

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
(освітня програма «Комп'ютерна графіка та розробка ігор»)
на тему:

«Розробка комп'ютерної гри в жанрі Soulslike»

Виконав студент групи КН-20-2
МАТЯШ Олександр Вікторович

Керівник роботи:
МАРЧУК Галина Вікторівна

Рецензент:
ПЕТРОСЯН Руслан Валерікович

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

«ЗАТВЕРДЖУЮ»

Зав. кафедри комп'ютерних наук
Марина ГРАФ
«25» січня 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **МАТЯШ Олександр Вікторович**

Керівник роботи: **МАРЧУК Галина Вікторівна**

Тема роботи: **«Розробка комп'ютерної гри в жанрі Soulslike»**,

затверджена Наказом закладу вищої освіти від **«25» січня 2024 р., №22/с**

Вихідні дані для роботи: ігрові механіки та архітектура ігор у жанрі «soulslike».

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів,
що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Марчук Галина Вікторівна	27.02.2024р.	20.03.2024р.
2	Марчук Галина Вікторівна	29.03.2024р.	09.05.2024р.
3	Марчук Галина Вікторівна	10.05.2024р.	10.06.2024р.

РЕФЕРАТ

Кваліфікаційна робота бакалавра складається з розробки комп'ютерної гри в жанрі Soulslike. Пояснювальна записка містить 68 сторінок, 37 ілюстрацій та 13 таблиць.

Головною метою даної роботи є дослідження, відтворення і вдосконалення складних ігрових механік гри у жанрі «soulslike» з використанням ігрового рушія Unreal Engine.

В роботі детально розглянуто основні завдання та етапи проектування ігрового продукту. Проведено аналітичний огляд існуючих рішень. Представлено основні сценарії роботи програмного забезпечення, його структуру та архітектуру. Детально описано використані алгоритми та інструменти розробки. Додатково наведено результати проведення тестування системи.

КЛЮЧОВІ СЛОВА: ГРА, ШТУЧНИЙ ІНТЕЛЕКТ, ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР, SOULSLIKE.

					КН.КР.Б – 122 – 5 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Розробка комп'ютерної гри в жанрі Soulslike	Літ.	Арк.	Аркушів
Розроб.		Матяш О.В.						
Керівник		Марчук Г.В.					3	59
Рецензент		Петросян Р.В.				Житомирська політехніка, група КН-20-2		
Зав. каф.		Граф М.С.						

ABSTRACT

The bachelor's qualification work consists of developing a computer game in the Soulslike genre. The explanatory note for the thesis contains 68 pages, 37 illustrations, and 13 tables.

The main goal of this work is to research, replicate, and enhance complex game mechanics of a Soulslike game using the Unreal Engine game engine.

The thesis thoroughly examines the main tasks and stages of game product design. An analytical overview of existing solutions is conducted. The main software scenarios, its structure, and architecture are presented. Detailed descriptions of the algorithms and development tools used are provided. Additionally, the results of system testing are included.

KEYWORDS: GAME, ARTIFICIAL INTELLIGENCE, PERSONAL COMPUTER, SOULSLIKE.

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

ЗМІСТ	5
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП	7
РОЗДІЛ 1. ДОСЛІДЖЕННЯ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ У РОЗРОБЦІ SOULSLIKE ІГОР.....	10
1.1. Опис проекту та постановка задачі	10
1.2. Аналітичний огляд існуючих аналогів	12
1.3. Аналіз вимог і обґрунтування вибору засобів середовища розробки	18
Висновки до першого розділу.....	22
розділ 2. ПРОТОТИПУВАННЯ, ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ гри	23
2.1. Визначення варіантів використання та структури системи	23
2.2. Проектування та реалізація основної роботи системи.....	31
2.3. Проектування та реалізація важливих алгоритмів	41
2.4. Налаштування графічних компонентів гри.....	44
Висновки до другого розділу	51
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПЗ	52
3.1. Порядок встановлення та налаштування роботи	52
3.2. Тестування програмного продукту	55
Висновки до третього розділу	57
ВИСНОВКИ.....	59
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	61
ДОДАТКИ.....	64

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

RPG – рольова гра

ООП – об'єктно-орієнтоване програмування

NPC – неігровий персонаж

ШІ – штучний інтелект

ПК – персональний комп'ютер

ПЗ – програмне забезпечення

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

За останнє десятиліття відеоігри перетворилися на провідну індустріальну та економічну силу, культурний феномен і платформу для творчого експериментування. Сьогодні у всьому світі налічується близько 3,09 мільярда активних гравців, а мільярди доларів і годин, витрачених на розробку та гру у відеоігри, відображають їх величезний вплив на культуру сьогодні. Така популярність пояснюється тим, що сучасна та культурно-історична психологія людини наполягає на існуванні глибинного зв'язку між середовищем, у якому людина живе, та уявним світом, створеним комп'ютерними іграми. Фантастична реальність відкриває перед гравцями можливості для втілення різноманітних ролей, дозволяє безпосередньо проживати сюжет, управляти подіями та експериментувати з різними аспектами своєї ідентичності. [1, 3, 17]

Одним з найбільш популярних жанрів в ігровій індустрії є «soulslike». Цей жанр виник на основі серії ігор «Dark Souls» від японської студії FromSoftware, яка встановила високі стандарти якості та оригінальності для подібних проектів. Здебільшого «soulslike» ігри являють собою суміш жанрів RPG і action. Їх із загального ряду виділяє приголомшлива складність і особлива атмосфера. Завдяки популярності жанру, щодня на ринку з'являється все більше нових творінь, кожне із яких намагається відділитися від своїх попередників. Проте, шлях до успіху не завжди простий, адже перед розробниками лежить дуже непросте завдання: навіть у найвищої складності повинна бути якась межа, можливість для гравця нарешті вирішити задачу і відчувати належне задоволення від гри. [8]

В останні роки технологічний прогрес, зокрема в області графіки, штучного інтелекту та мережевих технологій, відкриває нові горизонти для розробників, дозволяючи створювати більш реалістичні ігрові світи, складніші взаємодії та глибші сюжетні лінії. Очевидно, що цей процес є важким і вимагає не тільки гарних технічних навичок, але й глибокого розуміння психології гравців, креативності та здатності адаптуватися до тенденцій, що постійно

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

змінюються. У зв'язку із цим дослідження процесу розробки відеоігор набуває все більшої актуальності, оскільки допомагає визначити найефективніші методи створення якісних ігор та виявити ключові фактори їх успіху.

Метою роботи є дослідження, відтворення і вдосконалення складних ігрових механік гри у жанрі «soulslike» з використанням ігрового рушія Unreal Engine.

Для досягнення поставленої мети потрібно вирішити наступні завдання:

- вивчити характерні риси та основні механіки ігор у жанрі «soulslike»;
- ознайомитися з можливостями та інструментами підходящих технологій для розробки ігор;
- створити базовий прототип гри з основними механіками жанру «soulslike»;
- розробити та оптимізувати бойові системи з використанням ігрового інтелекту;
- провести тестування та аналіз реалізованої гри.

Об'єктом дослідження є ігрові механіки та архітектура ігор у жанрі «soulslike».

Предметом дослідження є ознайомлення з методами і технологіями розробки складних ігрових механік, таких як бойові системи, штучний інтелект ворогів, дизайн рівнів у «soulslike» з використанням ігрового рушія Unreal Engine.

Практична значимість розробки полягає у виявленні ефективних методів та інструментів для розробки ігор у жанрі «soulslike», що можуть бути використані як індивідуальними розробниками, так і ігровими студіями. Це включає вдосконалення механік бойових систем, оптимізацію штучного інтелекту ворогів, дизайн рівнів та користувацьких інтерфейсів, що в результаті сприятиме підвищенню якості ігрового процесу та загального досвіду гравців.

Методи дослідження. У роботі застосовувались емпіричні методи дослідження (проведення експериментальних тестів і прототипування для

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

перевірки ефективності розроблених ігрових механік та архітектури у віртуальному середовищі) та теоретичні методи дослідження.

Технічні підходи та джерела інформації. У роботі було використано метод ітеративної розробки, який передбачає поступове вдосконалення прототипу на основі отриманих результатів тестування і зворотного зв'язку, застосовано принципи ООП. Інформаційна складова дослідження базується на актуальних наукових публікаціях з розробки відеоігор, технічній документації і матеріалах по Unreal Engine.

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ У РОЗРОБЦІ SOULSLIKE ІГОР

1.1. Опис проекту та постановка задачі

Жанр «soulslike» відомий своєю високою складністю, стратегічними боями, розвитком персонажів і заплутаними рівнями. Даний проект націлений на дослідження, відтворення та удосконалення складних ігрових механік та концепцій даного жанру за допомогою Unreal Engine. Основним завданням є реалізація бойової системи з впровадження ігрового інтелекту для створення прототипу гри. [8]

Програмний продукт повинен відповідати всім стандартам ігор жанру «soulslike». Для реалізації цього завдання в проєкт мають бути включені наступні пункти:

- вороги мають бути складними та вимагати від гравця уважності, терпіння та навичок;
- різні типи зброї та спорядження з різними властивостями та ефектами;
- обмежені точки збереження (наприклад, багаття);
- сповіщення та підказки: функція, яка дозволяє гравцям отримувати сповіщення про важливі події в грі, такі як знаходження вогнища, відкриття коротких шляхів, активація пасток тощо;
- темний, похмурий світ;
- використання витривалості для атак, ухилення та блокування;
- різноманітні предмети та можливість взаємодії з ними.
- процес гри повинен здійснюватися від третьої особи, тому камера має бути закріплена відповідним чином;

Важливість інтерфейсу в розробці гри важко переоцінити. Інтерфейс – це комплекс засобів, методів і правил взаємодії між гравцем та грою. Він може стати вирішальним фактором атмосфери та отриманого задоволення від гри, адже сьогодні конкуренція в цій індустрії надзвичайно висока. Було сформовано наступні вимоги до інтерфейсу:

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

- інтерфейс має використовувати сучасні стандарти дизайну, також враховувати тематику та настрій гри;
- кольори, шрифти, іконки та інші елементи мають підкреслювати характер світу гри, створюючи унікальну атмосферу;
- інтерфейс має бути простим та зрозумілим, не відволікати гравця від геймплею, неоднозначність компонентів має бути відсутня;
- мають використовуватися уніфіковані та стандартизовані елементи, що розміщуються в однакових місцях на різних екранах;
- важливим є використання контрастних кольорів, чітких шрифтів, відступів та ієрархії, щоб виділити важливі елементи, такі як текст, кнопки, сповіщення тощо;
- інтерфейс має пристосовуватися до різних розмірів екрану, орієнтації, роздільної здатності та пристроїв.

Для того щоб успішно виконати поставлені завдання необхідно пройти наступні етапи роботи:

1. Дослідження жанру та визначення вимог.
 - 1.1. Аналіз існуючих ігор жанру soulslike для визначення ключових механік та особливостей.
 - 1.2. Визначення вимог до геймплею, графіки та інтерфейсу користувача.
 - 1.3. Дослідження основних концепцій, механік та структур гри.
 - 1.4. Визначення основних засобів та інструментів для реалізації програмного продукту.
2. Розробка ігрових механік.
 - 2.1. Реалізація бойової системи з використанням витривалості для атак, ухилення та блокування.
 - 2.2. Реалізація різних типів зброї з унікальними властивостями та ефектами.
 - 2.3. Створення системи AI для ворогів, яка включає складну поведінку та реакції на дії гравця.

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

2.4. Тестування та коригування AI для забезпечення балансу та виклику для гравця.

2.5. Планування, розробка та оптимізація клієнтської частини застосунку.

3. Реалізація системи збереження та взаємодії з предметами.

3.1. Впровадження обмежених точок збереження (наприклад, багаття) з можливістю відпочинку та відновлення ресурсів.

3.2. Створення системи взаємодії з предметами.

3.3. Оптимізація системи після тестування для забезпечення її стабільності та надійності.

4. Створення та налаштування ігрового світу.

4.1. Наповнення світу різноманітними об'єктами.

4.2. Реалізація темного, похмурого світу з детальним оточенням та атмосферою.

4.3. Створення базового рівня

5. Тестування та налагодження.

5.1. Проведення тестування для виявлення та виправлення помилок.

5.2. Оптимізація продуктивності та покращення ігрового процесу.

По завершенню виконання усіх етапів визначеного плану буде представлено готову гру, що матиме всі ключові елементи притаманні жанру «soulslike» та забезпечуватиме захоплюючий ігровий досвід користувачів.

1.2. Аналітичний огляд існуючих аналогів

Огляд та аналіз аналогічних розробок є важливою складовою для створення успішного проекту. На ринку популярними стали «soulslike» ігри, які вирізняються високою складністю та унікальною атмосферою. Цей жанр приваблює як новачків, так і відданих прихильників. Гравці цінують ці ігри за несподіваність, глибокий геймплей та потужні емоції. «Soulslike» ігри не

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

просто розвиваються як самостійний жанр, а й встановлюють нові стандарти в індустрії відеоігор.

Під час аналізу аналогічних програмних продуктів було розглянуто кілька ігор, а саме: «Dark Souls» (серія ігор), «Sekiro: Shadows Die Twice», «Elden Ring» та «BloodBorne», – які пропонують схожі механіки бою та прогресії. Водночас кожна з них має унікальні риси, що робить їх особливими в цьому жанрі.

«Dark Souls» – це епічна серія фентезійних екшн-рольових ігор, що розроблена компанією FromSoftware та видана Bandai Namco Entertainment. Вона відрізняється від більшості інших рольових ігор своїм непередбачуваним і темним світом, де гравець стикається з викликами, що вимагають високої уваги, реакції та стратегічного мислення. Гра відмовляється від широких пояснень щодо своєї історії, механіки та мети, надаючи гравцеві можливість самостійно досліджувати і розкривати секрети. На рисунку 1.1 зображено фрагмент гри:



Рисунок 1.1. – Гра «DarkSouls II»

Це одна з найвпливовіших і популярних серій в історії відеоігор, що встановила нові стандарти для жанру екшн-рольових ігор та надихнула безліч

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

розробників. «Dark Souls» отримала високі оцінки як від критиків, так і від гравців, а також здобула численні нагороди і визнання.

«Sekiro: Shadows Die Twice» – належить до жанру «Soulslike», але має свої особливості і відмінності. Події у грі розгортаються в Японії епохи Сенгоку, де гравець керує ніндзя з протезованою рукою, який намагається врятувати свого молодого господаря і помститися за свою смерть. «Sekiro: Shadows Die Twice» поєднує в собі стелс, вертикальний рух, ближній бій, а також використання різноманітних інструментів.

Геймплей гри складається з трьох основних аспектів: таємниць, дослідження та бою. Користувач може використовувати протезовану руку, яка наділена різними функціями, такими як гачок, меч, вогнемет, парасолька та інші, для переміщення по рівнях, уникнення або убивства ворогів і застосування різноманітних тактик. Гравець також може використовувати здатність воскресати після смерті, з метою шокувати ворогів або втекти в безпечне місце. Однак, ця здатність має свою ціну, так як вона може призвести до поширення хвороби серед дружніх персонажів. На рисунку 1.2 зображено ігровий світ та самого персонажа.



Рисунок 1.2. – Гра «Sekiro: Shadows Die Twice»

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		14

Бій в «Sekiro: Shadows Die Twice» ґрунтується на системі поступки, яка вимірює баланс сил між персонажем і ворогом. Гравець повинен атакувати, блокувати і контратакувати, щоб зменшити поступку ворога і зламати його захист, що дозволить завдати смертельного удару.

Одним з найважливіших елементів геймплею є битви з босами, які є дуже складними і незабутніми. Боси мають різні фази, атаки, слабкі місця і сюрпризи, які вимагають від геймерів глибокого розуміння механіки гри і вміння використовувати свої ресурси і навички.

«Elden Ring» – це нова фентезійна гра від творців серії «Dark Souls», Hidetaka Miyazaki та George R. Martin. Гра відбувається в світі, який був розбитий на шматки внаслідок руйнування загадкового Елден Рингу. Гравець виступає в ролі Потьмареного, одного з багатьох вигнаних і забутих істот, які мають знайти своє місце в цьому хаотичному світі

«Elden Ring» є духовним наступником «Dark Souls», але з більш відкритим і різноманітним геймплеєм. Гра поєднує елементи екшену, рольової гри, пригоди та виживання. Гравець може досліджувати величезну карту, яка складається з різних локацій, таких як поля, ліси, гори, болота, пустелі, замки, печери та інше. Також можна взаємодіяти з різними персонажами, які мають свої історії, мотиви та завдання. Персонаж має можливість використовувати різні види зброї, броні, магії та предметів, щоб пристосуватися до різних ситуацій та ворогів.

Однією з ключових особливостей геймплею є елемент верхової їзди на коні, що дозволяє швидко пересуватися по віртуальному світу та виконувати атаки згори (Рис. 1.3). Гра також підтримує унікальний онлайн-режим, який дозволяє співпрацювати або змагатися з іншими гравцями в асинхронний спосіб. «Elden Ring» має високу ступінь складності та виклику, як і попередні ігри від FromSoftware, але також надає гравцям більше свободи та вибору.

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		15



Рисунок 1.3. – Верховна їзда у грі «Elden Ring»

«BloodBorne» – це захоплююча гра в стилі «Soulslike», яка розповідає про мисливця, який опиняється в місті Ярнам, де жителі страждають від кровної хвороби, яка перетворює їх на жахливих істот. Гравець повинен знайти джерело хвороби, борючись з ворогами. Гра відзначається високою складністю, атмосферою, звуковим дизайном та темами «Лавкрафта».

«BloodBorne» також є духовним наступником серії «Dark Souls», але з більш динамічним і агресивним геймплеем. Гра фокусується на стратегічному використанні зброї, вогнепальної, магії та предметів, а також на дослідженні різних локацій, взаємодії з персонажами і розгадці таємниць.

Однією з головних особливостей гри є механіка, яка дозволяє гравцеві відновлювати частину свого здоров'я, швидко контратакуючи після отримання удару. Це стимулює гравця до агресивного стилю гри і ризикованих дій. Гра також має механіку перетворення гравця на різних істот. Це дозволяє адаптуватися до різних ситуацій і використовувати різні комбінації атак, а також різні представлення персонажа, наприклад, вовк, кріль, ворона, кальмар тощо, отримуючи нові можливості і слабкості.

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

Нелінійна структура гри дозволяє гравцеві вибирати, куди йти і що робити, а також впливати на хід подій і закінчення. Велика кількість діалогів та виборів впливають на відносини з персонажами і на їхню долю. У грі є ковенанти – різні фракції і ідеології, до яких можна приєднатися, отримавши певні ті чи інші переваги та недоліки. Фрагмент гри «BloodBorne» наведено на рисунку 1.4.



Рисунок 1.4. – Гра «BloodBorne»

Поряд із позитивними сторонами даних ігор було виявлено і недоліки, а саме:

- хоча кожна гра пропонує глибокий та цікавий сюжет, час від часу можна зустріти незавершені або суперечливі деталі про історію, персонажів або світ;
- здавалося б, що кожна гра має свою унікальну атмосферу та стиль, однак кожне нове рішення зазвичай будується на повторюваності, та одноманітності локацій, ворогів або завдань;
- доволі часто зустрічаються ситуації, коли гравець зазнає несправедливої смерті або втрати ресурсів через помилки геймдизайну, баги або невдале балансування.

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

Для кращого порівняння було розроблено таблицю 1.1, яка враховує різні аспекти функціональності та можливостей кожної гри. Ця таблиця сприяє об'єктивному аналізу представлених аналогів порівняно з власною розробкою.

Таблиця 1.1

Порівняльна таблиця аналогів

Властивості	Dark Souls	Sekiro: Shadows Die Twice	Elden Ring	BloodBorne	Власна розробка
Фізичний рушій	+	+	+	-	+
Складність механік	-	+	+	+	+
Графічне оформлення	-	+	+	-	+
Взаємодія з ворогами	+	+	+	+	+
Реалістичність боїв	+	+	+	+	+
Відкрита карта	-	-	+	-	-

Хоча обмеженість ресурсів може суттєво вплинути на якість та можливості розробки, для власної гри було вдало виокремлено і поєднано основні елементи жанру, необхідні для задоволення ключових потреб користувачів та досягнення визначеної мети.

1.3. Аналіз вимог і обґрунтування вибору засобів середовища розробки

Розробка ігор – це комплекс дій, спрямованих на створення інтерактивних та захоплюючих віртуальних світів для різних пристроїв, як от консолей, комп'ютерів, смартфонів тощо. Вибір правильного середовища розробки є першочерговою потребою. Ним може бути, наприклад, спеціалізований ігровий рушій, програмна платформа, мова програмування чи інші технічні рішення. Для цього необхідно врахувати такі фактори, як: можливості середовища, вимоги гри, бюджет розробки тощо. Статистика показує, що найпопулярнішими середовищами розробки ігор в усьому світі є Unity та Unreal Engine та Godot Engine. Існує багато інших середовищ, та все ж

основними є згадані раніше, тому засоби розробки ігор мають базуватися на них.

Unity – це ігровий рушій, який був заснований у 2004 році з метою зробити розробку ігор універсально та доступно. Сьогодні він відомий як "зроби будь-яку гру" рушій і ідеально підходить для інди-розробників. Unity використовує C# як основну мову програмування, яка повинна бути досить знайомою для всіх, хто займається розробкою ігор. В свою чергу Unity надає простий та інтуїтивний інтерфейс, який дозволяє швидко та легко створювати прототипи та експериментувати з ідеями. Для нього наявна велика кількість асетів, які можна завантажити з Asset Store, що допомагає збільшити продуктивність та якість розробки. Однак він також має свої недоліки, такі як менша продуктивність, менша якість графіки, менша підтримка від спільноти та менша гнучкість в налаштуванні рушія [11].

Unreal Engine – це ігровий рушій, який був розроблений засновником Epic Games Тімом Свіні в 1998 році. Він запустився разом зі своєю дебютною грою "Unreal" і був унікальним тим, що дозволяв гравцям модифікувати гру вперше. Сьогодні Unreal Engine асоціюється з "кращою графікою" і пропонує великостудійну AAA-якість іграм, які його використовують. Unreal Engine використовує C++, але при побудові коду гри використовується комбінація Blueprint (власної мови, унікальної для продуктів Epic) та C++. Двигун має потужний та гнучкий інтерфейс, який дозволяє створювати вражаючі та реалістичні світи, складну фізику, детальну анімацію, багатий звук та інші елементи геймплею. Unreal Engine також має велику підтримку від спільноти та безкоштовний доступ до всього вихідного коду рушія. Однак, також він має свої недоліки, такі як висока крива навчання, складність оптимізації, високі вимоги до системи та відсутність асетів для завантаження [2].

Godot Engine – це багатофункціональний кросплатформенний ігровий двигун для створення 2D і 3D ігор за допомогою єдиного інтерфейсу. Він надає повний набір загальних інструментів, щоб користувачі могли зосередитися на створенні ігор. Програмні продукти можна експортувати одним клацанням

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

миші на низку платформ, включаючи основні платформи настільних ПК (Linux, macOS, Windows), мобільні платформи (Android, iOS), а також веб-платформи та консолі. Розробка Godot є повністю незалежною та керованою спільнотою, що дає користувачам можливість допомогти сформувати свій двигун відповідно до їхніх очікувань. Він підтримується некомерційною організацією Godot Foundation . Для програмування можна використовувати як GDScript так і C++ [16].

Загальне порівняння Unreal Engine, Unity та Godot Engine за допомогою оцінок від 1 до 5, де 5 – дуже добре, наведено в таблиці 1.2.

Таблиця 1.2

Порівняльна таблиця ігрових двигунів

Характеристика	Unreal Engine	Unity	Godot Engine
Графіка	5	4	3
Фізика	5	3	3
Анімація	5	4	3
Легкість вивчення	4	4	5
Розширюваність функціоналу	5	4	4
Спільнота підтримки	5	5	3
Підтримка мов програмування	5	4	4
Інтеграція з сторонніми сервісами та SDK	5	4	4
Інструменти для візуалізації	5	4	3

Порівняючи дані ігрові двигуни, можна зробити висновок, що усі рушії мають свої переваги та недоліки, і вибір засобів проектування залежить від багатьох факторів, таких як: тип гри, рівень досвіду, бюджет, цільова платформа тощо. Для гри в жанрі «soulslike», було обрано Unreal Engine, тому що:

1. Unreal Engine має кращу графіку та освітлення, що допомагає створити темну та атмосферну естетику, яка характерна для «soulslike» ігор.

Unity також може створювати гарну графіку, але вимагає більше роботи та додаткових асетів, що може збільшити вартість розробки.

2. Unreal Engine має більш розвинуту фізику та анімацію, що допомагає створити реалістичну та відчутну бойову систему, яка є ключовою для souls like ігор. Unity також може похвалитися даними можливостями, але це вимагає більше кодування та налаштування, що може збільшити складність розробки.

3. Unreal Engine має Blueprint, який дозволяє створювати логіку гри без написання коду, що може спростити розробку та дозволити більше експериментувати з ідеями. Unity та Godot Engine також мають подібні засоби, але вони не є вбудованими в рушій, потребують додаткового встановлення та не є повноцінно вдосконаленими.

Blueprint в Unreal Engine – це повна система скриптингу геймплея, яка базується на концепції використання вузлового інтерфейсу для створення ігрових елементів зсередини редактора Unreal. Як і багато загальних мов скриптингу, вона використовується для визначення об'єктно-орієнтованих класів або об'єктів в двигуні. Blueprint дозволяє використовувати майже повний спектр концепцій і інструментів, які зазвичай доступні тільки програмістам. Blueprint працює за допомогою графів вузлів для різних цілей - конструювання об'єктів, окремих функцій і загальних подій геймплею – які є специфічними для кожного екземпляра Blueprint, щоб реалізувати поведінку та іншу функціональність. Найбільш поширеними типами є LevelBlueprint і Blueprint Class [13].

Level Blueprint виконує ту ж роль, що і Kismet в Unreal Engine 3, і має ті ж можливості. Кожен рівень має свій власний Level Blueprint, і він може посилатися на акторів у рівні, керувати кінематографічними елементами за допомогою Matinee Actor, і керувати такими речами, як рівень потокової передачі, контрольні точки, і інші системи, пов'язані з рівнем. Level Blueprint також може взаємодіяти з Blueprint Class, розміщеними на рівні, наприклад,

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

читати, впливати на будь-які змінні або запускати користувацькі події, які вони можуть містити [20].

Blueprint Class – це повноцінний клас, який створюється за допомогою Blueprint, і може бути розміщений в рівні або використовуватися як базовий клас для інших Blueprint. Blueprint Class може містити змінні, функції, події, інтерфейси, компоненти та інше. Blueprint Class може бути будь-яким типом актора, від простого статичного меша до складного персонажа з анімацією, фізикою та логікою [12].

Висновки до першого розділу

Аналіз теоретичних та функціональних вимог предметної області дав можливість визначити основні ідеї та етапи реалізації проекту, а саме гри у жанрі «soulslike».

Було виконано порівняння розробок, що вже існують на ринку, як от: «Dark Souls», «Sekiro: Shadows Die Twice», «Elden Ring», «BloodBorne». Це дозволило виділити найбільш важливі функції та особливості, на які варто звернути увагу при створенні гри. Особливо варто зосередити увагу на тому, що:

- вороги мають бути складними;
- у грі мають бути різні типи зброї;
- кількість точок збереження має бути обмеженою;
- світ має бути темним і похмурим;
- процес гри повинен здійснюватися від третьої особи.

Також було обгрунтовано вибір засобу реалізації програмного продукту, ним став рушій Unreal Engine, що дозволяє розробляти безліч різноманітних, масштабних та унікальних комп'ютерних ігор.

РОЗДІЛ 2. ПРОТОТИПУВАННЯ, ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ГРИ

2.1. Визначення варіантів використання та структури системи

Для того щоб зрозуміти як функціонування програмного комплексу узгоджено з різноманітними сценаріями взаємодії користувачів з грою було створено діаграму варіантів використання. Цей тип діаграм є важливим інструментом для ідентифікації та візуалізації можливостей взаємодії різних типів гравців із готовим програмним продуктом, без обов'язкового зазначення порядку виконання дій чи конкретного функціоналу [2424].

Діаграму варіантів використання гри представлено на рисунку 2.1.

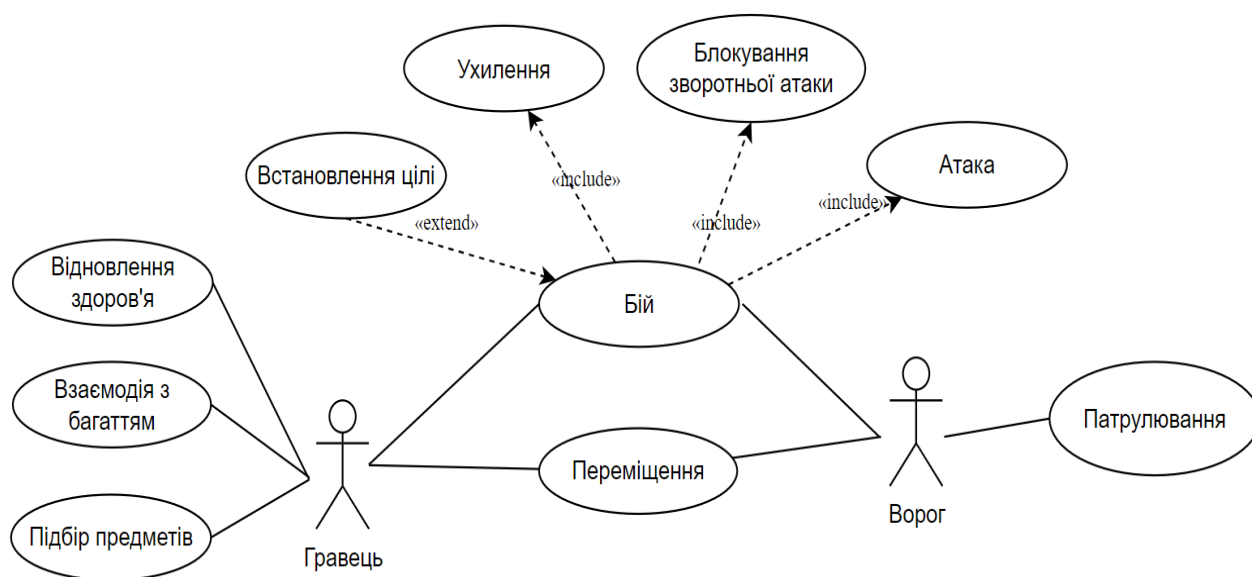


Рисунок 2.1. – Діаграма варіантів використання

Дане зображення вказує на те, що у поточній версії гри передбачено лише 2 типи акторів: гравець та ворог. Ворогом керує ігровий інтелект. Це дозволяє створити більш динамічну та захоплюючу ігрову атмосферу, де гравець стикається з непередбачуваними викликами. Кожен актор має свої унікальні можливості та поведінку. У таблицях 2.1 – 2.9 подано опис сценаріїв використання системи.

Таблиця 2.1

Прецедент «Відновлення здоров'я»

Назва	Відновлення здоров'я
Актори	Гравець
Опис сценарію	Гравець відновлює здоров'я свого персонажа для продовження гри. Для цього спершу персонаж має втратити здоров'я.

Таблиця 2.2

Прецедент «Підбір предметів»

Назва	Підбір предметів
Актори	Гравець
Опис сценарію	Гравець переміщається світом гри та знаходить предмет, який можна підібрати. Гравець наближається до предмета, який знаходиться на землі або в іншому доступному місці. Гравець натискає кнопку взаємодії, коли знаходиться достатньо близько до предмета. Сценарій завершується, коли гравець підібрав предмет і відбулась необхідна подія.

Таблиця 2.3

Прецедент «Переміщення»

Назва	Переміщення
Актори	Гравець, Ворог
Опис сценарію	Актор рухається по світу гри, досліджуючи локації та взаємодіючи з оточуючим середовищем. Для цього використовуються використовує клавіші керування або геймпад для переміщення, якщо це гравець, якщо це ворог – то сітка NavMesh. Сценарій закінчується, коли актор успішно досягнув потрібної точки.

Таблиця 2.4

Прецедент «Взаємодія з багаттям»

Назва	Взаємодія з багаттям
Актори	Гравець
Опис сценарію	Гравець підходить до багаття, яке є точкою збереження та відновлення ресурсів у грі. Натискає кнопку взаємодії. Відбувається збереження.

Таблиця 2.5

Прецедент «Патрулювання»

Назва	Патрулювання
Актори	Ворог
Опис сценарію	Ворог починає своє патрулювання відповідно до заданого маршруту або алгоритму (сітки NavMesh). Під час патрулювання ворог може реагувати на гравця. Якщо ворог виявляє гравця під час патрулювання, він починає преслідування або атаку. В залежності від ігрового інтелекту, ворог може реагувати на атаку, виконувати захисні дії тощо. Сценарій завершується, коли ворог повертається до початкової точки патрулювання або досягає кінцевої точки маршруту.

Таблиця 2.6

Прецедент «Встановлення цілі»

Назва	Встановлення цілі
Актори	Гравець, Ворог
Опис сценарію	Гравець знаходить ворога, та на певній відстані натискає клавішу для встановлення цілі. Сценарій завершується коли ціль встановлена.

Таблиця 2.7

Прецедент «Ухилення»

Назва	Ухилення
Актори	Гравець, Ворог
Опис сценарію	Гравець або ворог знаходяться у ситуації, де потрібно уникнути атаки або небезпеки. Ворог атакує гравця (чи навпаки) або наближається для нападу. Актор сприймає або виявляє загрозу, що наближається. Відбувається активація процес ухилення для уникнення атаки. Якщо ухилення успішне, атака ворога чи гравця не наносить шкоди. Сценарій завершується після виконання ухилення та оцінки наслідків цієї дії.

Таблиця 2.8

Прецедент «Блокування зворотньої атаки»

Назва	Блокування зворотньої атаки
Актори	Гравець, Ворог
Опис сценарію	Гравець або ворог готуються до атаки або реакції на атаку противника. Ворог атакує гравця або гравець спостерігає за атакою ворога. Гравець вибирає опцію блокування, натисканням відповідної клавіші або дією на геймпаді. Гравець виконує блокування, виконуючи захисну дію. Ворог також може блокувати атаку гравця або приймати захисну позицію. Якщо блокування успішне, атака ворога не наносить шкоди гравцеві або ворогу. Сценарій завершується після виконання блокування та оцінки наслідків цієї дії.

Прецедент «Атака»

Назва	Атака
Актори	Гравець, Ворог
Опис сценарію	Гравець або ворог готуються до виконання атаки. Гравець обирає ворога, якого він хоче атакувати, або ворог розпізнає гравця як ціль для своєї атаки. Гравець або ворог вибирає тип атаки і виконує її. Ворог або гравець отримує ушкодження або інші наслідки від атаки. В залежності від результатів атаки, сценарій може мати різні наслідки, такі як зниження здоров'я, знищення або відступ ворога, переміщення в іншу фазу бою тощо.

На жаль, даний тип діаграм не надає детального та комплексного представлення внутрішньої логіки процесів і дій, а слугує лише для високорівневого представлення взаємодії користувачів з системою. Щоб побачити як один процес впливає на інший – варто застосовувати діаграми активності. Діаграми активності мають кілька переваг над діаграмами використання, особливо в контексті детального моделювання і аналізу процесів [22].

Важливим процесом у роботі гри є оновлення поведінки ігрового інтелекту. Аби детально проаналізувати і впорядкувати його, було створено діаграму активності (Додаток А. Рис. А.1). Ця діаграма дозволяє візуально відобразити всі етапи оновлення поведінки ІІІ, починаючи від збору даних про поточну поведінку NPC, через етапи аналізу і вдосконалення алгоритмів, до впровадження нових моделей у гри. Завдяки діаграмі активності можна чітко побачити послідовність дій, визначити ключові точки прийняття рішень і паралельні процеси, що значно полегшує управління та оптимізацію роботи ігрового інтелекту.

Оновлення поведінки – початковий крок, який вказує на початок процесу оновлення ігрового інтелекту. Якщо він не проходить валідацію – то потік переходить до наступного кроку. Якщо проходить, то стейт визначається як

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

"Nothing" і потік завершується. Наступні кроки передбачають схожі дії, однак з іншими параметрами перевірки умов. Цей ітеративний процес оновлення ігрового інтелекту продовжується до досягнення відповідних критеріїв та виконання всіх необхідних завдань.

Якщо брати до уваги структуру системи, то чітка організація її файлів сприяє легкому знаходженню необхідних ресурсів, зменшує ймовірність конфліктів при редагуванні файлів і сприяє загальній ефективності розробки програмного забезпечення. Дотримання організованої структури файлів включає стандартизацію іменування файлів, групування за функціональним призначенням або модулями, а також документацію структури каталогів.

На рисунку 2.2 зображено основні каталоги проекту.

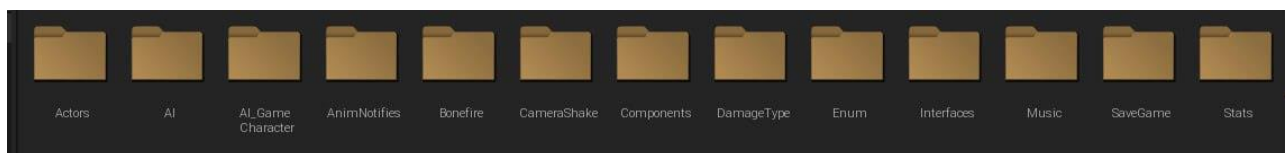


Рисунок 2.2. – Каталоги проекту

Опис важливих дирикторій наведено нижче:

1. Actors – містить об'єкти або сутності, які можна розміщувати на рівні гри і взаємодіяти з ними. Вони включають в себе такі речі, як об'єкти оточення (наприклад, багаття), а також предмети, що можуть бути використані, наприклад, зброя. Актори можуть мати власну поведінку, властивості та можливості, які впливають на ігровий процес і взаємодію з гравцем.

2. AI_GameCharacter – є каталогом, що включає класи, відповідальні за імплементацію штучного інтелекту персонажів у грі, їх поведінку та використання дерев рішень. Ці класи включають в себе рішення про рух, взаємодію з оточенням, вибір дій в залежності від умов та інші аспекти.

3. AnimNotifies – містить методи налаштування подій, які повинні відбуватися в конкретні моменти під час відтворення анімації в іграх. Наприклад, вони можуть використовуватися для створення колізій меча під час атаки або активації системи бою. Ці події інтегруються безпосередньо в

анімаційний потік і дозволяють точно синхронізувати різноманітні ігрові дії з рухами персонажів, що підвищує реалізм та іммерсивність геймплею.

4. **Components** – включає об'єкти, які можуть бути додані акторами як підоб'єкти в ігровому двигуні. Вони використовуються для реалізації різноманітних функціональних аспектів гри, таких як обробка статистики, обробка колізій, реалізація спеціальних ефектів та багато іншого. Компоненти дозволяють структурувати код актора, розділяючи різні аспекти його поведінки і функціональності на окремі модулі, що спрощує розробку, модифікацію та підтримку гри.

5. **Enum** – містить файли з наборами можливих значень, які можуть приймати змінні даного типу. В ігровій розробці **enum** використовується для визначення різних станів або варіантів, таких як типи атак, рівні навички персонажів, типи зброї тощо. Використання **enum** дозволяє чітко визначити і назвати можливі варіанти для певного аспекту гри.

6. **Interfaces** – містить класи інтерфейсів. Інтерфейси є корисним інструментом для того, щоб забезпечити, що набір (потенційно) непов'язаних класів може реалізувати загальний набір функцій без необхідності спільного предка. Це дозволяє структурувати програмний код, забезпечуючи єдність інтерфейсів для різних класів, які мають виконувати схожі операції чи мати спільну функціональність. Використання інтерфейсів сприяє зменшенню залежностей між класами і полегшує їхню заміну або розширення в майбутньому.

Однією з ключових вимог при проектуванні будь-якої системи є створення зрозумілої та легко розширюваної архітектури. Для того щоб гарантувати ефективний та результативний процес розробки було використано патерни проектування. Патерни представляють собою перевірені практикою шаблонні рішення, які сприяють створенню надійних, гнучких і масштабованих програмних систем. Вони відіграють важливу роль, допомагаючи проектувати програми більш організовано. Крім цього патерни також роблять код легшим для повторного використання та спрощують обслуговування системи.

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		29

Одним з таких став патерн «State». Цей шаблон проектування дозволяє об'єктам змінювати свою поведінку залежно від їх стану, створюючи враження зміни класу. Головна ідея патерну в тому, що програма може перебувати в одному з декількох станів, які постійно змінюються. Ці стани та переходи між ними заздалегідь визначені й обмежені. У різних станах програма може реагувати по-різному на одні й ті самі події. [7]

Приклад реалізації патерну наведено на рисунку 2.3.

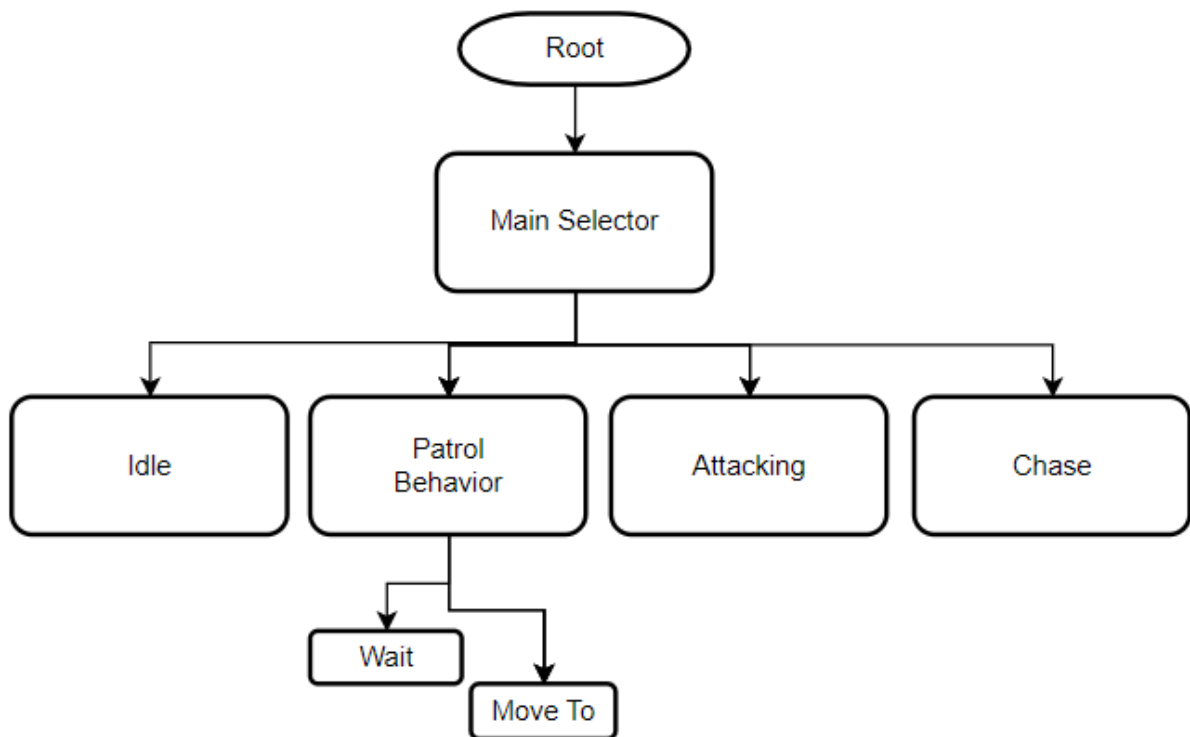


Рисунок 2.3. – Реалізація патерну «State»

MainSelector виконує роль контексту, тобто об'єкту, який має внутрішній стан і взаємодіє з об'єктами станів. Контекст зберігає посилання на об'єкт стану та делегує йому частину роботи, яка залежить від станів. Він працює з цим об'єктом через загальний інтерфейс станів. Контекст повинен мати метод для присвоєння йому нового об'єкта-стану. Вузли, які пов'язані з MainSelector є інтерфейсами станів, і вони визначають методи, що відповідають різним станам. Блоки, які є дочірніми до вузлів – і є конкретними станами. Вони реалізують поведінки, пов'язані з певним станом контексту. Іноді доводиться створювати цілі ієрархії класів станів, щоб узагальнити дублюючий код. В даному випадку конкретні стани відповідають за таку поведінку як рух,

патрулювання, очікування, атака тощо. На зміну стану впливає ряд чинників, як от: наявність гравця в полі зору, відстань між ворогом та гравцем тощо. Наприклад, якщо ворог бачить гравця, стан може змінитися з "патрулювання" на "атака". Подібним чином, якщо гравець зникає з поля зору, ворог може повернутися до "патрулювання". Такий підхід полегшує розширення функціональності та підтримку коду, забезпечуючи модульність і масштабованість системи.

2.2. Проектування та реалізація основної роботи системи

Процес проектування та реалізації основної роботи системи включає в себе розробку архітектури гри, визначення її основних компонентів і взаємодій між ними, а також реалізацію всіх необхідних ігрових механік. За допомогою візуального скриптового інструмента Blueprint було створено інтуїтивно зрозумілий інтерфейс для налаштування ігрової логіки без необхідності написання коду. Це дозволило прискорити процес розробки, забезпечуючи можливість легко вносити зміни та додавати нові функціональні можливості, що сприяє більш гнучкому та ефективному створенню кінцевого продукту.

Одним з таких сценаріїв стала обробка подій атаки у грі. На рисунку 2.4 зображено схему, яка демонструє логіку бойових дій.

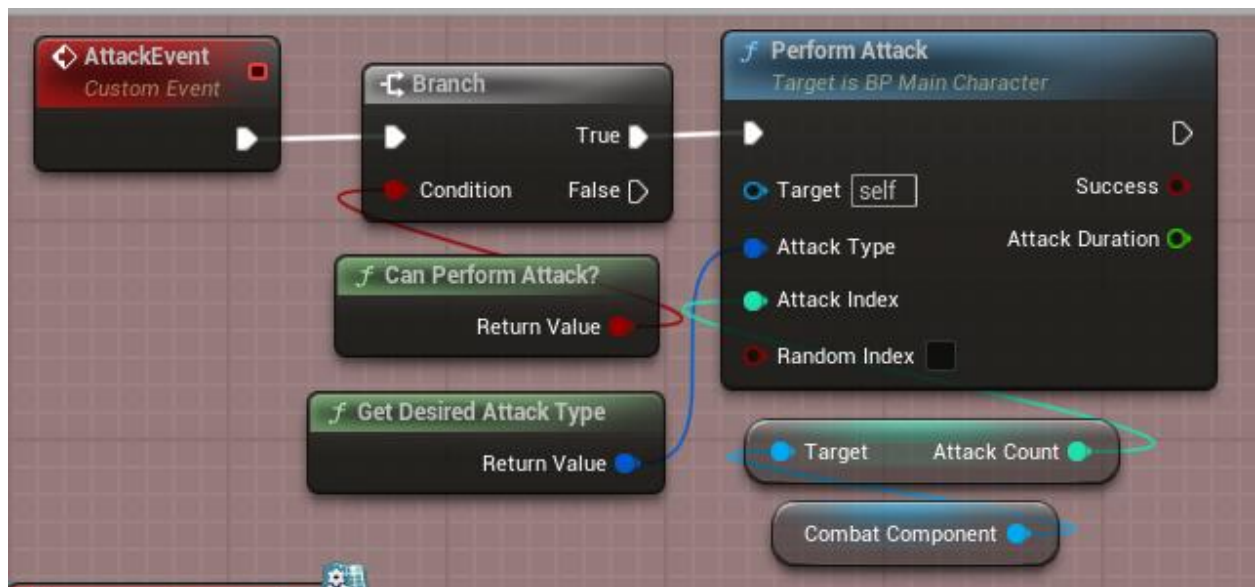


Рисунок 2.4. – Логіка бойових дій

За запуск та обробку всіх можливих атак та їх послідовності для різних комбінацій відповідає подія AttackEvent. Після відпрацювання AttackEvent відбувається перевірка умов для виконання атаки, таких як наявність достатньої витривалості та перебування персонажа у відповідному стані, за що відповідає вузол CanPerformAttack. У разі успішного проходження цих перевірок, гравець може атакувати. Процес атаки вимагає від системи керування витривалості персонажа, зменшуючи її на певну величину за кожен удар. Далі викликається функція "Perform Attack", яка ініціює атаку, враховуючи її тип, а також індекс. Відбувається перевірка через вузол "Branch", який перевіряє, чи бойовий режим увімкнений за допомогою функції "Is Combat Enabled?". Якщо бойовий режим увімкнено, сценарій продовжує потік виконання. Наступний етап включає отримання анімацій для атаки за допомогою функції. Відповідно через "Get Main Weapon" ми отримуємо зброю яку зараз використовуємо та передаємо цю інформацію в "Get Action Montages" де також приймаємо тип атаки і далі записуємо в локальну змінну "L Attack Montages". З використанням цих даних програється відповідна анімація атаки. Вузол "Select" визначає кількості послідовних атак, та відповідно до цього підбирає анімацію для програвання, записуючи її в локальну змінну, яку далі буде провалідовано. Визначається стан як "Attacking" та дія як тип атаки. Отримуючи персонажа відбувається програвання відповідної до нього анімації, також записується тривалість цієї атаки у відповідну змінну. Отримуючи кількість послідовних атак, можна визначити індекс відповідної анімації, та якщо набирається необхідна кількість то перевіряємо її і завершуємо виконання потоку з подальшою передачею інформації (Додаток Б. Рис. Б.1–Б.2) [14].

Для кращого розуміння виконання процесу було створено діаграму послідовності (Рис. 2.5). Даний тип діаграм є чудовим інструментом для детального опису операції що виконуються. Вони фіксують взаємодію між об'єктами в контексті співпраці. Діаграми послідовностей зосереджені на часі та показують порядок взаємодії, використовуючи вертикальну вісь діаграми для представлення часу, щоб пояснити які повідомлення надсилаються та коли [23].

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		32

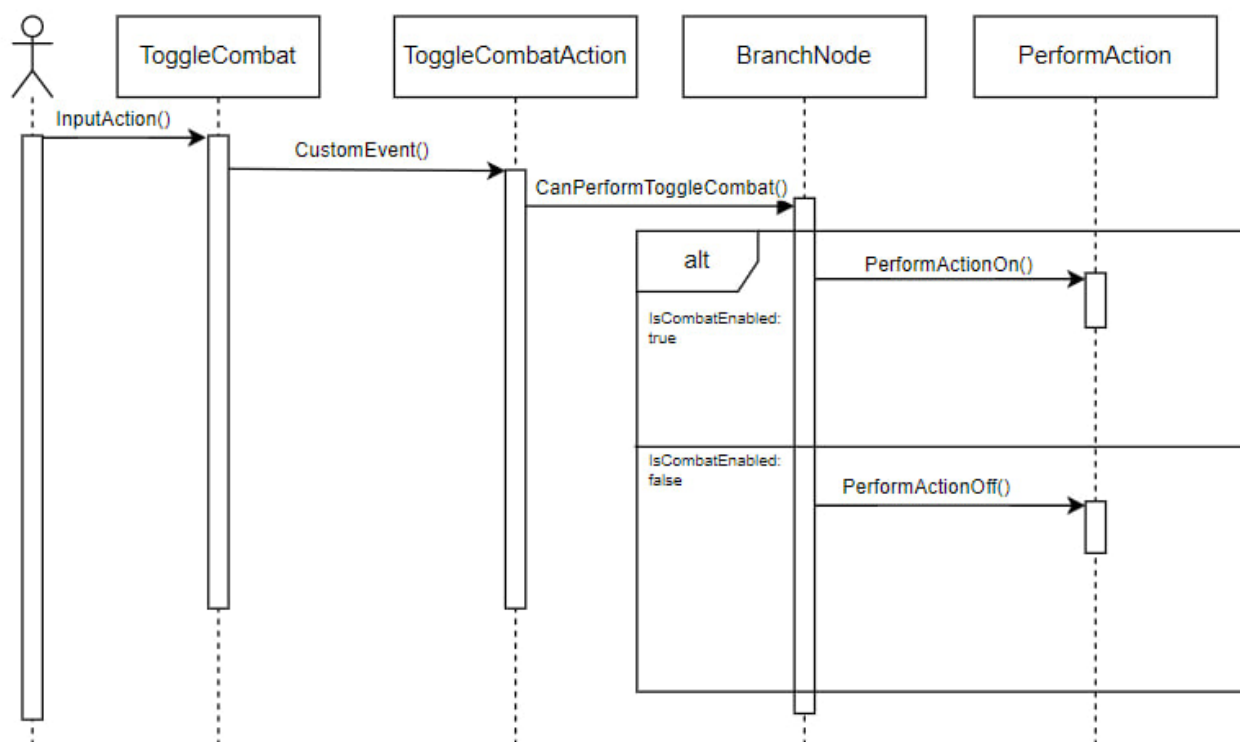


Рисунок 2.5. – Діаграма послідовності логіки бойових дій

Ваговою частиною бойової системи в іграх жанру soulslike є ухилення (Рис.2.6).

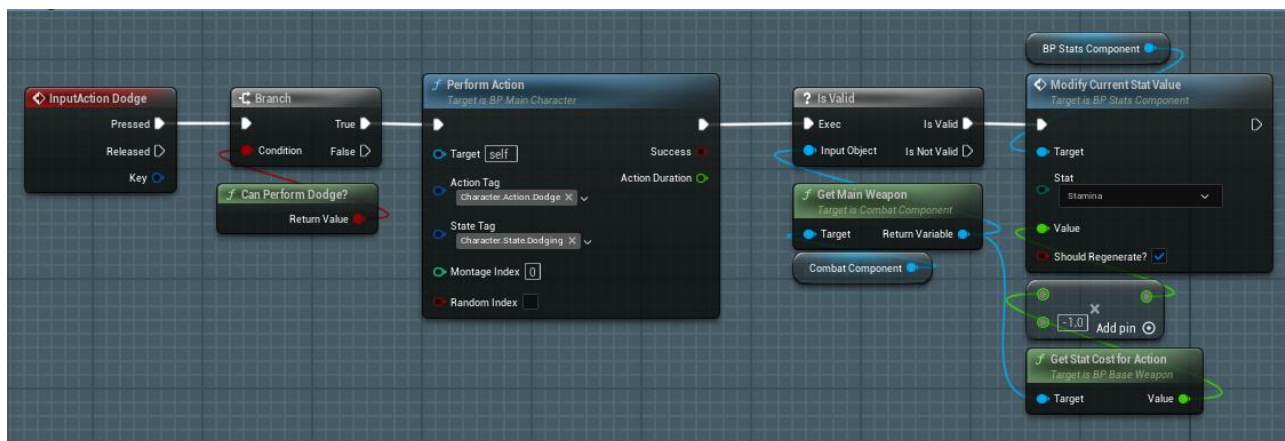


Рисунок 2.6. – Логіка роботи «Ухилення»

Сценарій починається з події "InputAction Dodge", яка активується при натисканні відповідної клавіші гравцем. Після цього відбувається перевірка через вузол "Branch", що перевіряє умову "Can Perform Dodge?". Якщо ця умова виконується, тобто персонаж може ухилитися, сценарій продовжується.

Наступний етап включає виконання дії ухилення через вузол "Perform Action". Тут задається дія "Dodging" для основного персонажа, з відповідним тегом дії та станом. Вказуються також індекси монтажу анімації для ухилення.

Після цього сценарій перевіряє валідність дії через вузол "IsValid". Якщо дія валідна, відбувається модифікація поточного значення витривалості персонажа через вузол "Modify Current Stat Value". Витривалість зменшується на величину, яка визначається вартістю ухилення, отриманою через вузол "Get Stat Cost for Action". Цей вузол повертає вартість витривалості для дії ухилення, яка вираховується з загальної витривалості персонажа.

Таким чином, представлений сценарій забезпечує виконання дії ухилення, враховуючи можливість її виконання та управління витривалістю персонажа, що є критично важливим для бойової системи в іграх жанру Soulslike. Це дозволяє створити більш реалістичний ігровий процес, де гравець повинен стратегічно використовувати свої ресурси для уникнення ворожих атак.

Для оптимізації постійного виявлення колізій об'єктів використовуються різні алгоритми. Це дозволяє уникнути промальовування колізій об'єкту на постійній основі та відмовитися від зайвих обчислень. У даному проекті для колізій зброї використовується функція, яка створює сфери від StartSocket до EndSocket зброї (Рис. 2.7-2.8).

Ми задаємо радіус цих сфер і перевіряємо, чи стикаються вони з об'єктами. При виявленні колізії об'єкт записується, що фіксує факт зіткнення.

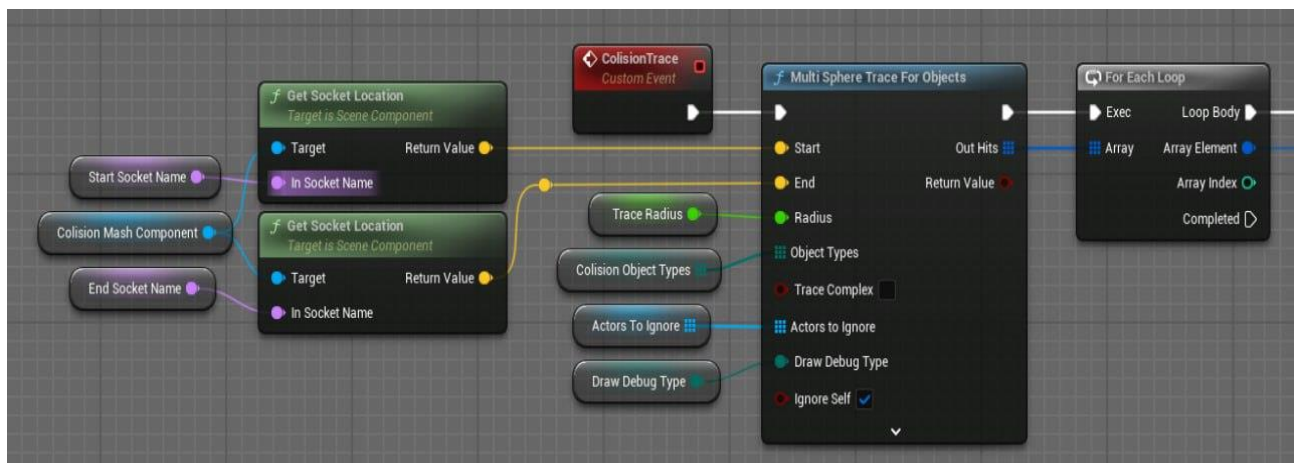


Рисунок 2.7. – Collision trace

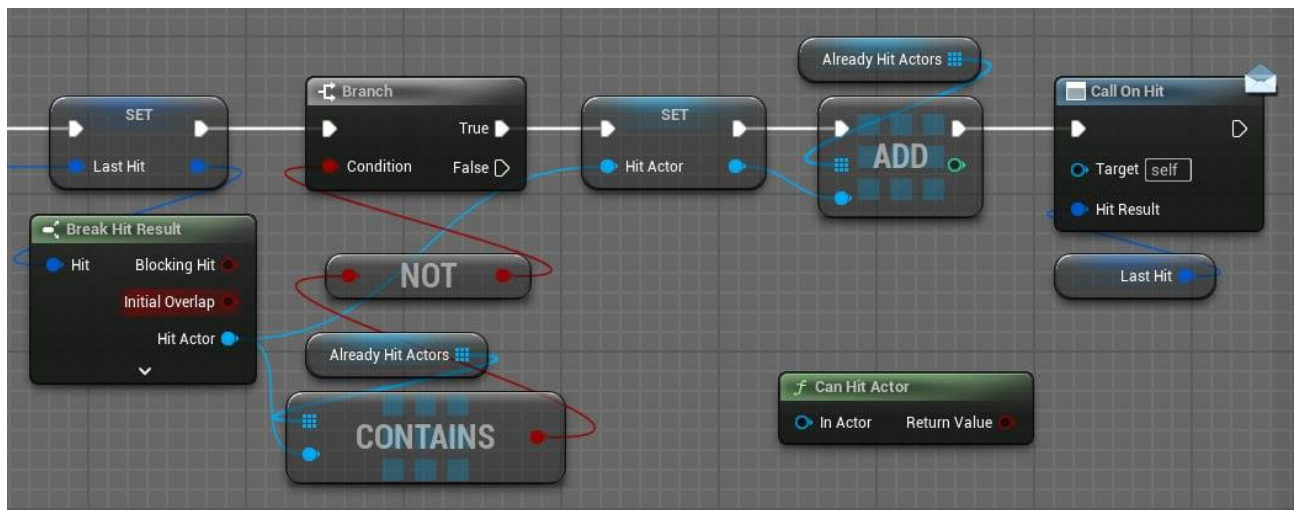


Рисунок 2.8. – Collision trace (продовження)

Для кращого розуміння, як працює наш персонаж в UnrealEngine, можна детальніше розглянути структуру його коду на прикладі нашого головного героя. Це дозволить зрозуміти, як визначаються і реалізуються основні функції та властивості.

Лістинінг коду наведено нижче:

```
public:
    FORCEINLINE class USkeletalMeshComponent* GetMesh() const { return Mesh; }
    static ENGINE_API FName MeshComponentName;

    template <class T>
    FORCEINLINE_DEBUGGABLE T* GetCharacterMovement() const
    {
        return CastChecked<T>(CharacterMovement, ECastCheckedType::NullAllowed);
    }
    FORCEINLINE UCharacterMovementComponent* GetCharacterMovement() const { return
    CharacterMovement; }

    static ENGINE_API FName CharacterMovementComponentName;

    FORCEINLINE class UCapsuleComponent* GetCapsuleComponent() const { return
    CapsuleComponent; }

    static ENGINE_API FName CapsuleComponentName;

#ifdef WITH_EDITORONLY_DATA
    class UArrowComponent* GetArrowComponent() const { return ArrowComponent; }
#endif

    ENGINE_API virtual void SetBase(UPrimitiveComponent* NewBase, const FName
    BoneName = NAME_None, bool bNotifyActor=true);

    UFUNCTION(BlueprintCallable, Category=Character)
    ENGINE_API virtual void CacheInitialMeshOffset(FVector MeshRelativeLocation,
    FRotator MeshRelativeRotation);

protected:
    UPROPERTY()
    struct FBasedMovementInfo BasedMovement;
```

```

UPROPERTY(ReplicatedUsing=OnRep_ReplicatedBasedMovement)
struct FBasedMovementInfo ReplicatedBasedMovement;

UPROPERTY(Replicated)
float AnimRootMotionTranslationScale;

public:
    UFUNCTION()
    ENGINE_API virtual void OnRep_ReplicatedBasedMovement();
    ENGINE_API virtual void SetReplicateMovement(bool bInReplicateMovement)
override;

protected:
    UPROPERTY()
    FVector BaseTranslationOffset;

    UPROPERTY()
    FQuat BaseRotationOffset;

    ENGINE_API virtual void BaseChange();

    UPROPERTY(Replicated)
    float ReplicatedServerLastTransformUpdateTimeStamp;

    UPROPERTY(ReplicatedUsing=OnRep_ReplayLastTransformUpdateTimeStamp)
    float ReplayLastTransformUpdateTimeStamp;

    UPROPERTY(Replicated)
    uint8 ReplicatedMovementMode;

    UPROPERTY(Replicated)
    FVector_NetQuantizeNormal ReplicatedGravityDirection;

    UPROPERTY()
    bool bInBaseReplication;

    FVector_NetQuantizeNormal PreNetReceivedGravityDirection;

```

Цей код представляє клас персонажа в Unreal Engine, який містить різні компоненти і методи для роботи з рухом та реплікацією даних.

Основні компоненти:

Mesh: Компонент сітки (USkeletalMeshComponent), який представляє модель персонажа. Метод GetMesh повертає вказівник на цей компонент, а MeshComponentName зберігає його ім'я.

CharacterMovement: Компонент руху персонажа (UCharacterMovementComponent). Існує два методи GetCharacterMovement: один шаблонний для перевірки типу, інший - звичайний. Ім'я компонента зберігається в CharacterMovementComponentName.

CapsuleComponent: Компонент капсули (UCapsuleComponent), що використовується для колізій. Метод GetCapsuleComponent повертає вказівник

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

на цей компонент, а його ім'я зберігається в CapsuleComponentName. Налаштування капсульного компонента можна побачити на рисунку 2.9

ArrowComponent: Компонент стрілки (UArrowComponent), доступний лише в редакторі. Метод GetArrowComponent повертає вказівник на цей компонент.

Методи:

SetBase: Встановлює новий базис для персонажа, на якому він стоїть. Використовується для того, щоб персонаж міг слідувати за динамічними об'єктами.

CacheInitialMeshOffset: Кешує початкове зміщення сітки від капсули для згладжування руху.

OnRep_ReplicatedBasedMovement: Метод для обробки реплікації відносного руху.

SetReplicateMovement: Встановлює, чи буде рух персонажа реплікуватися на клієнтські машини.

BaseChange: Подія, що викликається після зміни базису актора.

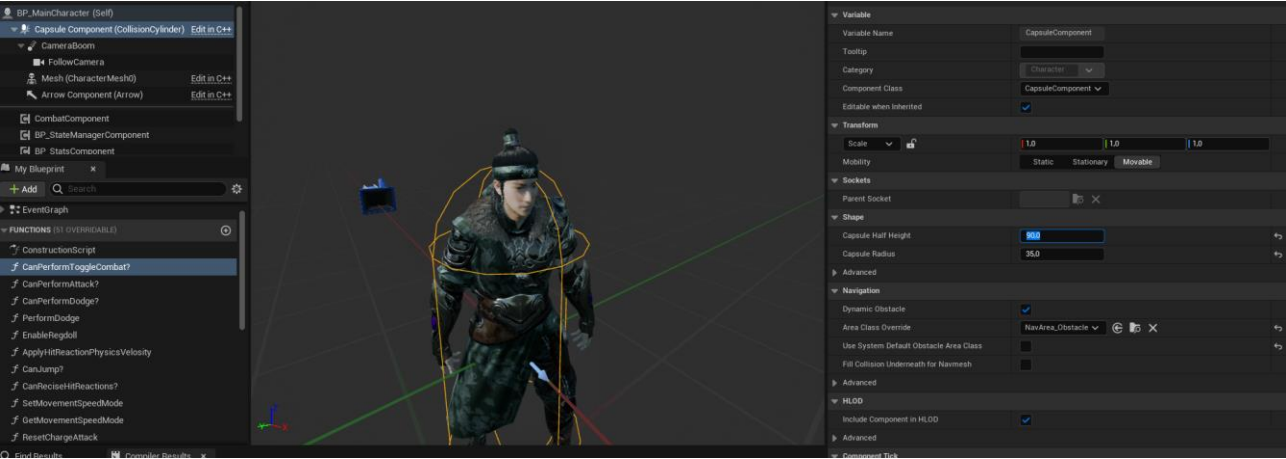


Рисунок 2.9. – Капсульний копонент персонажа

Інтерфейс користувача (UI) — це набір засобів і елементів, що дозволяють гравцю взаємодіяти з грою. Він включає меню, панелі, індикатори, кнопки та інші візуальні елементи, які надають гравцю інформацію і дозволяють здійснювати дії. Ось основні причини, чому UI важливий у іграх:

Надання інформації: UI забезпечує гравця важливою інформацією, такою як здоров'я персонажа, рівень боєприпасів, місцезнаходження на карті, завдання та їх прогрес. Це допомагає гравцю приймати обґрунтовані рішення під час гри.

Зручність управління: Добре спроектований інтерфейс робить гру більш інтуїтивно зрозумілою і зручною. Гравцю легше освоїтися з механікою гри та виконувати дії, коли елементи керування зрозумілі і доступні.

Занурення у гру: Елементи UI можуть підсилювати атмосферу гри та сприяти більшому зануренню гравця у ігровий світ. Наприклад, стилізований інтерфейс може підкреслювати тематику гри і допомагати створити правильний настрій.

Для того, щоб перевіряти статистичні дані користувача, такі як витривалість та здоров'я, було реалізовано функцію, яка отримує всю статистику користувача, переводить її у відсотки, при кожній зміні порівнює значення з максимально можливим, та виводить дані користувачу на візуальний інтерфейс, що робить взаємодію користувача з грою інтуїтивно зрозумілою. Налаштування користувацького інтерфейсу наведено на рисунках 2.10. – 2.11.

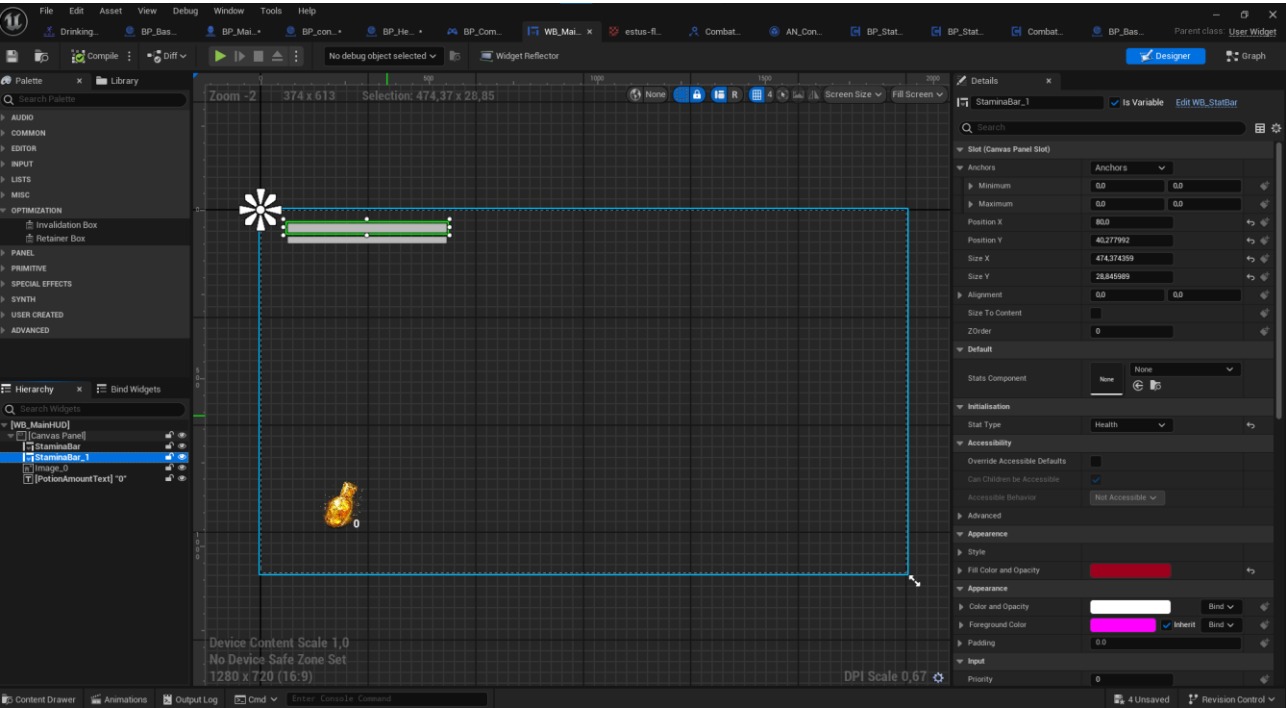


Рисунок 2.10. – Створення UI

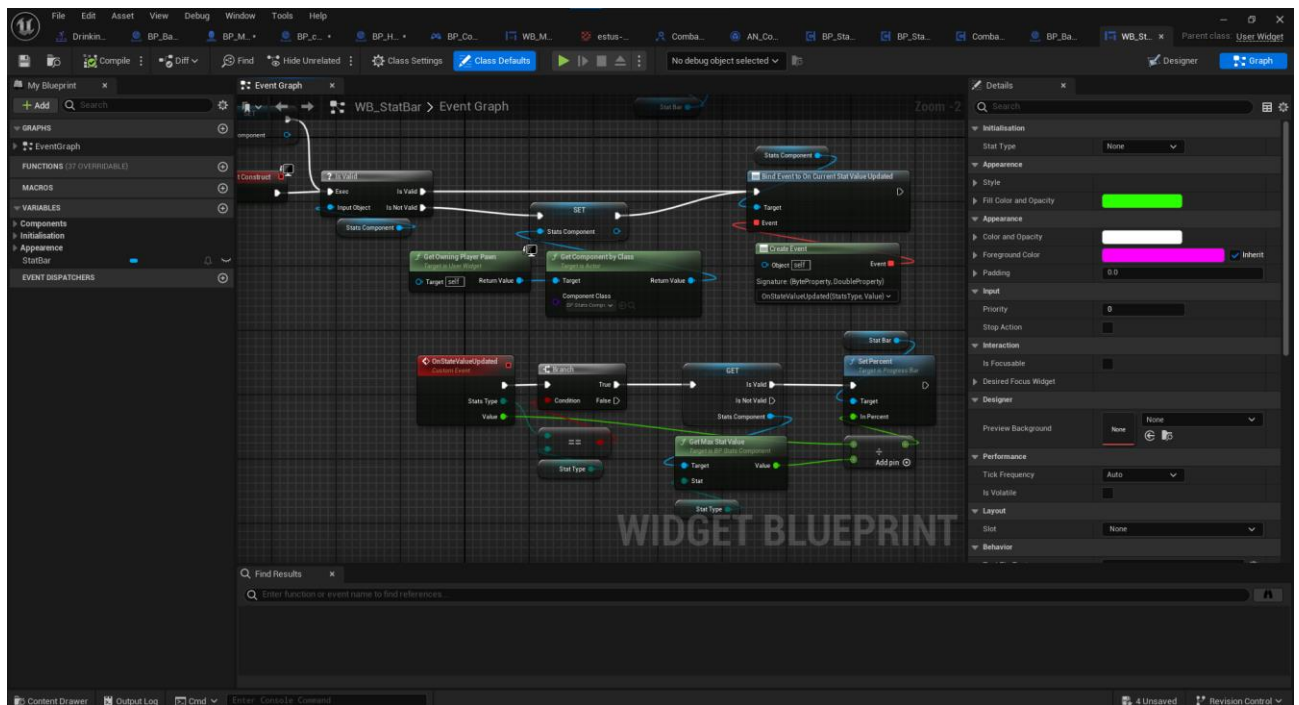


Рисунок 2.11. – Оновлення даних в UI

Для детального розуміння як працює виведення інтерфейсу на екран гравця розглянемо що відбувається на системному рівні.

Лістинінг коду наведено нижче:

Метод AddToViewport

```
void UUserWidget::AddToViewport(int32 ZOrder)
{
    if (UGameViewportSubsystem* Subsystem = UGameViewportSubsystem::Get(GetWorld()))
    {
        FGameViewportWidgetSlot ViewportSlot;
        if (bIsManagedByGameViewportSubsystem)
        {
            ViewportSlot = Subsystem->GetWidgetSlot(this);
        }
        ViewportSlot.ZOrder = ZOrder;
        Subsystem->AddWidget(this, ViewportSlot);
    }
}
```

1. $UGameViewportSubsystem$ $Subsystem$ $=$ $UGameViewportSubsystem::Get(GetWorld())$
 - Отримує підсистему व्यюпорта гри ($UGameViewportSubsystem$), яка керує व्यюпортами, використовуючи метод Get і поточний світ ($GetWorld()$).
2. $FGameViewportWidgetSlot$ $ViewportSlot$
 - Створює об'єкт $FGameViewportWidgetSlot$, який представляє слот для віджета у व्यюпорті.

3. if (bIsManagedByGameViewportSubsystem)
 - Перевіряє, чи віджет керується підсистемою вьюпорта. Якщо так, то отримує відповідний слот для віджета з підсистеми.
4. ViewportSlot.ZOrder = ZOrder
 - Встановлює порядок відображення (ZOrder) для віджета.
5. Subsystem->AddWidget(this, ViewportSlot)
 - Додає віджет до вьюпорта за допомогою підсистеми.

Метод AddToPlayerScreen

```
bool UUserWidget::AddToPlayerScreen(int32 ZOrder)
{
    if (UGameViewportSubsystem* Subsystem = UGameViewportSubsystem::Get(GetWorld()))
    {
        if (ULocalPlayer* LocalPlayer = GetOwningLocalPlayer())
        {
            FGameViewportWidgetSlot ViewportSlot;
            if (bIsManagedByGameViewportSubsystem)
            {
                ViewportSlot = Subsystem->GetWidgetSlot(this);
            }
            ViewportSlot.ZOrder = ZOrder;
            Subsystem->AddWidgetForPlayer(this, GetOwningLocalPlayer(),
            ViewportSlot);
            return true;
        }
        else
        {
            FMessageLog("PIE").Error(LOCTEXT("AddToPlayerScreen_NoPlayer",
            "AddToPlayerScreen Failed. No Owning Player!"));
        }
    }
    return false;
}
```

1. UGameViewportSubsystem Subsystem = UGameViewportSubsystem::Get(GetWorld())*
 - Отримує підсистему вьюпорта гри.
2. if (ULocalPlayer LocalPlayer = GetOwningLocalPlayer())*
 - Перевіряє, чи є локальний гравець, якому належить цей віджет.
3. FGameViewportWidgetSlot ViewportSlot
 - Створює об'єкт FGameViewportWidgetSlot, який представляє слот для віджета у вьюпорті.
4. if (bIsManagedByGameViewportSubsystem)
 - Перевіряє, чи віджет керується підсистемою вьюпорта. Якщо так, то отримує відповідний слот для віджета з підсистеми.

5. ViewportSlot.ZOrder = ZOrder

- Встановлює порядок відображення (ZOrder) для віджета.

6. Subsystem->AddWidgetForPlayer(this, GetOwningLocalPlayer(), ViewportSlot)

- Додає віджет до екрану конкретного гравця за допомогою підсистеми.

7. FMessageLog("PIE").Error(LOCTEXT("AddToPlayerScreen_NoPlayer", "AddToPlayerScreen Failed. No Owning Player!"))

- Якщо немає власного гравця, виводить повідомлення про помилку в лог.

Загальний підсумок

- AddToViewport: Додає віджет до вьюпорта з певним порядком відображення (ZOrder).
- AddToPlayerScreen: Додає віджет до екрану конкретного гравця, якщо він існує, і виводить повідомлення про помилку, якщо власний гравець відсутній.

Обидва методи використовують підсистему вьюпорта гри (UGameViewportSubsystem) для управління віджетами і встановлення їх порядку відображення на екрані.

2.3. Проектування та реалізація важливих алгоритмів

Дане програмне забезпечення було створено з використанням великої кількості алгоритмів, що зберігаються у відповідних класах-менеджерах для запобігання дублюванню коду. Ці класи використовуються іншими компонентами системи для виконання бізнес-логіки, що спрощує структуру алгоритмів і зменшує ймовірність помилок.

Ігровий інтелект є найважчою та найважливішою частиною розробки даного проекту. Він відповідає за всіх можливих ворогів забезпечуючи їхню поведінку, взаємодію з гравцем і навколишнім середовищем. Система штучного інтелекту в грі включає кілька ключових аспектів [5]:

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

1. Патрулювання та пересування. Вороги повинні мати здатність патрулювати територію, дотримуючись визначених маршрутів або точок. Для цього можна використовувати навігаційні системи, такі як навігаційна сітка (NavMesh), яка дозволяє ворогам переміщуватися по території з урахуванням перешкод.

2. Виявлення та розпізнавання гравця. Вороги повинні мати можливість виявляти гравця. Це може бути реалізовано через системи виявлення на основі зору. Коли ворог помічає гравця, він повинен змінювати свою поведінку, переходячи в стан «бойової готовності».

3. Преслідування та атака. Якщо ворог виявляє гравця, він повинен переслідувати його та намагатися атакувати. Для цього можна використовувати поведінкові дерева для керування станами ворога (пошук, переслідування, атака тощо). Наприклад, ворог може мати різні атаки, в залежності від відстані до гравця або наявності спеціальних умов.

Unreal Engine дозволяє реалізувати дану логіку за допомогою вбудованих засобів. Основним інструментом реалізації алгоритму є дерево ухвалення рішень (Behavior Tree). Це модель для перемикання між кінцевим набором завдань, яка представляє собою ієрархічний набір вузлів, які можуть контролювати потік прийняття рішень. Дерево поведінки можна використовувати, для того щоб ігровий інтелект думав самостійно та приймав власні рішення. У той час як дерево поведінки виконує логіку, окремий актив, який називається Blackboard, використовується для зберігання інформації про те, яку поведінкове дерево має знати, щоб приймати обґрунтовані рішення. У нашій аналогії деревом поведінки є мозок [10].

Для реалізації ігрового інтелекту, було використано “Body-soul-brain” патерн штучного інтелекту, де:

- “Body” – візуальне представлення персонажа;
- “Soul” – сутність, яка його контролює (Controller);
- “Brain” – те що визначає поведінку (BehaviorTree);

- “Memory” – місце де ігровий інтелект зберігає інформацію, необхідну для функціонування (Blackbord).

Ігровий рушій надає потужні інструменти для розробки ігрового інтелекту, але програміст повинен реалізувати більшість запитів, обробників і відповідних дій персонажа самостійно. Для розуміння логіки поведінки персонажів достатньо дослідити головний алгоритм ігрового інтелекту, його структуру і виконання. Усе це можна побачити рисунку 2.12 [5].

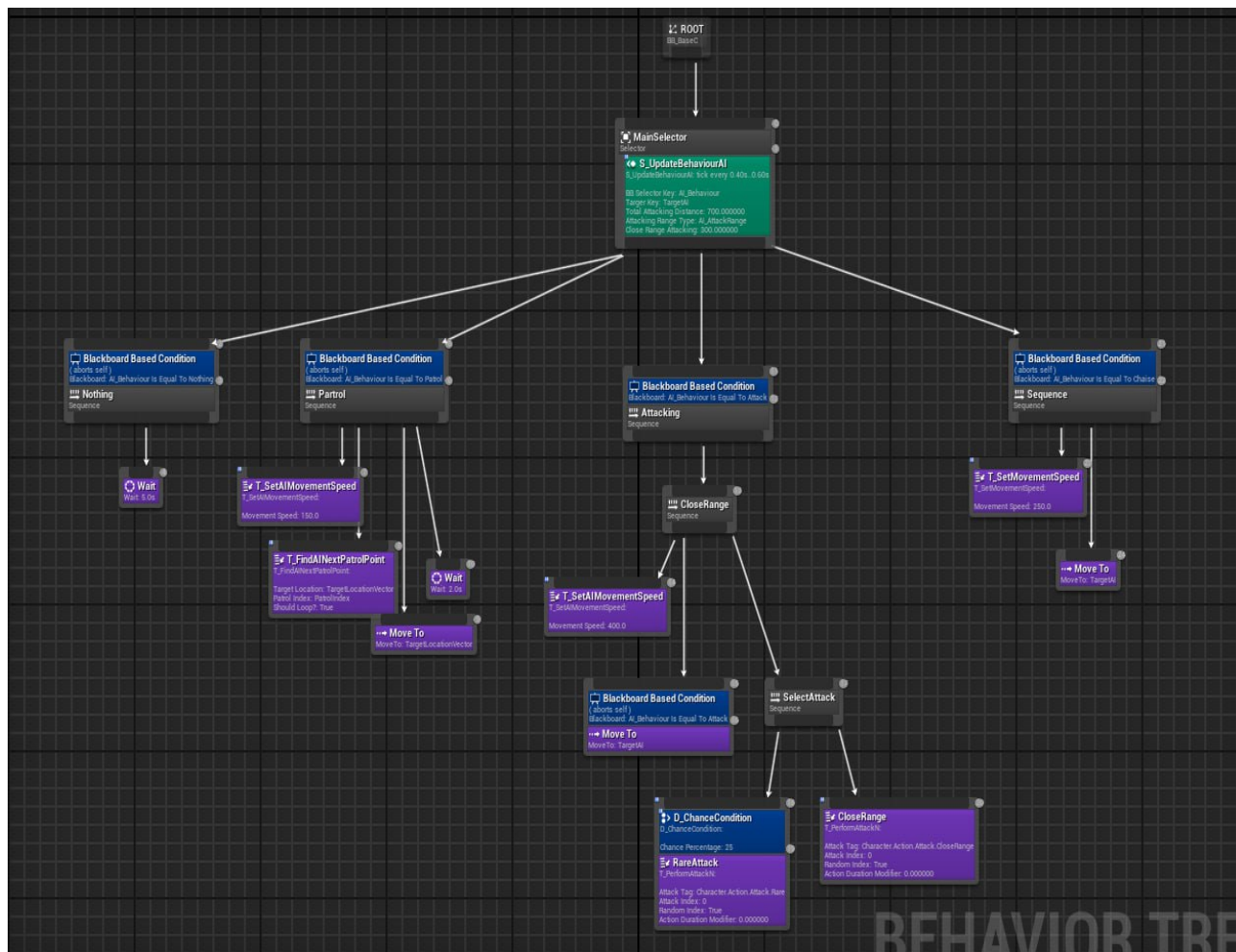


Рисунок 2.12. – Дерево ігрового інтелекту

На верхньому рівні знаходиться кореневий вузол "ROOT" де зберігається інформація необхідна для функціонування так звана пам'ять. Від нього відходить головний селектор "MainSelector", який включає поведінкове оновлення "S_UpdateBehaviourAI", що допомагає визначити перелік подій та їх виконання. Дерево має в собі декілька вузлів які визначають конкретні дії як патрулювання, яке передбачає перехід по зазначених точках на карті, атака, яка

спрацьовує, якщо ворог поруч з гравцем та переслідування, що змінюються залежно від стану поведінки та умов, до прикладу якщо ворог побачиш гравця. Це дозволяє ворогу динамічно реагувати залежно від ситуації в грі.

2.4. Налаштування графічних компонентів гри

Для більшого занурення гравця у ігровий світ, було здійснено налаштування графічних компонентів, що включають розміщення моделей, проектування ворогів, додавання ефектів та освітлення.

Вороги, це один з основних компонентів гри, тому їм потрібно приділяти значну увагу, додаючи до них відповідні анімації та текстури. Так як розробити та налаштувати скелет персонажа це велика частина роботи, яка займає величезну кількість часу, було використано відповідні ассети, які полегшують даний процес та заощаджують часові ресурси. Знаходяться ассети у вільному доступі на UnrealEngine Marketplace під назвою Infinity Blade:Adversaries [18].

Процес налаштування моделі та текстур головного боса у «soulslike» грі визначався кількома етапами. Спочатку було визначено загальний вигляд і характер персонажа. Далі було підібрано ассети, що відповідають концептуальним задумам персонажа та вже включають базові моделі, текстури та кістяки. Після цього моделі було адаптовано під специфічні вимоги гри: коригувалися пропорції, додавалися унікальні елементи тощо. На етапі текстурування застосовувались наявні в ассетах текстури, а також додавалися допоміжні шари, щоб підкреслити унікальні риси боса. По завершенню було прикріплено необхідні анімації. На рисунках 2.13–2.14 зображено процес налаштування боса.

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

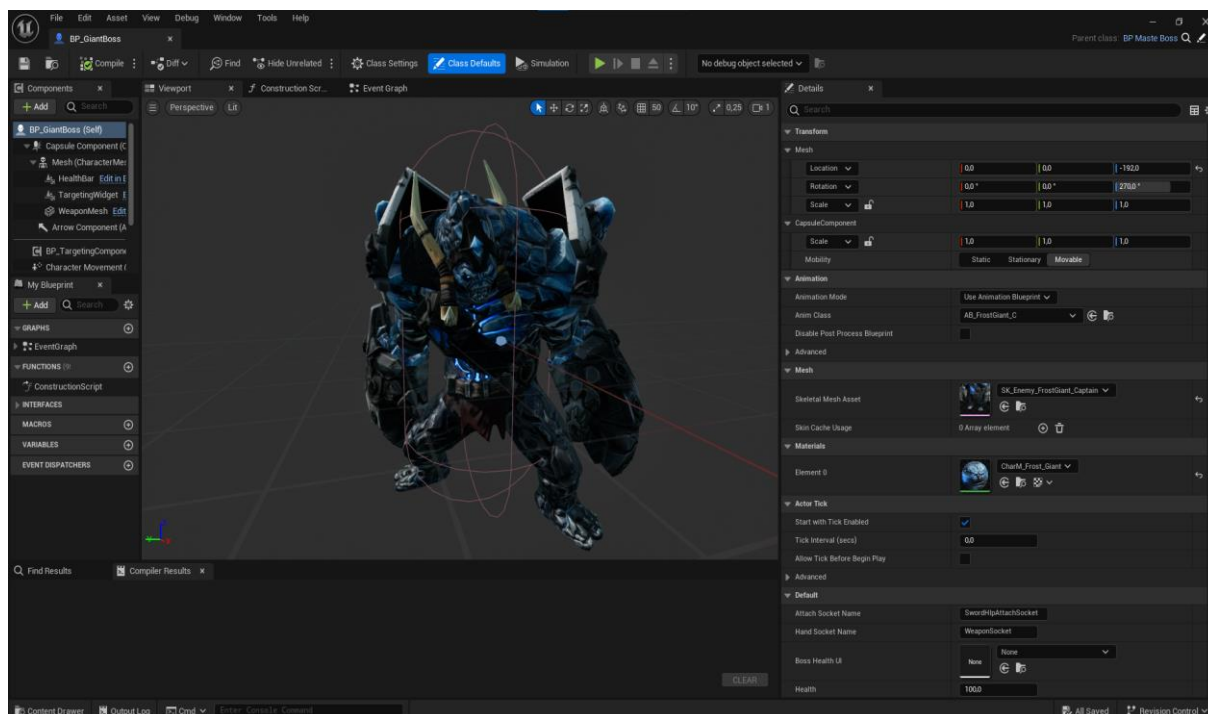


Рисунок 2.13. – Налаштування моделі та текстур головного боса

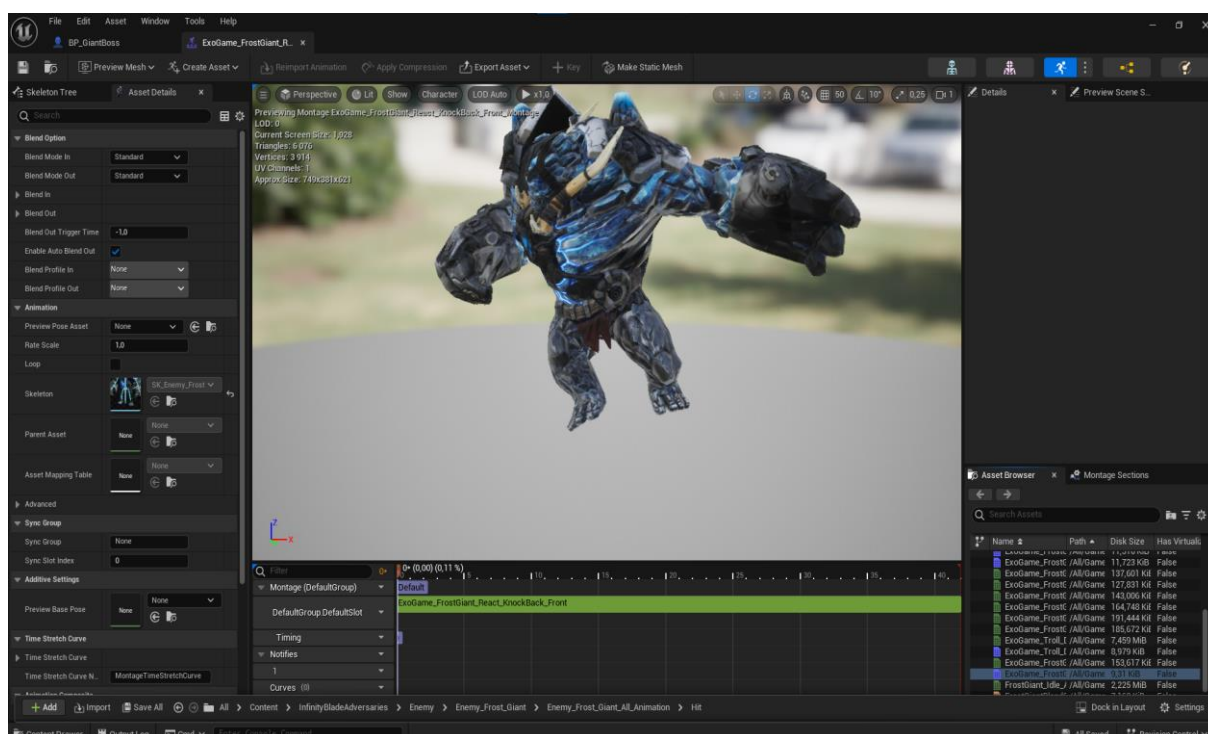


Рисунок 2.14. – Додавання анімацій до моделі боса

Ігровий світ в «soulslike» – темний та похмурий. Для його розробки потрібно приділити значну увагу деталям, об’єктам та освітленню. Атмосферні локації, опрацьовані до дрібниць, додають грі глибини та сприяють зануренню в гру користувача, роблячи кожну нову область відчуттям відкриття та подолання перешкод. Такий підхід дозволяє створити унікальний геймплейний

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

досвід, де атмосфера та світогляд стають невід'ємною частиною загального враження від гри. Тому для розробки було використано набір асетів Infinity Blade:Fire Lands [19].

Використовуючи ассети з набору Infinity Blade: Fire Lands, було інтегровано ряд готових моделей будівель, ландшафтів та об'єктів, які були налаштовані у відповідності до стилістики гри. Вогонь відіграв ключову роль в створенні напруженої атмосфери: палаючі руїни та вогнища додали відчуття небезпеки і драматизму. Додатково, було додано унікальні елементи та текстури, щоб кожна область мапи мала свої відмінності і відчувалася живою та інтерактивною [19].

На рисунку 2.15 зображено фрагмент створення локації.

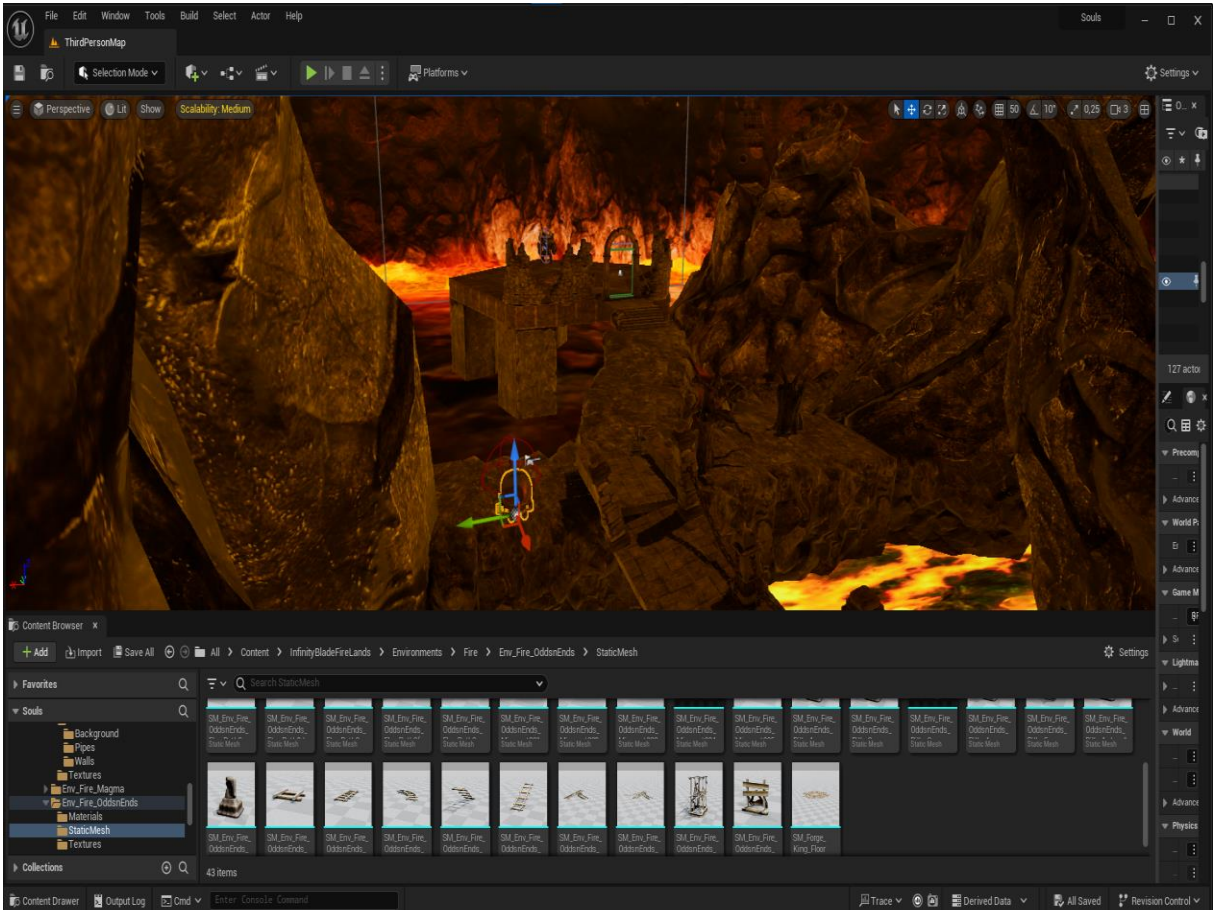


Рисунок 2.15. – Створення локації

Важливим елементом ігрового світу є багаття. Воно служить для збереження гравця, на місці, якого він дістався у грі. Це багаття не лише слугує точкою збереження прогресу, але й виконує важливу геймплейну функцію, дозволяючи гравцеві відновити здоров'я та перезарядити фляги з лікувальними

еліксирами. Крім того, багаття часто є місцем відпочинку і стратегічного планування, де гравець може обміркувати свої подальші дії, підготуватися до нових викликів. Такі точки збереження створюють необхідний баланс між дослідженням світу та необхідністю зберегти досягнутий прогрес, що підсилює відчуття небезпеки та важливості кожного рішення в грі.

Багаття також було використано з відповідних ассетів. На рисунках 2.16–2.17 можна побачити процес налаштування та розміщення об’єктів багаття на мапі.

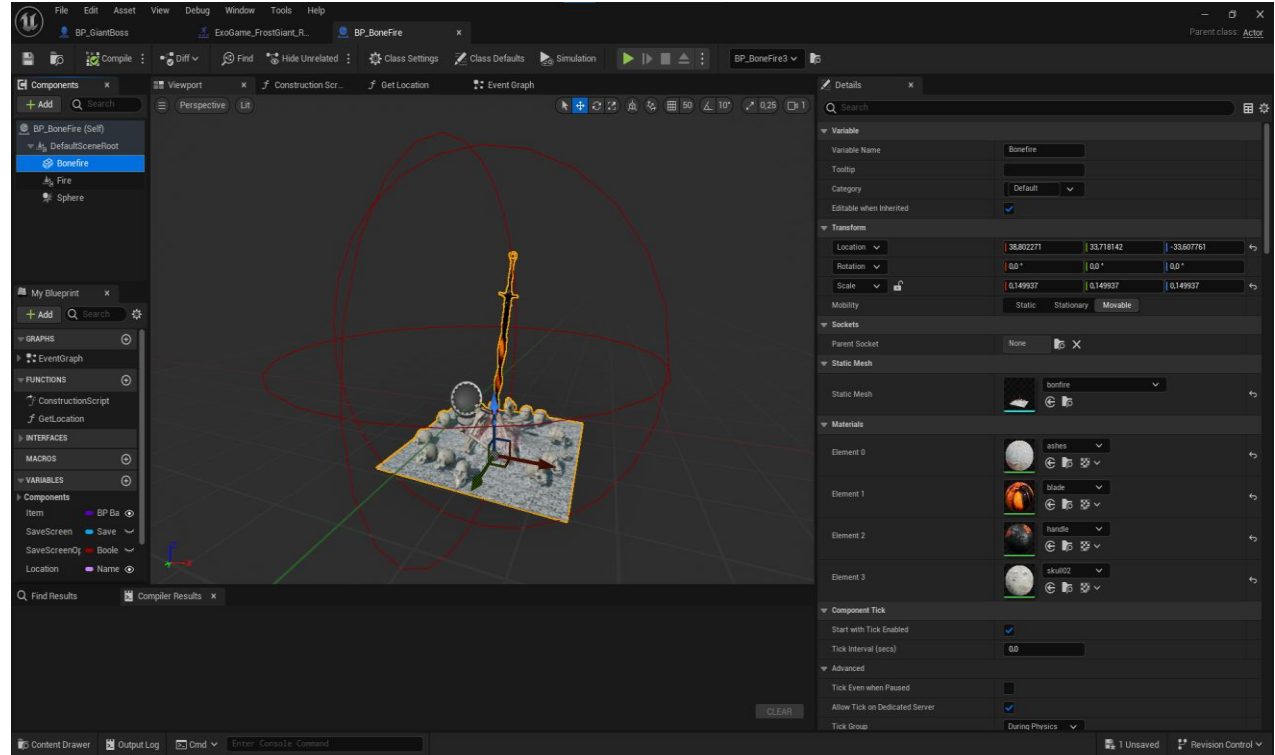


Рисунок 2.16. – Налаштування багаття, додавання сфери колізії для взаємодії

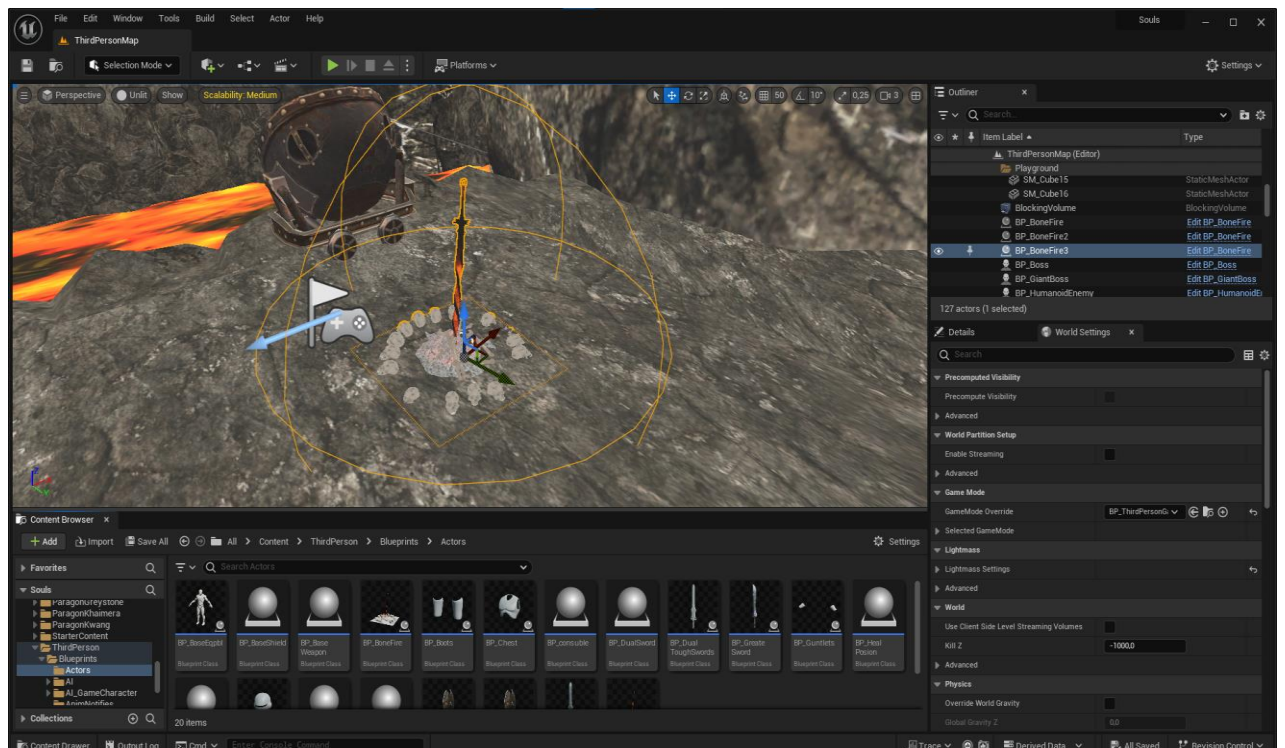


Рисунок 2.17. – Розміщення багаття на локації

Головному гравцеві та його герою також було приділено значну увагу. Для граця було використано асет персонажу з ParagonKwang. На рисунку 2.18 представлено зовнішній вигляд ігрового персонажа [21].

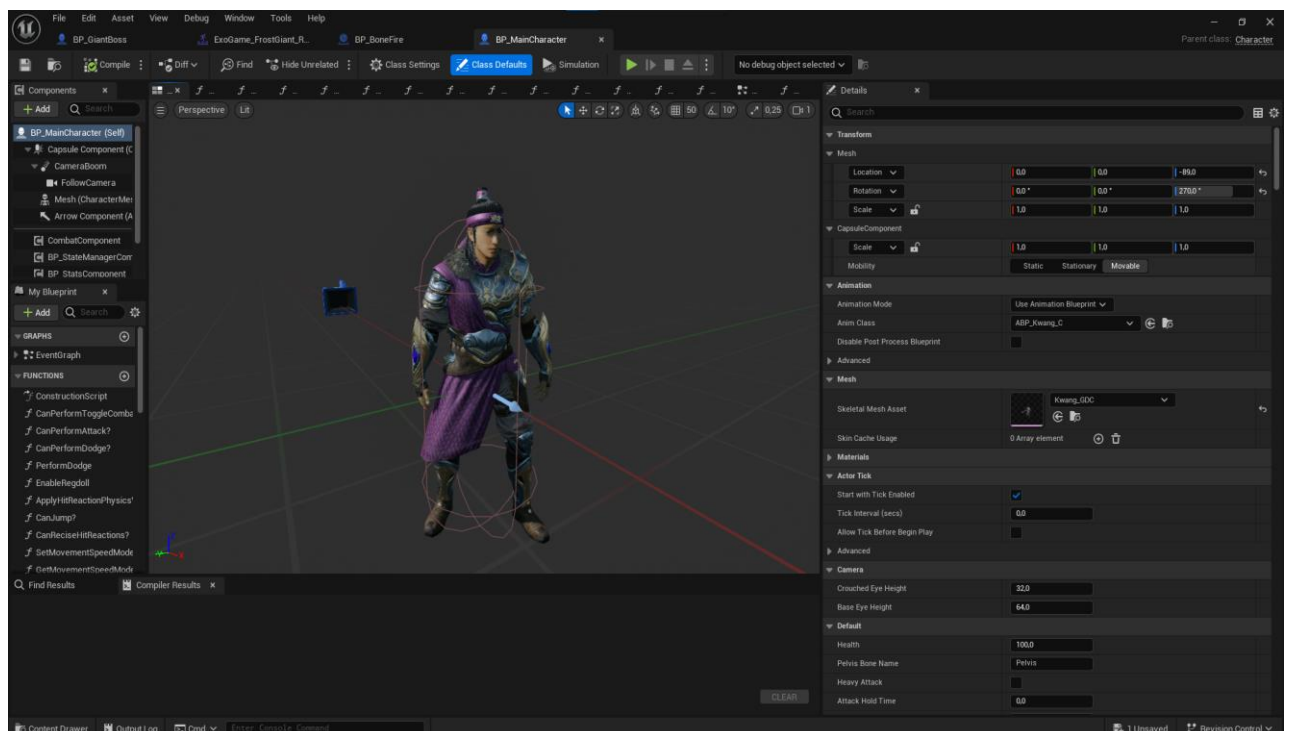


Рисунок 2.18. – Вигляд персонажа

Для даного об'єкта було виконано ряд основних налаштувань:

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

- анімації для відповідних дій, включаючи атаки, захист, переміщення, використання предметів та взаємодію з навколишнім світом. Це дозволяє створити плавний та реалістичний рух персонажа, що підвищує рівень занурення у гру;

- визначено попередньо встановлені об'єкти, з якими з'являється персонаж, тобто меч та щит. Це базове обладнання дає гравцю стартові можливості для захисту та атаки;

- налаштовано капсульний компонент для колізії об'єкта. Капсульна колізія забезпечує коректну взаємодію персонажа з оточенням, дозволяючи точно визначати зіткнення з різними об'єктами, такими як стіни, підлога, вороги тощо;

- встановлено слоти для зброї, що дозволяють гравцю змінювати озброєння під час гри.

- визначено положення камери, що забезпечує оптимальний огляд та керування персонажем. Камера слідує за гравцем, забезпечуючи широкий огляд оточення, допомагаючи краще орієнтуватися в просторі та своєчасно реагувати на загрози.

На рисунку 2.19 представлено приклада налаштування анімації ігрового персонажа.

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		49

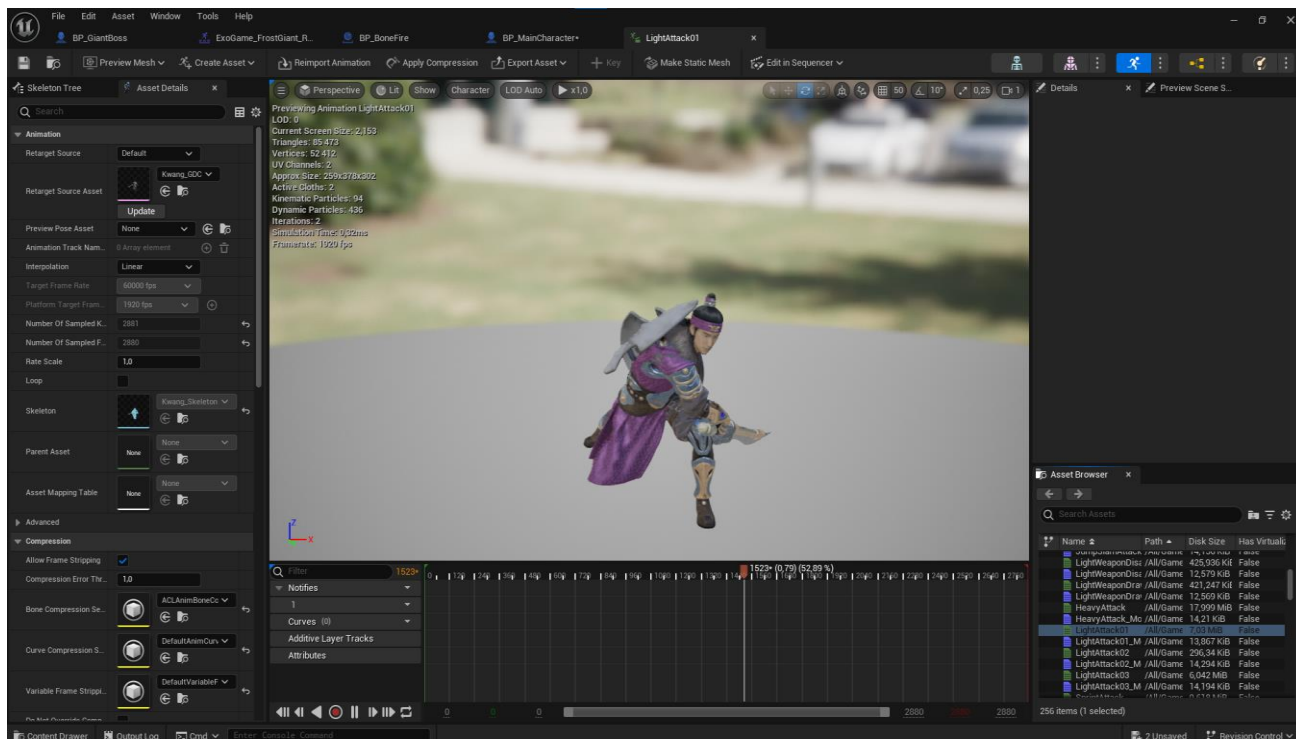


Рисунок 2.19. – Налаштування анімації ігрового персонажа

Зброя була розроблена як певний актор, з яким користувач може взаємодіяти. Так як систему було проєктовано під легку зміну налаштувань та анімацій. Кожного такого актора можна налаштовувати окремо, роблячи його дочірнім класом певного об'єкта. Приклад налаштування зброї наведено на рисунку 2.20

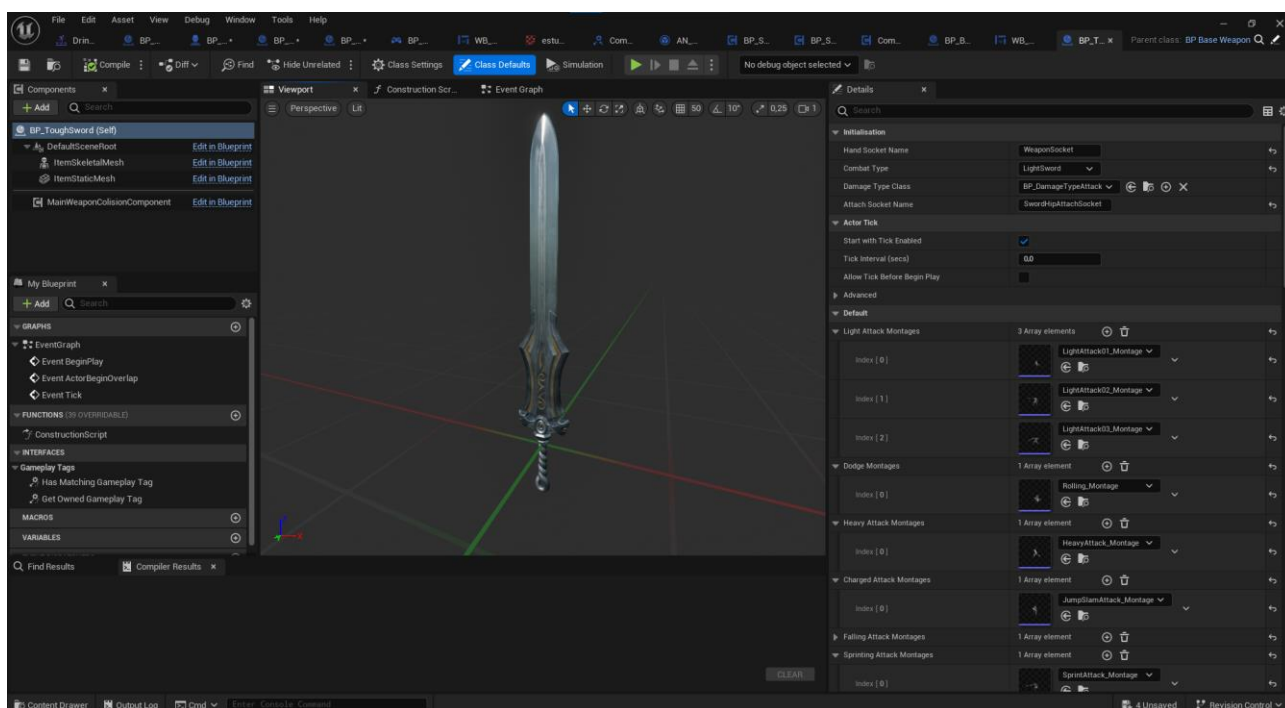


Рисунок 2.20 – Налаштування зброї

		Матіш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		50

Висновки до другого розділу

Під час написання другого розділу проекту було здійснено ряд заходів для проектування та реалізації важливих частин і алгоритмів програмного продукту. Був проведений аналіз варіантів використання системи, який було окремоено за допомогою відповідної діаграми. Такий підхід дозволив наочно описати, як користувачі можуть взаємодіяти із системою та виявити можливі проблемні місця програмного продукту. Було описано структуру системи та шаблони проектування, використані для створення гнучкого та масштабованого проекту. Важливими елементами стали опис та проектування ігрового інтелекту та опис створення ПЗ.

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПЗ

3.1. Порядок встановлення та налаштування роботи

Зважаючи на обмеженість ресурсів, не вдалося реалізувати всі заплановані функції в рамках цієї версії проекту. Однак, основні завдання були виконані, і продукт цілком готовий до первинного використання, а в майбутньому передбачається вдосконалення і розширення можливостей гри.

Першим кроком до початку роботи з програмним продуктом є його встановлення. Для встановлення гри користувачу потрібно виконати такі дії:

1. Завантажити архів з грою на свій комп'ютер.
2. Розархівувати завантажений архів.
3. Запустити файл «Souls.exe».

Перш за все потрібно переконатися, що комп'ютер відповідає всім вимогам до апаратного забезпечення, а саме:

- має встановлену операційну систему Windows 10 x64;
- містить пакет бібліотек DirectX 11.

Усі інші бібліотеки, необхідні для запуску гри, включені в архів.

Після запуску гри користувач потрапляє в головне меню з трьома різними опціями. Це «почати гру», «вийти з гри», або «зайти в налаштування». В налаштуваннях гравець може змінити роздільну здатність екрану, відредагувати налаштування графіки, увімкнути або вимкнути вертикальну синхронізацію та зберегти зміни. На рисунку 3.1 зображено приклад головного меню гри.

Після запуску програми, користувач потрапляє на локацію, яку йому потрібно пройти. Щоб зробити це необхідно подолати усіх ворогів. Для цього користувачу одразу видається стандартне спорядження, а саме меч та щит. На рівні розміщено декілька ворогів стандартного класу та головний бос, що знаходиться в кінці локації. Дані вороги мають різний рівень складності, а також вони керуються ігровим інтелектом, який адаптується до дій гравця. Це дозволяє створити більш динамічний ігровий процес. Кожен тип ворогів

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

змушує гравця використовувати різноманітні підходи для їх подолання. Таке рішення щодо розробки ворогів робить гру цікавою та непередбачуваною.



Рисунок 3.1. – Головне меню

Бій з босом часто є кульмінацією рівня або завдання, де гравцю потрібно використовувати всі набуті навички та стратегії для перемоги. Боси мають унікальні механіки, які вимагають від гравця ретельного планування і швидкої реакції. У даній версії гри також передбачено останній бій з найсильнішим ворогом. Щоб запустити даний сюжет необхідно переміститись у відповідне місце локації. Коли робота боса активується внизу екрана з'явиться відповідна шкала здоров'я. На рисунку 3.2 зображено фрагмент бою з головним ворогом.



Рисунок 3.2. – Битва з босом

Керування персонажем гравця відбувається за допомогою спеціальних клавіш. Основні клавіші керування описані в таблиці 3.1

Таблиця 3.1

Клавіші керування

Дія	Клавіша
Рухатись вперед	W
Рухатись назад	S
Рухатись вліво	A
Рухатись вправо	D
Біг	Left Shift
Змінити швидкість на шаг	Left Alt
Взаємодіяти	E
Дістати/Сховати зброю	1
Прицілювання	Middle Mouse Button
Легкий удар	Left Mouse Button
Тяжкий удар	V
Блокування	Right Mouse Button
Використання предмету	R
Ухиляння	SPACE

За допомогою даних можливостей, та бойової системи, користувач повинен пройти рівень та перемогти боса. На карті розміщені багаття при взаємодії з якими користувач встановлює нову точку з якої буде починати рівень

після сметрї. Так, як цей жанр передбачає складність, після сметрї гравця всі вороги з'являються знову.

3.2. Тестування програмного продукту

Перш ніж випустити гру на ринок та зробити її доступною для широкого кола користувачів, її варто протестувати, щоб переконатися, що вона працює як треба, і відповідає вимогам. Тестування допомагає виявити дефекти, помилки та інші проблеми на різних етапах розробки.

Для тестування гри було використано кілька підходів та інструментів. Було проведено функціональне та ручне тестування, що дозволило перевірити гру на наявність помилок і дефектів [4, 9].

З ціллю полегшення проведення тестів було розроблено чеклісти, що сприяли ефективному плануванню, перевірці та виконанню тестування. Приклад чеклістів наведено в таблиці 3.2

Таблиця 3.2

Чекліст тестування гри

№	Тестовий сценарій	Очікуваний результат	Фактичний результат
1.	Перевірка якості та коректності відображення ефектів	Усі ефекти відбуваються плавно, анімації відтворюються без затримок і багів	Усі ефекти відбуваються плавно, анімації відтворюються без затримок і багів
2.	Балансування складності ворогів	Вороги мають адекватний рівень складності, що відповідає жанру гри	Вороги мають адекватний рівень складності, що відповідає жанру гри
3.	Перевірка ігрового інтелекту ворогів	Вороги реагують на персонажа, коли той потрапляє в їх поле зору	Вороги реагують на персонажа, коли той потрапляє в їх поле зору
4.	Відображення візуальних ефектів атак	Ефекти відображаються без артефактів	Ефекти відображаються без артефактів

Продовження Таблиці 3.2

5.	Взаємодія персонажа з оточенням	Персонаж правильно взаємодіє з об'єктами оточення	Персонаж правильно взаємодіє з об'єктами оточення
6.	Швидкість завантаження	Рівень завантажується швидко і без затримок	Рівень завантажується швидко і без затримок
7.	Відображення елементів інтерфейсу	Всі елементи інтерфейсу відображаються правильно	Всі елементи інтерфейсу відображаються правильно
8.	Перевірка текстур оточення	Текстури завантажуються правильно і без затримок	Текстури завантажуються правильно і без затримок
9.	Відновлення здоров'я гравця	Відновлюється правильно	Відновлюється правильно
10.	Стабільність роботи гри	Гра працює стабільно	Гра працює стабільно

Завдяки пройденим тестам вдалося виявити дефекти та конфлікти у роботі системи і виправити їх ще на початкових етапах розробки. В основному вони стосувалися відображення графічного інтерфейсу та взаємодії персонажів із системою. Усі ці недоліки були виправлені до початку фінальної стадії.

Не менш важливим стало навантажувальне тестування на систему та її окремі компоненти. Замірювання використаних ресурсів здійснювалося за допомогою програм MSI Afterburner та Windows Task Manager, що дозволило отримати детальні дані про використання CPU, GPU, оперативної пам'яті та інших системних ресурсів. Це було критично важливим для забезпечення оптимальної продуктивності системи під час розробки та налаштування і для виявлення можливих проблем з ресурсами ще до релізу [6].

Тестування проводилося на системі з такими характеристиками:

- ОС: Windows 10 x64;
- Процесор: Intel Core i5-12400F;
- Оперативна пам'ять: 16Gb DDR4 на частоті 3200;
- Відеокарта: Geforce GTX 1650;

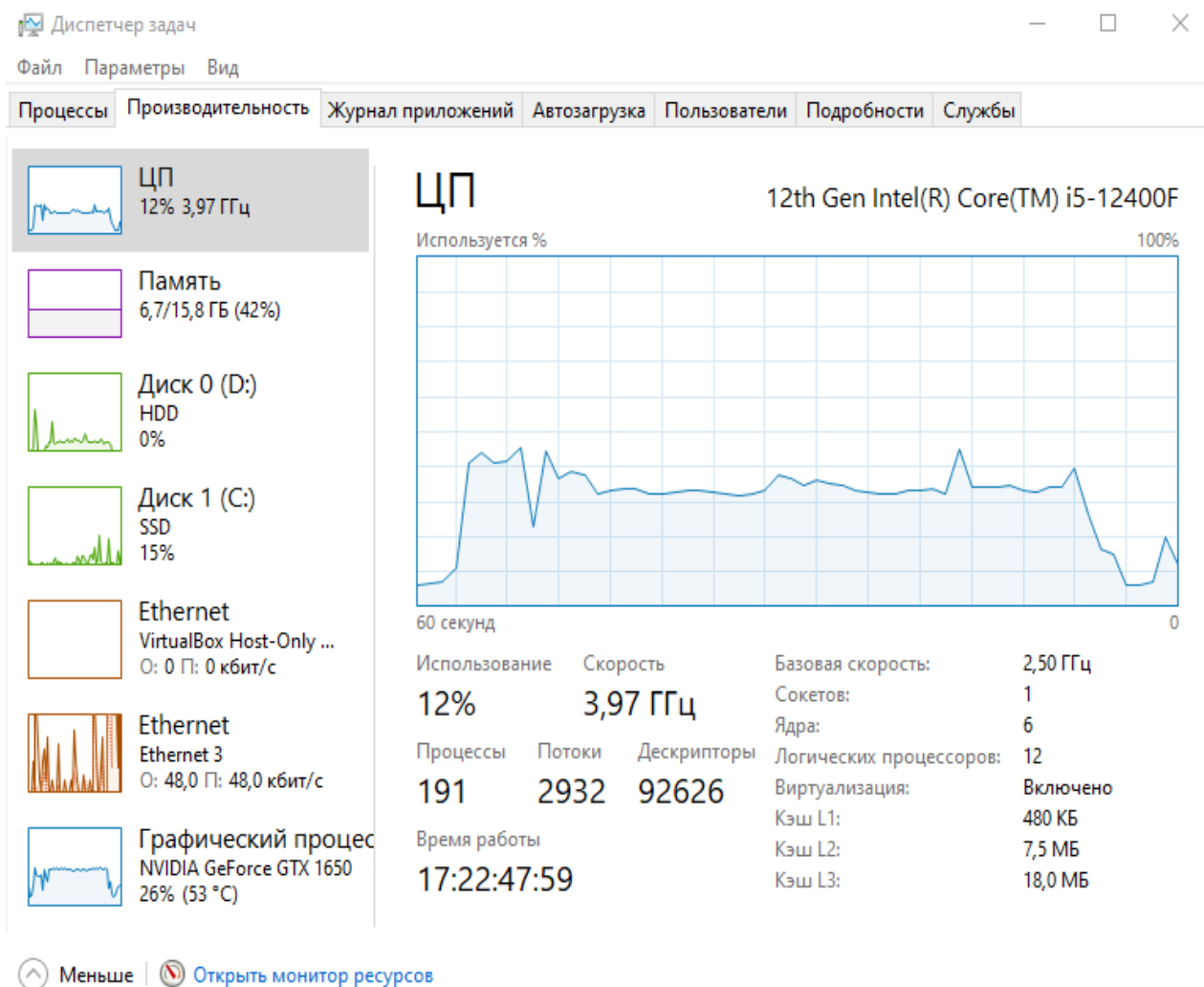


Рисунок 3.3. – Графік споживання ресурсів системи

У результаті тестування навантаження були отримані гарні показники. Частота кадрів трималась на рівні 80-95 кадрів на секунду, що є гарним показником. Навантаження на процесор складало не більше 40%. Споживання оперативної пам'яті було на рівні 6.6 ГБ . Навантаження відеокарти складо 50%. Це все доводить те що система відповідає мінімальним вимогам ігрового продукту, видаючи при цьому стабільну частоту кадрів (Рис.3.3).

Висновки до третього розділу

У третьому розділі кваліфікаційної роботи було виконано реалізацію та тестуванню програмного продукту.

Було описано як користувачі можуть взаємодіяти із готовим програмним забезпеченням та як відбувається ігровий процес. Також було окреслено

інструкцію, щодо роботи з програмою, в якій описані ключові елементи інтерфейсу гравця, а також наведено опис можливостей керування персонажем.

Щоб представити продукт належної якості було виконано декілька видів тестування. Це дозволило виявити критичні місця та оперативно їх скорегувати.

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

ВИСНОВКИ

В рамках кваліфікаційної роботи освітнього ступеню бакалавр було проведено ключові етапи циклу розробки гри у жанрі «soulslike» з використанням ігрового рушія Unreal Engine, та представлено першу версію програмного продукту.

У першому розділі було сформовано теоретичні та функціональні вимоги до програмного забезпечення. Завдяки дослідженню сучасних тенденцій було встановлено жанр та технології розробки. Важливим етапом на шляху до цього став пошук та аналіз аналогічних програмних продуктів. Було виокремлено важливі елементи, необхідні для створення якісної та конкурентоспроможної гри.

У другому розділі було визначено та окреслено сценарії використання системи. Описано основні ролі та ключові прецеденти системи. Було описано архітектуру та структуру програми, які забезпечують ефективність та масштабованість програмної системи, що відкриє можливість додавати новий функціонал у майбутньому. Ключовими моментами у даному розділі стали проектування і реалізація ігрового інтелекту та основної роботи системи.

У третьому розділі, який став фінальним, було описано основний варіант встановлення та налаштування гри. Для цього було сформовано інструкцію. Також було описано основні моменти та процеси у грі, представлено приклади інтерфейсу. Було наведено частину плану тестування, в якому було наведено результати перевірок з використанням різних технік та підходів. Виявлено та виправлено помилки.

Результатом виконання всіх завдань кваліфікаційної роботи стала готова гра у жанрі «soulslike», яка ілюструє ключові елементи ігор цього жанру. Гра може бути використана як для особистих потреб, так і для подальшого дослідження в галузі відеоігор або для комерційної розробки.

У майбутніх версіях гри передбачається реалізація додаткових ігрових механік, покращення графічної і аудіо-складових, а також розширення

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

геймплейних можливостей для збільшення інтерактивності та емоційної глибини гравців.

Під час створення даного проекту довелось зіштовхнутись з великою кількістю труднощів. Найважчим у даній роботі виявилось використання нових технологій та складних інструментів, які були обрані для створення гри. Наприклад, використання Unreal Engine на деяких етапах розробки стало складним викликом через його складність і величезний обсяг можливостей, а також відсутність попереднього досвіду розробки. Не легкою стала і реалізація складної системи бою та впровадження ігрового інтелекту, які потребували ретельної уваги.

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		60

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гордієнко А. В. Комп'ютерні ігри та їхні позитивні психологічні ефекти / А. В. Гордієнко // Особистість у просторі проблем ХХІ століття / А. В. Гордієнко. – Київ, 2019. – С. 31–35.
2. Документація до ігрового рушія Unreal Engine [Електронний ресурс] // Unreal Engine – Режим доступу до ресурсу: https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5-4-documentation?application_version=5.4.
3. Кіберспорт: як відеоігри стали професією [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: <https://maxnet.ua/blog/kibersport-yak-videoigri-stali-profesiyeyu/>.
4. Мануальне тестування [Електронний ресурс] – Режим доступу до ресурсу: <https://qalight.ua/baza-znaniy/ruchne-ta-avtomatizovane-testuvannya/>.
5. Мероник Т. Ігровий штучний інтелект – підходи до побудови [Електронний ресурс] / Тетяна Мероник. – 2020. – Режим доступу до ресурсу: <https://ekmair.ukma.edu.ua/items/a34a07d6-7cea-4c86-8d41-a597a176a542>.
6. Навантажувальне тестування [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: <https://training.qatestlab.com/blog/technical-articles/load-testing-tools/>.
7. Поведінковий патерн State [Електронний ресурс] – Режим доступу до ресурсу: <https://refactoring.guru/ru/design-patterns/state>.
8. Феномен souls-like жанру [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: <https://migdb.com/uk/statti/fenomen-souls-like-zhanru>.
9. Функціональне тестування [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: <https://training.qatestlab.com/blog/technical-articles/difference-between-functional-and-non-functional-testing/>.
10. BehaviorTree [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.unrealengine.com/4.27/en-US/InteractiveExperiences/ArtificialIntelligence/BehaviorTrees/>.

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

11. Best practices for organizing your Unity projec [Електронний ресурс] // Unity – Режим доступу до ресурсу: <https://unity.com/en/how-to/organizing-your-project>.
12. Blueprint Class [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/UserGuide/Types/ClassBlueprint/>.
13. Blueprints Quick Start Guide [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/QuickStart/>.
14. Combat System (Documentation) [Електронний ресурс] – Режим доступу до ресурсу: <https://forums.unrealengine.com/t/combat-system-documentation/1157772>.
15. Controller [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.unrealengine.com/4.27/en-US/InteractiveExperiences/Framework/Controller/>.
16. Godot Engine [Електронний ресурс] – Режим доступу до ресурсу: <https://github.com/godotengine/godot>.
17. How Many Gamers Are There? [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: <https://explodingtopics.com/blog/number-of-gamers>.
18. Infinity Blade: Adversaries [Електронний ресурс] – Режим доступу до ресурсу: <https://www.unrealengine.com/marketplace/en-US/product/infinity-blade-enemies>.
19. Infinity Blade: Fire Lands [Електронний ресурс] – Режим доступу до ресурсу: <https://www.unrealengine.com/marketplace/en-US/product/infinity-blade-fire-lands>.
20. Level Blueprint [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/UserGuide/Types/LevelBlueprint/>.
21. Paragon: Kwang [Електронний ресурс] – Режим доступу до ресурсу: <https://www.unrealengine.com/marketplace/en-US/product/paragon-kwang>.

- 22.UML Activity Diagram [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/>.
- 23.UML Sequence Diagrams [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/>.
- 24.UML Use Case Diagram [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/>.

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

ДОДАТКИ

		Матяш О.В.			КН.КР.Б – 122 – 5 – ПЗ	Арк.
		Марчук Г.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

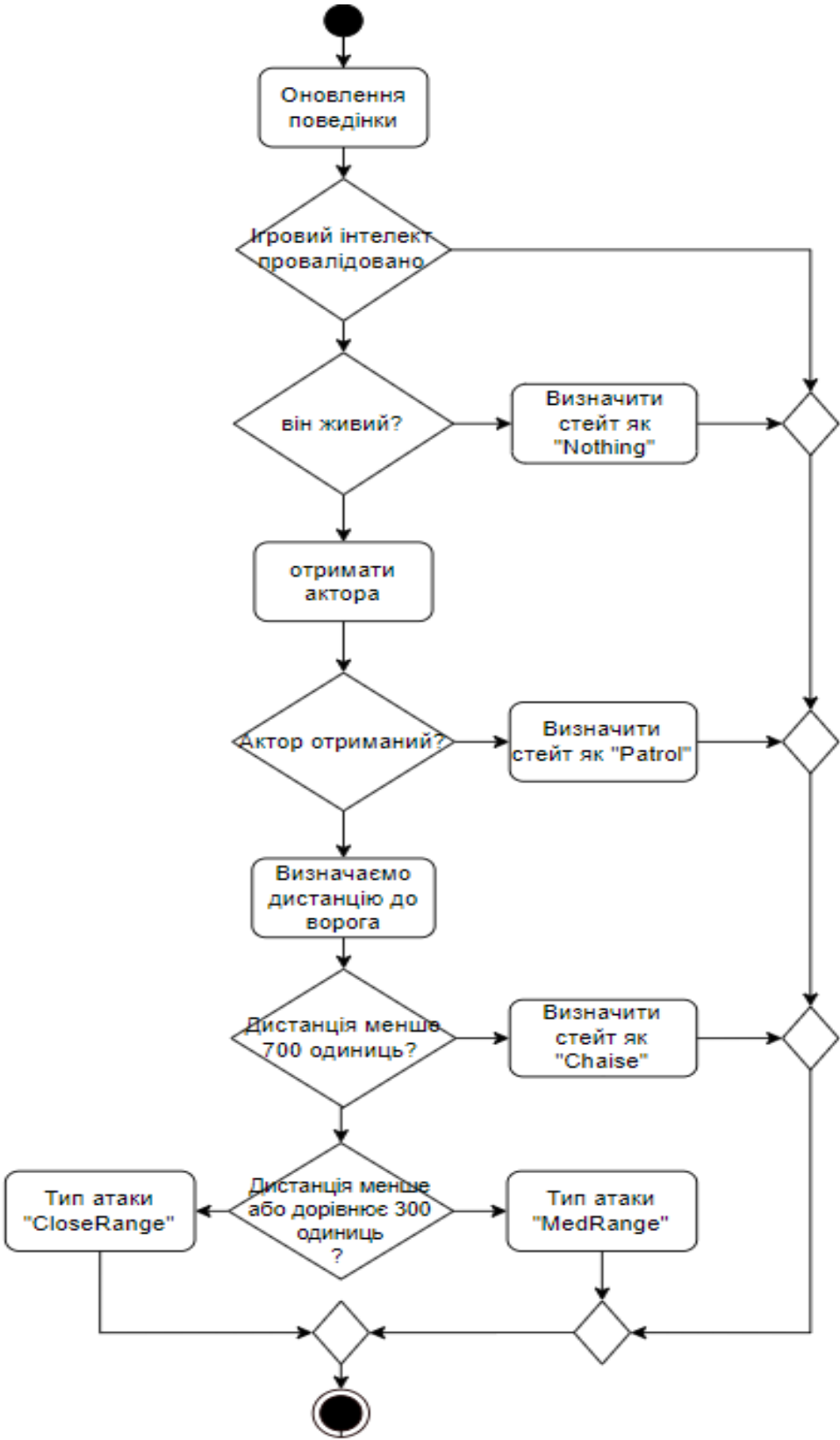


Рисунок А.1 – Діаграма активності оновлення поведінки ігрового інтелекту

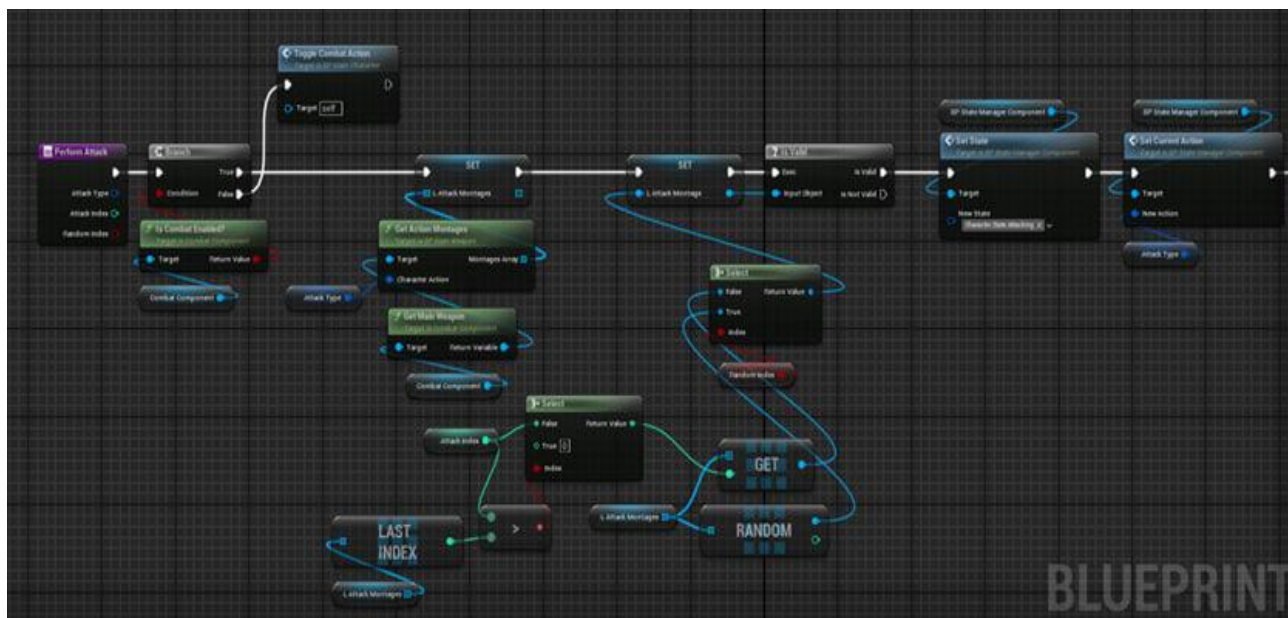


Рисунок Б.1 – Refrom attack, частина 1

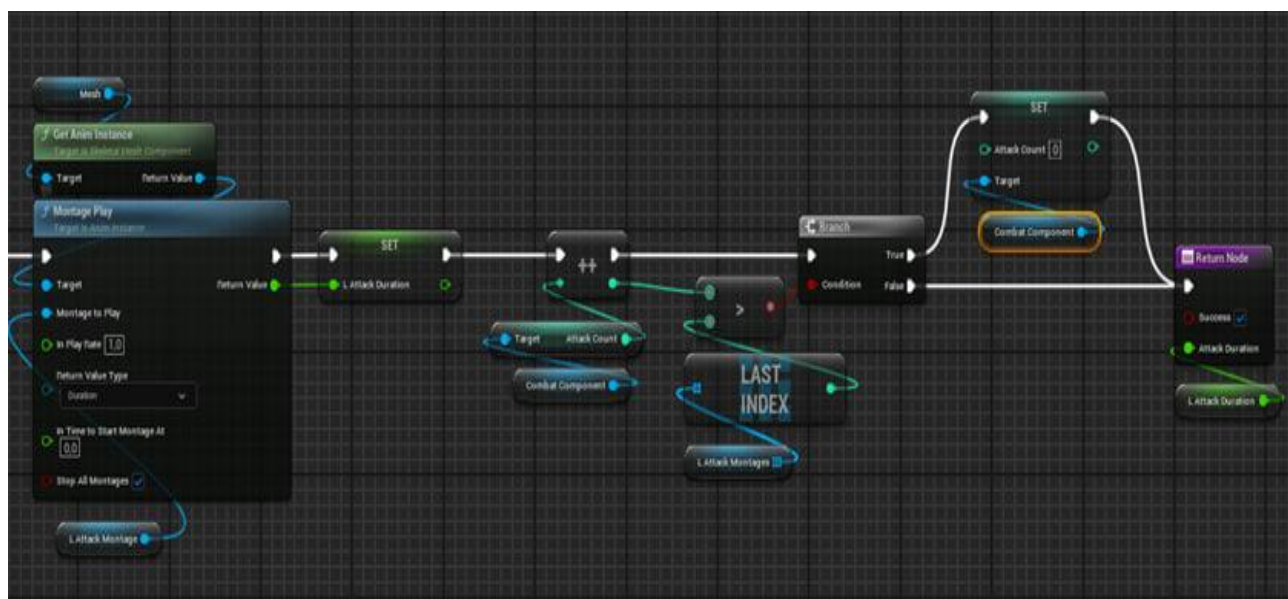


Рисунок Б.2 – Refrom attack, частина 2

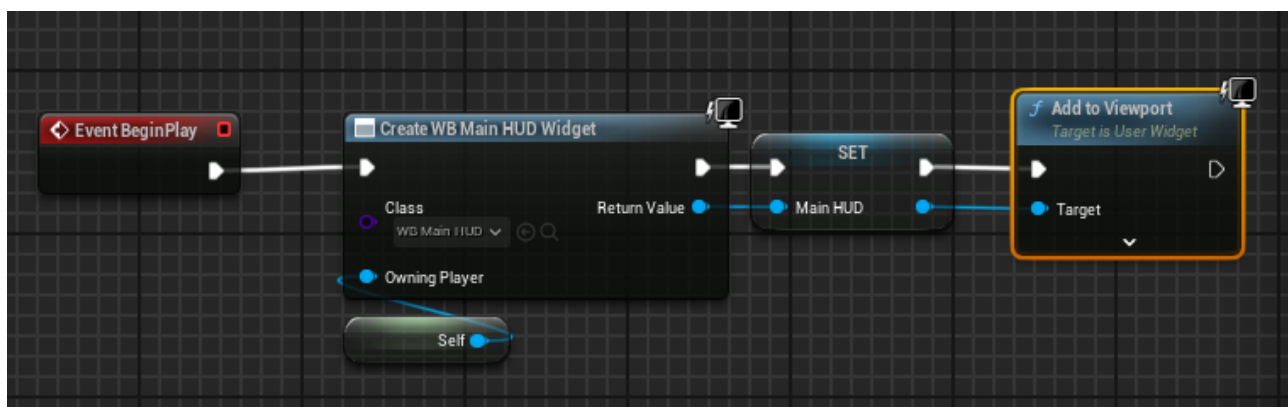


Рисунок Б.3 – Створення та виведення UI

		Матяш О.В.		
		Марчук Г.В.		
Змн.	Арк.	№ докум.	Підпис	Дата

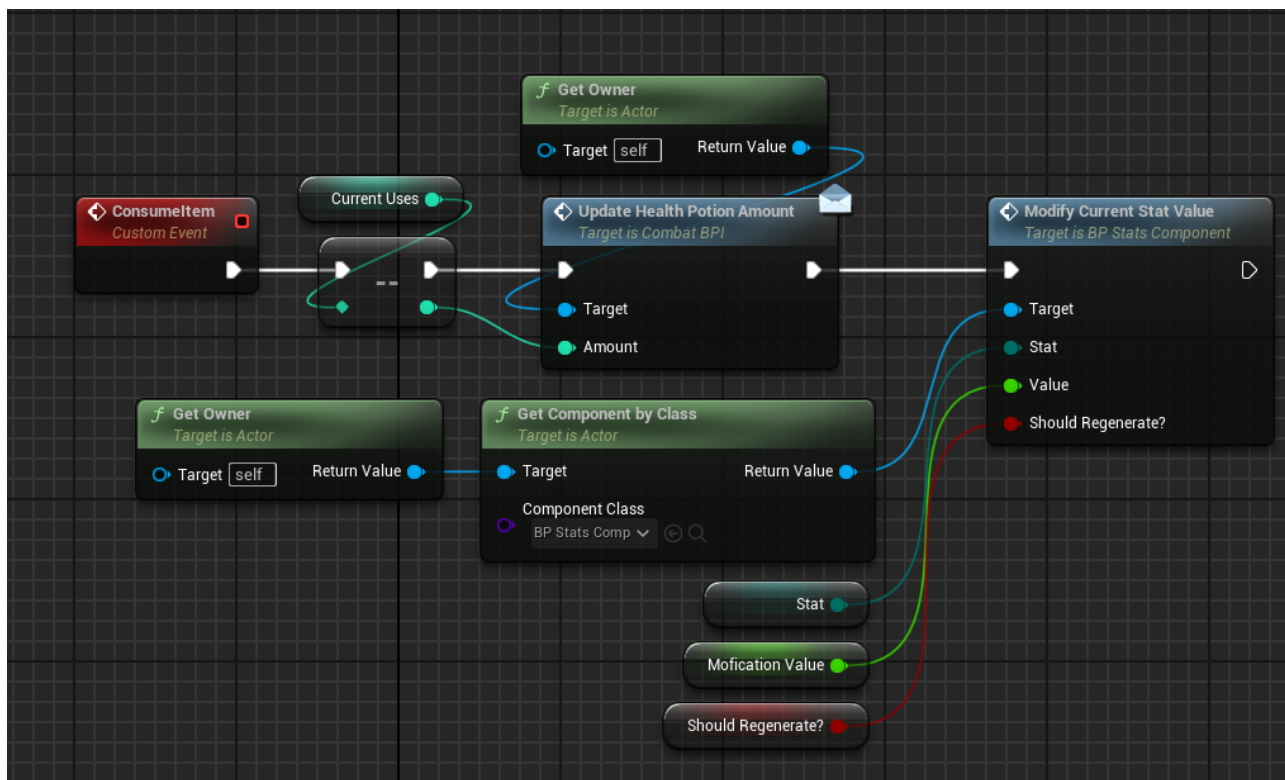


Рисунок Б.4 – Використання та підрахунок лікувального зілля

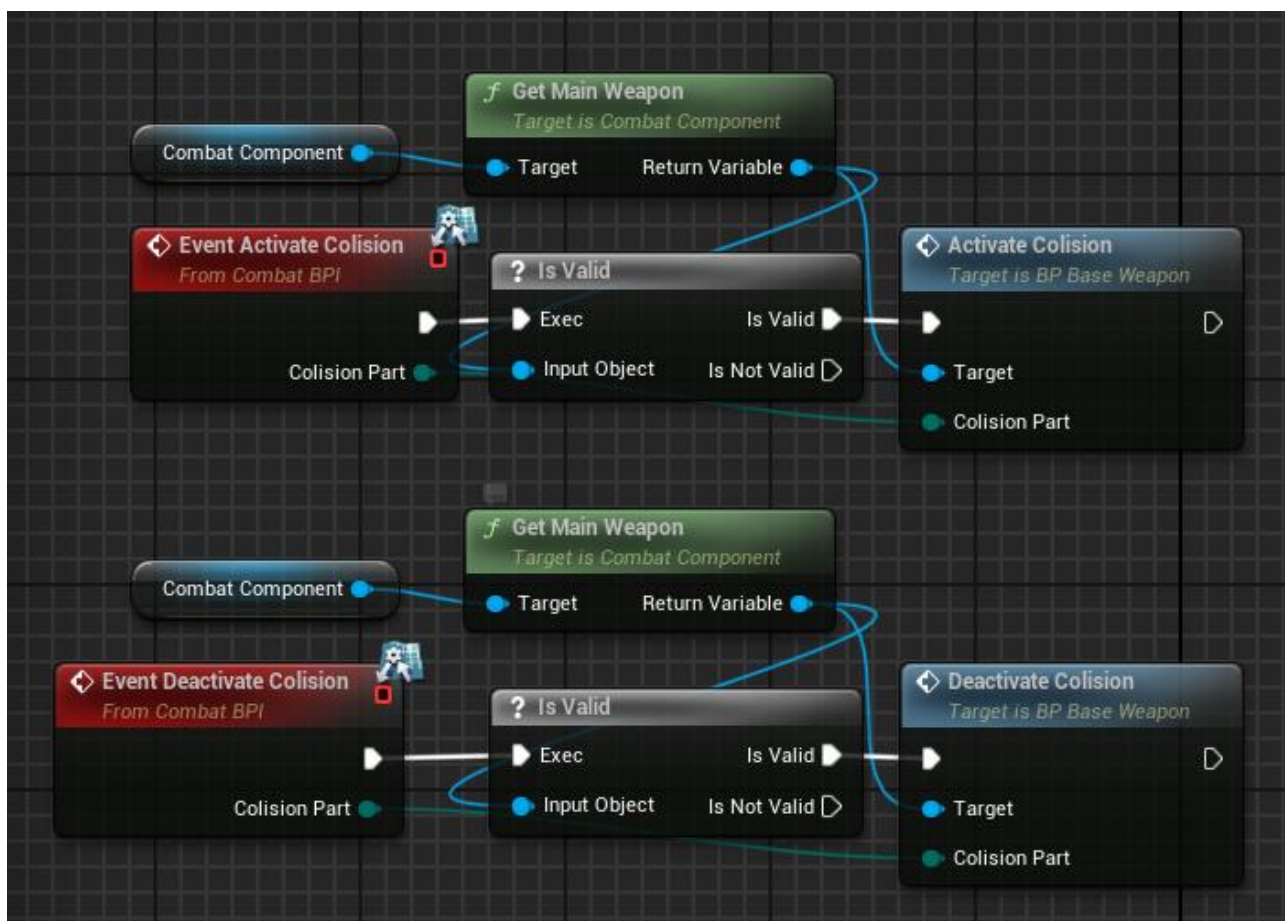


Рисунок Б.5 – Ввімкнення та вимкнення колізії зброї

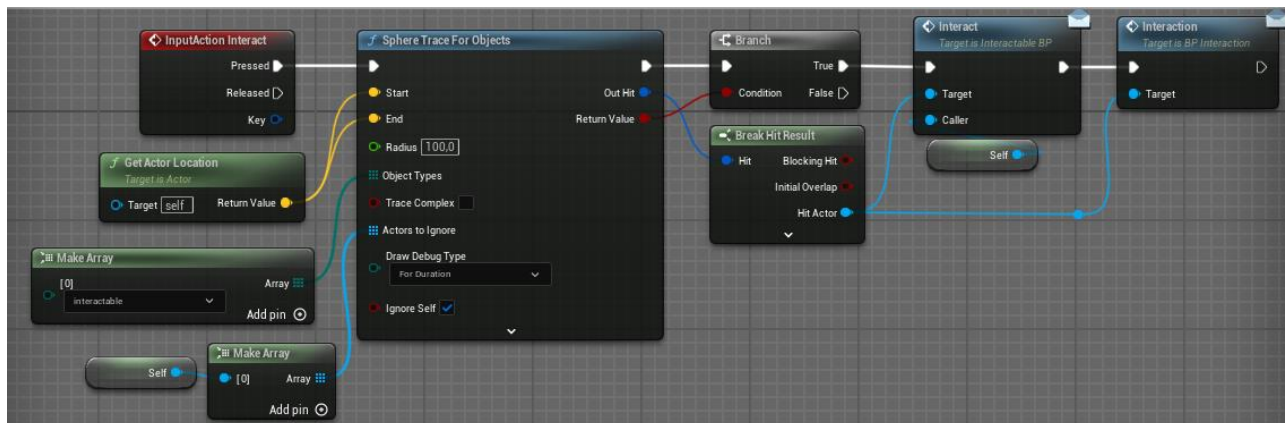


Рисунок Б.6 – Взаємодія з оточуючими об'єктами

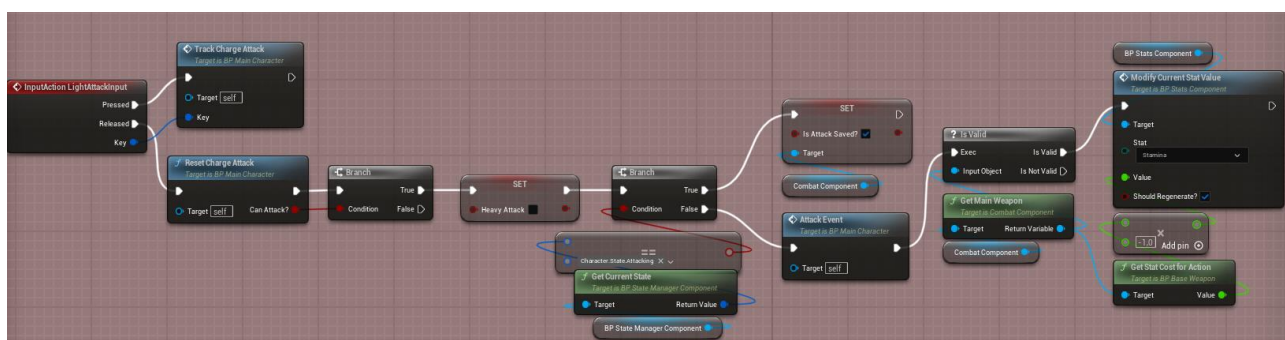


Рисунок Б.7 – Легка атака

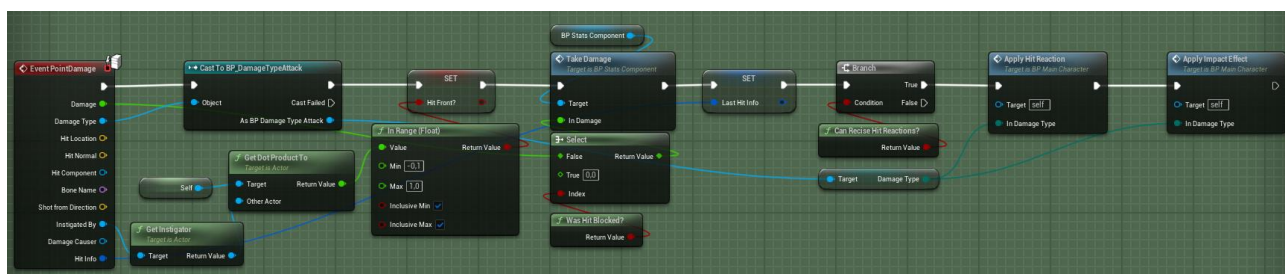


Рисунок Б.8 – Прорахунок отримання шкоди

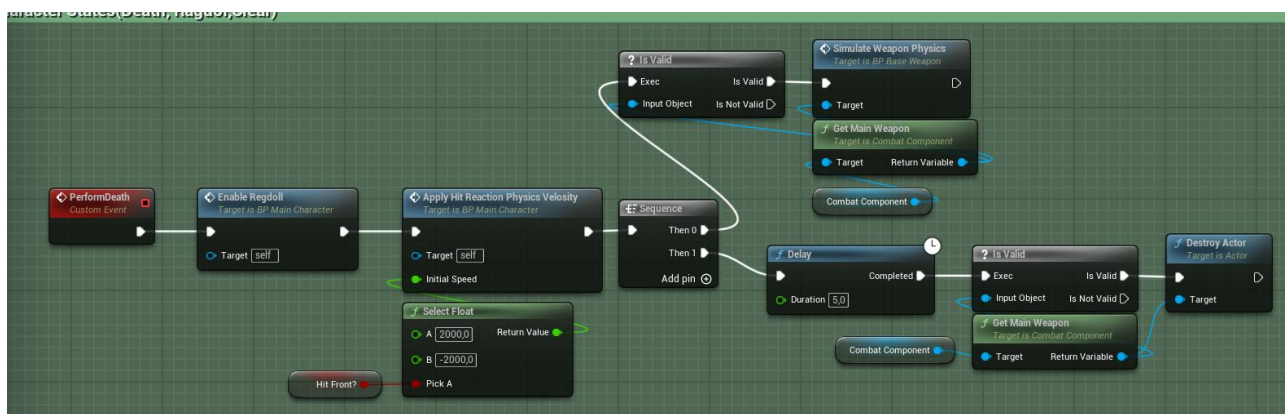


Рисунок Б.9 – Стан смерті