Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська дипломна робота на тему:

«РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ АВТОРСЬКОЇ ГРИ НА ІГРОВОМУ РУШІЇ UNREAL ENGINE 5»

Виконав: студент 4 курсу, групи КН-20Б
Спеціальність 122 —Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Суботіна І. А.
(призвище та ініціали)

Керівник: к.т.п., доцент каф.КН
Денисюк В. О.
(призвище та ініціали)

«07» 06 2024 р.

Рецензент: к.т.н., доцент каф. САІТ
Варчук І. В.
(призвище та ініціали)

«07» 06 2024 р.

Допущено до захисту
Зав. кафедри _____ Яровий А. А.
«07» 06 2024 р.

Вінниця ВНТУ — 2024 рік

Факультет інтелектуальних інформаційних технологій та автоматизації

Кафедра комп'ютерних наук

Рівень вищої освіти перший (бакалаврський)

Галузь знань – 12 Інформаційні технології

Спеціальність – 122 Комп'ютерні науки

Освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А.А.
«07» 03 2024р.

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Суботіну Іллі Андрійовичу

1. Тема роботи: Розробка програмного модуля авторської гри на ігровому рушії Unreal Engine 5

Керівник роботи: д.т.н., доцент Денисюк В. О. затверджені наказом вищого навчального закладу "11" 03 2024 року №80

2. Строк подання студентом роботи 27 05 2024 р.

3. Вихідні дані до роботи :

Кількість гравців – 2

Частота кадрів – 60 кадрів на секунду

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): аналіз предметної області, обґрунтування актуальності розробки програмного модуля авторської гри на ігровому рушії Unreal engine 5, аналіз існуючих концепцій програмного модуля авторської гри на ігровому рушії Unreal engine 5, розробка і проєктування програмного модуля авторської гри на ігровому рушії Unreal engine 5.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):схеми користувацького інтерфейсу, алгоритми загальної роботи та роботи ігрової системи, модель реалізації ігрової системи.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Денисюк В. О., к.т.н., доцент		

7. Дата видачі завдання 07.03.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітки
		початок	закінчення	
1	Аналіз предметної області	06.05.2024	10.05.2024	
2	Постановку задачі	11.05.2024	15.05.2024	
3	Проектування моделі авторської гри;	15.05.2024	24.05.2024	
4	Розробку програмного модуля авторської гри;	25.05.2024	29.05.2024	
6	Оформлення додатків	30.05.2024	02.06.2024	
7	Розробка інструкції користувача	03.06.2024	04.06.2024	
8	Оформлення матеріалів до захисту БДР	05.06.2024	06.06.2024	

Студент _____ Суботін І. А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Денисюк В.
(підпис) (прізвище та ініціа

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 69 сторінок формату А4, на яких є 65 рисунки 3 додатки список використаних джерел містить 15 найменувань.

У бакалаврській дипломній роботі було проведено детальний аналіз сфери ігрової розробки та програмних засобів ігрової розробки. Описано об’єкт, предмет, завдання та методи дослідження. Сформульовано мету дослідження – іновація програмного модуля авторської гри з використання ігрового рушія Unreal engine 5. Проаналізовано аналоги та зроблено висновок, що дана розробка є актуальною.

Гра була реалізована на ігровому рушію Unreal engine 5 з використання візуальної мови програмування Blueprint. Кінцевий програмний продукт відповідає завданням та меті.

ANNOTATION

The bachelor’s thesis consists of 69 pages of A4 format, on which there are 65 figures, 3 additions, the list of used sources contains 15 names.

A detailed analysis of the field of game development and game development software was carried out in the bachelor thesis. The object, subject, tasks and research methods are described. The purpose of the research is formulated - the innovation of the software module of the author's game using the game engine Unreal engine 5. Analogues are analyzed and it is concluded that this development is relevant.

The game was implemented on the Unreal engine 5 game engine using the Blueprint visual programming language. The final software product meets the tasks and purpose.

Зміст

ВСТУП

Актуальність. Сучасний світ комп'ютерних ігор є однією з найбільш динамічних та швидкозростаючих галузей індустрії розваг. З кожним роком ігрові технології вдосконалюються, відкриваючи нові можливості для розробників і забезпечуючи гравцям захоплюючі та інноваційні досвід. У цьому контексті створення авторської гри є складним, але водночас надзвичайно цікавим і творчим завданням.

Особливо при використанні Unreal Engine 5 - останньої версії одного з найпотужніших і найпопулярніших ігрових рушіїв. З моменту свого випуску UE5 революційно змінив підхід до створення ігор завдяки новаторським технологіям і покращеному інструментарію для розробників.

Unreal Engine 5 відомий завдяки своїм широким спектром інструментів для розробки ігор. Рушій підтримує розробку для різних платформ, включаючи ПК, консолі, мобільні пристрої та віртуальну реальність. Це дозволяє розробникам створювати ігри будь-якого масштабу — від невеликих інді-проектів до масштабних ААА-релізів.

Окрім стандартних інструментів розробки ігор Unreal Engine 5 також володіє унікальними технологіями, як nanite та lumen. Перший дозволяє аналізувати та обробляти меші спеціальним чином суттєво знижуючи навантаження на апарат користувача не впливаючи на деталізацію проекту.

Lumen - це система глобального освітлення на основі трасування променів, що дозволяє створювати високоякісне та реалістичне освітлення.

Тема актуальна обумовлена тим, що сучасні технології дозволяють розробникам створювати ігри високої якості навіть без великих ресурсів, а авторські проекти часто стають джерелом нових ідей та тенденцій в індустрії.

Мета та завдання дослідження є іновація програмного модуля авторської гри з використання ігрового рушія Unreal engine 5, що призвиде до

збільшить рівень продуктивності та зменшить апаратні вимоги вимоги та реалізувати можливість багатокористувацької гри.

Основними задачами роботи є:

- Провести аналіз існуючих ігор у вибраному жанрі та методи їх реалізацій.
- Розробити структуру модуля авторської гри на ігровому рушії Unreal engine 5.
- Обґрунтувати вибір алгоритмів роботи програмного модуля авторської гри на ігровому рушії Unreal engine 5.
- Розробка інтуїтивно зрозумілого користувацького інтерфейсу та його інтеграція у гру на ігровому рушії Unreal engene 5.
- Розробити алгоритми модуля авторської гри на ігровому рушії Unreal engine 5;
- Виконати програмну реалізацію модуля авторської гри на ігровому рушії Unreal engine 5.

Предметом дослідження – алгоритми та програмні засоби для створення програмного модуля авторської гри на ігровому рушії Unreal engine 5.

Об'єктом дослідження є процес створення програмного модуля авторської гри на ігровому рушії Unreal engine 5.

Очікується, що результати цієї роботи зроблять внесок у розуміння ефективних підходів до розроки ігор на Unreal engine 5, а також допоможуть розробникам створювати більш якісні та комерційно успішні продукти.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ІГРОВА РОЗРОБКА

1.1 Аналіз галузі ігрової розробки

Історія ігрової розробка бере свій початок, аж з 1950-их років коли однією з найперших була створена гра "OXO" (1952) для комп'ютера EDSAC, а вже через 10 років з’явились перші комерційні аркадні ігри, але справжня ера аркадних ігор розпочалась 1972, коли компанія Atari випустила легендарну "Pong".

Наступні 10 років прозвуть золотим віком аркадних ігор з випуском "Space Invaders" від компанії Taito 1978 року. А вже через 2 роки вийде "Pac-Man" від Namco, яка стала однією з найуспішніших аркадних ігор усіх часів, після неї гра "Donkey Kong" від Nintendo представила світу персонажа Маріо у 1981 році.

Та вже у наступних роках це привиде до маштабної кризи аркадних ігор.Надмірна кількість низькоякісних ігор призвела до краху ринку відеоігор у Північній Америці. Багато компаній зазнали банкрутства, і індустрія потребувала змін.

І вони прийшли, адже у 1985 році випустились нові консолі Nintendo та культова "Super Mario Bros.", а всього через 5 років свої консолі випустять Sega Genesis, Super Nintendo Entertainment System (SNES), Sony PlayStation та Nintendo 64.

1990-ті - це початок переходу до 3D-графіки та перша поява таких мастадонтів ігрової розробки, як Unreal Engine та Quake. Ці двигуни не тільки змінили спосіб створення ігор, але й вплинули на зростання ігрових спільнот завдяки можливостям модифікації ігор [1].

Черговим переломним моментом стали 2000-ні саме цей період характеризується зростанням мобільних ігор та масовим поширенням інтернету, що сприяло розвитку мережевих мультиплеєрних ігор. Також

з'являються двигуни, такі як Unity, які надають інструменти для розробки ігор на різні платформи, що значно спрощує вихід на глобальний ринок.

Тоді ж у 2000-них роках в світ вийдуть шедеври комп'ютерного гейммингу таких як "The Sims" (2000), "Grand Theft Auto III" (2001) та "World of Warcraft" (2004) ознаменує початок ігрової експансії.

Сьогодні важко уявити собі світ без комп'ютерних ігор, в кожній сім'ї є одна або кілька людей , які захоплюються якоюсь грою, будь то World of Warcraft, чи League of legend чи якоюсь іншою, але їх вплив на сучасний світ не піддається сумнівам. Сотні міжнародних компаній витрачають мільйони доларів на розробку чергової гри, тисячі програмістів створюють нові та нові ігрові світи та радують фанатів по всьому світу.

Кожного року виходять все новіші та новіші ігри та кожна із них привносить в індустрію щось нове, у 2003 році вийшла DotA, що тоді ще була невеликою фанатською картою, але всього за пару років вона покладе початок одному з найпопулярніших жанрів на наступних 10 років -MOBA.

А вже на наступний рік вийшла не меньш відома Half-Life 2, а на наступний Resident Evil 4, що можна сказати пр наступні роки. З кожним роком ігрова індустрія росте і розвивається у 2010-ті розробники отримали доступ до сучасних технології VR та AR, а також постійне оновлення таких двигунів, як Unreal Engine 5 та Unity, що відкрило нові можливості для іммерсивного геймплею. Розвиток штучного інтелекту та машинного навчання також починає впливати на розробку ігор, роблячи їх більш реалістичними та адаптивними до поведінки гравців.

На даний момент близько 40% населення планети грає у комп'ютерні, геймпаді чи телефоні ігри. Це більше 3 мільярдів людей по всьому світу. З них майже 90% припадає на комп'ютерний або телефонний геймінг, та лише 10% на геймпади та інші пристрої [2].

Комп’ютерні ігри дозволяють розвивати моторику рук, реакцію, стратигічне мислення… За допомогою них можна розказувати історію та вчити водіню.

Однак ніхто не замислюється наскільки важким та примхливим є процес розробки ігор адже для цього потрібна велика та злагоджена команда спеціалістів: розробників, аніматорів, сценаристів тощо…

Розробка ігор є складним і багатогранним процесом, який вимагає різноманітних навичок та інструментів. Ось основні елементи, які потрібні для розробки ігор:

1. Ідея та концепт гри:

- Сюжет і сценарій: для створення хорошої гри потрібно розробити історію персонажів, світу та навіть прості історії, які наповняють ігровий світ роблячи його живим.
- Механіка гри: потрібно також визначитись з цікавими та в міру складними правилами та механіками гри.
-Концепт-арт: не потрібно також забувати про візуальну складову ігрової розробки, а в наш час вимоги до графіки в іграх неймовірно завищенні.

2. Знання інструментів для розробки

- Ігровий рушій: програма для створення ігор. Популярні рушії включають Unity, Unreal Engine, Godot.
- Мови програмування: найпоширеніші мови для розробки ігор включають C#, C++, Python, JavaScript.
- Графічні редактори: програми для створення графіки і анімацій, такі як Adobe Photoshop, Blender, Autodesk Maya.

3. Розробка гри

- Програмування: написання коду для реалізації ігрової логіки, фізики, штучного інтелекту та інших аспектів гри.
- Дизайн рівнів: створення ігрових рівнів та середовищ, де гравці взаємодіють.
- Графіка та анімація: розробка 2D або 3D моделей, текстур, анімацій.

- Звук і музика: створення звукових ефектів та музичного супроводу.

4. Тестування

- Бета-тестування: залучення гравців для перевірки гри на помилки і збір відгуків.
- Виправлення помилок: виправлення знайдених помилок і оптимізація гри.

5. Реліз і маркетинг

- Випуск гри: Публікація гри на платформах, таких як Steam, App Store, Google Play.
- Маркетинг: Промоція гри через соціальні мережі, огляди, трейлери та інші маркетингові інструменти.

6. Пост-релізна підтримка

- Оновлення та патчі: Випуск оновлень для виправлення помилок, додавання нового контенту.
- Зворотній зв'язок: Робота з відгуками гравців для покращення гри.

Розробка ігор вимагає багатьох навичок, починаючи від програмування і графічного дизайну до звукового оформлення та маркетингу. Однак, з правильними інструментами і навчальними матеріалами, цей процес може бути дуже цікавим і захоплюючим.

1.2 Аналіз відомих ігрових рушіїв

Найпопулярніші ігрові рушії на даний момент це без сумніву Unreal engine 5 та Unity, за ними йдуть менш відомий Godot, але у чому їх відмінність?, та який із них найкращий?

Unreal Engine 5 — це новітня версія популярного ігрового двигуна від компанії Epic Games, яка пропонує численні технологічні інновації та поліпшення порівняно з попередніми версіями. Як і будь-яка технологія, вона має свої переваги та недоліки [3].

Переваги:

1. Графіка високої якості: Unreal Engine 5 використовує новітні технології, такі як Nanite і Lumen, які дозволяють створювати детальні візуальні ефекти та реалістичне освітлення у реальному часі.

2. Скейлинг продуктивності: Двигун оптимізований для різних платформ, від високопродуктивних ігрових ПК до мобільних пристроїв, дозволяючи розробникам створювати ігри, які працюють ефективно на багатьох типах обладнання.

3. Зручні інструменти для розробників: Unreal Engine 5 пропонує потужні інструменти для розробки, включаючи розширену систему блупринтів (blueprints), яка дозволяє розробникам створювати складні взаємодії і логіку без глибоких знань у програмуванні.

4. Широка підтримка спільноти та ресурсів: Оскільки Unreal Engine є одним із найпопулярніших ігрових двигунів, велика спільнота розробників та безліч доступних навчальних ресурсів можуть допомогти новачкам освоїти двигун.

5. Підтримка різних типів контенту: Unreal Engine 5 відмінно підходить не тільки для ігор, але й для розробки віртуальної реальності, анімації, симуляцій та інших інтерактивних додатків.

Недоліки:

1. Високі вимоги до обладнання: Для розробки та запуску ігор на Unreal Engine 5 потрібні потужні комп'ютери, особливо для роботи з передовими візуальними та освітлювальними ефектами.

2. Складність навчання: Хоча блупринти дозволяють спростити програмування, загальна складність інструментів та можливостей двигуна може бути складною для новачків або для тих, хто не має технічного досвіду.

3. Ресурсоємність: Ігри, створені на Unreal Engine 5, можуть вимагати значних ресурсів, особливо у візуальних та освітлювальних ефектах, що може вплинути на продуктивність на менш потужних системах.

4. Ліцензування: Хоча Unreal Engine безкоштовний для навчання та розробки, Epic Games вимагає певного відсотка від доходів від комерційних проектів, що перевищують встановлений поріг доходів.

Висновок:

Unreal Engine 5 продовжує бути одним із найпотужніших інструментів для розробки ігор, але важливо враховувати його потенційні складнощі та обмеження, особливо при плануванні масштабних або вимогливих проектів.

Unity — це популярний ігровий двигун, який використовується для створення ігор різного масштабу, від невеликих інді-проектів до великих комерційних ігор. Як і будь-який інструмент розробки, Unity має свої переваги та недоліки [3].

Переваги:

1. Кросплатформеність: Unity дозволяє розробникам легко експортувати ігри на більшість популярних платформ, включаючи Windows, MacOS, Linux, iOS, Android, а також на консолі та VR системи. Це робить його ідеальним вибором для проектів, що цілять на широку аудиторію.

2. Широка підтримка спільноти та ресурсів: Unity має одну з найбільших спільнот розробників, з багатою бібліотекою плагінів, готових активів та навчальних матеріалів, доступних через Unity Asset Store та інші платформи.

3. Дружній до початківців інтерфейс: Unity пропонує візуально орієнтований інтерфейс та систему перетягування компонентів, що робить його доступнішим для новачків та недосвідчених розробників.

4. Підтримка 2D та 3D: Unity відмінно підходить як для 2D, так і для 3D ігрових проектів, надаючи гнучкість у виборі стилю та формату ігри.

5. Вбудовані інструменти аналітики та монетизації: Unity пропонує ряд інструментів для відстеження поведінки гравців та ефективної монетизації ігор.

Недоліки:

1. Оптимізація: Ігри, створені на Unity, можуть страждати від проблем з продуктивністю, особливо на слабших або старіших пристроях. Це може вимагати додаткових зусиль з оптимізації від розробників.

2. Ліцензування: Хоча основна версія Unity безкоштовна, підписка на розширені функції та інструменти може бути дороговартісною, особливо для невеликих команд або інді-розробників.

3. Якість графіки: В той час як Unity може створювати візуально привабливі ігри, воно може відставати від більш потужних двигунів, таких як Unreal Engine, з точки зору графічних можливостей для дуже вимогливих проектів.

4. Стандартизація інструментів: Деякі інструменти та редактори в Unity можуть бути менш інтуїтивно зрозумілими або потребувати налаштувань для певних типів проектів, що може збільшити криву навчання.

Висновок:

Unity залишається одним із найпопулярніших ігрових двигунів на ринку, здатним задовольнити потреби різних типів проектів і розробників, але важливо зважати його сильні та слабкі сторони при виборі платформи для вашого наступного ігрового проекту.

Godot — це відкритий ігровий двигун, який швидко набирає популярність серед розробників ігор завдяки своїм унікальним функціям та гнучкості. Як і будь-який інструмент, Godot має свої переваги та недоліки, які важливо розглянути при виборі двигуна для вашого проекту [4].

Переваги:

1. Відкритий код: Godot є повністю відкритим двигуном з активною спільнотою, що постійно працює над поліпшенням і розширенням його можливостей. Це дозволяє розробникам вільно модифікувати і адаптувати двигун під свої потреби.

2. Підтримка кросплатформеності: Ігри, розроблені в Godot, можуть бути експортовані на багато платформ, включаючи Windows, MacOS, Linux, iOS, Android, HTML5 та інші.

3. Гнучка система сцен та вузлів: У Godot використовується унікальна система управління сценами та вузлами, яка дозволяє легко створювати комплексні ієрархічні структури, забезпечуючи велику гнучкість при проектуванні ігор.

4. Інтегрована мова скриптування GDScript: Godot використовує GDScript, мову, схожу на Python, яка легка в навчанні та ефективна для розробки ігор.

5. Мінімалістичний та інтуїтивно зрозумілий інтерфейс: Інтерфейс Godot є чистим і організованим, що сприяє легкому навчанню та ефективній роботі.

Недоліки:

1. Обмежені візуальні можливості: Хоча Godot підходить для багатьох типів ігор, його графічні можливості можуть бути обмежені порівняно з більш потужними двигунами, як Unreal чи Unity, особливо для проектів з високою візуальною складністю.

2. Менша кількість ресурсів та готових активів: Хоча спільнота Godot активно розвивається, наразі вона менша порівняно з Unity чи Unreal, що може означати меншу кількість доступних ресурсів, навчальних матеріалів та готових активів.

3. Підтримка сторонніх інтеграцій: Godot може мати обмеження у підтримці плагінів та інтеграції зі сторонніми сервісами, які часто використовуються в інших двигунах.

4. Масштабування проектів: Великі та вимогливі проекти можуть зіткнутися з труднощами через обмеження двигуна, особливо в контексті продуктивності та управління ресурсами.

Висновок:

Godot є чудовим вибором для розробників, які шукають відкритий, гнучкий і доступний ігровий двигун, але важливо розуміти його обмеження, особливо при плануванні складних або візуально вимогливих проектів.

Усі наведенні рушії мають свої переваги та недоліки в особливості для розробки мобільних додатків використання складного Unreal Engine 5 є недоцільним, як і використання Godot та Unity для великих складних проектів.

1.3 Вибір жанра гри та аналіз існуючих аналогів

Серед нескінченості різних жанрів комп'ютерних ігор таких як: гонки, рпг, файтинг, стратегія, шутер, симулятор та інші… Мою увагу приволік такий жанр, як roguelike.

Roguelike – це жанр гри в якій, визначними особливостями якого є випадкове, процедурне створення рівнів та перманентна смерть персонажа у випадку поразки.

Як правило під час цього гравець перемагає ворогів та в якості винагороди отримає досвід та підвищення рівня. Збільшим рівнем збільшується різноманіття здібностей, які може вивчити персонаж. Але й вороги з часом стають сильніші та їх кількість зростає, це дозволяє створити складності впродовж усього часу гри.

Вперше подібна ідея була реалізована у грі 'Vampire Survivor' там до елементів жанру "roguelike" було додано система рівнів та нескінченний потік ворогів, що створило неймовірне поєднання простоти входу та великий простір для поліпшення своїх навичок.

Рисунок 1.1 — Vampire Survivor.

Після неї було створено безліч інтерпретацій геймплею Vampire Survivor, однак однією найвідоміших з них є Magic survivor. Вона не привнесла багато нового, лише доробила усі аспекти попередника.

Розширена система рівнів та здібностей дозволяла створити такого бійця, якого ти сам захочеш, а урізноманітнення ворогів додало стратегічної глибини та на останок було значно покращена графіка, все це зробило гру відомою.

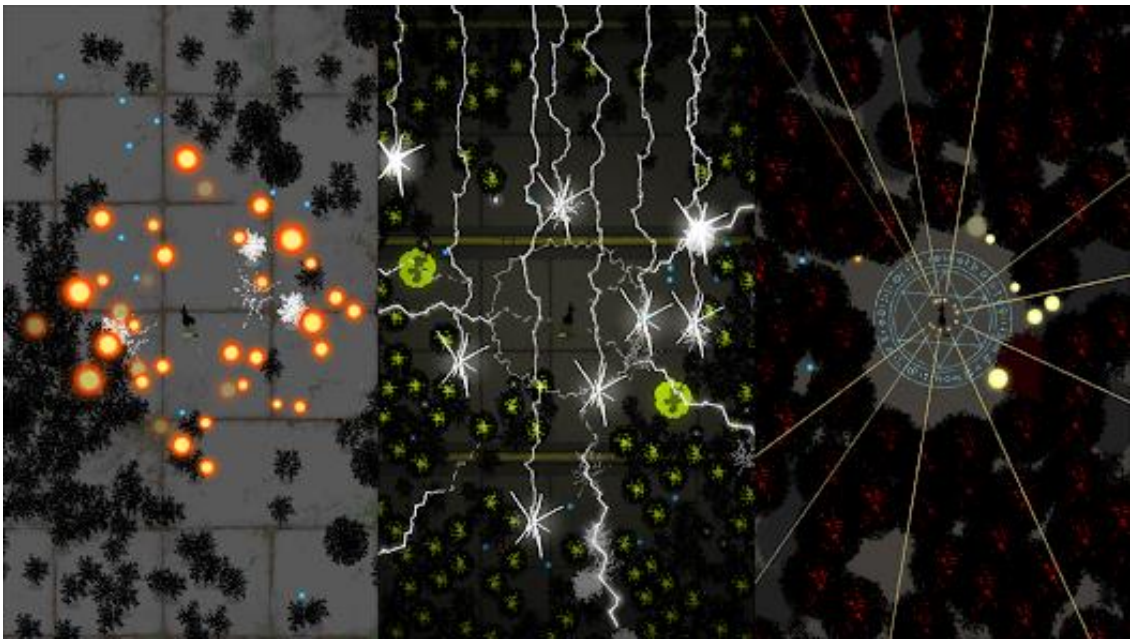

Рисунок 1.2 — Magic Survivor.

З більш відалених аналогів подібного: Archero. Відмовившись від нескінчених орд ворогів Archero поділив гру на рівні із зростаючим рівнем складності, також додавши систему обладнання, боса кожну 5-ту кімнату та зміну локації гри. На жаль із обновлення гемплею та графіки також було встроєна велика кількість різноманітної реклами та пропозиціями витратити гроші.

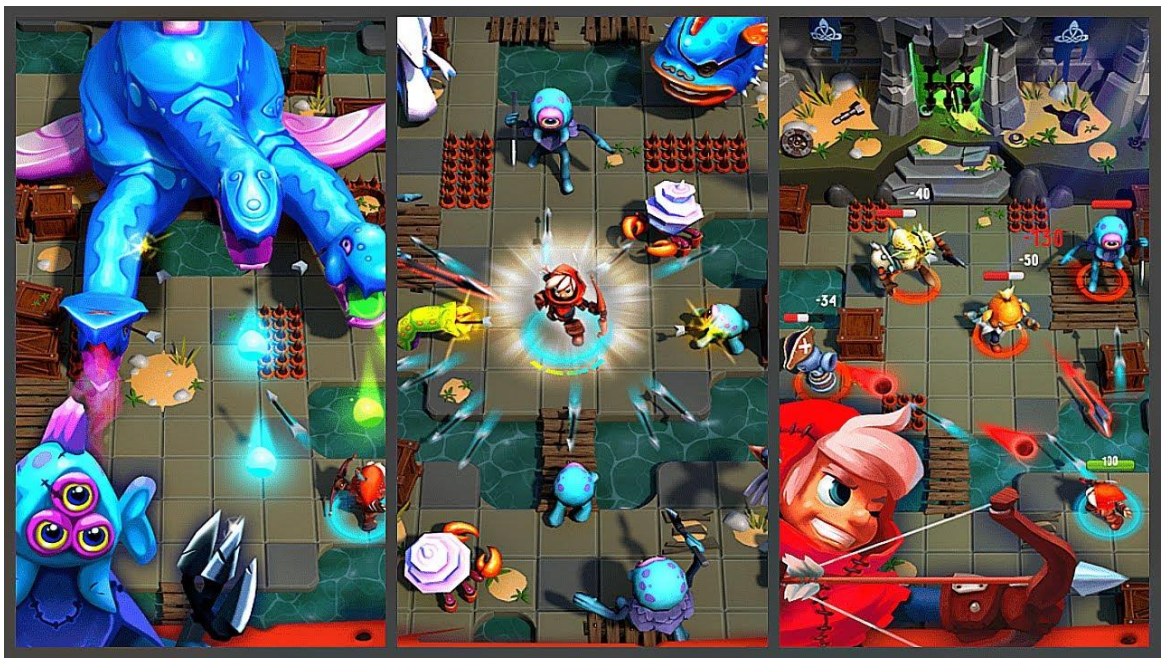

Рисунок 1.3 — Archero.

Також несподіваний погляд на цей жанр представила Soul Knight. Вона не тільки поділила гру на рівні, але і рівні поділила на кімнати в яких відбувалась сутичка, добавив до цього великий арсенал найрізноманітнішої зброї та перелік небезпечних ворогів.

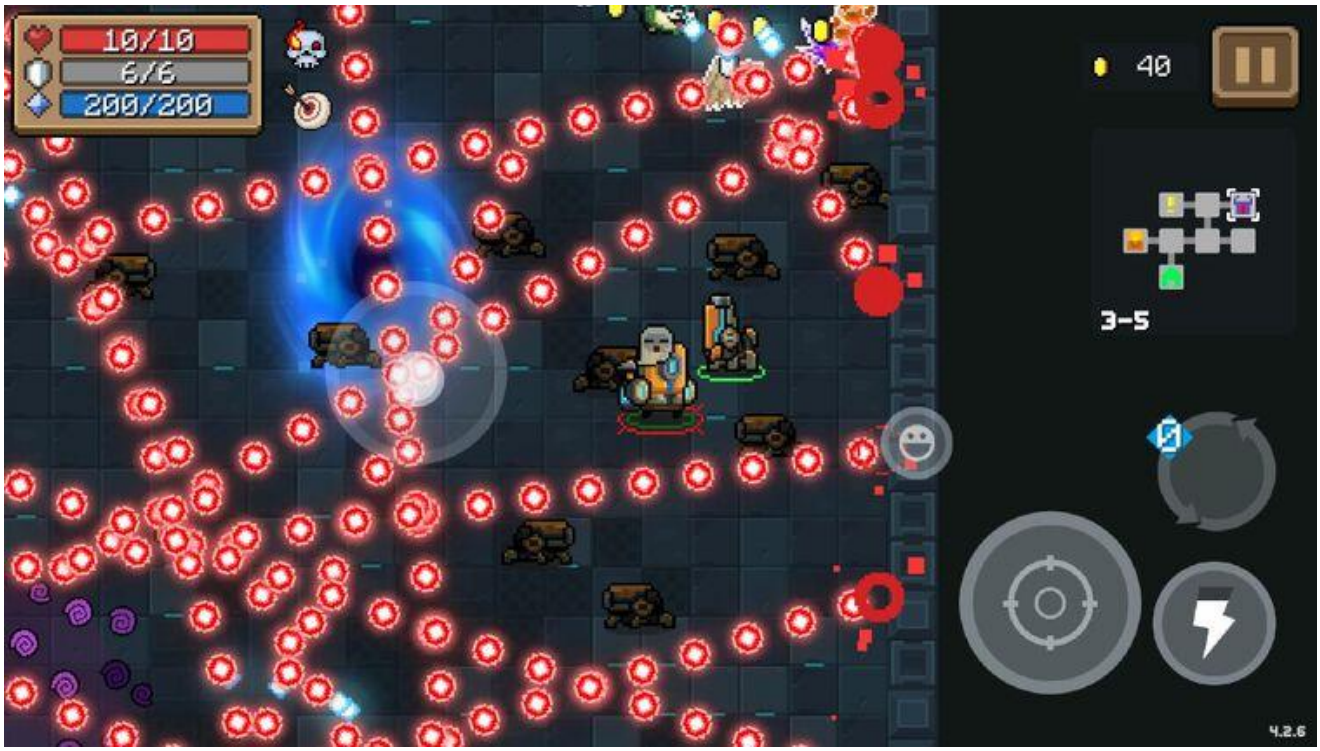

Рисунок 1.4 — Soul Knight.

Для комп’ютера в такому жанрі, відзначається такі ігри, як: The Binding of Isaac та Hades їхня суть мало чим відрізняється від попереднім: інші здібності персонажів, інша зброя та монстри.

Але навіть тут з’явились нові елементи, наприклад у The Binding of Isaac - є не тільки закручене та неймовірно схована сюжетна лінія, а й неймовірно велика кількість випадкових, дивних та навіть моторошних предметах, які до того ж навіть не мають опису, тому зрозуміти що робить той чи інший предмет

неймовірно складно. Та саме це наділяє The Binding of Isaac своїм унікальним шармом.

Рисунок 1.5 — The Binding of Isaac.

Hades" — це популярна відеогра, розроблена і видана студією Supergiant Games. Вона отримала високі оцінки від гравців та критиків за кілька ключових аспектів.

Захоплюючий сюжет: гра базується на грецькій міфології, що додає їй особливого шарму. Гравець бере на себе роль Загрея, сина Аїда, який намагається втекти з підземного світу.

Високий рівень реіграбельності: завдяки випадковій генерації рівнів і постійного зростання складності. Кожен забіг відрізняється від попереднього, що підтримує інтерес гравців навіть після багатьох годин гри

Геймплей та механіки бою: Бої в "Hades" швидкі, динамічні та дуже задовільні. Гравці можуть обирати різноманітну зброю та здібності, що дозволяє їм експериментувати з різними стилями бою.

Рисунок 1.6 — Hades .

Як ми можемо спостерігати усі ігри у 2D або 2.5D, не одна з них не була реалізована для 3D, також більш новіші ігри, як Soul Knight та Archero, мають суттєву проблему з продуктивністю, тому було прийняте рішення обрати жанром авторської гри саме Roguelike. При використанню ігрового рушію Unreal Engine 5 виправлення проблеми продуктивності та перехід до 3D графіки є можливим завдяки широкому спектру інструментів для розробки ігор [5].

1.4 Висновок

У розділі було розглянуто та проаналізовано галузь ігрової розробки. Розглянуті частина навичок та інструментів потрібних щоб зробити хорошу гру Детально розглянуто найпопулярніші ігрові рушії та порівняно їх між собою. Були проаналізовані плюси та мінуси найвідоміши ігрових рушіїв.

Також було проаналізовано найвідоміші ігри в жанрі roguelike для телефону та комп’ютера та виділенні їх основні ознаки.

Оскільки всі аналоги ігор в жанрі roguelike зроблені в 2d та переважно на мобільних пристроях створення гри в жанрі «roguelike» на комп’ютер є доцільним при використання 3d та ігрового рушію unreal engine 5.

2. РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ ПРОГРАМНОГО МОДУЛЯ ГРИ НА UE5

2.1 Огляд особливостей обраного ігрового рушія

Unreal Engine 5 (UE5) — це сучасний і високотехнологічний рушій для створення відеоігор, розроблений компанією Epic Games. Він відомий своїми передовими можливостями, які значно спрощують процес розробки ігор, забезпечуючи при цьому високу якість графіки і продуктивність. Ось кілька ключових аспектів Unreal Engine 5.

Серед різних сторін рушія одним з перших на очі попадається унікальна система візуального програмування Blueprint, яка дозволяє втілювати складні системи навіть без глибоких навичок у програмуванні, що значно розширює можливості при розробці.

Після Blueprint, слід згадати про унікальну систему наслідування, завдяки якій і досягається висока продуктивість та структурованість у великих проектах на Unreal Engine 5 [6].

Ось основні класи Unreal Engine 5 та їх призначення:

1. Actor основний клас для всіх об'єктів, які можуть бути розміщені у світі. Усі об'єкти, які існують у грі, є похідними від цього класу.

2. APawn спеціалізований клас для будь-яких об'єктів, якими можуть керувати гравці чи штучний інтелект. Це підклас AActor, який від нього наслідується.

3. ACharacter підклас APawn, призначений для персонажів, які мають функціональність руху, включаючи анімацію та колізію.

4. UObject базовий клас для всіх об'єктів у UE5. Він надає базову функціональність і серіалізацію.

5. UComponent базовий клас для всіх компонентів, які можуть бути додані до актора для надання додаткової функціональності.

6. USceneComponent підклас UComponent, який має трансформацію (позицію, обертання та масштабування) у світі.

7. USkeletalMeshComponent спеціалізований компонент для рендерингу і анімації скелетних сіток.

8. UStaticMeshComponent компонент для рендерингу статичних сіток, які не мають анімації.

9. APlayerController клас, який відповідає за прийом введення від гравця і керування Pawns. Зазвичай використовується для управління камерою і обробки вводу.

10. AGameMode клас, який визначає правила гри, геймплейні механіки і керування рівнями.

11. AGameState клас для зберігання інформації про стан гри, доступної всім клієнтам у багатокористувацькій грі.

12. APlayerState клас для зберігання інформації про стан конкретного гравця, яка синхронізується між клієнтом і сервером.

с13. UUserWidget базовий клас для створення користувацьких інтерфейсів за допомогою Unreal Motion Graphics (UMG).

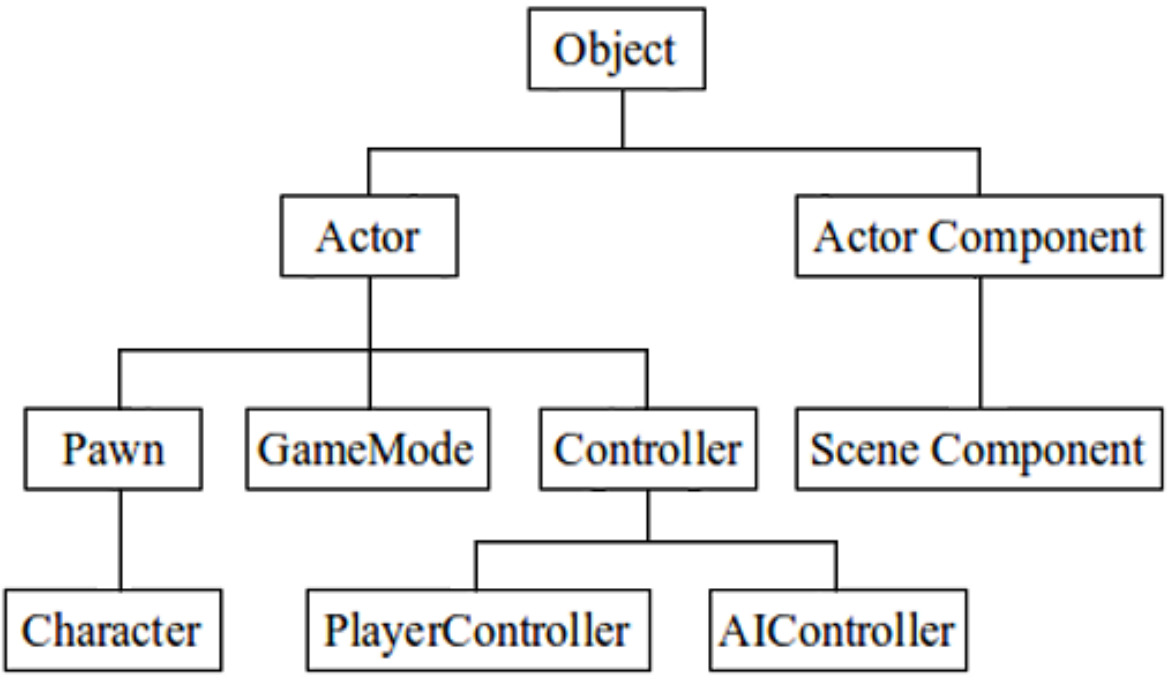

Рисунок 2.1 — Дерево насладування класів Unreal Engine 5

2.2 Вибір мови програмування

Порівняння C++ та Blueprint в контексті розробки ігор, особливо використовуючи Unreal Engine, відкриває цікаві аспекти обох підходів до програмування. C++ є могутньою загальнопризначеною мовою програмування, тоді як Blueprint — це візуальна скриптова система, розроблена Epic Games для Unreal Engine [7].

C++

Переваги:

1. Швидкодія: Код на C++ зазвичай виконується швидше, оскільки він компілюється в машинний код. Це важливо для ігор, де вимоги до продуктивності високі.

2. Гнучкість та контроль: C++ надає розробникам повний контроль над системними ресурсами, включаючи обробку пам'яті та багаторівневу архітектуру програми.

3. Широке використання: C++ є однією з найбільш використовуваних мов у розробці високопродуктивного програмного забезпечення, включаючи ігри, що забезпечує велику кількість бібліотек і інструментів.

Недоліки:

1. Складність: C++ може бути складним для вивчення та використання через свою складну синтаксичну структуру та вимоги до управління пам'яттю.

2. Час розробки: Розробка на C++ займає більше часу, особливо при налагодженні та управлінні пам'яттю.

Blueprint

Переваги:

1. Доступність: Blueprint дозволяє розробникам швидко і інтуїтивно створювати ігрову логіку за допомогою візуального інтерфейсу, що особливо корисно для дизайнерів та початківців.

2. Швидкість розробки: Розробка за допомогою Blueprint може бути швидшою, оскільки вона забезпечує миттєву відповідь і легке налагодження.

3. Інтеграція з Unreal Engine: Blueprint глибоко інтегрований з Unreal Engine, що забезпечує гарну сумісність і оптимізацію.

Недоліки:

1. Обмежені можливості: Незважаючи на велику гнучкість, Blueprint має обмеження у виконанні складних завдань порівняно з C++.

2. Залежність від двигуна: Blueprint прив'язаний до Unreal Engine, що обмежує його використання тільки цим двигуном.

Після детального порівння двох мов програмування було принято рішення обрати мовою розробки Blueprint. Впершу чергу рішення обумовленне часом розробки гри, але також сутевою перевагою у сторону Blueprint є протота і доступність у використані

2.3 Розробка структури інтерфейсу гри

При проектуванні користувацького інтерфейсу необхідно враховувати кілька факторів: колір не повинен заважати сприйняттю тексту; інформативні елементи мають розташовуватися в полі зору та у відповідних місцях, їх колір не повинен зливатися з фоном. Це стосується, зокрема, функцій, доступ до яких повинен бути забезпечений, параметрів, які необхідно показати, та багатьох інших важливих для зручності користувача аспектів [8].

Розроблювана гра в жанрі «roguelike» призначена для використання на персональних комп'ютерах, тому користувачі матимуть у своєму розпорядженні великий монітор, а також пристрої введення типу миша та клавіатура.

Для проектування користувацького інтерфейсу складено список необхідних сцен для відображення елементів інтерфейсу.

1. Сцена “Головне меню” відображає головне меню.

2. Сцена “Гра” відображає інтерфейс гравця(його життя, досвід та витривальсть)

3. Сцена “Меню пауза” відображає меню під час паузи.

4. Сцена “ Підвищення рівня ” відображує меню із виборем складності.

На основі висунутих вимог розроблено схематичні зображення користувацького інтерфейсу ігрового додатку.

Схема сцени “Головне меню” приведене на рисунку 2.2

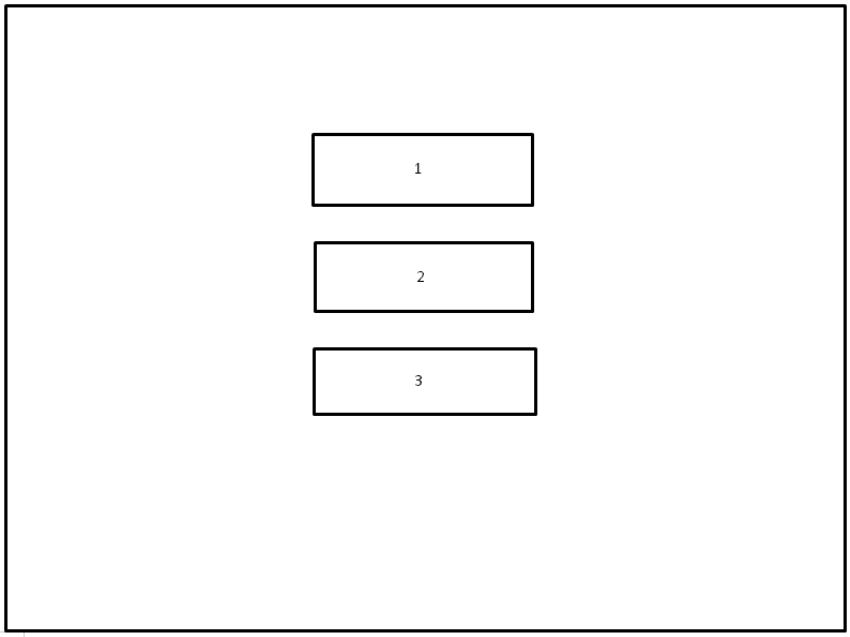

Рисунок 2.2 — Схема користувацького інтерфейсу сцени “Головне меню”

Опис елементів схеми відповідно до їх нумерації на рисунку 2.2:

1. Кнопка початку гри.

2. Кнопка вибору налаштувань.

3. Кнопка вихід з гри.

Схема сцени “Гра” приведене на рисунку 2.3

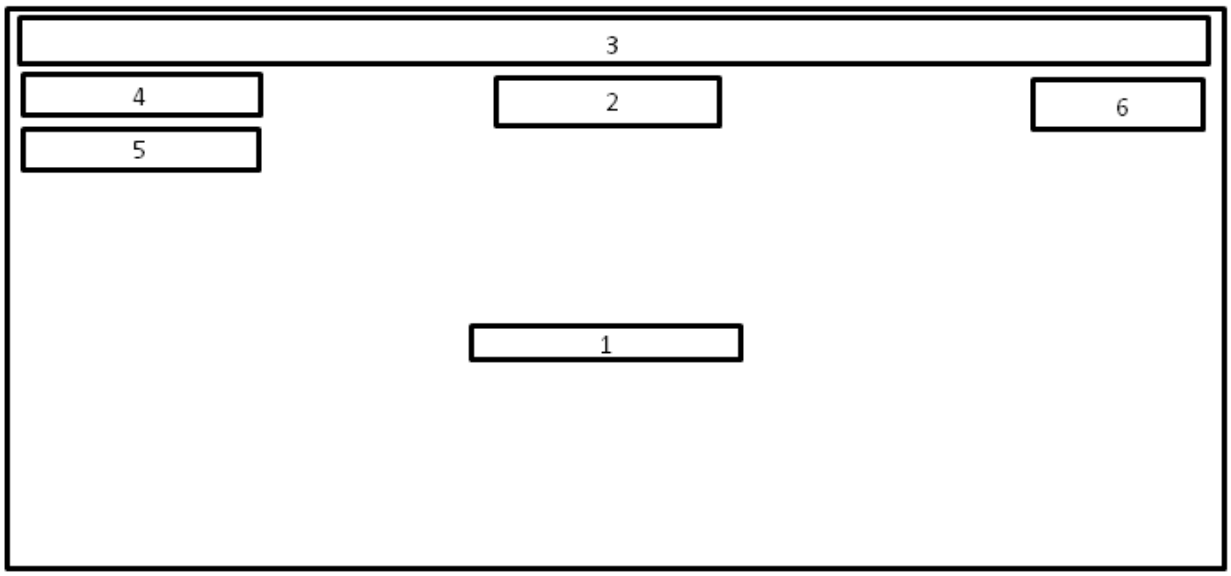

Рисунок 2.3 — Схема користувацького інтерфейсу сцени “Гра”

Опис елементів схеми відповідно до їх нумерації на рисунку 2.3:

1. Шкала здоров’я персонажа
2. Кількість часу з початку гри
3. Шкала досвіду персонажу
4. Відображення вивчених активних здібностей та їх рівня
5. Відображення вивчених пасивних здібностей та їх рівня
6. Кількість золота

Схема сцени “Меню пауза” приведене на рисунку 2.4

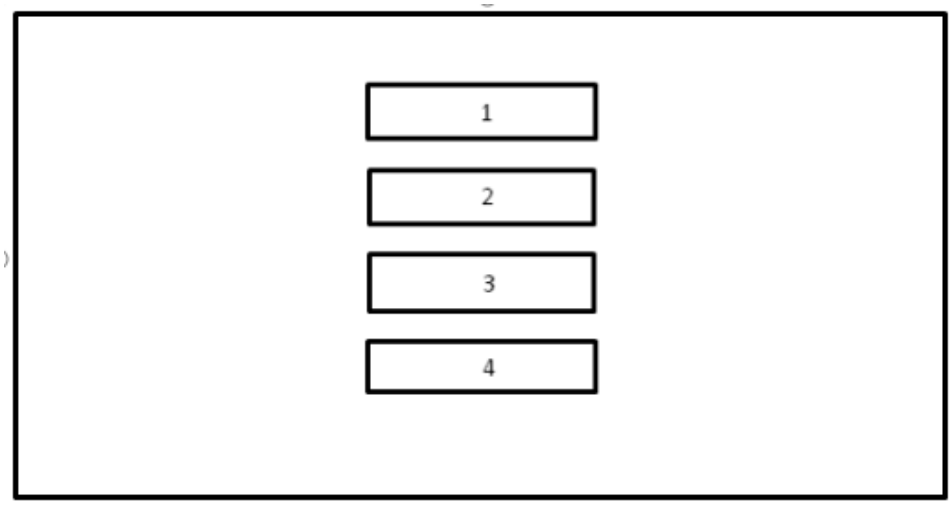

Рисунок 2.4 — Схема користувацького інтерфейсу сцени “Меню пауза”

Опис елементів схеми відповідно до їх нумерації на рисунку 2.4:

1. Кнопка продовжити гру
2. Кнопка перезапустити рівень
3. Кнопка вийти в головне меню
4. Кнопка вийти з гри

Схема сцени “Підвищення рівня” приведене на рисунку 2.5:

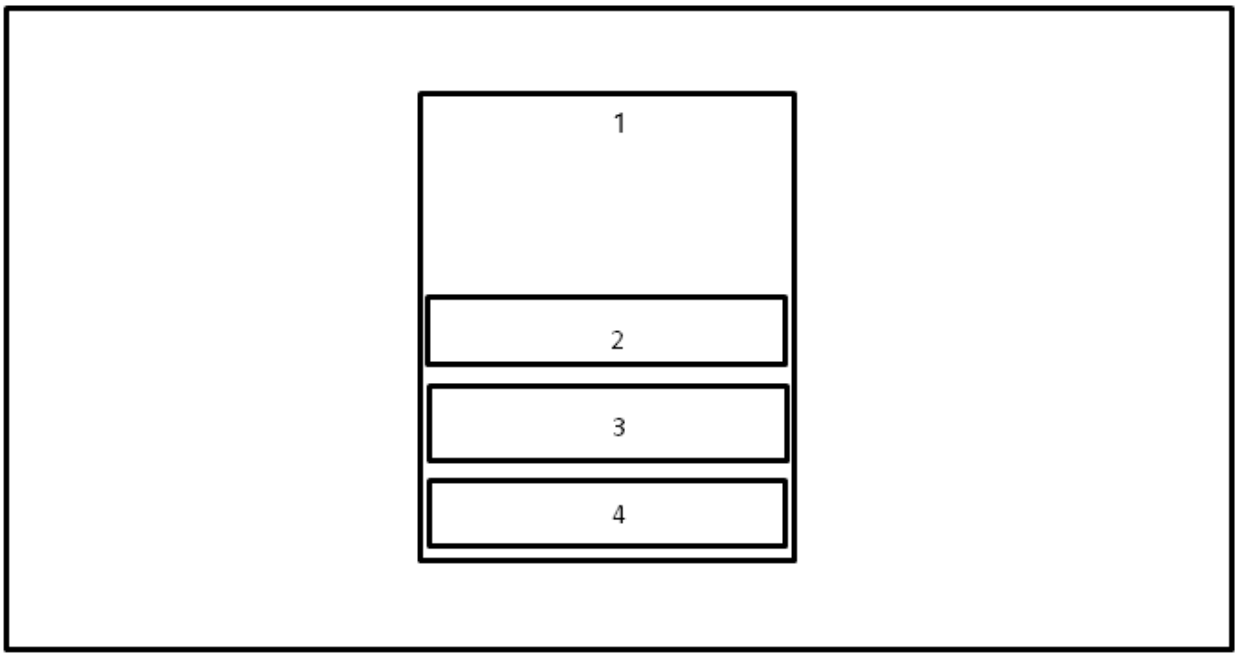

Рисунок 2.5 – Схема користувацького інтерфейсу сцени “ Підвищення рівня ”

Опис елементів схеми відповідно до їх нумерації на рисунку 2.5:

1. Вікно підвищення рівня
2. Вибір здібності 1 з описом
3. Вибір здібності 2 з описом
4. Вибір здібності 3 з описом

2.4 Розробка моделі авторської гри

Модель реалізації гри в жанрі «roguelike» розпочинається з сцени “Головне меню”, яка надає доступ до трьох кнопок: “Вихід з гри”, ”Вибір

складності” ”Початок гри”. Кнопка “Вихід з гри” здійснює завершення роботи гри, ”Вибір складності” дозволяє вибрати необхідний складність, а ”Початок гри” дозволяє відкрити ігрову карту гри та перейти до її сцени. Вихід з ігрової карти здійснюється шляхом переходу до сцени “Меню пауза” та подальшого переходу в “Головне меню” або виходу з гри. Також для виходу із ігрової сцени може здійснюватись через вивід результату гри.

На рисунку 2.6 наведено загальний алгоритм роботи гри в жанрі «roguelike».

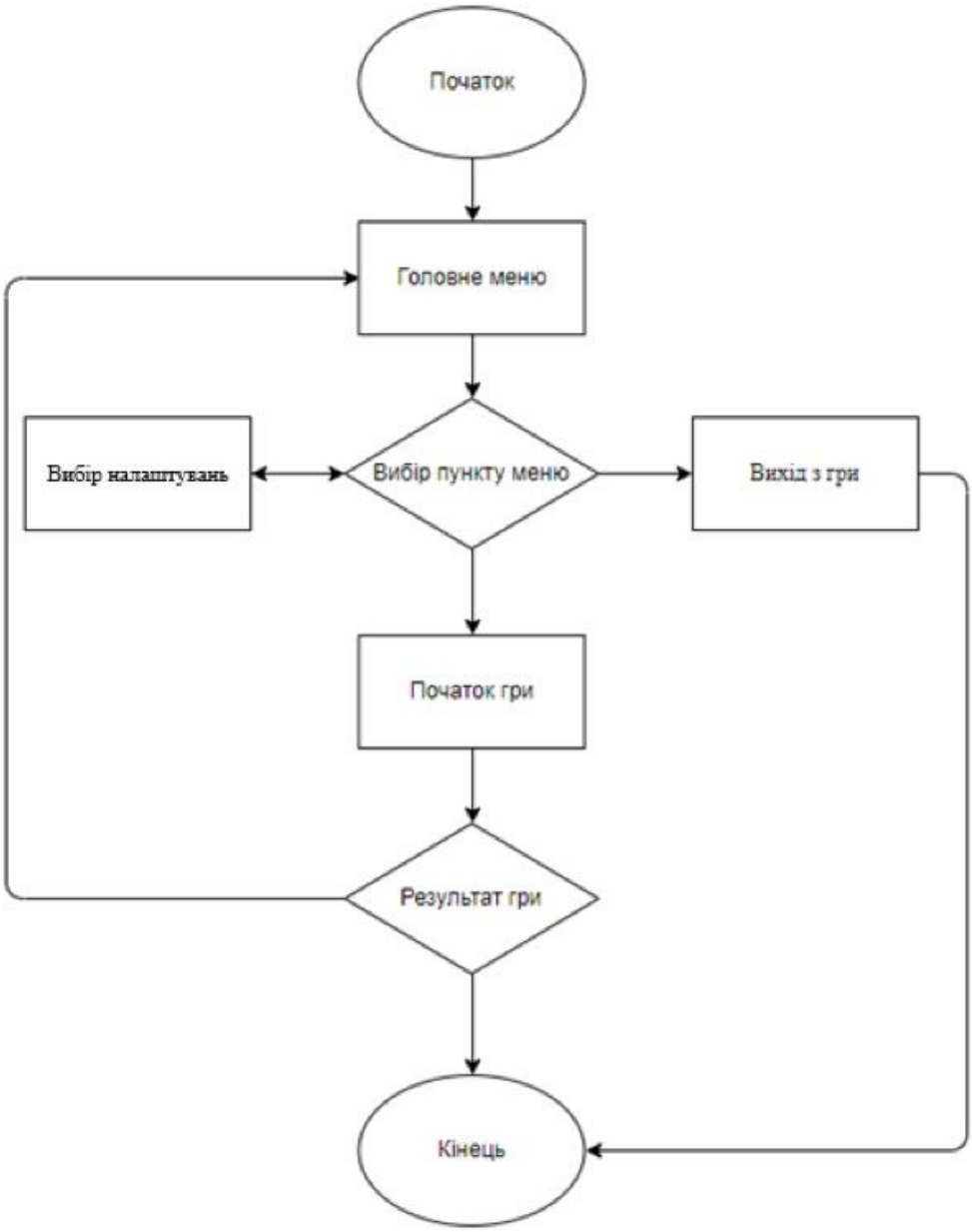

Рисунок 2.6 — Загальний алгоритм роботи гри в жанрі «roguelike»

Під час гри першим чином запускається таймер, який рахує час поточної гри. Далі всі вороги будуть ставати сильнішими в залежності від пройденого часу таймера, після чого гравець матиме змогу знайти ворогів та перемогти їх, отримавши бали досвіду за це. При отриманні достатньої кількості балів досвіду персонаж збільшить свій рівень та стане сильніше. Після чого настане повторення цього процесу до поразки гравця, далі у гравця є вибір перейти у головне меню, чи повторити ігровий сценарій ще раз [9].

На рисунку 2.7 наведено алгоритм реалізації гри в жанрі «roguelike»

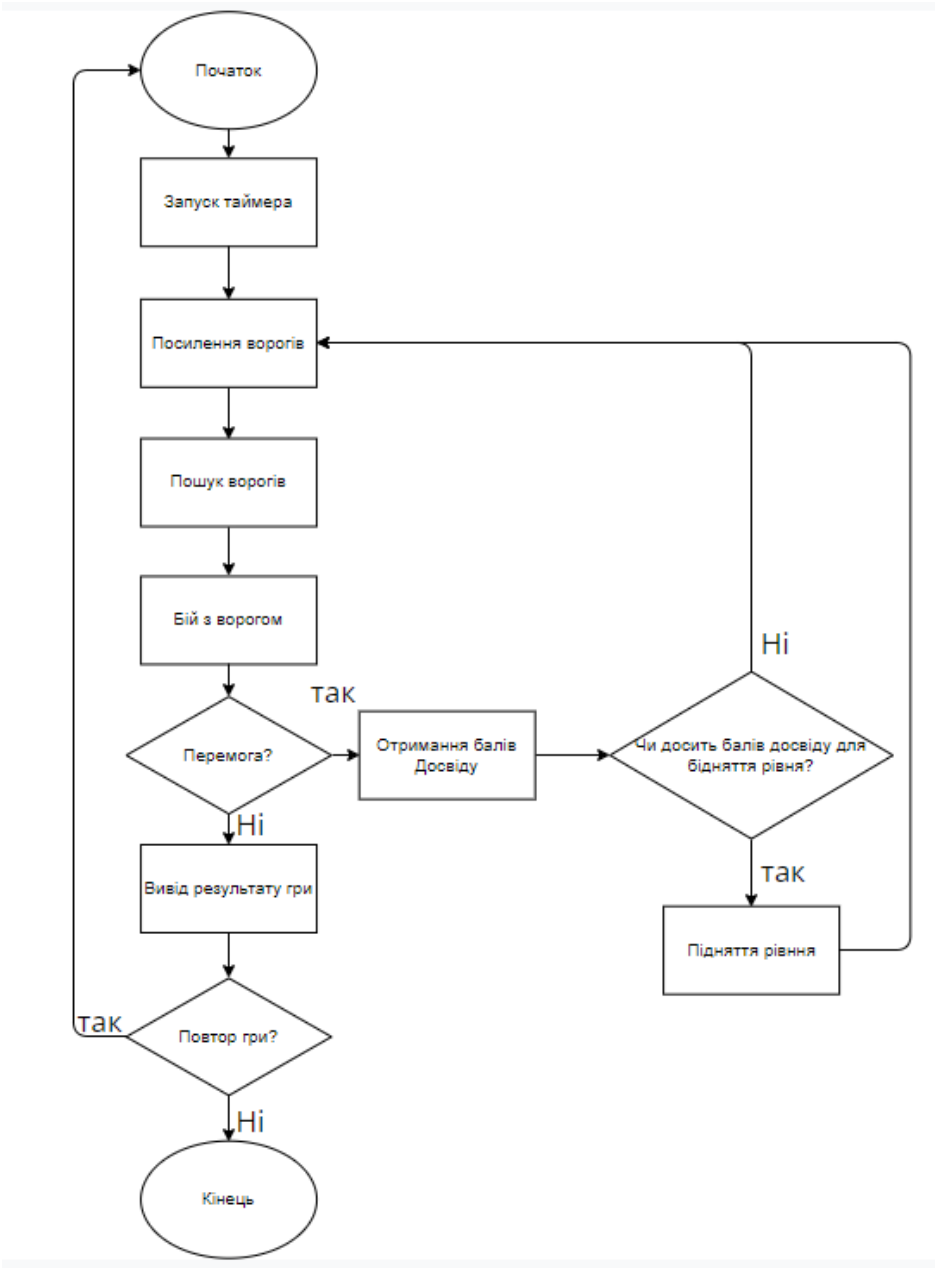

Рисунок 2.7 — Алгоритм реалізації гри в жанрі «roguelike»

Перед тим як гравець потрапляє на ігрову сцену, активується модуль налаштування рівня. Після цього ігрова сцена взаємодіє з модулем обробки рухів головного персонажа, який, у свою чергу, обмінюється даними з модулем обробки дій гравця. Завершення рівня веде до активації модуля, який підраховує результати [10].

На рисунку 2.8 представлена модель реалізації ігрової системи.

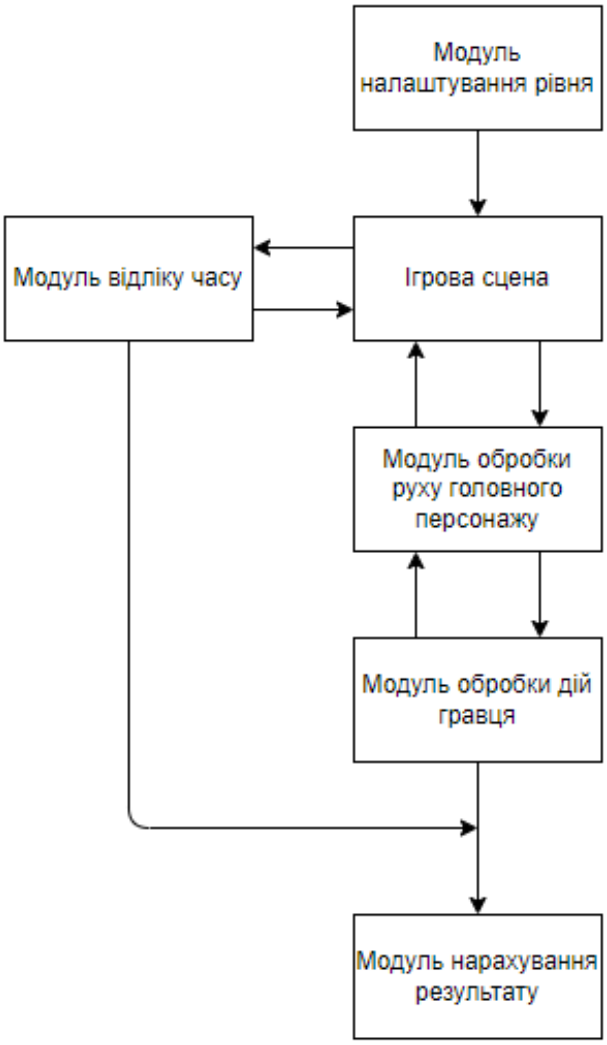

Рисунок 2.8 — модель реалізації гри в жанрі «roguelike»

2.5 Висновок

У другому розділі був здійснений вибір програмних засобів розробки гри. А саме був обрано ігровий рушій Unrial engine 5 завдяки своїй продуктивності та простоті розробки, також розглянуті особливості цього ігрового рушія, зокрема дерево наслідування класів та систему візуального програмування Blueprint. Було проведене порівняння Blueprint з C++, та обрано мовою розробки Blueprint через простоту використання та швидкість розробки.

Далі було проведена розробка та опис структури інтерфейсу гри, а саме “Головного меню”, Гри”, “Меню паузи” та “Вибору складності”. А також проведена розробка моделі реалізації ігрової системи з алгоритмами реалізації ігрової система та загальний алгоритм роботи гри.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІГРОВОГО ДОДАТКУ

3.1 Розробка модуля ігрового персонажа

Перед створенням персонажу потрібно створити проект та світ, для цього заходимо в ігровий рушій Unreal Engine 5, та оскільки метою роботи є створення гри в жанрі roguelike, то обираєм розділ Games – рисунок 3.1 в якому створюємо тип світу Blank – це пустий світ де створена тільки базова карта, також обирає Blueprint та відключаєм starter content. Результат створеного світу показаний на рисунку 3.2

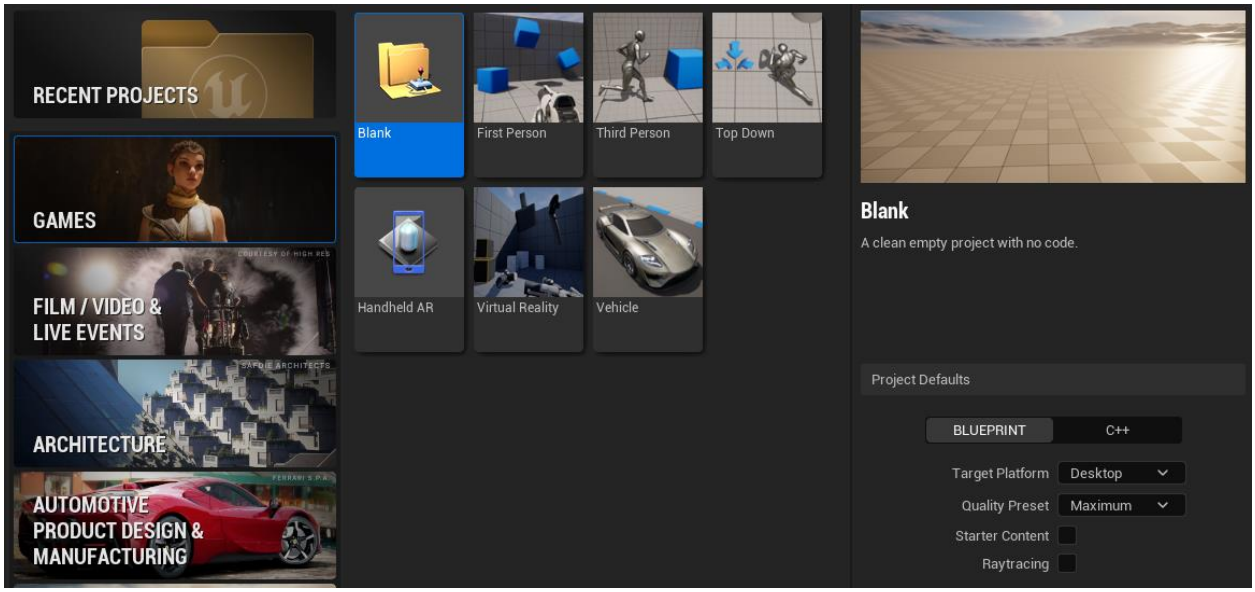

Рисунок 3.1 — Меню створення проекту Unreal Engine 5

Рисунок 3.2 — Створений світ

Далі створимо папку Blueprint та у ній створимо блупринт клас Character – це буде головний персонаж, яким буде керувати гравець. Назвемо його BP_Base_Character. Результат на рисунку 3.3

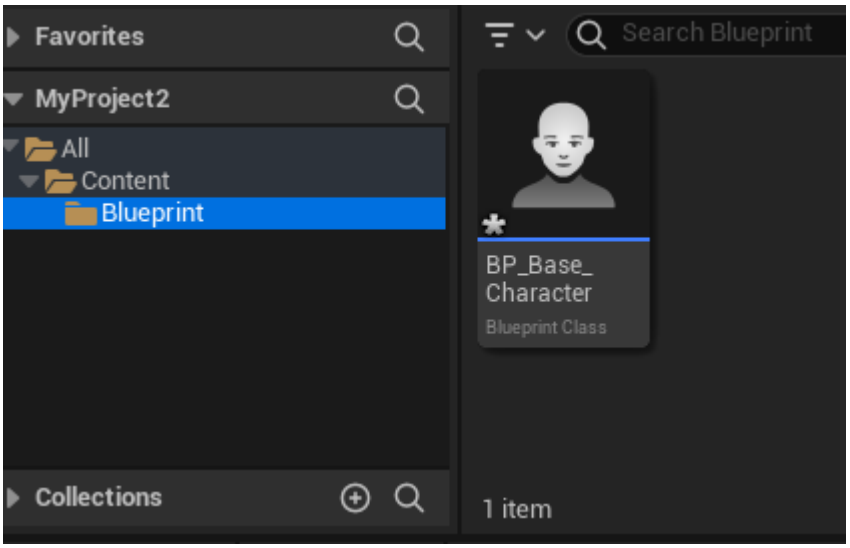

Рисунок 3.3 — Створення BP_Base_Character

Далі зайдемо у нього та у вікні Component добавимо компоненти SpringArm та Camera, за допомогою, яких ми налаштуємо камеру персонажа. Оскільки гра розрахована на вид зверху піднімаєм та віддаляєм камеру, рисунок 3.4

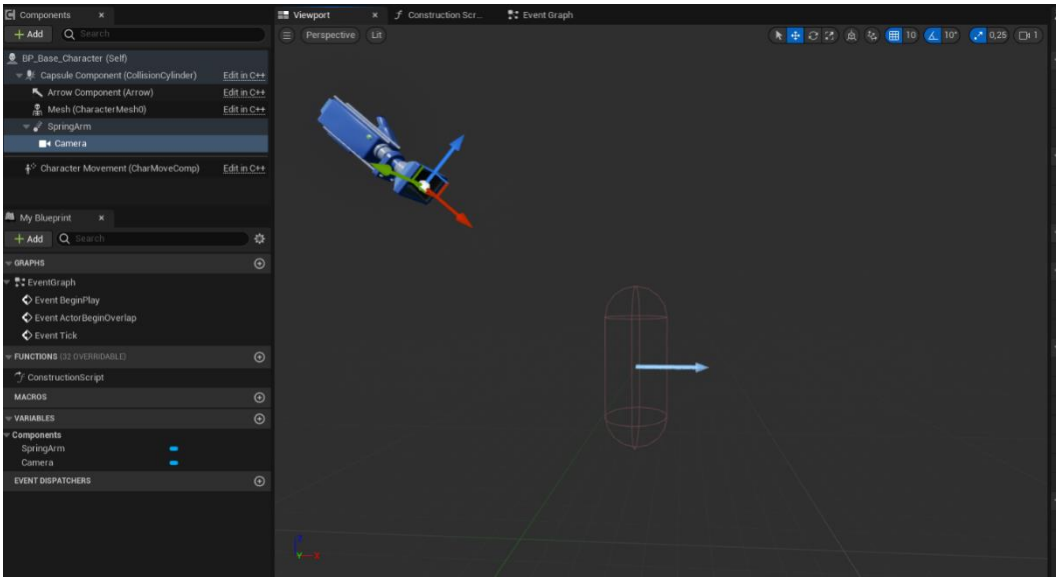

Рисунок 3.4 — Розділ Viewport BP_Base_Character

Далі додаймо персонажу можливість рухатись на кнопки WASD та стрілочки, для цього відкриєм Project Settings та зайдем у розділ Input, там відкриваєм Axis Mappings та додаємо 2 нових Axis: «Move Forward / Backward» та «Move Right / Left», після чого назначаємо клавіши для «Move Forward / Backward»: W, S, стрілки верх та вниз та аналогічно – «Move Right / Left»: A, D, стрілки вправо та вліво. Для S, D, стрілки вниз та вліво у пункті Scale введем значення -1, для інших залишим – 1.

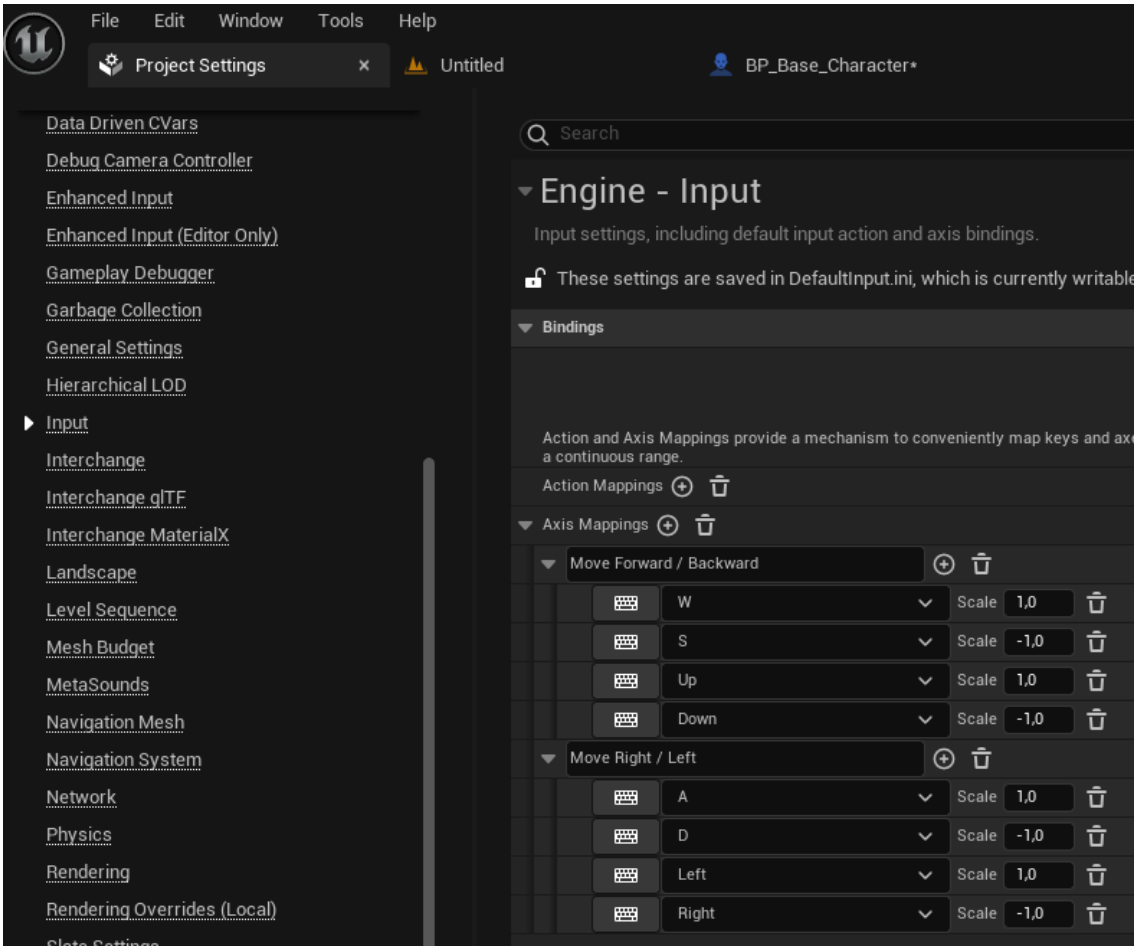

Рисунок 3.5 — Додаємо Input

Далі повертаємось у BP_Base_Character та переходим у розділ Event Graph, де створюємо Івенти «InputAxis» та за допомогою ноди «AddMovementInput», якій передаємо значення переднього/правого вектора гравця та значення Axis Value рисунок 3.6

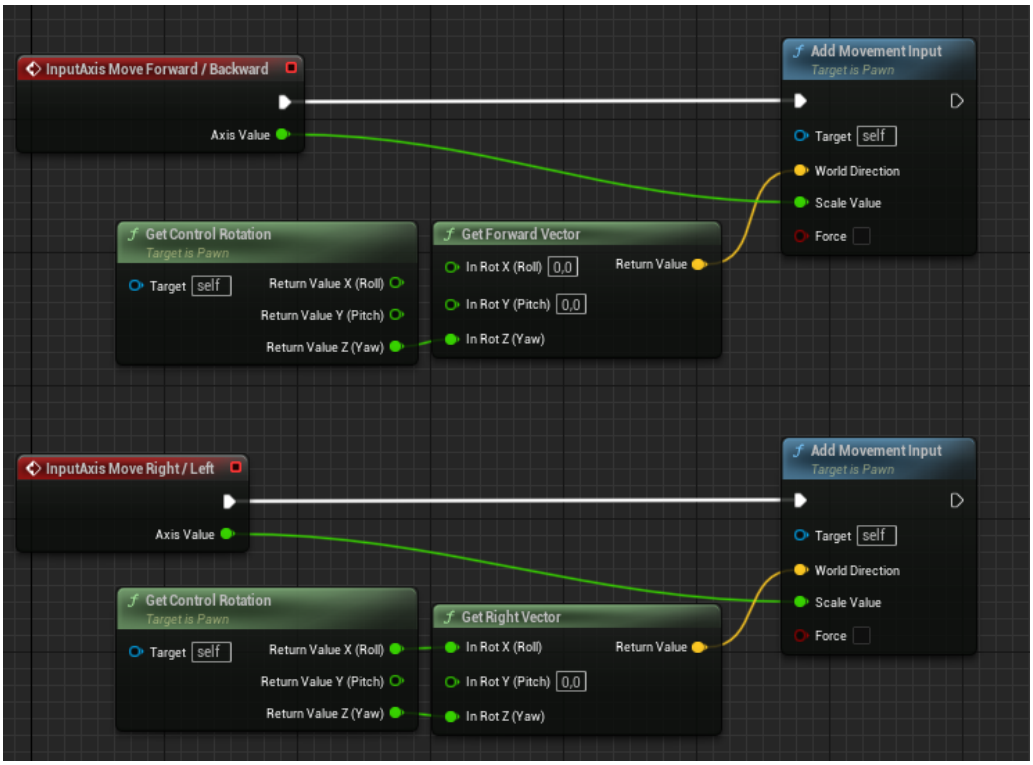

Рисунок 3.6 — Скрипт для пересування персонажа

Далі створимо віджет для відображення hp гравця, для цього створюємо нову папку Widgets та створюємо новий віджет WB_HealthBar, додаєм «Size Box» та «ProgressBar», після чого переходим у «ProgressBar» та робимо з пункту percent зміну за допомогою неї ми будемо регулювати рівень заповнення шкали.

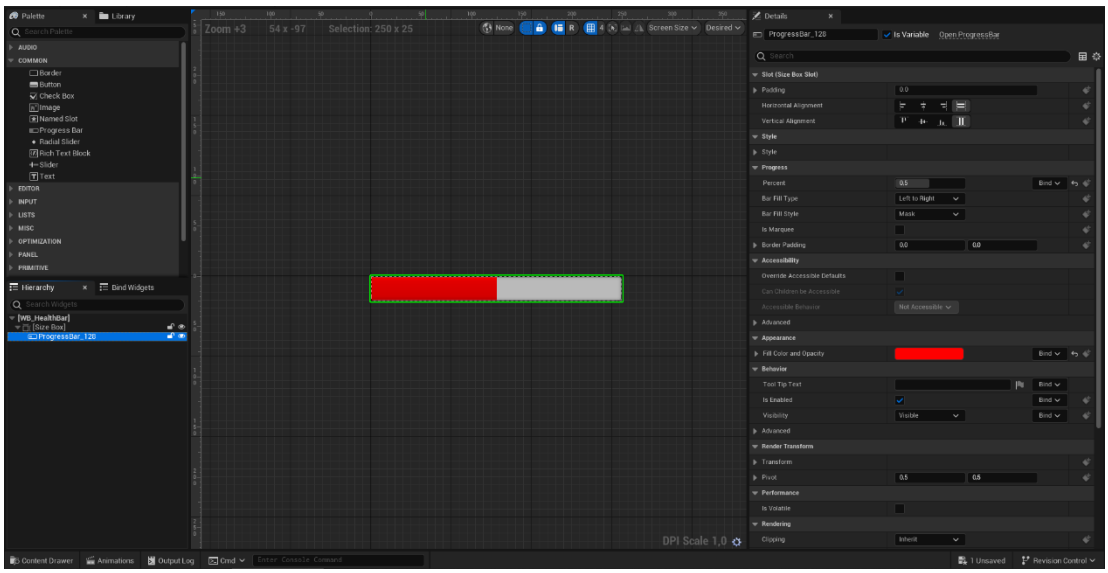

Рисунок 3.7 — віджет WB_HealthBar

Повертаючись у BP_Base_Character створюємо дві зміні: MaxHealth, CurrentHealth та функцію CreateHealthWidget, де функція CreateHealthWidget буде підключати віджет WB_HealthBar.

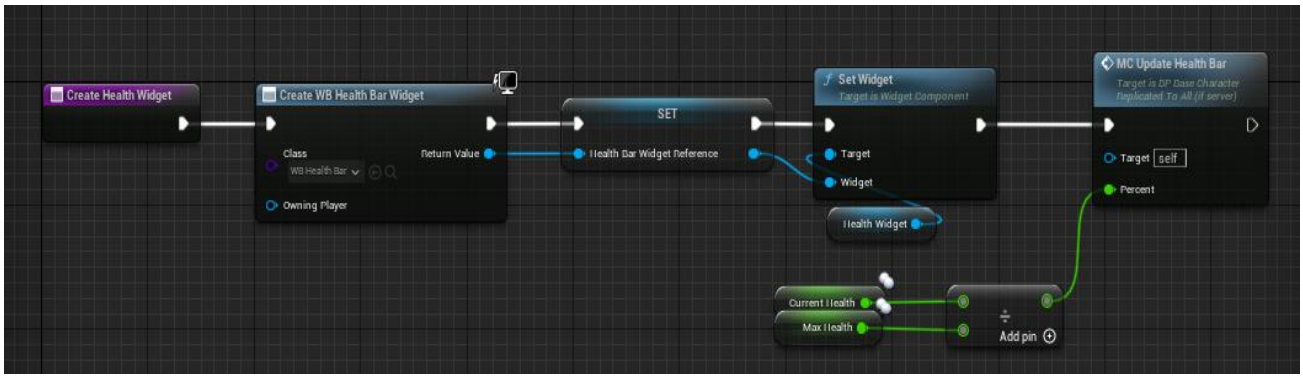

Рисунок 3.8 — функція CreateHealthWidget

Також реалізуємо оновлення шкали здоров’я за допомогою івента MC_UpdateHealthBar, який буде викликатись при створенні WB_HealthBar, отриманні шкоди та відновлення здоров’я.

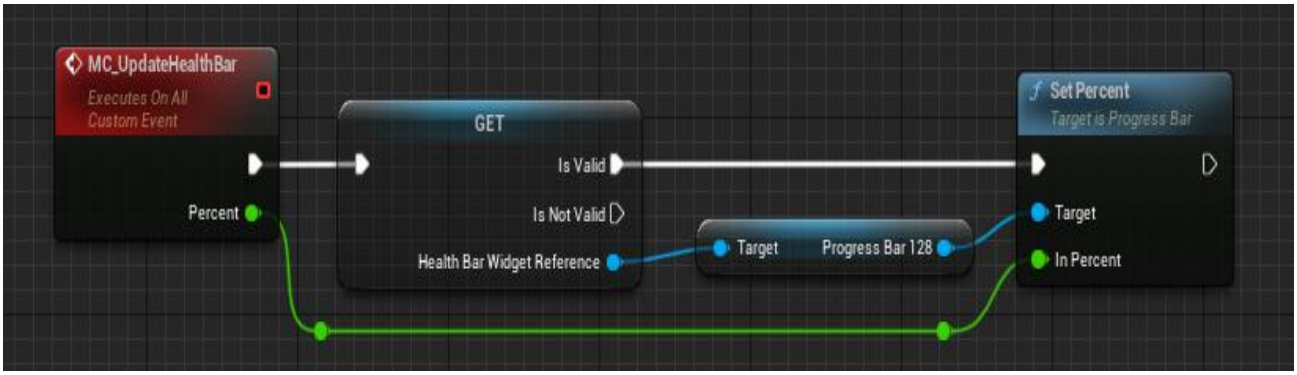

Рисунок 3.9 — івент MC_UpdateHealthBar

Тепер створимо івенти отримання шкоди та відновлення здоров’я, а саме івенти: AnyDamage рисунок 3.10 та S_RestoreHealth рисунок 3.11

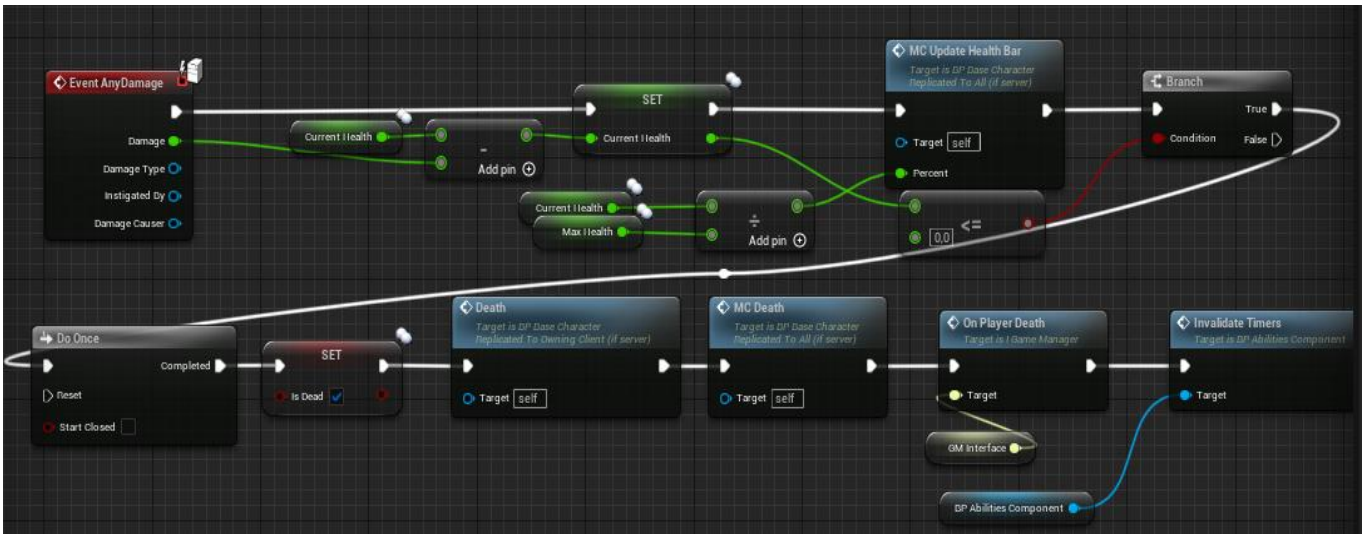

Рисунок 3.10 — івент AnyDamage

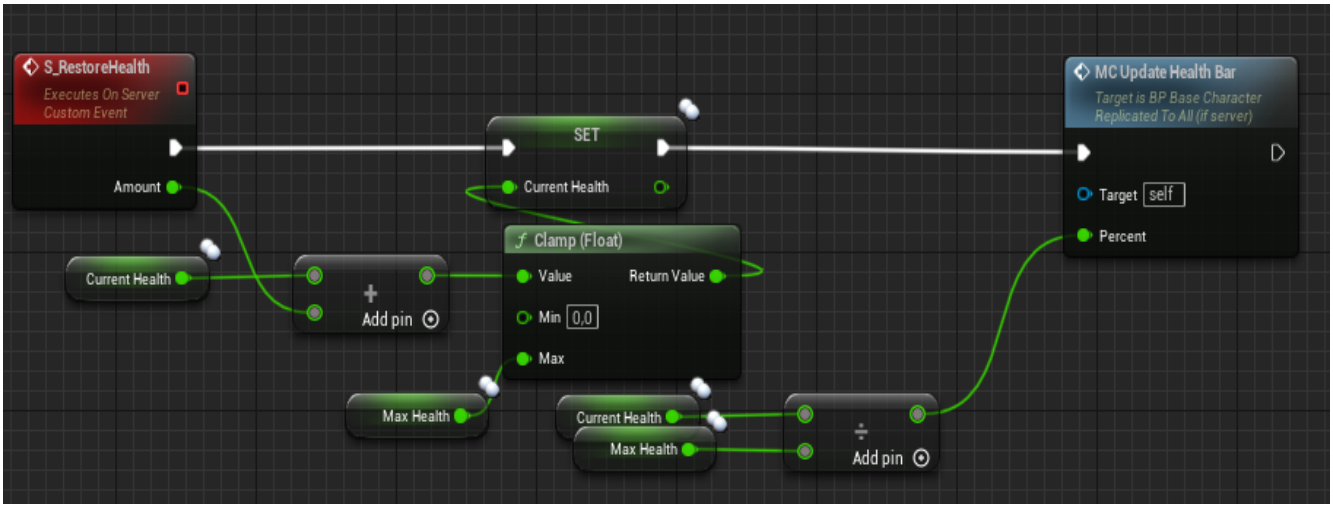

Рисунок 3.11 — івент S_RestoreHealth

Тепер прийшла пора підключити анімації для нашого гравця, для цього створюємо папку Animations та в неї імпортуємо анімації для персонажа, далі створюємо Animation Blend Spase, а у ньому зміну speed. Додаєм анімації залежності від швидкості персонажа, якщо швидкість 0 – анімація спокою, для більшої швидкості – анімація ходьби, а для найбільшої – бігу рисунок 3.12

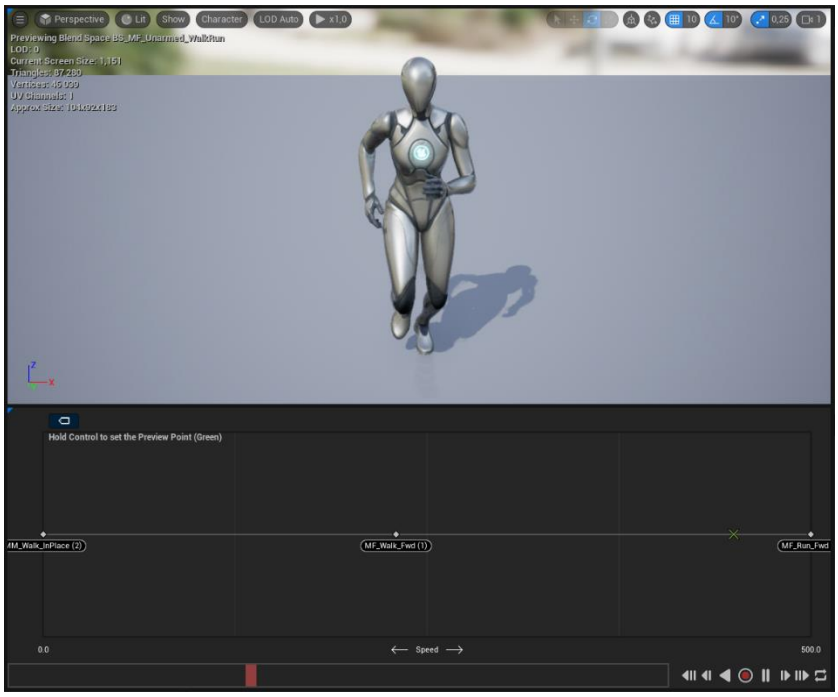

Рисунок 3.12 — Animation Blend Spase

Після цього створюємо Animation Blueprint де підключаємо наші анімації та налаштуймо умови переходу.

На останок додаймо івент який запускає анімацію смерті коли здоров’я дорівнює 0.

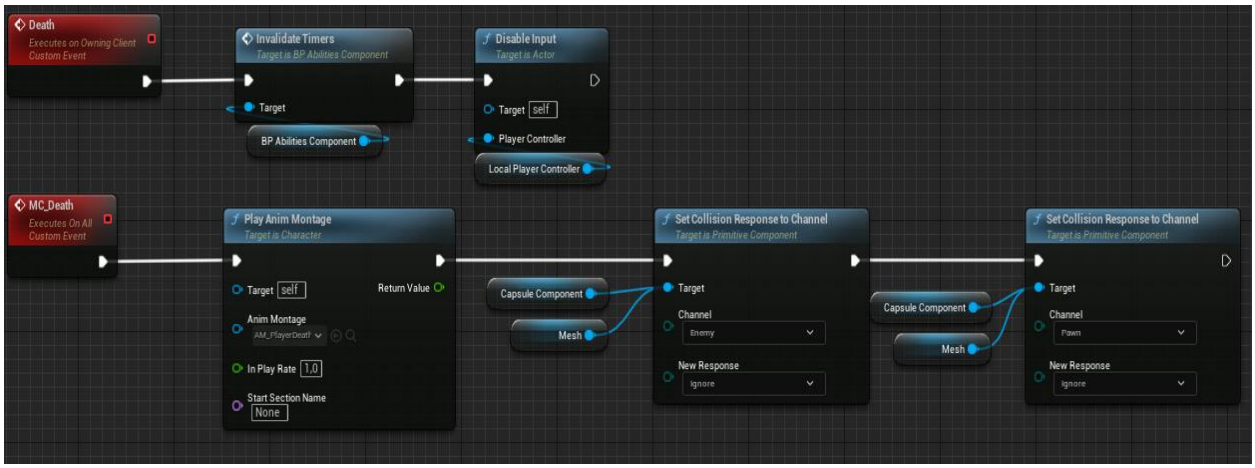

Рисунок 3.13 — івент Death

Також у вкладці налаштувань Charactera потрібно заповнити пункт Replication. Саме ця технологія дозволяє кільком гравцям бачити та взаємодіяти один одного, це працює по простій схемі: спочатку дія гравця передається на сервер з клієнта, це робиться за допомого двох однакових івентів перший з яких визивається як клієнт, а другий як сервер. Після чого сервер мультикастом розсилає дані по всім клієнтам, за допомогою 3 івента з мультикастом, який визивається з сервера.

3.2 Розробка модуля ворога

Оскільки у грі повинні бути вороги давайте займімся їх створенням для створенням, для початку імпортуємо модель ворогів та їх анімації, та створимо blueprint class для них та назвемо його BP_Base_Enemy.

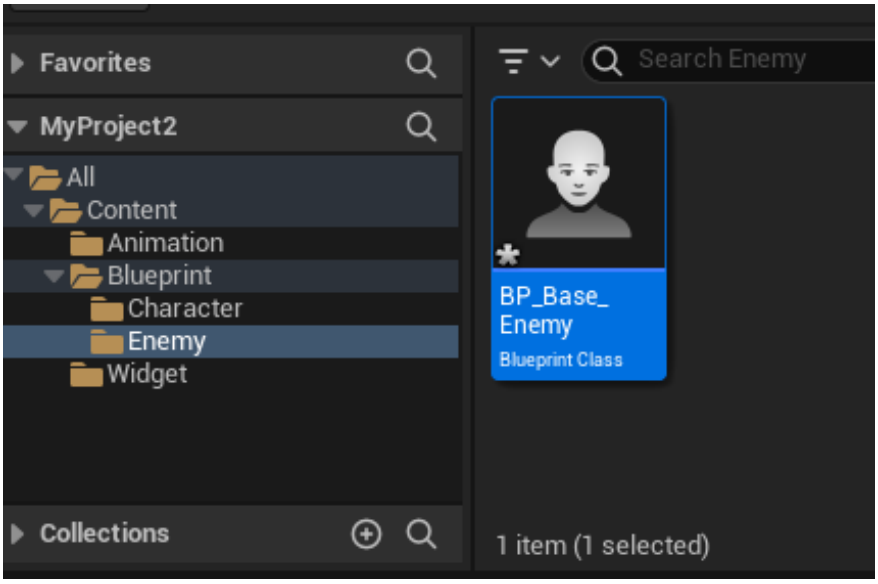

Рисунок 3.14 – Створення BP_Base_Enemy

Зайдем у BP_Base_Enemy та вставимо імпортований меш моделі ворога та додаймо кілька компонентів.

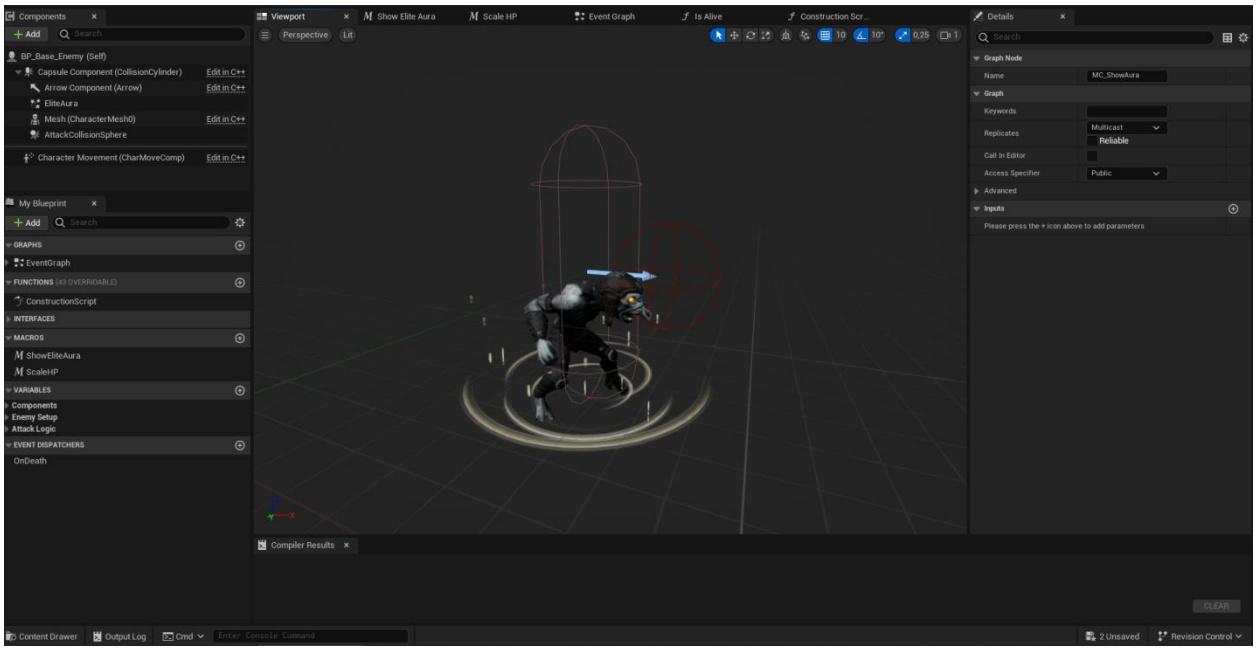

Рисунок 3.15 — Розділ Viewport BP_Base_Enemy

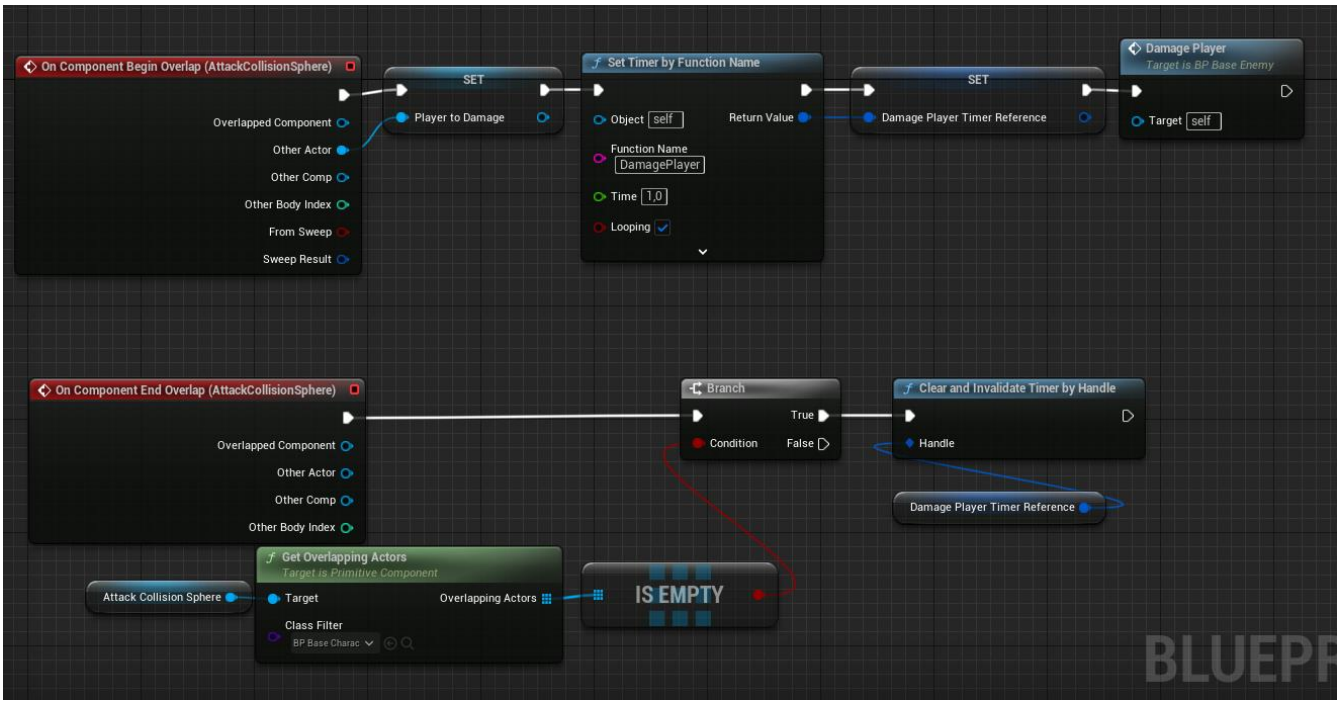

Рисунок 3.16 – Івенти OnComponentBeginOverlap та OnComponentEndOverlap

Потім створимо логіку для нанесення школи персонажу гравця. За це відповідає кілька івентів: OnComponentBeginOverlap, OnComponentEndOverlap, DamagePlayer та MC_EnemyAttack. З них OnComponentBeginOverlap та OnComponentEndOverlap відповідають за попадання по персонажу коли він пересікає сферу колізії ворога, а DamagePlayer та MC_EnemyAttack за завдання шкоди та програвання анімації атаки. Результат зображен на рисунку 3.16

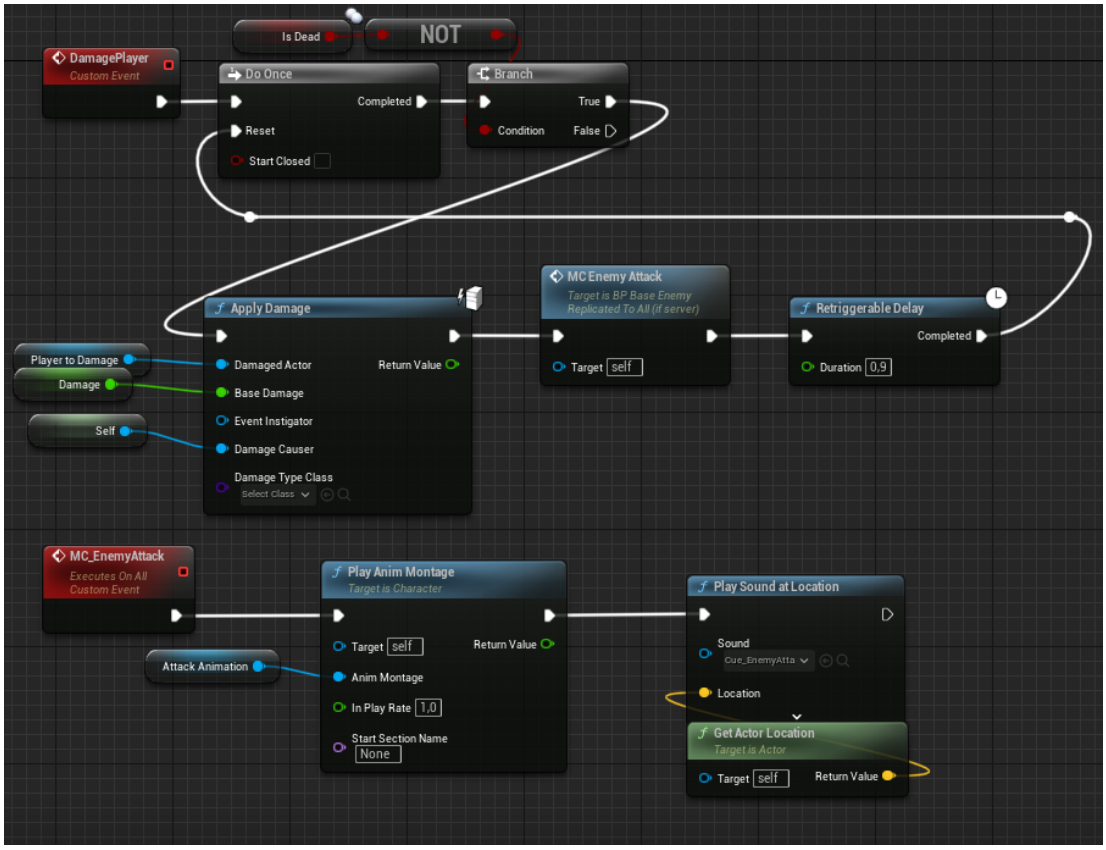

Рисунок 3.17 – Івенти DamagePlayer та MC_EnemyAttack

Тепер можна створити логіку отримання шкоди ворогом, вона буда знаходитись в івенті AnyDamage, також добавим показ шкоди під час атаки персонажа, реакцію ворога на атаку у вигляді звуку та зупинки ворога з програванням анімації отримання шкоди. Також оновлення значення здоров’я, крім того перевірка на смерть та запуск івента MC_Enemy_Death , який

відключає колізію зі всіма об’єктами та запускає запускає анімацію смерті та знищує цей об’єкт.

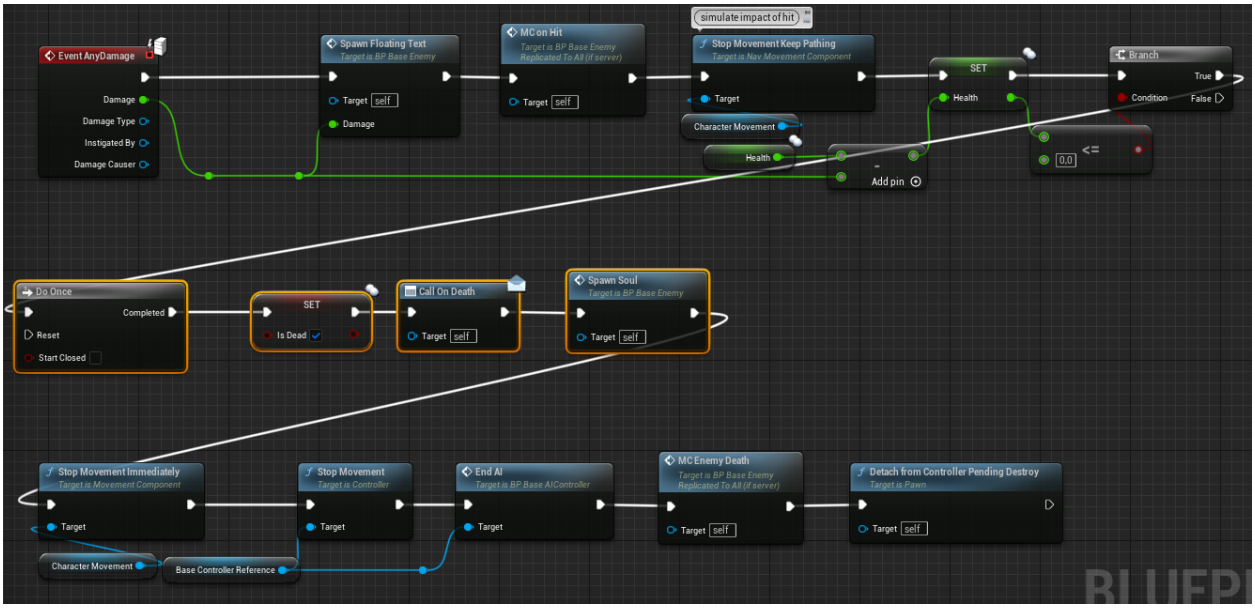

Рисунок 3.18 – Івент AnyDamage

Ще було створено два івента: SpawnFloatingText та SpawnSoul, де SpawnFloatingText генерує цифри отриманої шкоди від гравця, а SpawnSoul – створює «душу» на місці смерті, яка підвищує шкалу досвіду гравця при піднятті.

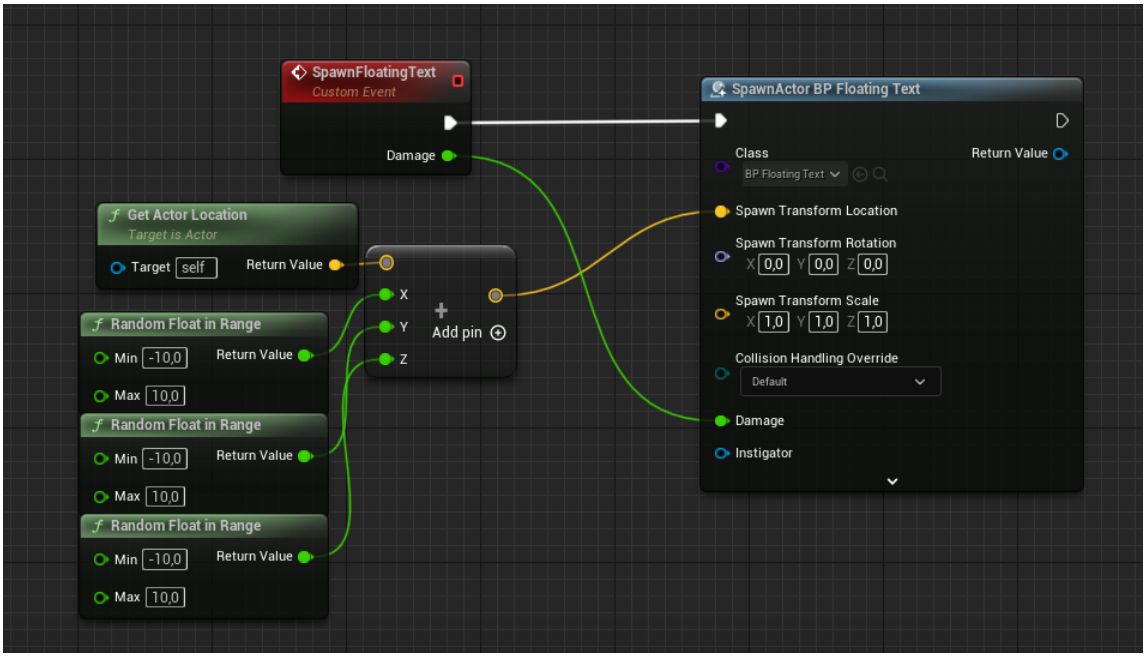

Рисунок 3.19 – Івент SpawnFloatingText

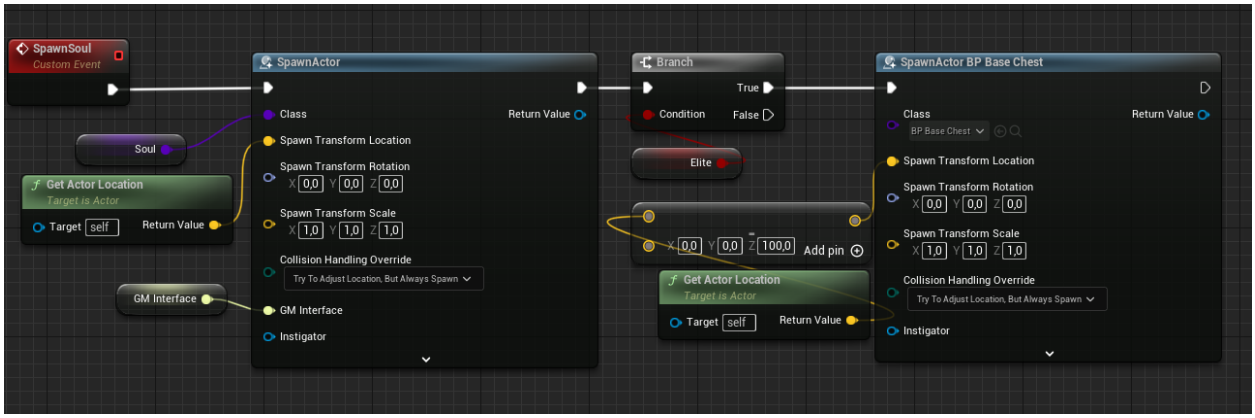

Рисунок 3.20 – Івент SpawnSoul

Тепер потрібно навчити ворогів шукати та нападати на гравця, переслідувати його. Для цього у Unreal Engine 5 – є свої інструменти. Тому для початку створимо AIControler, Behavior Tree та Blackboard. У Blackboard створим три ключа: SelfActor – референс на самого себе (об’єкт BP_Enemy) Target – референс на гравця, та Behavior – який показує які види ворогів можуть з’являтись, а які ні.

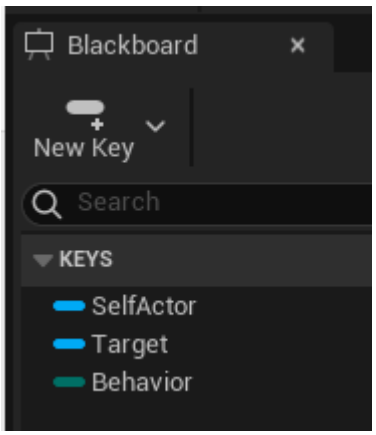

Рисунок 3.21 – Blackboard

Далі створюємо Behavior Tree, де зазначаєм, що вороги будуть чекати 5 секунд, а потім рухатись до Target або гравця, також сюди ми підключаєм вже створений Blackboard для отримання ключа Target.

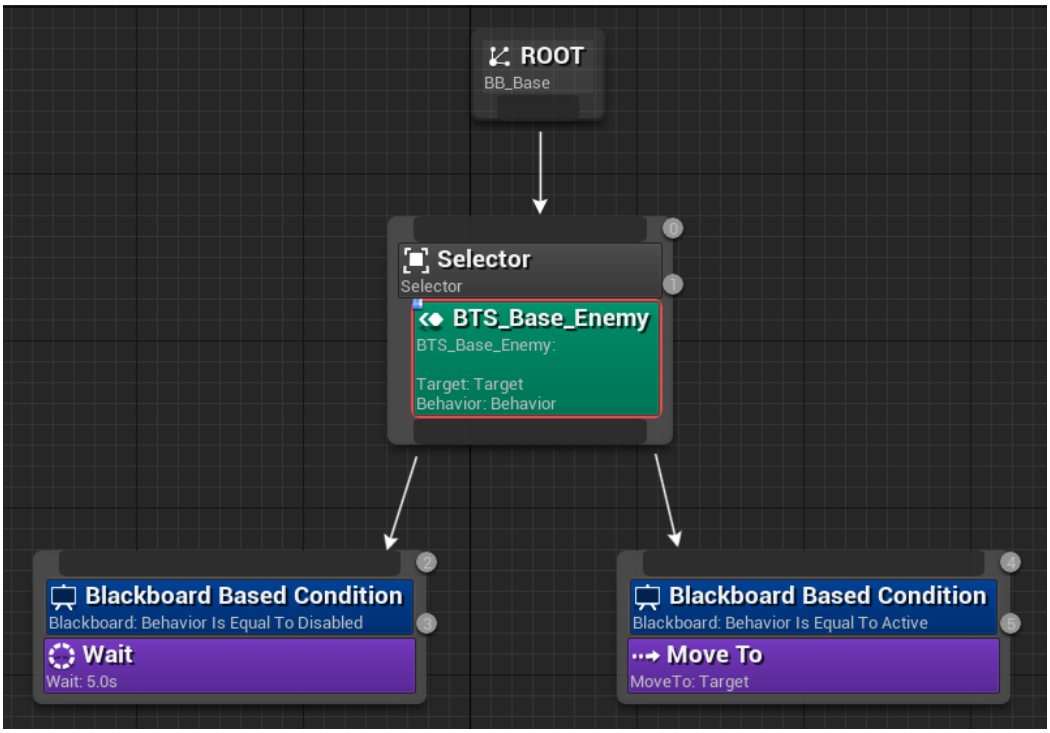

Рисунок 3.22 – Behavior Tree

В класі AIControler спочатку викликається івент BeginAI, який визиває інший івент FindTarget кожну секунду. Він в свою чергу перевіряє, чи живий гравець та обновляє масив гравців, після чого записує його референс у ключ Target. Інший івент On Possess запускає Behavior Tree та івент BeginAI.

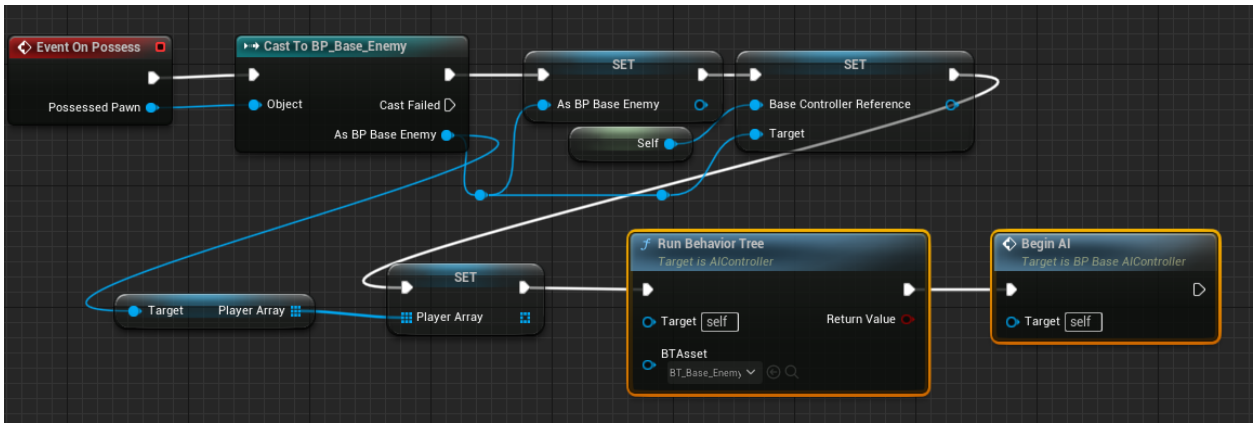

Рисунок 3.23 – Івент On Possess

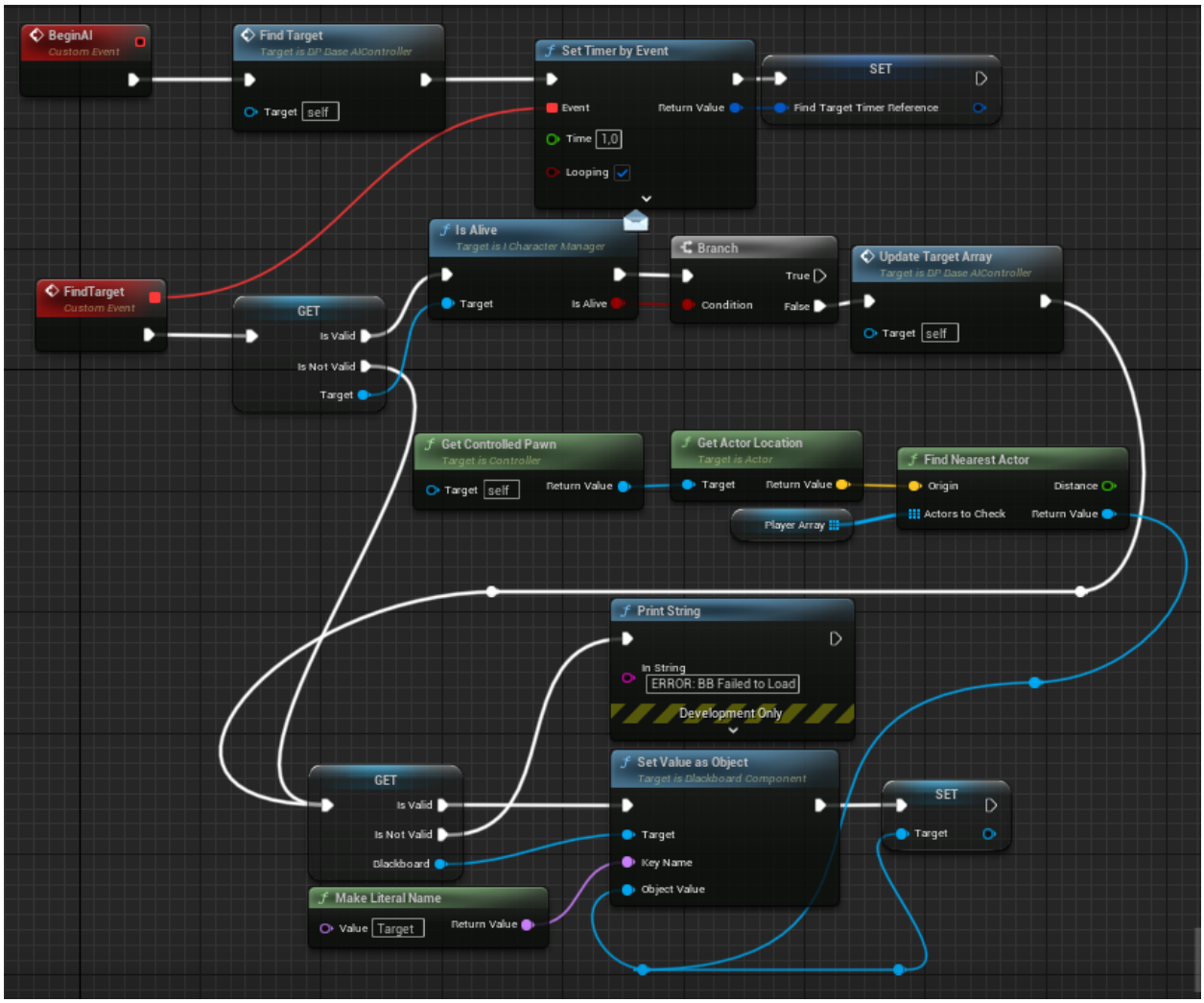

Рисунок 3.24 – Івент BeginAI

3.3 Розробка модуля активних здібностей гравця

Свої здібності (Abilities) гравець буде отримувати з підвищенням рівня, тому потрібно створити компонент, який буде додавати нові здібності до персонажа, також необхідне створення віджетів для досвіду, підвищення рівня та вибору здібності.

Для початку створимо віджет для відображення прогресу рівня та назвем його WB_PlayerHud. Далі всередині нього створим Canvas Panel, Overlay,

ProgressBar, TextBlock для відображення рівня та Vertical Box з 2 HorizontalBox для активних та пасивних здібностей, на останок добавим TextBlock для відображення часу

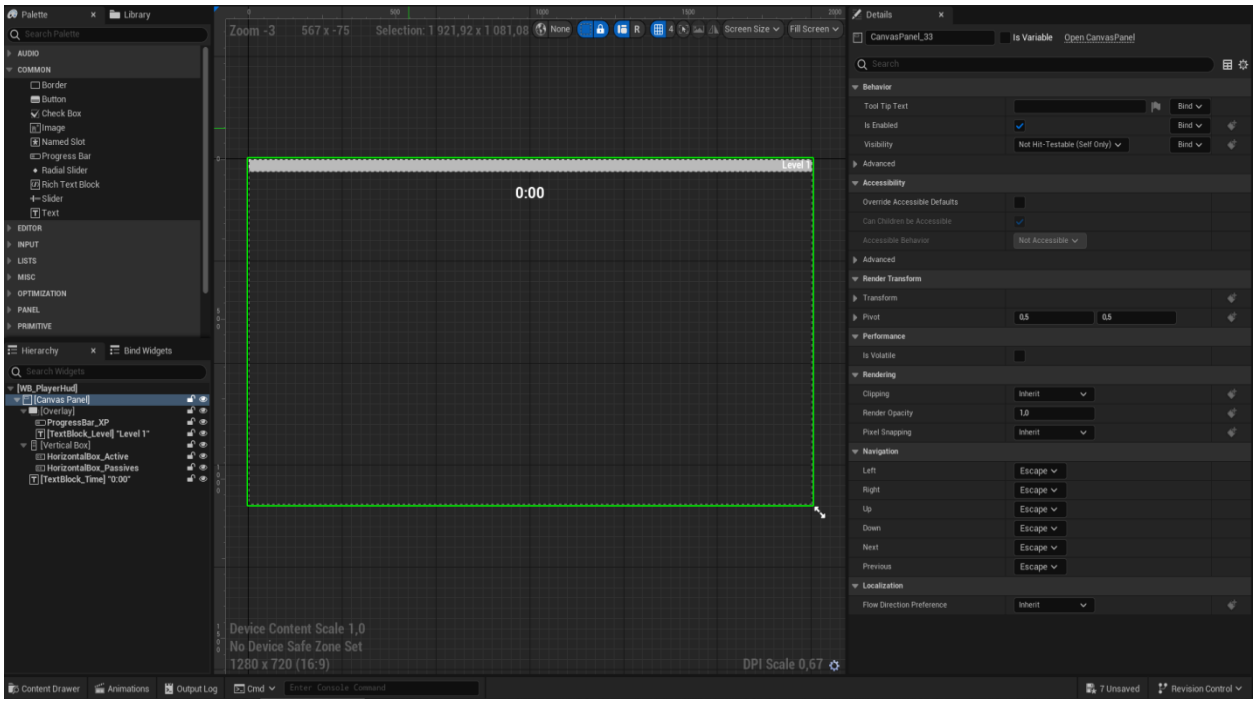

Рисунок 3.25 – Віджет WB_PlayerHud

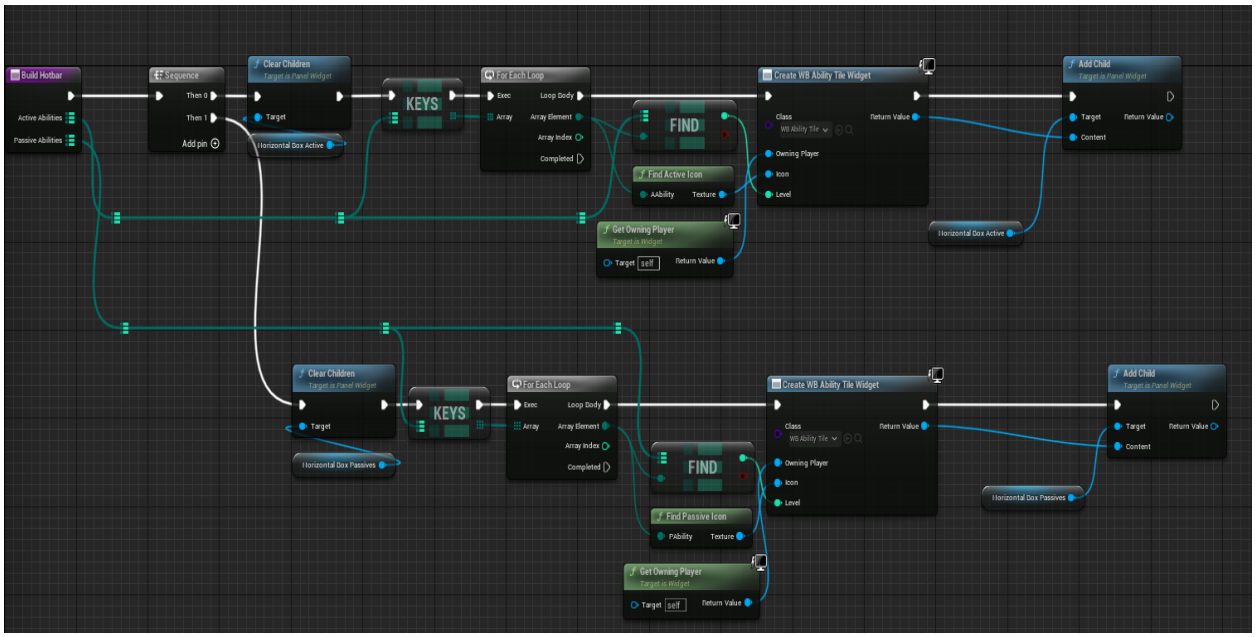

Рисунок 3.26 – Функції Build Hotbar

В розділі Event Graph створюєм 2 функції: Build Hotbar та UpdateTime. Перша для створення віджета вивчених здібностей та розміщення їх у HorizontalBox в залежності від їх типу. А друга для відліку часу від початку гри.

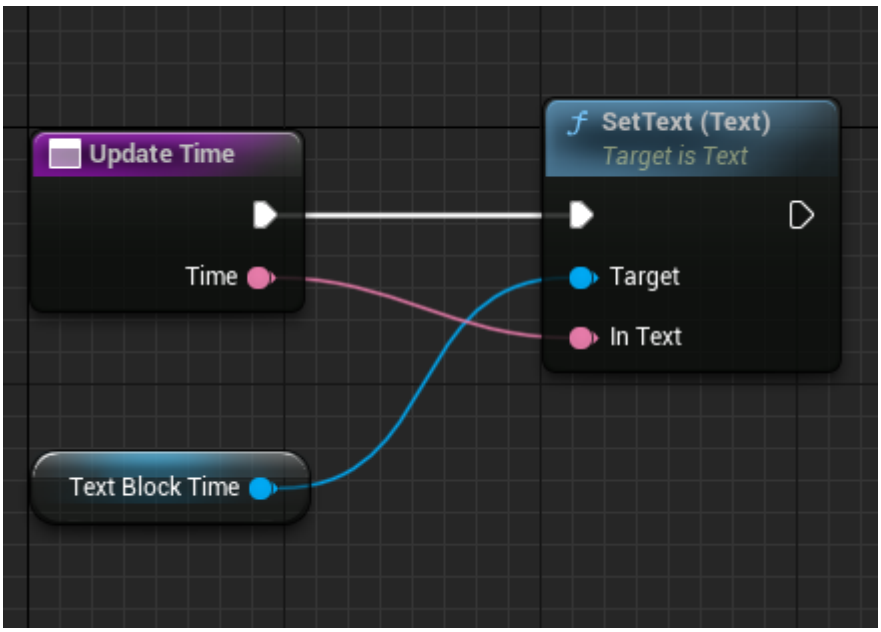

Рисунок 3.27 – Функції UpdateTime

Build Hotbar використовує ще один створений віджет WB_AbilityTile, який буде створюватись та розміщуватись у один з двох HorizontalBox. В залежності від обраної здібності та її рівня віджет буде змінюватись вставляючи потрібне зображення та текст. Самі зображення були імпортовані завчасно.

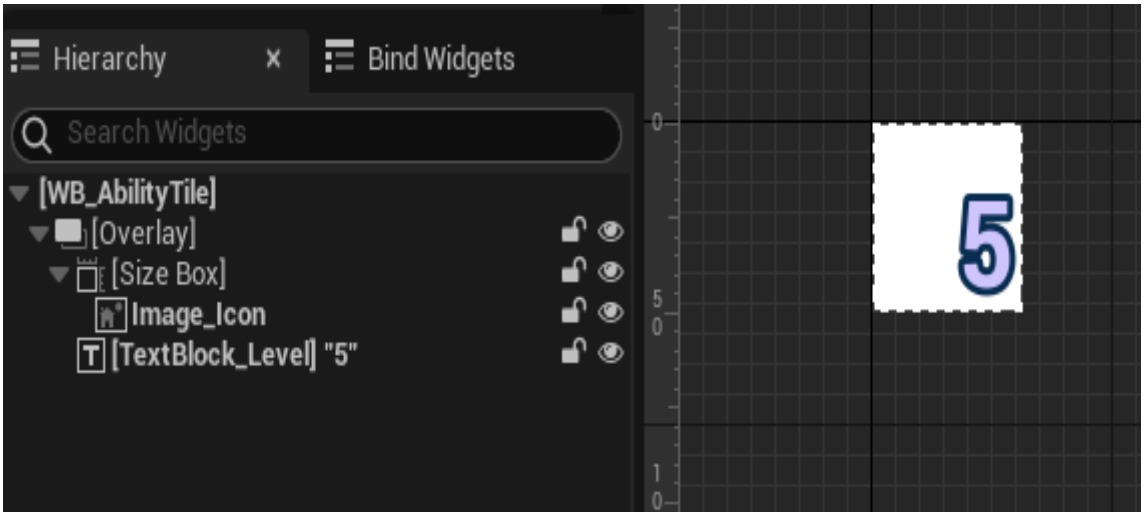

Рисунок 3.28 – Віджет WB_AbilityTile

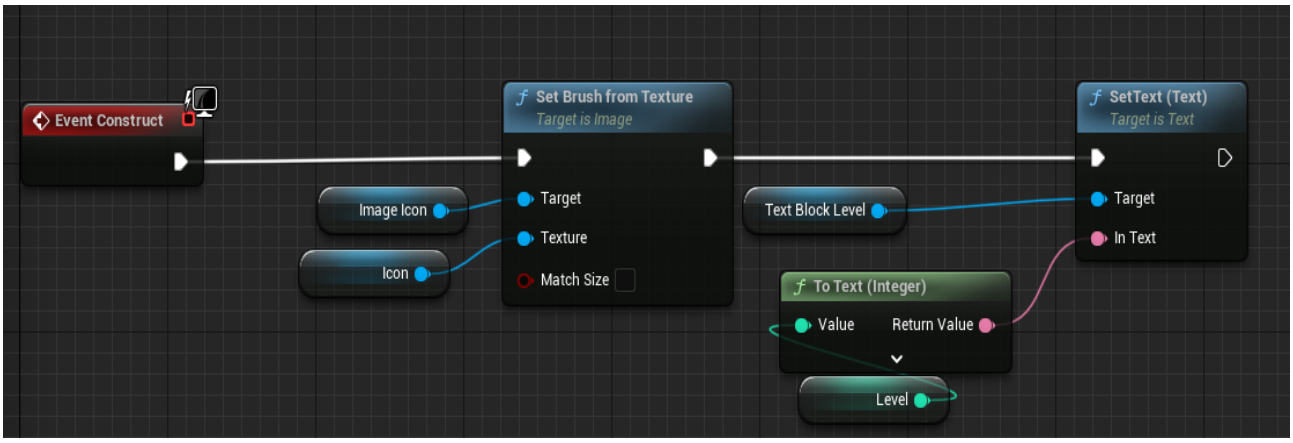

Рисунок 3.29 – Event Graph віджета WB_AbilityTile

Прийшов час створити віджет для підняття рівня та обрання здібності. Назвем їх WB_LevelUpMaster та WB_LevelUpItems. В першому просто повідомлення про підвищення рівня, а другий дає на вибір чотири здібності з картинкою, описом та рівнем здібності.

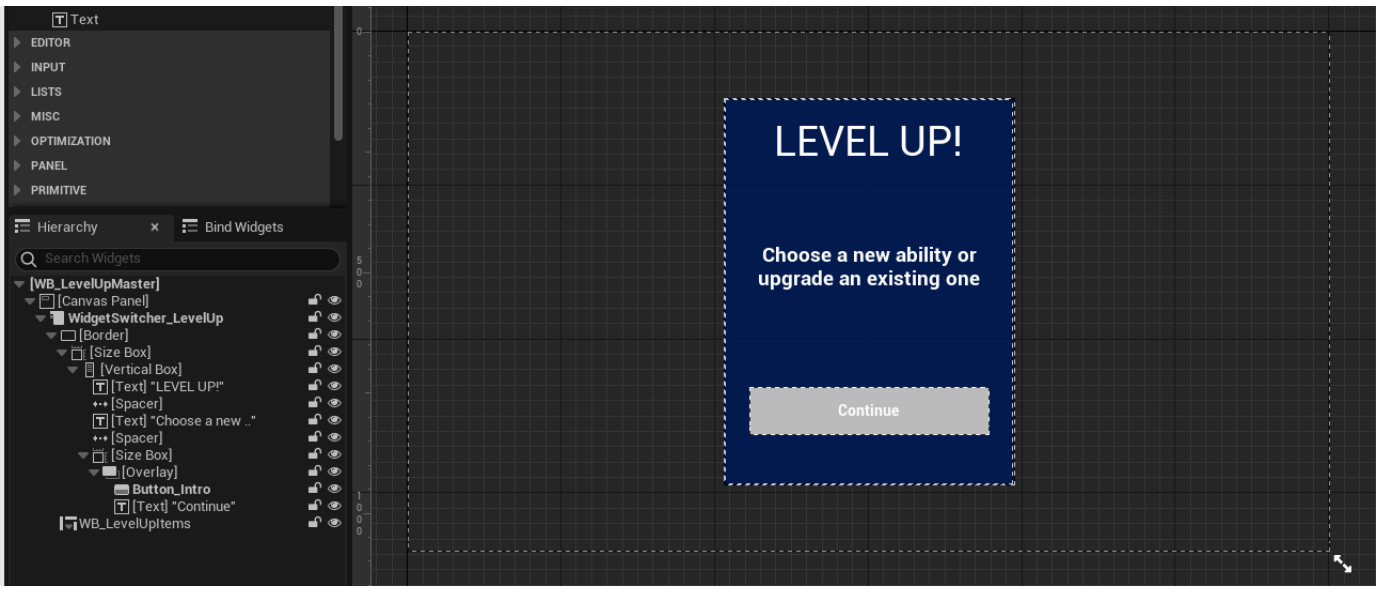

Рисунок 3.30 – Віджет WB_LevelUpMaster

На кнопку Continue визивається віджет WB_LevelUpItems. В якому будуть створюватись чотири віджета WB_LevelUpCard, як варіанти вибору.

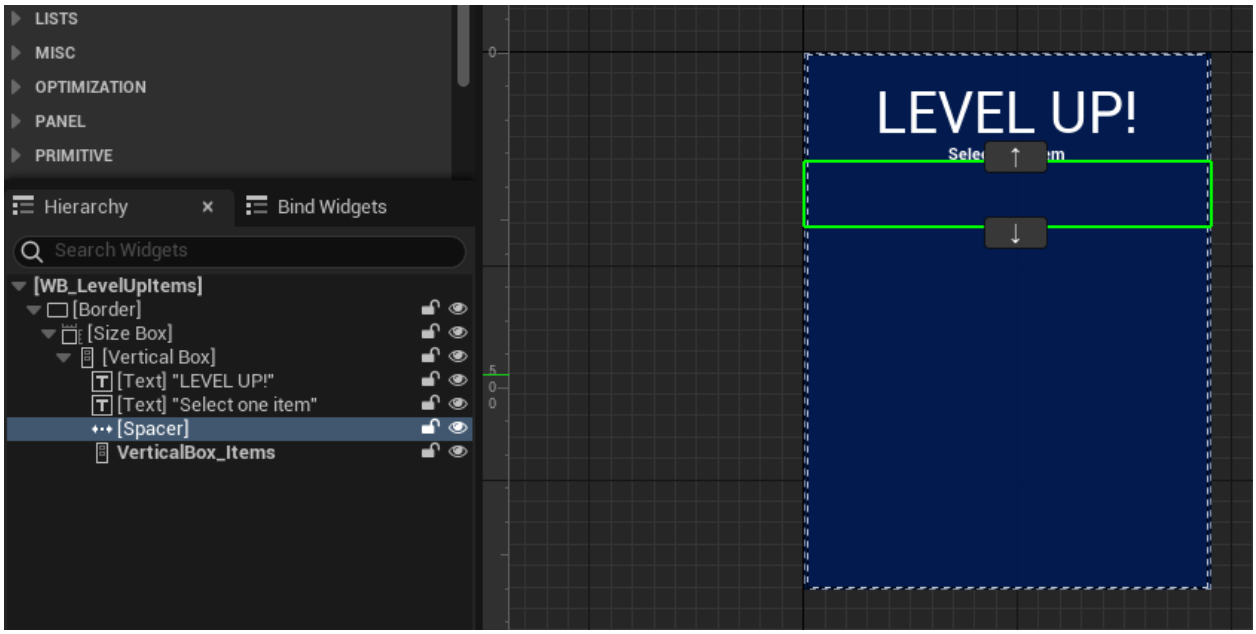

Рисунок 3.31 – Віджет WB_LevelUpItems

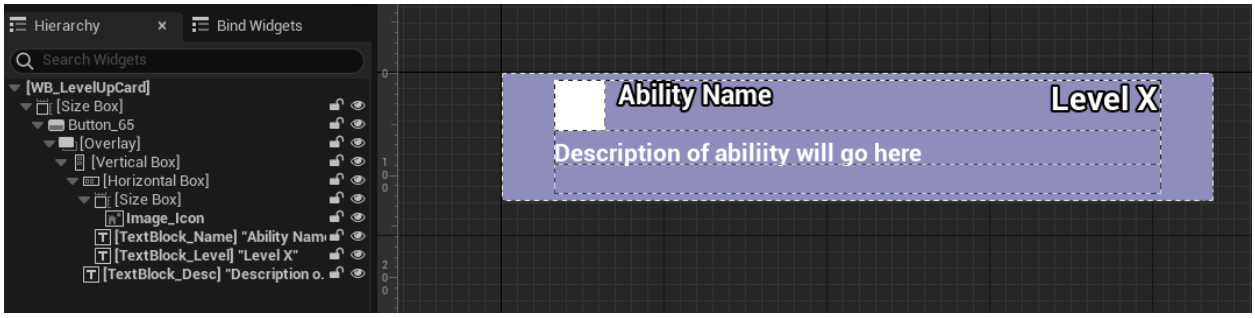

Рисунок 3.32 – Віджет WB_LevelUpCard

Тепер їх потрібно підключити до персонажа, для цього можна використати PlayerControler. В якому будуть підключаються віджети: WB_PlayerHud, WB_LevelUpMaster та WB_LevelUpItems. Також тут будуть оновлюватись ці та інші віджити. Тут же реалізована логіка підвищення рівня.

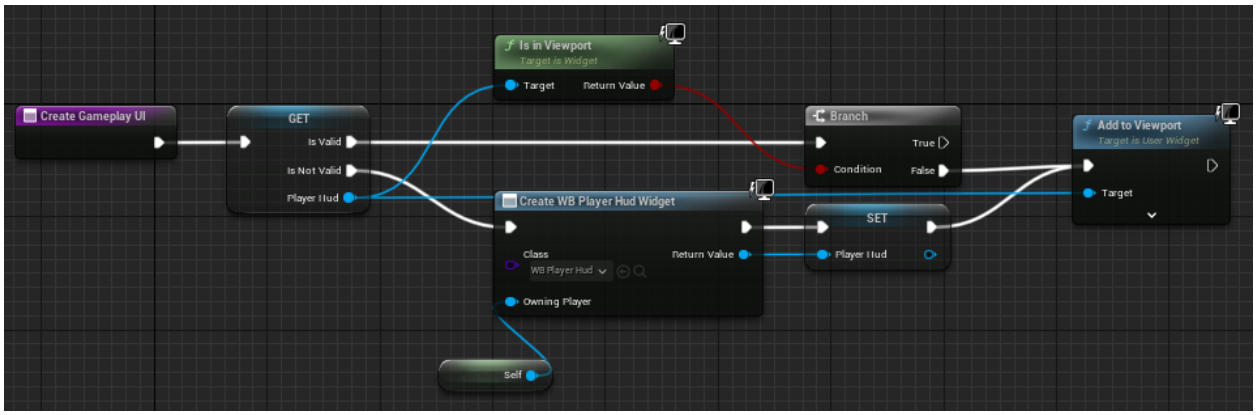

Рисунок 3.33 – Функція підключення віджета WB_PlayerHud

Рисунок 3.34 – Функція оновлення віджету WB_PlayerHud

Тепер коли всі потрібні віджети створені можна приступити до створення самих здібностей для цього – створимо Actor Component, який добавимо до BP_Base_Character для додавання нових здібностей. Всього буде чотири активних здібності: Hammer, Frost bolt, Lighting bolt та Fireball.

Розпочнемо з Hammer, він повинен створювати молот, який крутиться навколо нас певний час та атакує ворогів. Для цього спочатку створимо сферу з трасованих лучів, вона перевіряє чи є в радіусі об’єкт вказаного класу, місцем створення буде координати персонажу. Далі якщо якийсь ворог попав у сферу, він втратить здоров’я, але лише раз навіть якщо він находиться в сфері, або заходить і виходить із неї. І в кінці створюємо модель молота навколо нас

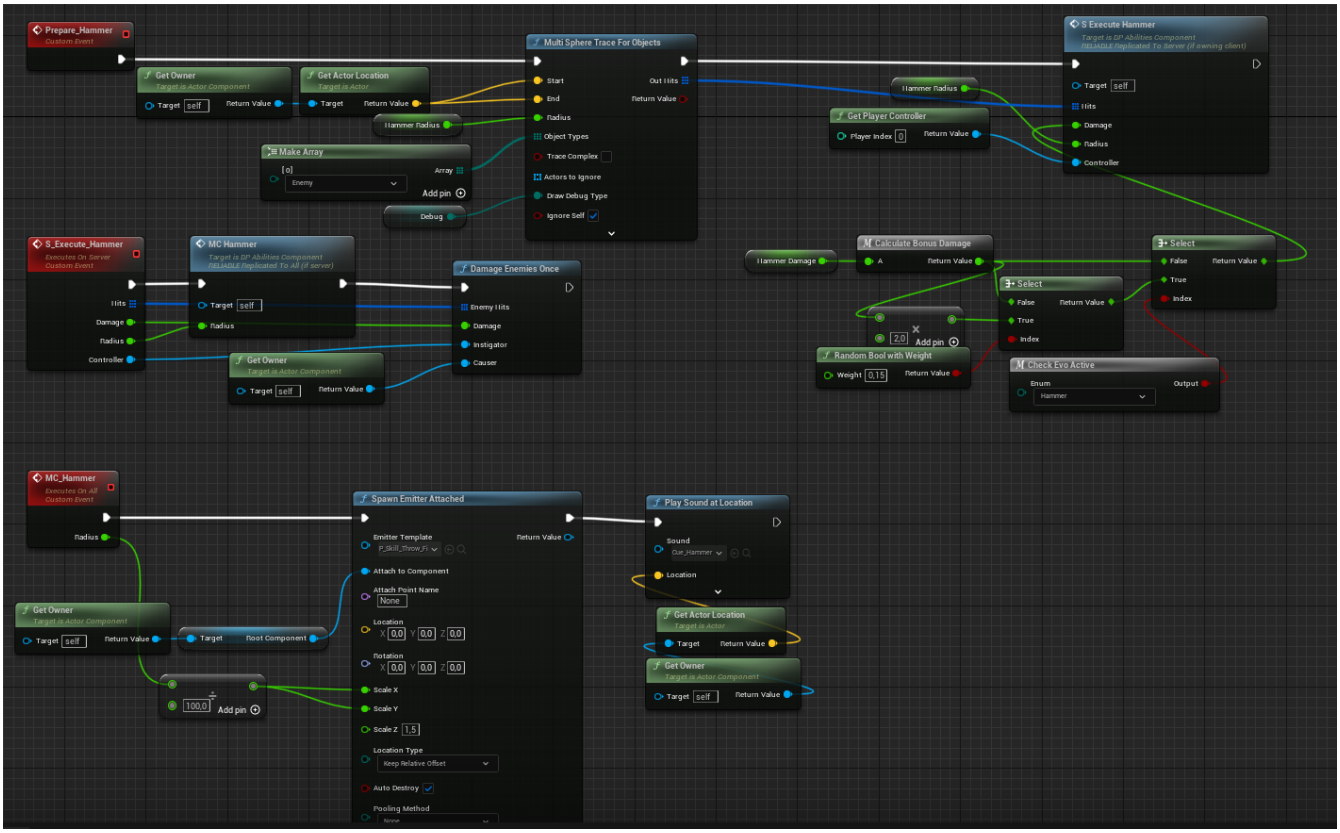

Рисунок 3.35 – Скрипт роботи Hammer

Далі розглянемо Frost bolt, логіка його роботи простіша. Спочатку створюється сфера колізії та вона перевіряється на перетин з класом ворог, після чого спочатку створюється модель Frost bolt, а потім кількості існуючих моделей додається, ще одна допоки кількість існуючих моделей не стане рівним максимальній кількості моделец, тоді наступна створиться тільки після попадання та знищення попередньої.

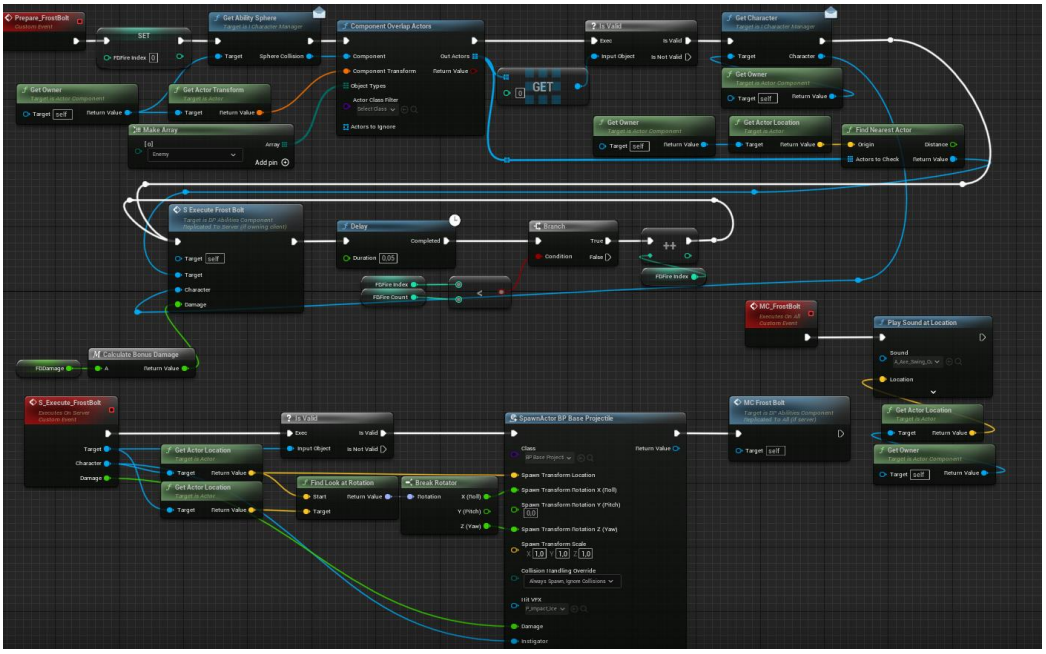

Рисунок 3.36 – Скрипт роботи Frost bolt

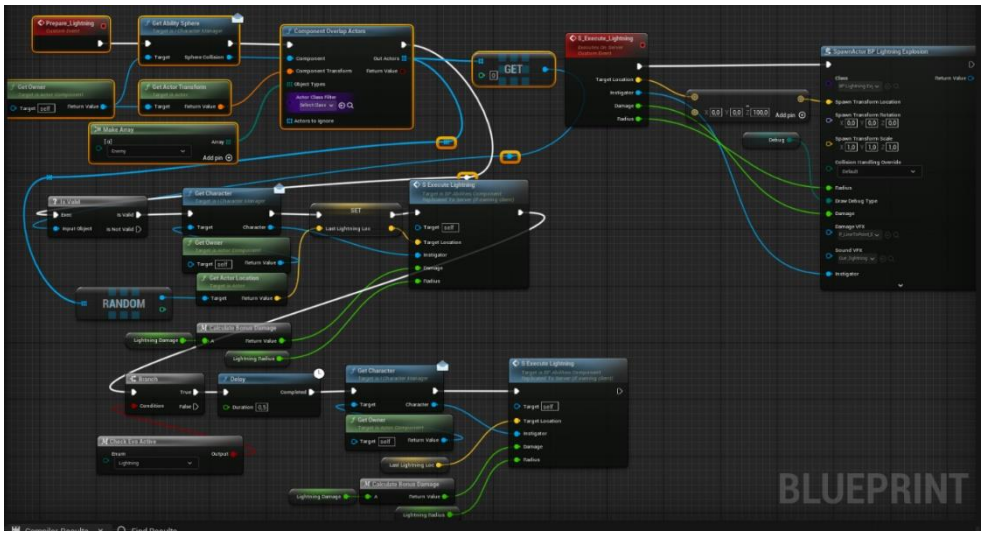

Рисунок 3.37 – Скрипт роботи Lighting bolt

Наступний Lighting bolt, він ще простіший за Frost bolt. На відміну від останньої в Lighting bolt просто створюється сфера колізії та наносить шкоду після створення моделі Lighting bolt. Єдина перевага перед Frost bolt полягає у еволюції Lighting bolt коли він атакує повторно.

Останнім залишився, який зроблений так само як і 2 попередніх, окрім еволюції яка дозволяє запускати ще один Fireball. А в усьому іншому вони ідентичні

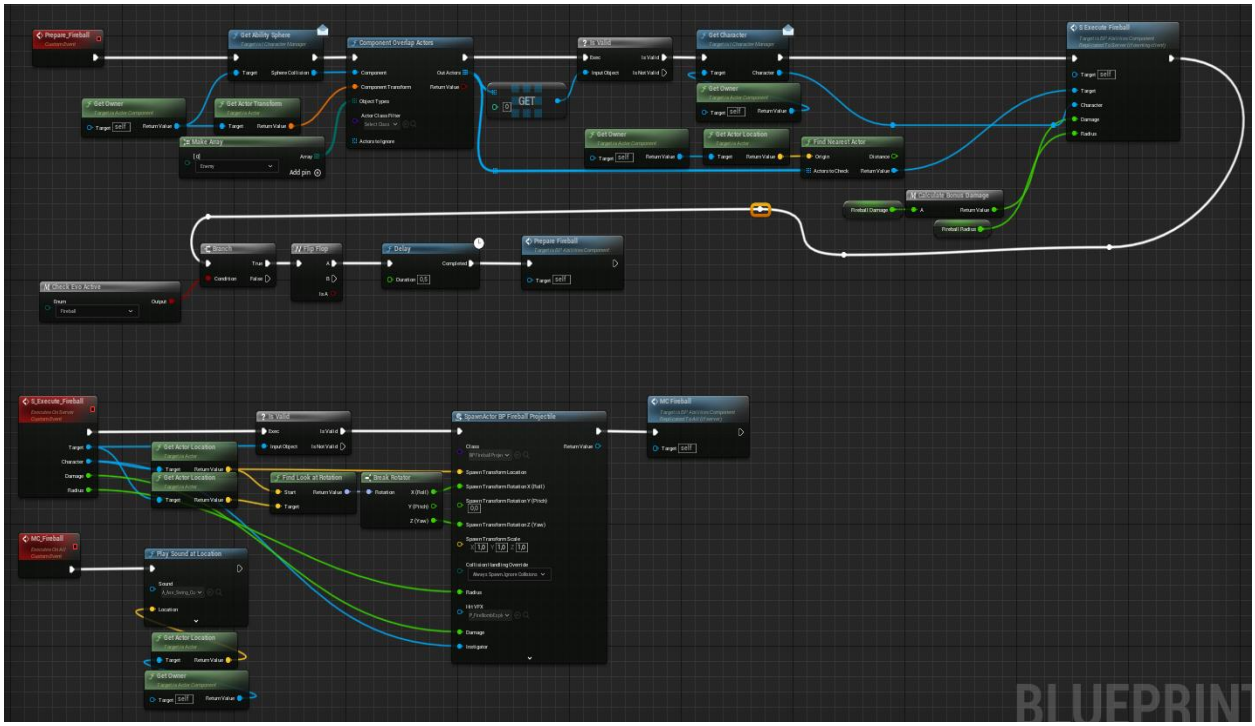

Рисунок 3.38 – Скрипт роботи Fireball

В кінці ми створюємо функцію SetStartingAbility, яка добавляє нові здібності або покращує вже вивчені для об’єкта до якого прикріплений цей компонент. Для цього він витягує інформацію з бази даних, де є інформація про кожен вид здібності та добавляє в нову базу, яка відповідає за персонажа гравця.

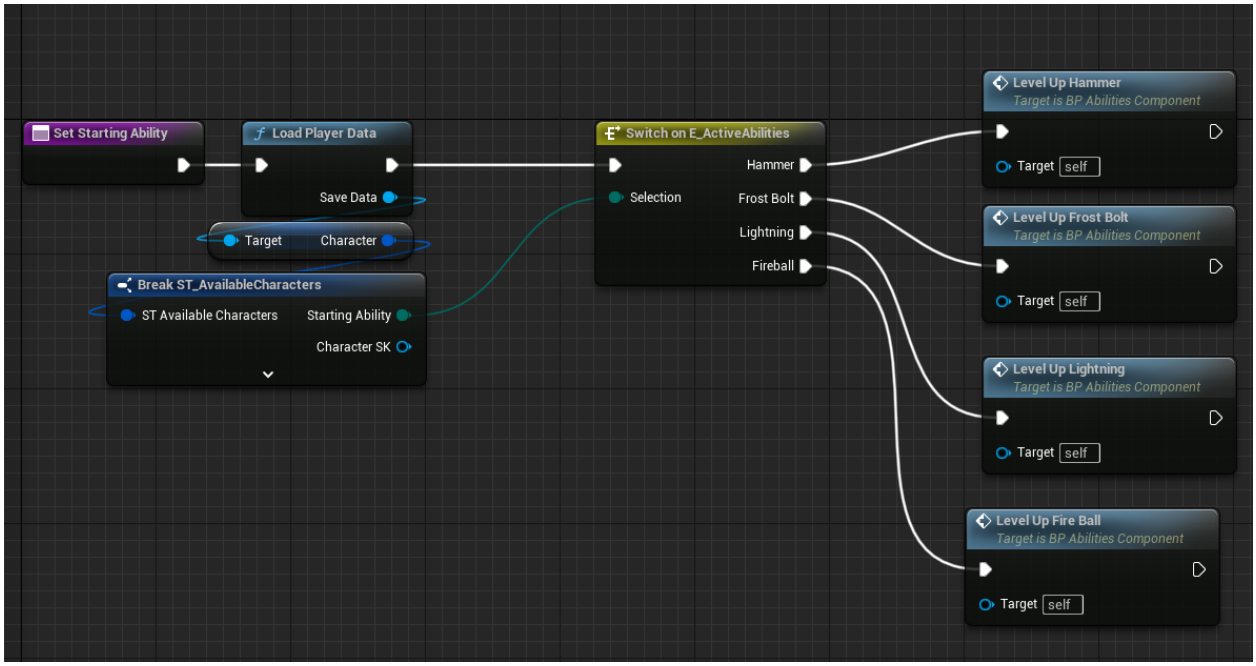

Рисунок 3.39 – Функцію SetStartingAbility

3.4 Створення ігрового рівня

Для створення ігрового рівня, можна використати прості кубічні меші та розставити їх щоб створити арену для гри. Потім на цю арену розтянути «Nav Mesh Bounds Volume» для роботи штучного інтелекту. Також в центрі створюємо точку спавна гравця

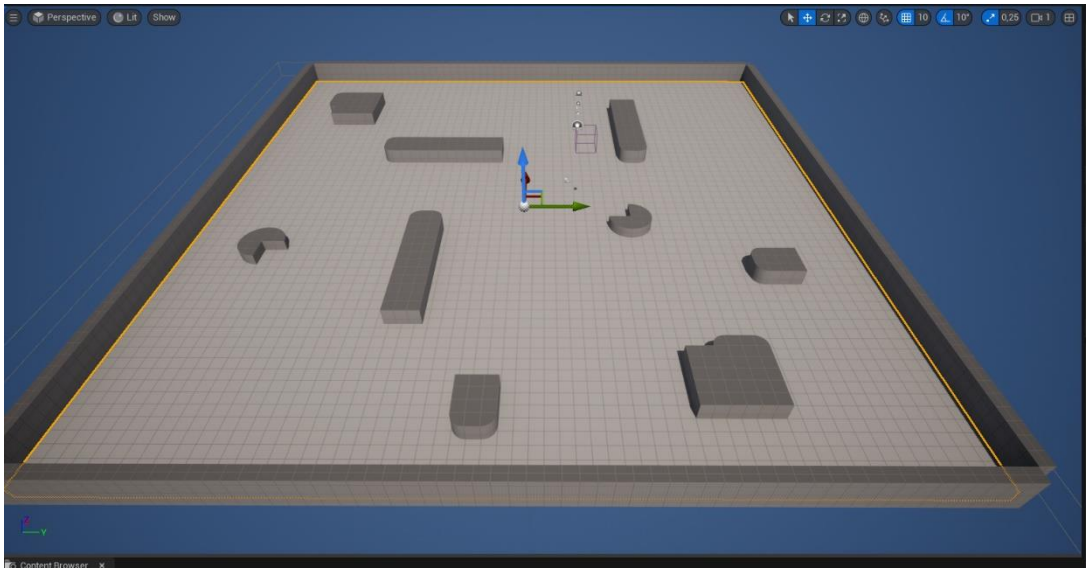

Рисунок 3.40 – Карта ігрового рівня\

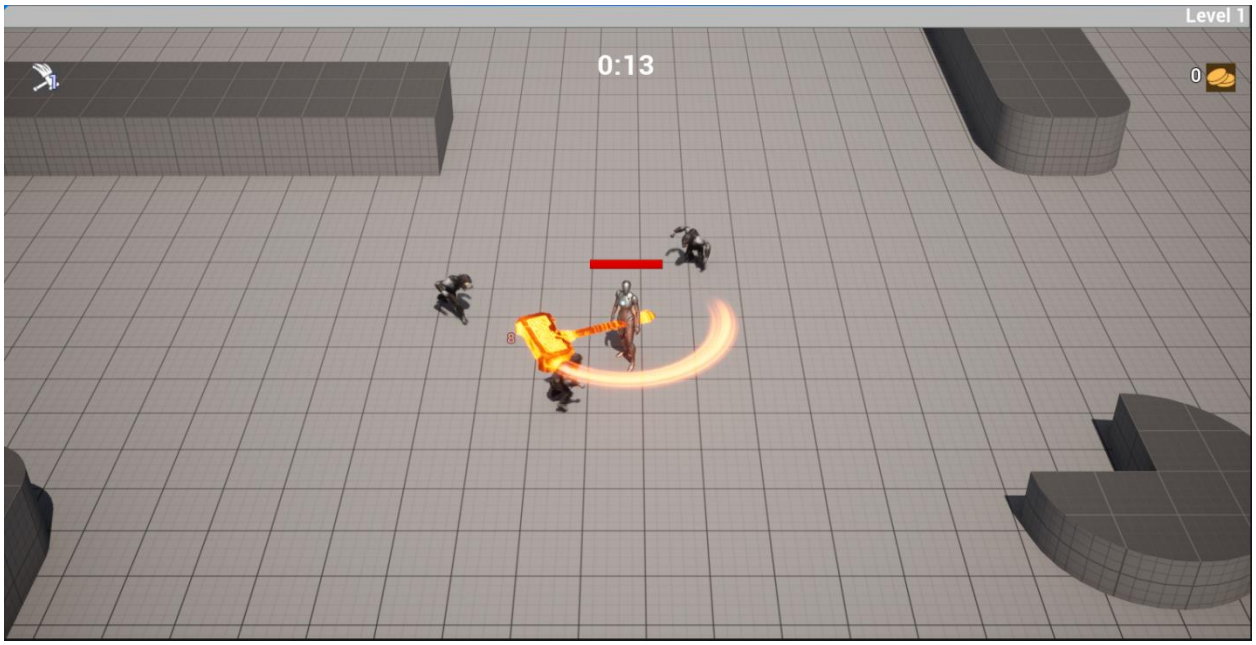

Рисунок 3.41 – Карта ігрового рівня під час гри

3.5 Висновок

В третьому розділі було створено персонажа гравця та реалізовано базове керування, створені скрипти для отримання шкоди та відновлення здоров’я, також реалізований кінець гри у випадку опускання здоров’я до нуля та підключені анімації руху.

Було створено кілька віджетів для відображення різної інформації про гравця, або ігровий прогрес.

Було створено персонажа ворога, реалізовано метод нанесення шкоди гравцю, отримання шкоди від гравця, смерть ворога, анімації його пересування, атаки та смерті, створено дерево поведінки та контролер штучного інтелекту, який шукає переслідує та атакує гравця.

Було створенні чотири унікальні здібності, які гравець міг вибрати під час підвищення рівня, реалізовано логіку їх роботи, нанесення шкоди, додавання до персонажу, та підвищення рівня.

Також була створена карта ігрового рівня.

ВИСНОВКИ

У рамках виконання бакалаврської кваліфікаційної роботи було проведено всебічну роботу над створенням програмного модуля модуля авторської гри на ігровому рушії Unreal engine 5. Проект включав проектування користувацького інтерфейсу, розробку алгоритмів роботи, розробки моделі роботи, створення ігрового циклу, розробки усіх основних модулів гри та ігрового рівня.

Було проведено детальне дослідження розвитку області ігрової розробки, виділені основні інструменти потрібні для розробки ігор та Було розглянуто існуючі аналоги ігор в обраному жанрі та проаналізовано на сильні та слабкі сторони, також були виявленні найвиразніші особливості жанру. Також було проаналізовані ігрові рушії їхні сильні та слабкі сторони. Для розробки була обрана візуальна мова програмування Blueprint, причиною даного рішення було прийнято легкість використання мови та швидкість розробки програми на ній

Одним з ключових аспектів роботи було проектування користувацького інтерфейсу, що передбачало визначення основних елементів, які забезпечують інтуїтивно зрозумілий доступ до функцій системи. На основі проведених досліджень та тестувань було створено макети інтерфейсу, що дозволяють користувачам легко орієнтуватися у додатку та ефективно використовувати його функції.

Розробка алгоритмів роботи охопила усі основні алгоритми гри на ігровому рушії Unreal engine 5, зокрема алгоритма реалізації ігрової системи. Також була створена модель гри на рушії Unreal engine 5 та завдяки розробленим алгоритмам було досягнуто перехід від 2.5d до 3d без великих затрат продуктивності, з підвищенням рівня візуального оформлення роботи.

Розробка самої гри включала в себе створення моделі персонажа, ворога, віджетів та здібностей гравця. Створено штучний інтелект, який дозволяв шукати та переслідувати гравця. Та група унікальних здібностей, які можливо випробувати на ворогах

На основі проведеної роботи можна зробити висновок, що основні етапи проектування та розробки програмного модуля авторської гри на ігровому рушії Unreal engine 5 виконані успішно. Досягнуті результати закладають міцний фундамент для подальшої роботи над проектом та забезпечують високу якість кінцевого продукту. Усі поставлені цілі бакалаврської кваліфікаційної роботи були досягнуті, що підтверджує ефективність обраних методів та підходів. Наступним кроком є завершення розробки модуля, його тестування та підготовка до випуску на ринок, з урахуванням отриманого досвіду та висновків, зроблених під час практики.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Thompson, S. Unreal Engine Essentials: Building Interactive Applications, [Електронний ресурс]. Режим доступу: https://www.springer.com/gp/book/9781484242673

2. Martinez, A. Mastering Unreal Engine: Advanced Techniques for Game Development, [Електронний ресурс]. Режим доступу: https://www.pearson.com/store/p/mastering-unreal-engine/P100002648644

3. Brown, L. Unreal Engine: From Beginner to Pro, [Електронний ресурс]. Режим доступу: https://www.oreilly.com/library/view/unreal-engine-from/9781838645569/

4. Fisher, D. Blueprints Visual Scripting for Unreal Engine: The Faster Way to Build Games, [Електронний ресурс]. Режим доступу: https://www.packtpub.com/product/blueprints-visual-scripting-for-unreal-engine/9781839216540

5. White, M. Unreal Engine Animation and Cinematics: Creating High Impact Visuals, [Електронний ресурс]. Режим доступу: https://www.wiley.com/en-us/Unreal+Engine+Animation+and+Cinematics-p-9781119651789

6. Green, S. Advanced AI Design for Unreal Engine, [Електронний ресурс]. Режим доступу: https://www.cambridge.org/core/books/advanced-ai-design-for-unreal-engine/9781108476459

7. Harris, J. Unreal Engine Lighting and Rendering Essentials, [Електронний ресурс]. Режим доступу: https://www.elsevier.com/books/unreal-engine-lighting-and-rendering-essentials/harris/978-0-12-821222-2

8. Nelson, B. Unreal Engine Game Development Cookbook, [Електронний ресурс]. Режим доступу: https://www.crcpress.com/Unreal-Engine-Game-Development-Cookbook/Nelson/p/book/9780367331589

9. Carter, E. Unreal Engine VR Cookbook: Developing Virtual Reality with UE4, [Електронний ресурс]. Режим доступу: https://www.apress.com/gp/book/9781484217237

10. Mitchell, T. Blueprints and C++: Programming for Unreal Engine, [Електронний ресурс]. Режим доступу: https://www.routledge.com/Blueprints-and-C-Programming-for-Unreal-Engine/Mitchell/p/book/9780367511432

11. Lee, H. Unreal Engine Physics Essentials, [Електронний ресурс]. Режим доступу: https://www.springer.com/gp/book/9781484253099

12. Gomez, F. Building an RPG with Unreal, [Електронний ресурс]. Режим доступу: https://www.packtpub.com/product/building-an-rpg-with-unreal/9781838645560

13. Wallace, K. The Unreal Engine Developer's Guide to Mobile Platforms, [Електронний ресурс]. Режим доступу: https://www.oreilly.com/library/view/the-unreal-engine/9781838828844/

14. Ford, J. Unreal Engine Game Optimization, [Електронний ресурс]. Режим доступу: https://www.cambridge.org/core/books/unreal-engine-game-optimization/9781108417469

15. Peterson, R. Interactive 3D Graphics in Unreal Engine 4, [Електронний ресурс]. Режим доступу: https://www.wiley.com/en-us/Interactive+3D+Graphics+in+Unreal+Engine+4-p-9781119561798

ДОДАТОК А

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка програмного модуля авторської гри на ігровому рушії Unreal Engine 5

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 99,11% Схожість 0,89%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Озеранський В.С.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Суботін І.А.

Керівник роботи _____ Денисюк В.О.

ДОДАТОК Б

ІНСТРУКЦІЯ КОРИСТУВАЧА

1. Відкривши гру перше, що побаче користувач буде головне меню. Щоб розпочати гру потрібно натиснути кнопку host game, а щоб приєднатись до готової гри join game.

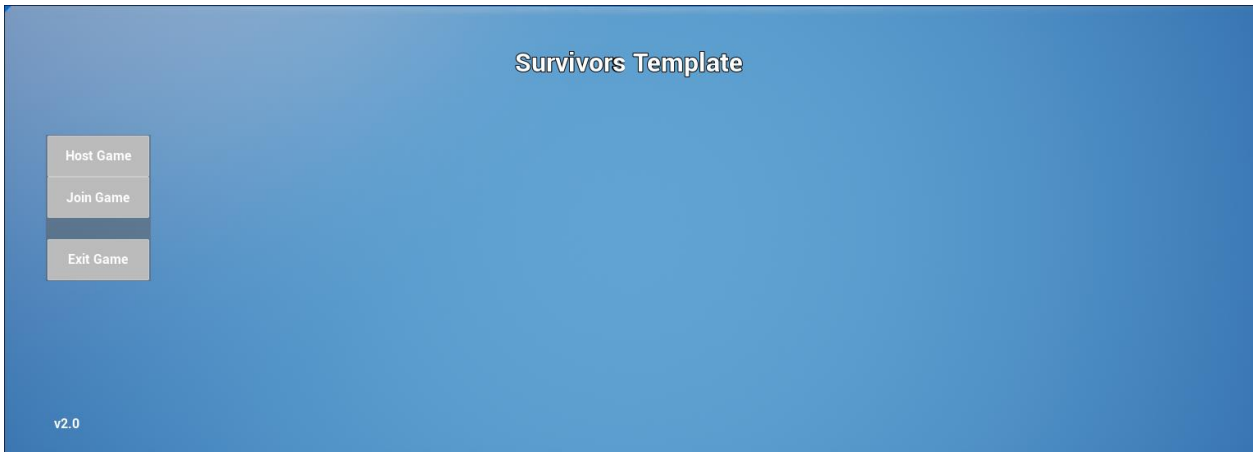

Рисунок Б.1 – Головне меню гри

2. Далі після початку гри на карті вороги розпочнуть атакувати гравця, перемагаючи їх гравець отримає досвід та підвищення рівня, отримання нових здібностей.

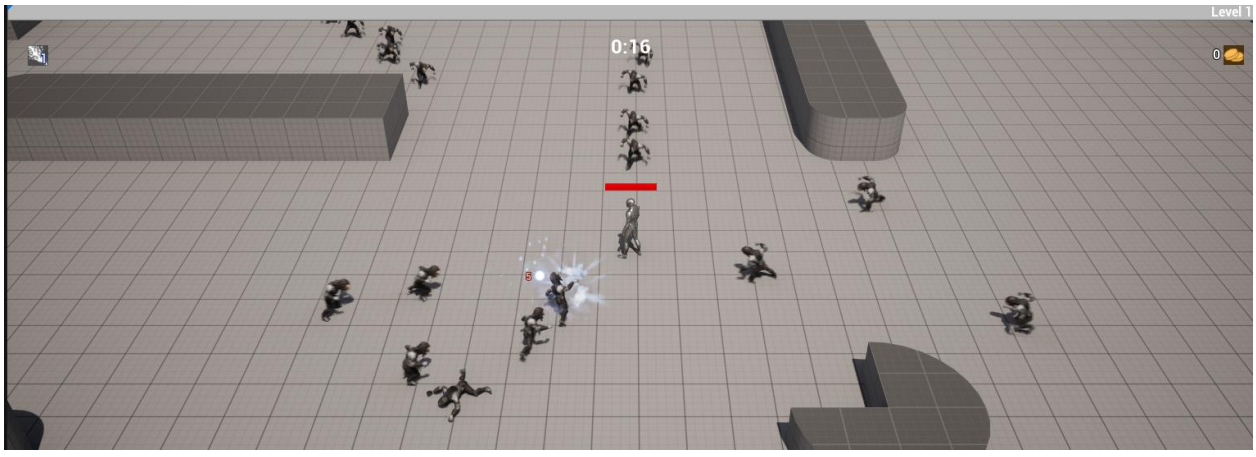

Рисунок Б.2 – Сцена початку гри

3. Після поразки гравця з’являється результат гри та через деякий час гравця перекидає назад до головного меню.

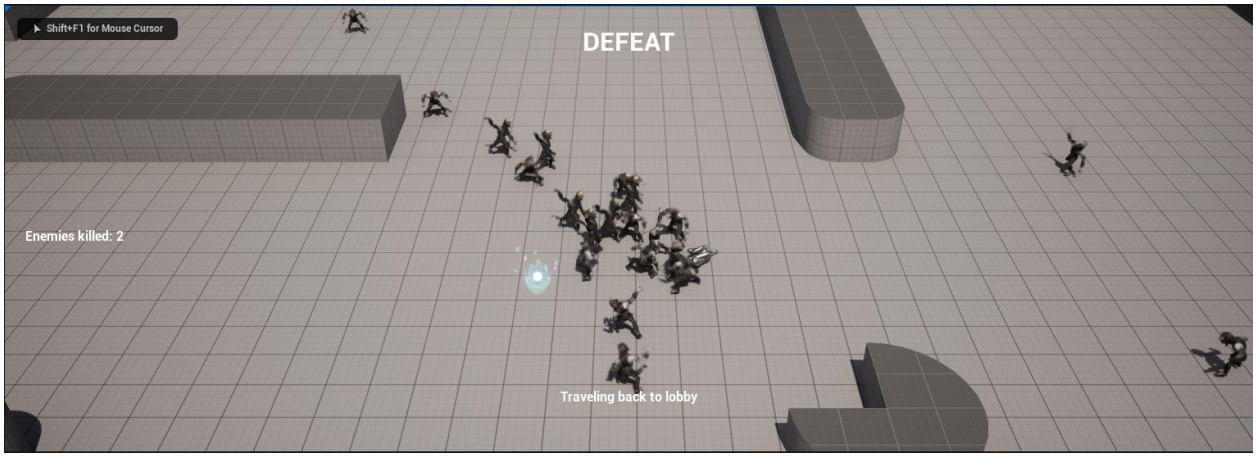

Рисунок Б.3 – Кінець гри

ДОДАТОК В

ГРАФІЧНИЙ ДОДАТОК

Рисунок Г.1 — Назва роботи

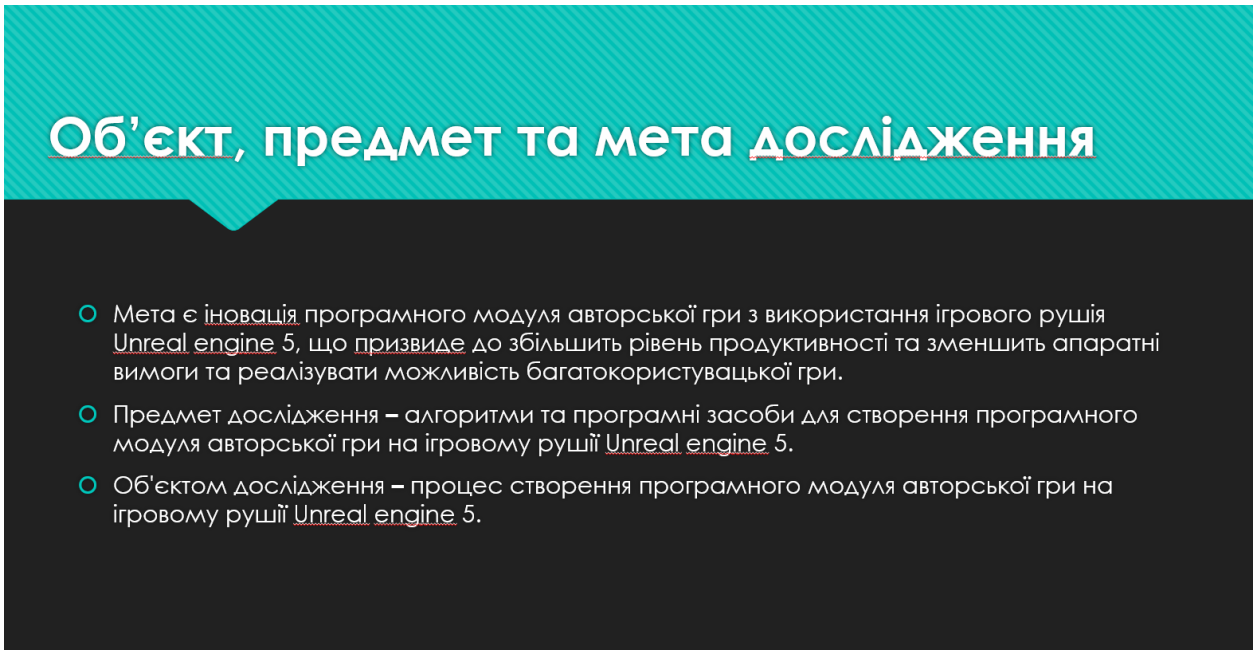

Рисунок Г.2 — Мета, об’єкт і предмет дослідження

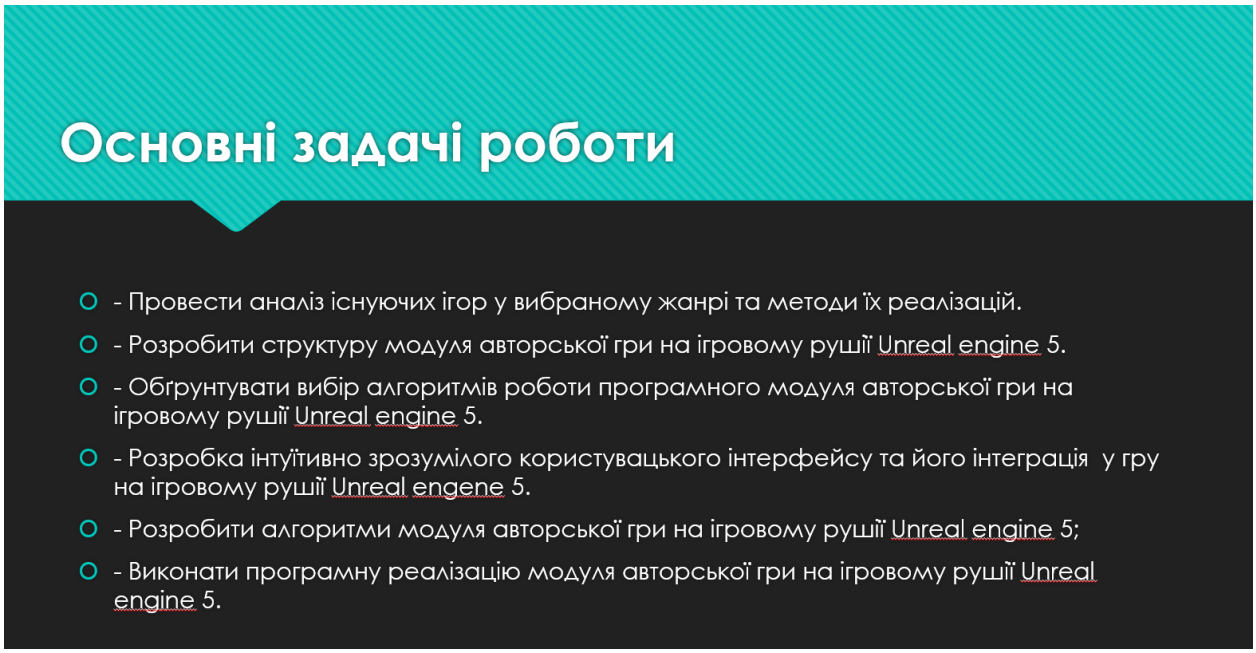

Рисунок Г.3 — Задачі бакалаврської дипломної роботи

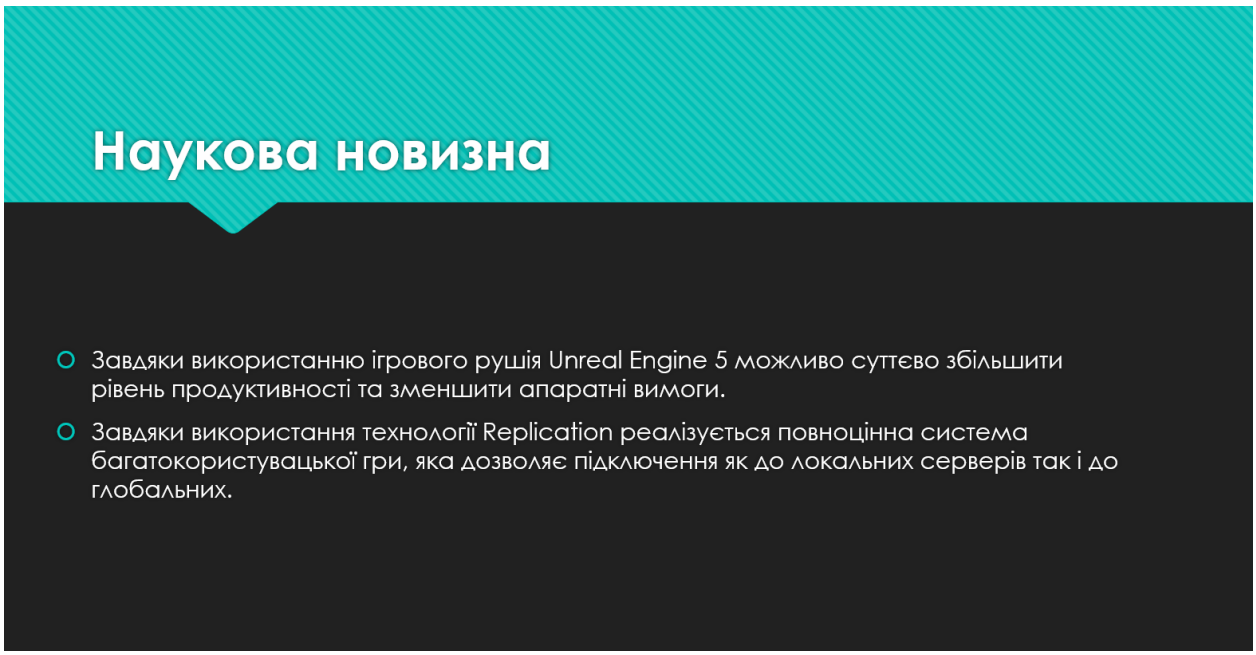

Рисунок Г.4 — Наукова новизна

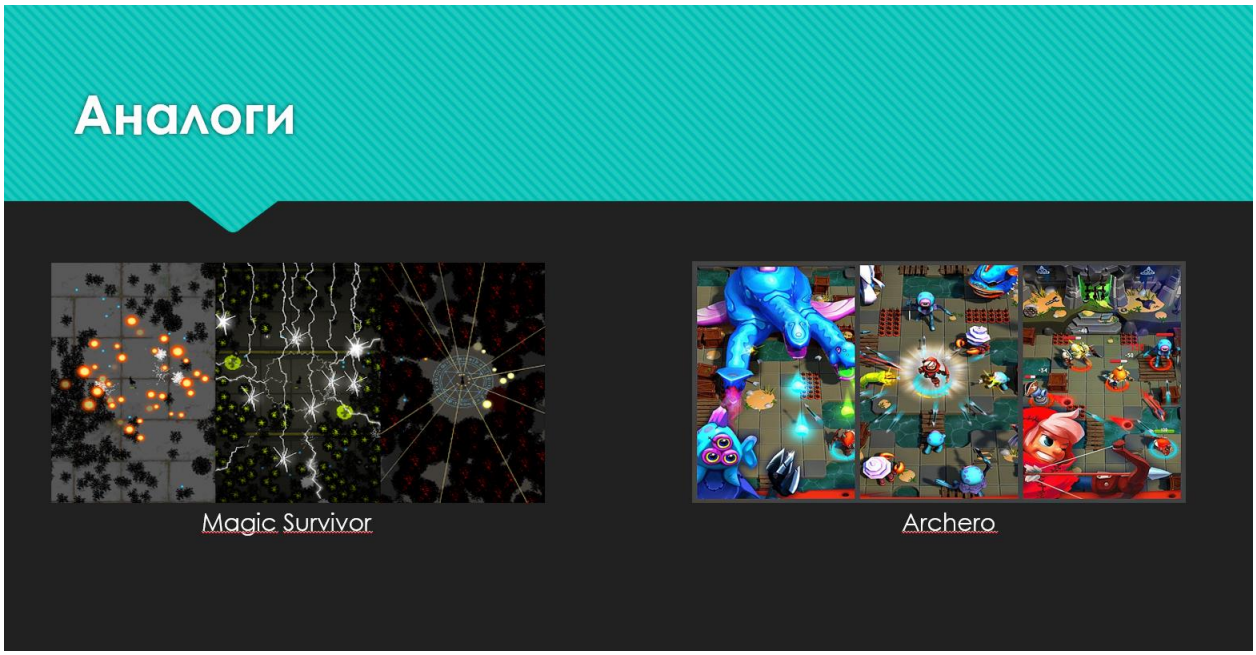

Рисунок Г.5 – Аналоги

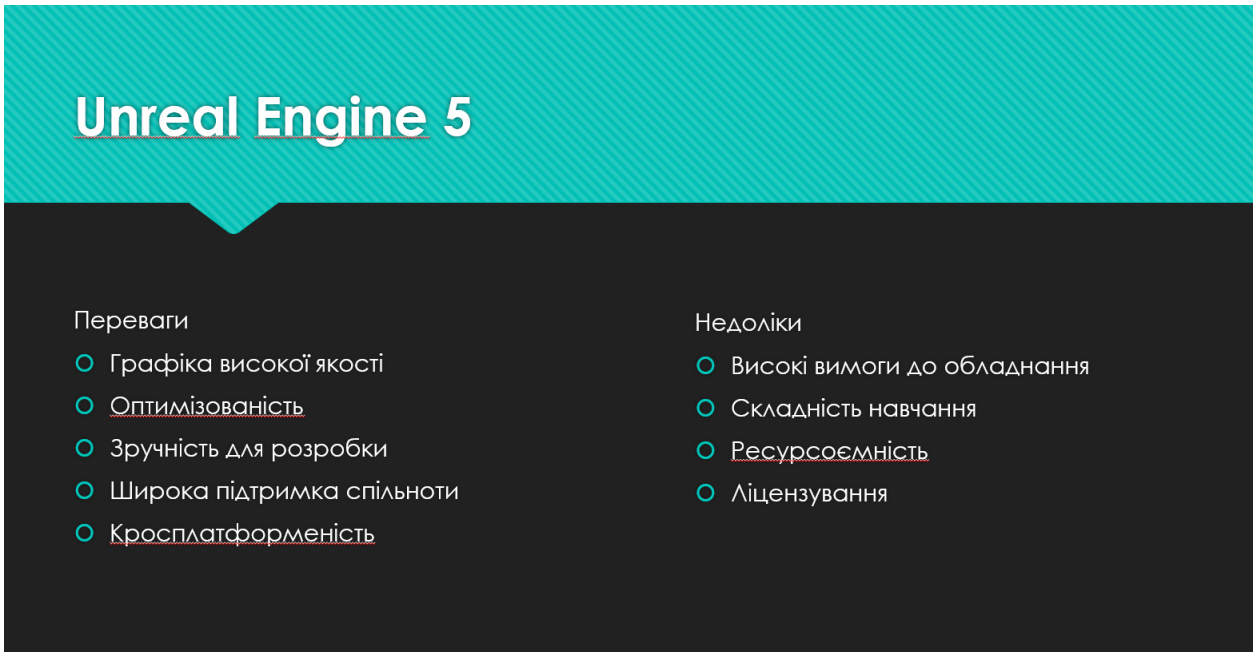

Рисунок Г.6 – Переваги та недоліки Unreal Engine 5

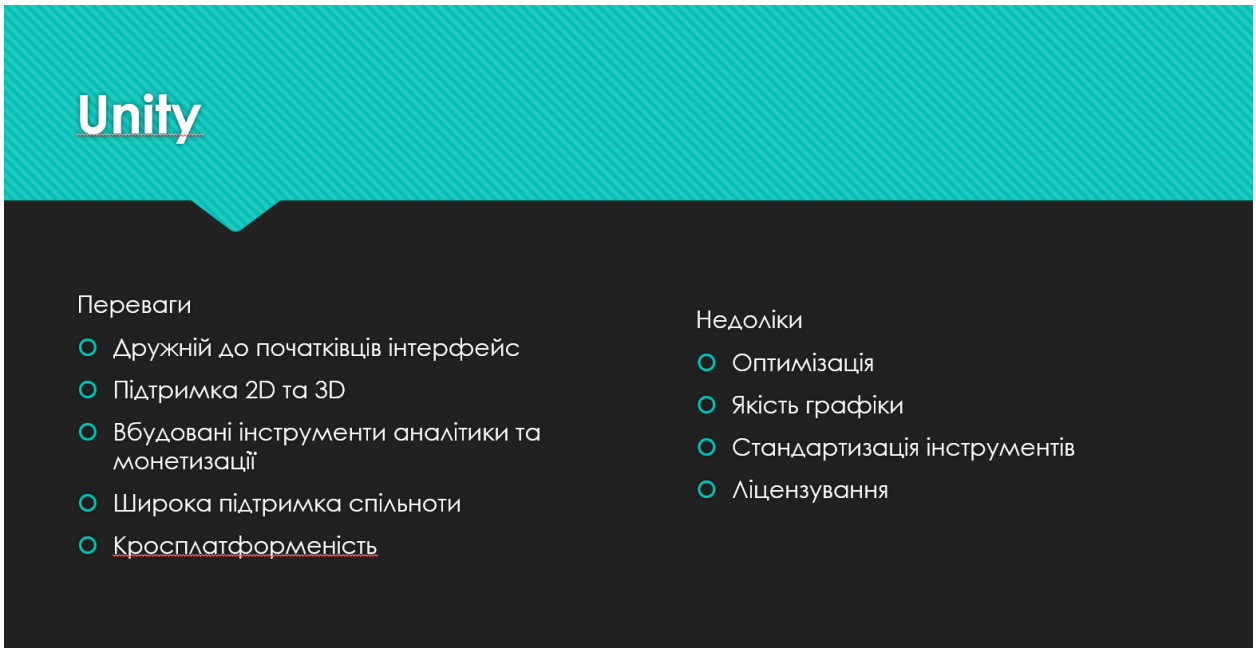

Рисунок Г.7 – Переваги та недоліки Unity

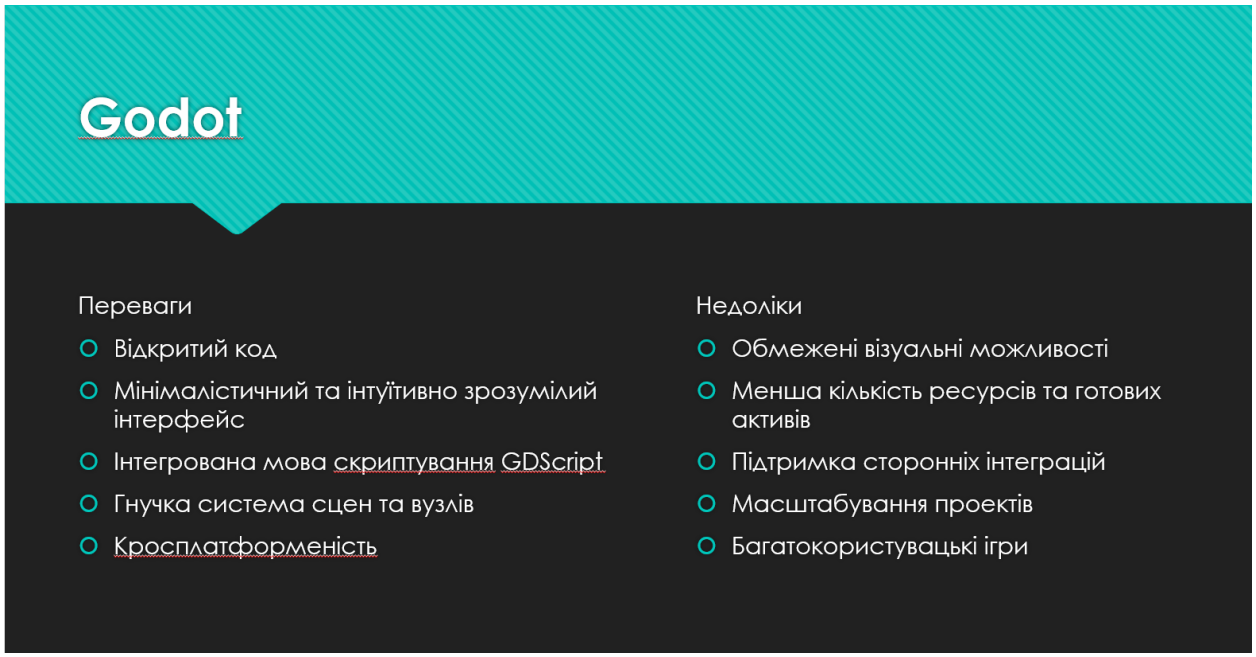

Рисунок Г.8 – Переваги та недоліки Godot

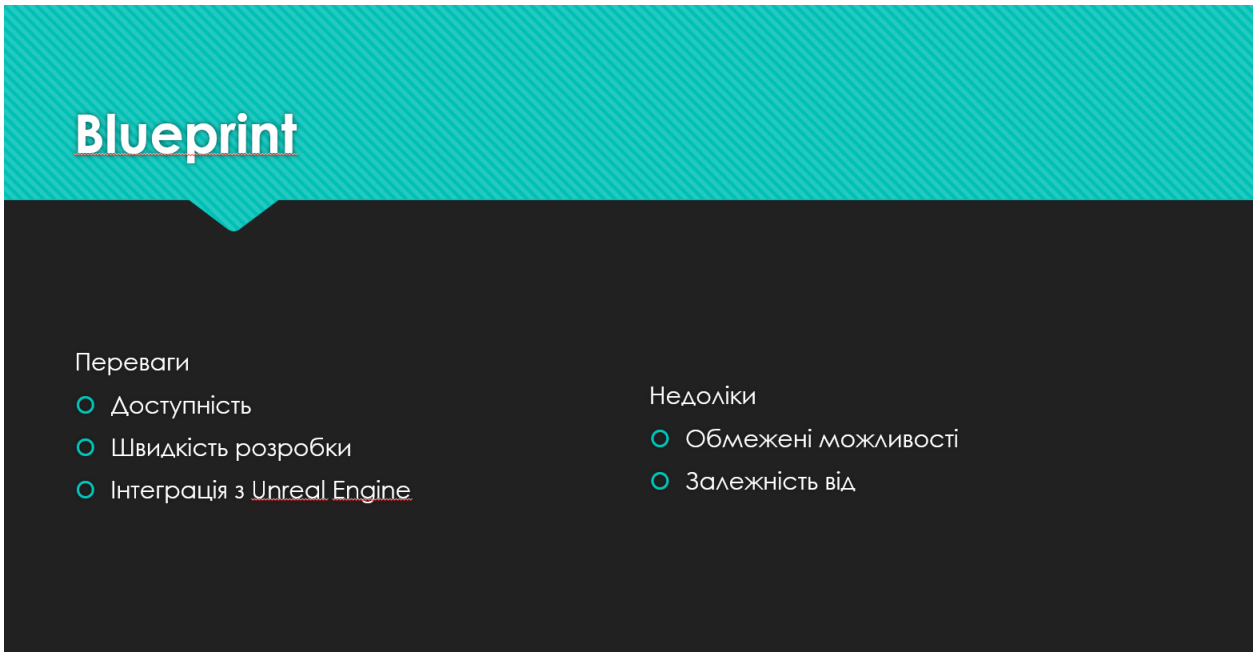

Рисунок Г.9 – Переваги та недоліки Blueprint

Рисунок Г.10 – Дерево наслідування класів Unreal Engine 5

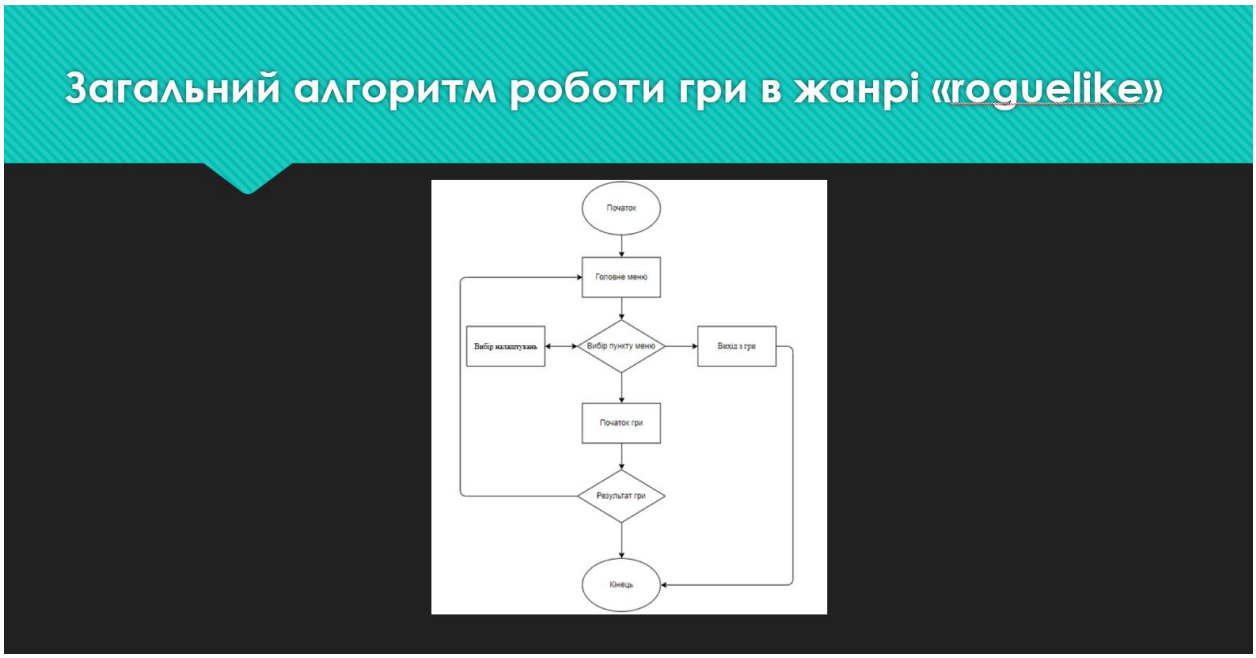

Рисунок Г.11 – Загальний алгоритм роботи гри в жанрі «roguelike»

Рисунок Г.12 – Алгоритм реалізації гри в жанрі «roguelike»

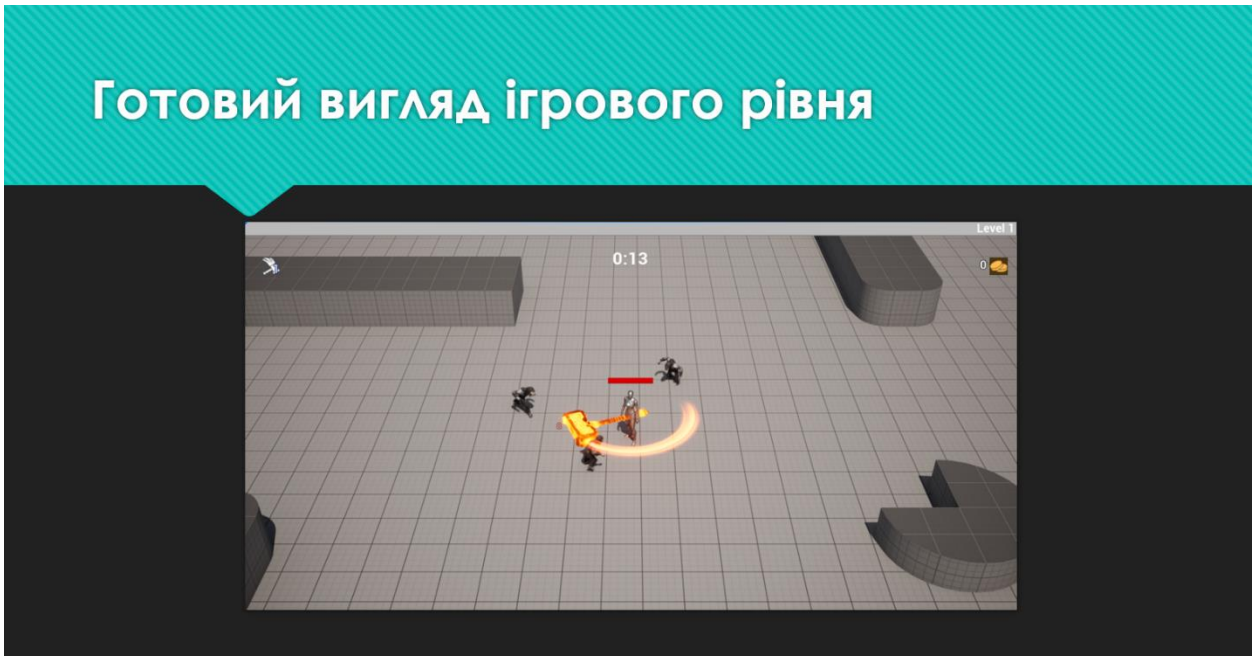

Рисунок Г.13 – Готовий вигляд ігрового рівня

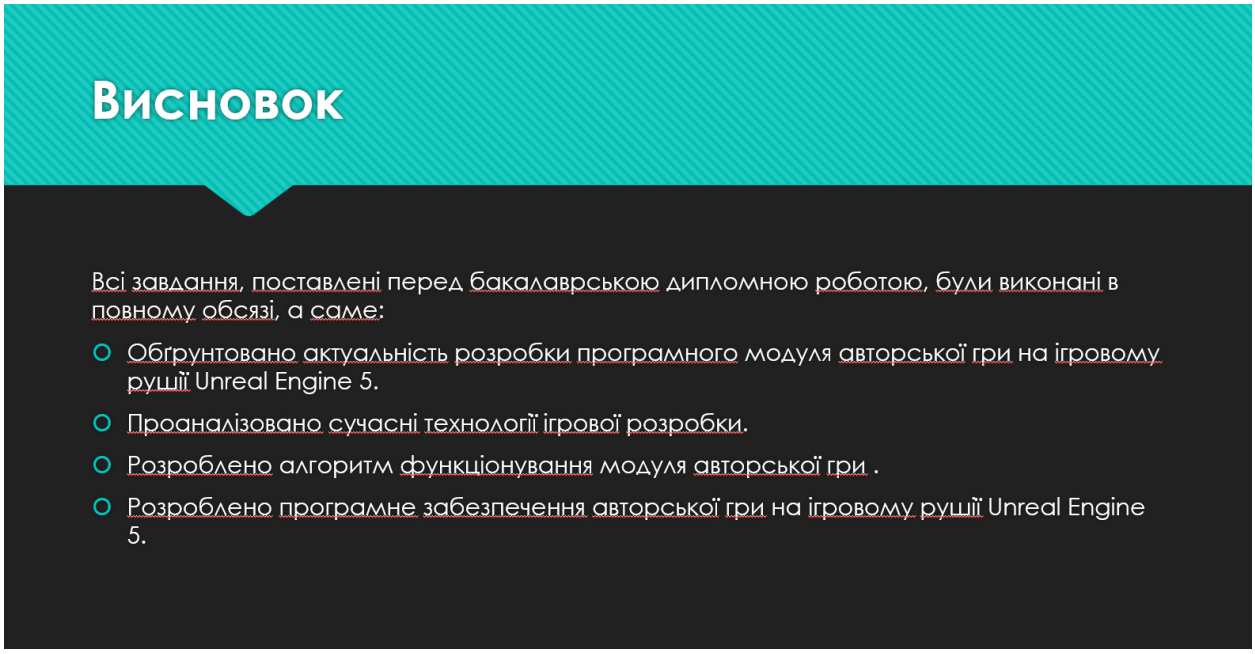

Рисунок Г.14 – Висновок

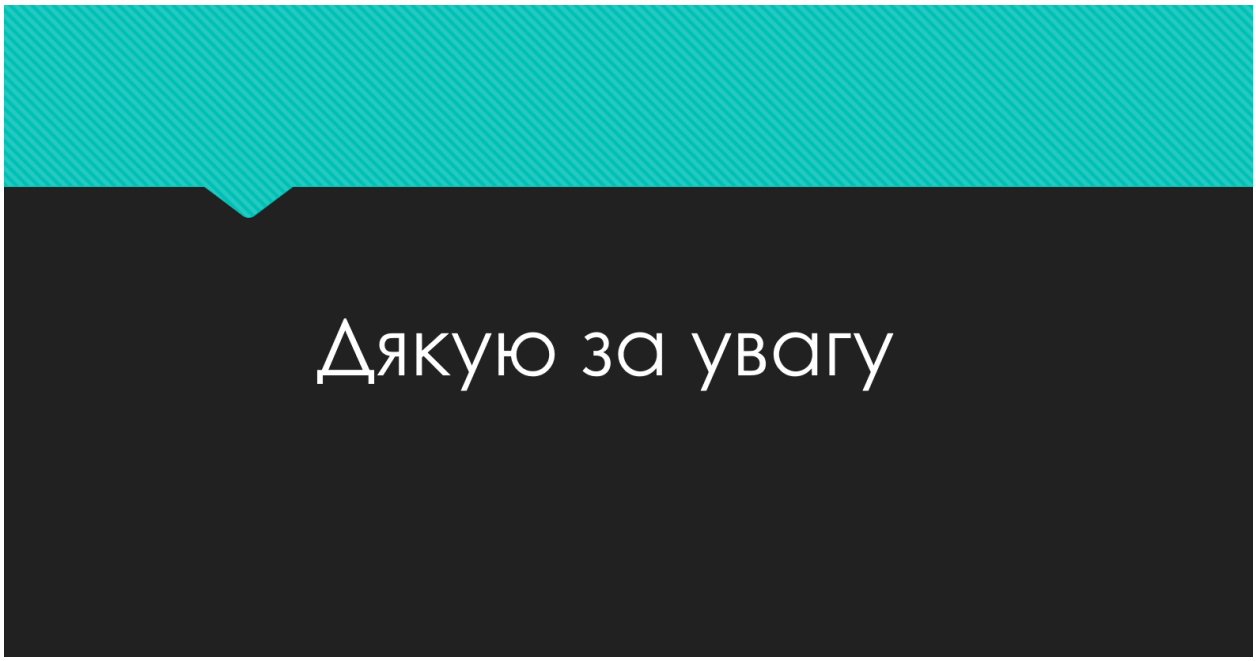

Рисунок Г.15 – Дякую за увагу