev/uml-versions-and-uml-modelling-toolssoftware (дата звернення: 02.05.2024);

8. Допоміжні діаграми взаємодії. URL: https://prezi.com/mc6rlyo9echx/presentation/ (дата звернення: 02.05.2024);

9. Варіанти використання та сценарії (Use Cases and Scenarios). URL: https://www.maxzosim.com/use-cases-and-scenarios/ (дата звернення: 02.05.2024);

10. Діаграма діяльності в UML: символ, компоненти та приклад. URL: https://www.guru99.com/uk/uml-activity-diagram.html (дата звернення: 02.05.2024);

11. Створення схеми діаграми станів UML. URL: https://support.microsoft.com/uk-

office/%D1%81%D1%82%D0%B2%D0%BE%D1%80%D0%B5%D0%BD%D0%BD%D1%8F-%D1%81%D1%85%D0%B5%D0%BC%D0%B8-%D0%B4%D1%96%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%B8-%D1%81%D1%82%D0%B0%D0%BD%D1%96%D0%B2-uml-2e46fd66-e861-4e8c-9188-36255395ebf3 (дата звернення: 02.05.2024);

12. Unreal engine: переваги та основні можливості ігрового двигуна. URL: https://avada-media.ua/ua/services/unreal-engine-preimushchestva-i-osnovnyye-vozmozhnosti-igrovogo-dvizhka/ (дата звернення: 27.05.2024);

13. Unity проти Unreal Engine — який рушій обрати для гри та чому. URL: https://gamedev.dou.ua/articles/unity-or-unreal-engine/ (дата звернення: 27.05.2024);

14. Документація до Godot Engine 4.2. URL: https://docs.godotengine.org/uk/4.x/getting_started/introduction/introduction_to_godot.html (дата звернення: 27.05.2024);

15. Чи підходить CryEngine для новачків? URL: https://moshka.zapisi.cx.ua/ukraincyam/chi-pidkhodit-cryengine-dlya-novachkiv.html (дата звернення: 27.05.2024);

16. Чому game maker studio краще? URL: https://gamemaker.ucoz.com/publ/1-1-0-100 (дата звернення: 27.05.2024);