МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА РОЛЬОВОЇ ВІДЕОГРИ ЗА ДОПОМОГОЮ РУШІЯ UNREAL ENGINE»**

Виконав: студент 4 курсу, групи 6.1220
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми Комп'ютерні науки
(назва освітньої програми)
Р.Д. Крикунов
(ініціали та прізвище)

Керівник завідувач кафедри комп'ютерних наук, доцент, д.т.н. Шило Г.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри програмної інженерії, доцент, к.ф.-м.н., Лісняк А.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти бакалавр

Спеціальність 122 Комп'ютерні науки
(шифр і назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук,
д.т.н., доцент

_____ Шило Г.М.
(підпис)

" 25 " грудня 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Крикунову Роману Дмитровичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка рольової відеогри за допомогою рушія Unreal Engine

керівник роботи Шило Галина Миколаївна, доцент кафедри комп'ютерних наук, к.т.н.
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 21 » грудня 2023 року № 2180-с

2. Строк подання студентом роботи 05.06.2024

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Постановка задачі.
2. Основні теоретичні відомості.
3. Розробка гри на рушій Enreal Engine 5
4. Презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 25.12.2023

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	18.03.2024	
2.	Збір вихідних даних.	20.03.2024	
3.	Обробка методичних та теоретичних	5.04.2024	
	джерел.		
4.	Розробка першого та другого розділу.	15.04.2024	
5.	Розробка третього розділу.	05.05.2024	
6.	Оформлення та нормоконтроль	04.06.2024	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	22.06.2024	

Студент _____ Р.Д. Крикунов
(підпис) (ініціали та прізвище)

Керівник роботи _____ Г.М. Шило
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г.Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка рольової відеогри за допомогою рушія Unreal Engine»: 61 с., 42 рис., 4 джерел, 12 додатків.

АНІМАЦІЯ ПЕРСОНАЖА, ГРАФІЧНИЙ ДВИГУН, КЕРУВАННЯ ПЕРСОНАЖЕМ, ОПТИМІЗАЦІЯ ПРОДУКТИВНОВСТІ, ТЕХНІЧНА ПІДТРИМКА, ТОЧНІСТЬ УПРАВЛІННЯ.

Об’єкт дослідження – технічні можливості двигуна Unreal Engine для створення 2D платформерів, оптимізація продуктивності гри на Unreal Engine для досягнення стабільної швидкодії та мінімального обсягу пам'яті та вивчення можливостей різних методів управління персонажем у 2D платформерах.

Мета роботи: розробка 2D платформеру з використанням рушія Unreal Engine, з метою дослідження можливостей цього рушія для створення ефективних та захоплюючих ігрових середовищ.

Метод дослідження – було використано такі методи дослідження:

– аналіз існуючих 2D платформерів та можливостей використання Unreal Engine для створення;
– експериментальний метод: створення прототипів ігрових рівнів для вивчення ефективності різних інструментів та технік розробки у середовищі Unreal Engine;
– аналіз геймплею та взаємодії користувача з грою для виявлення сильних та слабких сторін розробленого платформера.

SUMMARY

Bachelor's Qualifying Theses "Development of a role-playing video game using the Unreal Engine": 61 pages, 42 figures, 4 references, 12 supplements.

CHARACTER ANIMATION, GRAPHICS ENGINE, CHARACTER CONTROL, PERFORMANCE OPTIMIZATION, TECHNICAL SUPPORT, PRECISION CONTROL.

Object of the study – technical capabilities of the Unreal Engine for creating 2D platformers, optimizing the performance of a game on Unreal Engine to achieve stable performance and minimum memory usage, and study of the possibilities of different methods of character management in 2D platformers.

Aim of the study: development of a 2D platformer using the Unreal Engine to explore the capabilities of this engine to create efficient and engaging gaming environments.

Method of research – the following research methods were used:

- analysis of existing 2D platformers and the possibilities of using Unreal Engine for creation;
- experimental method: creating prototypes of game levels to study the effectiveness of various development tools and techniques in the Unreal Engine environment;
- analysis of gameplay and user interaction with the game to identify the strengths and weaknesses of the developed platformer.

ЗМІСТ

ВСТУП

У сучасному світі відеоігри стали не лише розважальними продуктами, а й важливим елементом культури та мистецтва. Швидкий технологічний розвиток та поширення потужних інструментів розробки відкривають безмежні можливості для створення захоплюючих ігрових світів, які вражають своєю глибиною та реалізмом.

Одним із найпопулярніших та потужних інструментів для розробки відеоігор є рушій Unreal Engine. Його використання дозволяє розробникам створювати ігри різного жанру та рівня складності, від простих інді-проектів до великих AAA-тайтлів. Особливою популярністю користується розробка рольових ігор, які здатні занурити гравців у фантастичні світи, де вони можуть відчути себе героями незвичайних пригод.

Мета роботи: розробка 2D платформеру з використанням рушія Unreal Engine та дослідження можливостей цього двигуна для створення ефективних та захоплюючих ігрових середовищ.

Задачі дослідження включають в себе детальний аналіз функціональних можливостей рушія Unreal Engine для створення рольових ігор, визначення оптимальних стратегій розробки геймплею та механік гри, а також вивчення методів оптимізації продуктивності та підвищення якості готового продукту.

Особлива увага буде приділена технічним аспектам розробки, таким як оптимізація продуктивності та використання ресурсів, щоб забезпечити плавний та стабільний геймплей.

1 АНАЛІЗ ІСНУЮЧИХ ПОПУЛЯРНИХ RPG ІГОР

1.1 Огляд сучасних трендів у галузі рольових ігор

Сучасний світ відеоігор вражає уяву своїм калейдоскопом жанрів та ігрових концепцій, що постійно розвиваються та еволюціонують. Цей динамічний процес не оминув і сферу рольових ігор (RPG), де з'являються все нові та цікаві тренди, які формують сучасний ландшафт цього захоплюючого жанру.

Традиційні RPG, що ґрунтуються на покроковому бої, дослідженні підземель та прокачці персонажа, все ще залишаються популярними. Проте, розробники постійно шукають нові шляхи розвитку жанру, розширюючи його межі та пропонуючи гравцям свіжі враження:

– інновації в бойовій системі: Покрокові бої поступаються місцем динамічним та видовищним сутичкам, де гравцям треба швидко приймати рішення та комбінувати різні навички;
– глибші сюжети та персонажі: RPG все більше нагадують кінематографічні твори, пропонуючи гравцям не лише захоплюючі пригоди, але й складні моральні дилеми та багатогранних персонажів;
– соціальні елементи: Багатокористувацькі режими стають невід'ємною частиною RPG, дозволяючи гравцям об'єднуватися у команди, досліджувати віртуальні світи разом та спільно долати виклики.

Сучасні RPG пропонують неймовірне розмаїття піджанрів, що задовольнить будь-якого гравця:

– фентезі: Класичні RPG, що переносять гравців у світи магії, драконів та відважних лицарів;

- наукова фантастика: RPG, де майбутнє переплітається з високими технологіями, космічними пригодами та міжзоряними війнами;
- історичні RPG: RPG, що дозволяють поринути в атмосферу минулих епох, відчути себе учасником історичних подій та дослідити культури різних часів;
- пост-апокаліптика: RPG, де гравці досліджують руїни зруйнованого світу, борються за виживання та шукають нове місце у ворожому середовищі;
- аніме-RPG: RPG, що черпають натхнення з японської анімації, пропонуючи яскравий візуальний стиль, динамічні бої та емоційні історії.

Жанр RPG продовжує динамічно розвиватися, поєднуючи в собі елементи інших жанрів, інтегруючи нові технології та пропонуючи гравцям все більш захоплюючий та незабутній ігровий досвід:

- віртуальна реальність: RPG у VR переносять гравців у віртуальні світи з неймовірною глибиною та реалістичністю, роблячи пригоди ще більш вражаючими;
- штучний інтелект: RPG з вдосконаленим штучним інтелектом пропонують гравцям більш динамічних та непередбачуваних супротивників, а також нові можливості для взаємодії з неігровими персонажами;
- хмарні технології: RPG у хмарі дозволяють грати в будь-який час і в будь-якому місці, без прив'язки до потужних ПК або консолей.

Таким чином, жанр RPG переживає справжній розквіт, пропонуючи гравцям невичерпне джерело захоплюючих історій, незабутніх пригод та емоцій. З постійним розвитком та появою нових технологій, RPG й надалі залишатимуться одним із найпопулярніших та улюблених жанрів у світі відеоігор.

1.1.1 Рогалики та роглайти: поєднання складності та повторного проходження

Одним із найвиразніших трендів є зростання популярності рогаликів (roguelikes) та роглайтів (roguelites). Ці піджанри RPG характеризуються низкою спільних рис:

- процедурна генерація рівнів: Кожне проходження гри генерується випадковим чином, роблячи ігровий процес динамічним та непередбачуваним;
- висока відтворюваність: Завдяки процедурній генерації та смерті персонажа, що кидає виклик, рогалики та рогалайти стимулюють повторне проходження, пропонуючи нові виклики та можливості з кожним новим запуском гри;
- надзвичайна складність: Ці піджанри RPG відомі своїм нещадним рівнем складності, який кидає виклик навіть досвідченим гравцям.

Прикладами популярних рогаликів та роглайтів є Hades, Skull: The Hero Slayer, Dead Cells, The Binding of Isaac та Rogue Legacy. Ці ігри завоювали визнання завдяки захоплюючому геймплею, глибині та складності, пропонуючи унікальний та захоплюючий досвід для любителів RPG. Кожна з цих ігор має свію унікальну особливість геймплею, яка і відрізняє її серед інших.

1.1.2 Рогалики та роглайти: поєднання складності та повторного проходження

Іншим ключовим трендом є зростання популярності RPG з відкритим світом. Ці ігри надають гравцям неперевершену свободу дій, дозволяючи досліджувати великі та детально пропрацьовані віртуальні світи на власний розсуд.

RPG з відкритим світом пропонують:

– нелінійне проходження: Гравці вільні обирати власний шлях, виконувати завдання в будь-якому порядку та досліджувати світ у власному темпі;
– різноманіття контенту: Відкриті світи рясніють квестами, персонажами, секретами та іншими цікавими елементами, які стимулюють дослідження та повторне відвідування;
– вплив на світ: Деякі RPG з відкритим світом дозволяють гравцям впливати на хід подій у світі, змінюючи його та роблячи його більш динамічним.

Прикладами популярних RPG з відкритим світом є The Legend of Zelda: Breath of the Wild, Red Dead Redemption 2, The Witcher 3: Wild Hunt, Skyrim та Elden Ring. Ці ігри вражають своїми масштабами, свободою та глибиною, пропонуючи гравцям незабутні пригоди у віртуальних світах.

1.1.3 Візуальний Стиль: Занурення у Світи, Створені з Фантазією

Візуальний стиль у сучасних RPG відіграє все більш важливу роль, стаючи не просто обгорткою для геймплею, а й самостійним елементом, що створює неповторну атмосферу та доповнює ігровий досвід. Розробники вдаються до найрізноманітніших візуальних прийомів, щоб занурити гравців у світи, створені з фантазією та вигадкою.

Сучасні RPG пропонують гравцям широкий спектр візуальних стилів, від мальовничого фентезі до темного фентезі, кіберпанку, наукової фантастики та інших жанрів. Кожен стиль має свої візуальні особливості, які створюють унікальну атмосферу та емоційне забарвлення гри:

– реалістичний стиль: Деталізовані текстури, реалістичне освітлення, 3D-моделі персонажів та оточення створюють ефект присутності у віртуальному світі, роблячи ігровий процес більш захоплюючим;

– стилізований візуал: Використання карикатурних персонажів, яскравих кольорів та незвичайних пропорцій дає можливість створити легку та веселу атмосферу, або ж підкреслити темні та серйозні теми;
– піксельний арт: Ретро-стиль піксельного арту, що нагадує про класичні ігри, користується популярністю завдяки своїй простоті, ностальгічній атмосфері та безмежним можливостям для креативу.

Прикладами ігор які мають свій власний, ні на що не схожий візуал можна назвати:

Disco Elysium

Ця детективна RPG вражає не лише глибоким сюжетом та цікавими персонажами, але й оригінальною графікою, що нагадує комікси. Розробники використовують ручний стиль малювання, що підкреслює темну та похмуру атмосферу гри. Деталізовані портрети персонажів та мальовничі фони створюють відчуття того, що ви опинилися у справжньому детективному романі (див. рис. 1.1)

Рисунок 1.1 – Постер гри Disco Elysium

Hades

Roguelike Hades відрізняється яскравою та барвистою графікою, що нагадує грецьку міфологію. Персонажі та оточення оформлені в стилі мультфільмів, що робить гру візуально привабливою та динамічною. Цей

візуальний стиль підкреслює веселий та захоплюючий геймплей, а також додає трохи гумору до серйозної теми міфів та легенд (див. рис. 1.2).

Рисунок 1.2 – Постер гри Hades

Cuphead

"Run and gun" RPG Cuphead зачаровує своїм унікальним візуальним стилем, що імітує мультфільми 1930-х років. Ручний стиль малювання, яскраві кольори та експресивні персонажі створюють атмосферу ностальгії та веселощів. Цей візуальний стиль робить гру не лише візуально привабливою, але й додає їй складності, адже анімовані вороги та динамічний геймплей кидають виклик гравцям (див. рис. 1.3).

Рисунок 1.3 – Постер гри Cuphead

Ori and the Will of the Wisps

Платформер-RPG Ori and the Will of the Wisps вражає мальовничою графікою, що нагадує акварельні картини. Деталізовані фони, м'які кольори та елегантні анімації створюють атмосферу краси та спокою. Цей візуальний стиль підкреслює зворушливу історію про дружбу та самопожертву, а також робить гру візуально приємною для гравців будь-якого віку (див. рис. 1.4).

Рисунок 1.4 – Постер гри Ori and the Will of the Wisps

Hollow Knight

Metroidvania RPG Hollow Knight занурює гравців у темний та атмосферний світ, що нагадує готичні казки. Деталізовані фони, чіткі лінії та похмурі кольори створюють відчуття таємниці та небезпеки. Цей візуальний стиль підкреслює складний геймплей та таємничу атмосферу гри, а також робить її візуально привабливою для любителів готики та фентезі (див. рис. 1.5).

Рисунок 1.5 – Постер гри Hollow Knight

Візуальний стиль не лише створює атмосферу та емоційне забарвлення гри, але й може впливати на геймплей. Наприклад, темний та похмурий візуал може підкреслити атмосферу небезпеки та тривоги, а яскравий та барвистий візуал може зробити геймплей більш розслабленним та веселим.

1.2 Характеристики та особливості успішних проектів

Успішність рольових відеоігор визначається не лише їхнім геймплеєм, але й рядом інших факторів, які, подібно до мозаїки, складають захоплюючу та незабутню картину для гравців.

1.2.1 Глибина сюжету та харизматичні персонажі

Захоплююча історія: Сюжет RPG повинен бути не лише цікавим, але й захоплюючим, з несподіваними поворотами та емоційними моментами, які змушують гравців співпереживати героям та прагнути дізнатися, що ж буде далі.

Пам'ятні персонажі: Персонажі RPG повинні бути не лише добре прописаними, але й харизматичними, з власною історією, мотивами та емоціями. Гравці повинні мати можливість співпереживати їм, захоплюватися їхніми подвигами або сміятися над їхніми кумедними моментами і тоді рівень поглиблення у гру буде в рази більшим.

1.2.2 Якісна графіка та звукове супроводження

Візуальна краса: Графіка RPG повинна бути приємною для очей, відповідати атмосфері гри та створювати відчуття занурення у віртуальний світ.

Захоплюючий звук: Звукове супроводження RPG повинне підкреслювати атмосферу гри, емоції персонажів та динаміку боїв. Музика, звукові ефекти та озвучення повинні бути на високому рівні та не дратувати гравців.

1.2.3 Якісна графіка та звукове супроводження

Унікальні системи вибору: Ігри, де вибір гравця впливає на подальший розвиток сюжету або взаємовідносини персонажів, роблять геймплей більш цікавим та динамічним.

Глибока система розвитку персонажів: Можливість прокачувати персонажів, обирати для них навички та обладнання робить гру більш захоплюючою та дозволяє гравцям створювати унікальні збірки.

Висока ступінь іммерсії: Ігри, де гравці можуть повністю поглибитися у віртуальний світ, роблячи свій власний вибір та відчуваючи себе частиною історії, стають більш емоційними та незабутніми.

1.3 Постановка задачі

Метою роботи є розробка 2D-платформеру з використанням рушія Unreal Engine та дослідження можливостей цього двигуна для створення ефективних та захоплюючих ігрових середовищ.

Для досягнення цієї мети необхідно виконати наступні задачі:

– дослідження функціональних можливостей ігрових рушіїв;
– аналіз напрямів розвитку сучасної ігрової індустрії;
– розробка персонажу гри;
– розробка локації;
– розробка ворога;
– реалізація алгоритму руху ворога та взаємодії з персонажу;
– тестування 2D-платформеру.

2 ВСТУП ДО UNREAL ENGINE 5

Ігрові рушії – це комплексні програмні середовища, які надають розробникам набір інструментів для створення відеоігор. Вони включають в себе компоненти для обробки графіки, звуку, фізики, штучного інтелекту, мережевих функцій та інших аспектів, необхідних для створення інтерактивного ігрового досвіду.

2.1 Компоненти ігрових рушіїв

Основні компоненти ігрових рушіїв:

- графічний рушій: Відповідає за рендеринг зображень на екрані, використовуючи як 2D, так і 3D технології. Включає шейдери, текстури, освітлення та постобробку;
- фізичний рушій: Обробляє взаємодію об'єктів у грі, враховуючи закони фізики. Це включає в себе колізії, гравітацію, жорсткість тіл та інші фізичні явища;
- звуковий рушій: Забезпечує відтворення звукових ефектів та музики. Може підтримувати 3D-звук та реалістичні аудіоефекти;
- інструменти розробки: Набір інструментів для створення контенту, таких як редактори рівнів, системи сценаріїв та анімації, конструктори інтерфейсів користувача та інші;
- штучний інтелект (ШІ): Відповідає за поведінку неігрових персонажів (NPC), включаючи шляхобудування, прийняття рішень, реагування на дії гравця та інші аспекти ШІ;
- мережеві функції: Забезпечують можливість багатокористувацької гри через інтернет або локальні мережі.

Використання ігрового рушія надає багато переваг , як приклад економію часу та коштів, гнучкість розробки, масштабованість.

2.1.1 Популярні ігрові рушії

Серед найпопулярніших ігрових рушіїв можна виділити декілька, що використовуються дуже часто:

Unreal Engine

Один з найпотужніших рушіїв на сьогодняшній день який використовується для розробки ігор різних жанрів (див. рис. 2.1).

Рисунок 2.1 – Лого рушія Unreal Engine

Unity

Популярний ігровий рушій який використовується для створення широкого спектру ігор, від мобільних до AAA-проектів (див. рис. 2.2).

Рисунок 2.2 – Лого рушія Unity

Godot

Безкоштовний та відкритий ігровий рушій, який стає популярним завдяки своїй простоті у використанні (див. рис. 2.3).

Рисунок 2.3 – Лого рушія Godot

CryEngine

Потужний ігровий рушій від компанії Crytek який вона використовує у розробці своїх ігор (див. рис. 2.4).

Рисунок 2.4 – Лого рушія CryEngine

2.2 Переваги Unreal Engine

Unreal Engine – один з найвідоміших і найпотужніших ігрових рушіїв, який використовується для створення відеоігор, інтерактивних додатків, віртуальної реальності (VR) та доповненої реальності (AR). Цей рушій, розроблений компанією Epic Games, вперше був представлений у 1998 році. З моменту свого дебюту Unreal Engine зазнав численних оновлень і вдосконалень, перетворюючись на універсальний інструмент для розробників у різних галузях. Сьогодні актуальною версією є Unreal Engine 5, випущена в 2021 році.

Цей ігровий рушій став вибором багатьох студій, завдяки своїй універсальності, продуктивності та інноваціям.Це дозволяє створювати візуально вражаючі ігри та додатки з високим рівнем деталізації та реалістичності.

2.2.1 Основні переваги Unreal Engine

Однією з ключових переваг Unreal Engine є його універсальність. Він підтримує розробку для широкого спектру платформ, включаючи:

- пк (Windows, Mac, Linux);
- консолі (PlayStation, Xbox, Nintendo Switch);
- мобільні пристрої (iOS, Android);
- vr/ar гарнітури (Oculus, HTC Vive, HoloLens).

Це дозволяє розробникам створювати ігри та додатки, які можна легко адаптувати та випустити на різних пристроях, досягаючи широкої аудиторії.

Unreal Engine відомий своєю високою продуктивністю, що дозволяє створювати складні та графічно насичені проекти без значних компромісів у швидкодії. Він оптимізований для роботи з сучасними апаратними засобами, використовуючи переваги багатоядерних процесорів і потужних графічних процесорів (GPU).

Це досягається завдяки оптимізації для багатоядерних процесорів, використанню потужних графічних процесорів (GPU) та деяким інструментам оптимізації:

- profiling та оптимізація: Unreal Engine надає розробникам потужні інструменти для профілювання та оптимізації, такі як Unreal Insights та GPU Profiler. Ці інструменти дозволяють детально аналізувати продуктивність гри, ідентифікувати вузькі місця та оптимізувати використання ресурсів;
- automatic LOD (Level of Detail): Система автоматичного управління рівнями деталізації (LOD) дозволяє знижувати деталізацію об'єктів на відстані, зберігаючи високу продуктивність без втрати візуальної якості. Це особливо важливо для великих відкритих світів з великою кількістю об'єктів;
- efficient memory management: Unreal Engine ефективно управляє пам'яттю, оптимізуючи завантаження текстур, моделей та інших ресурсів. Це забезпечує зменшення затримок при завантаженні сцен та більш плавний ігровий досвід;
- vr/ar гарнітури (Oculus, HTC Vive, HoloLens);
- adaptive sync та variable rate shading (VRS): Підтримка технологій адаптивної синхронізації та змінної частоти шейдінгу дозволяє оптимізувати використання GPU, знижуючи навантаження в менш критичних зонах кадру та підвищуючи продуктивність без помітної втрати якості зображення.

2.2.2 Графічна потужність

Lumen – це революційна система динамічного глобального освітлення і відбиття світла, представлена в Unreal Engine 5. Вона забезпечує реалістичне освітлення в режимі реального часу, реагуючи на зміни в сцені, як-от переміщення об'єктів, зміни освітлення або часу доби. Lumen створює

високоякісне глобальне освітлення, яке значно покращує візуальне сприйняття гри без необхідності попереднього рендерингу або складної настройки освітлення (див. рис. 2.5).

Рисунок 2.5 – Приклад роботи системи Lumen

Nanite – це технологія віртуалізації геометрії, яка дозволяє рендерити величезні кількості полігонів у реальному часі без втрат продуктивності. Вона автоматично обробляє та оптимізує геометрію об'єктів, що дозволяє використовувати моделі з мільярдами полігонів без необхідності їх ручного зменшення або оптимізації (див. рис. 2.6).

Рисунок 2.6 – Приклад роботи з технологією Nanite

2.3 Особливості Unreal Engine

Unreal Engine вирізняється своїми передовими технологіями та можливостями, які роблять його вибором багатьох студій розробників по всьому світу. Ось деякі з ключових особливостей, які роблять Unreal Engine унікальним:

Фотореалістичний рендеринг

Завдяки Lumen та Nanite Unreal Engine 5 забезпечує неперевершену якість зображення. Lumen дозволяє отримати динамічне глобальне освітлення, яке реагує на зміни сцени в реальному часі. Nanite забезпечує детальність геометрії без необхідності оптимізації полігонів вручну.

Метасистеми та анімація

Рушій пропонує потужні інструменти для роботи з анімаціями, включаючи Control Rig для створення анімаційних рігів, та MetaHuman Creator для створення реалістичних персонажів (див. рис. 2.7).

Рисунок 2.7 – Приклад моделей створених за допомогою MetaHuman Creator

Інтеграція з реальними об’єктами

Unreal Engine активно використовується в кіноіндустрії для створення віртуальних декорацій та спецефектів. Наприклад, технологія віртуальних

виробничих майданчиків дозволяє поєднувати реальні та віртуальні елементи у реальному часі.

Blueprints і C++

Це візуальна мова програмування, яка дозволяє розробникам створювати ігрову логіку без написання коду. Це робить процес розробки більш доступним для дизайнерів та художників, які можуть працювати над ігровими механіками та взаємодіями без глибоких знань програмування. Розробники можуть обирати між візуальним програмуванням через Blueprints та традиційним програмуванням на C++. Це дозволяє гнучко підходити до створення логіки гри та оптимізації (див. рис. 2.8).

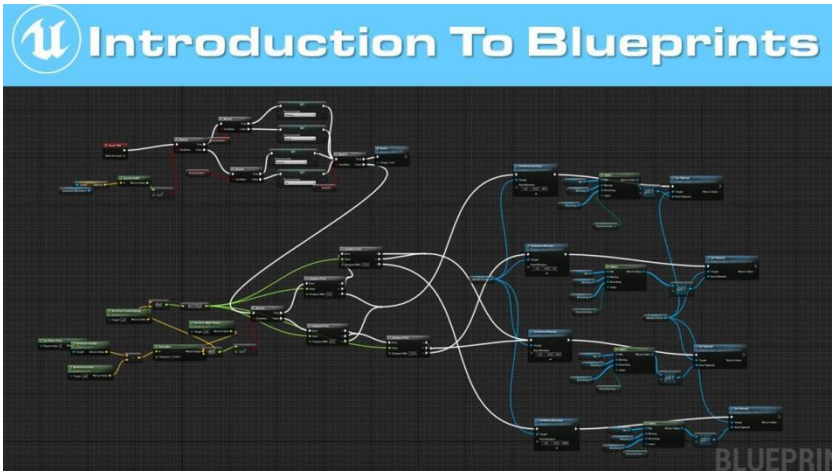

Рисунок 2.8 – Приклад роботи з використанням технології Blueprints

Підтримка VR та AR

Unreal Engine надає розширені можливості для розробки ігор та додатків віртуальної та доповненої реальності, забезпечуючи високий рівень інтерактивності та реалістичності.

Мережеві функції та мультиплеєр

Вбудовані інструменти для створення багатокористувацьких ігор, включаючи реплікацію об'єктів, синхронізацію станів та управління серверами.

Marketplace

Unreal Engine має власний магазин, де розробники можуть купувати та продавати різноманітні ресурси: моделі, анімації, текстури, звуки, шейдери та код. Це значно спрощує процес розробки, дозволяючи використовувати вже готові ресурси та зосередитися на унікальних аспектах гри.

Трасування променів у реальному часі (Ray Tracing)

Unreal Engine підтримує трасування променів, що дозволяє створювати високоякісні візуальні ефекти, такі як реалістичні відбиття, тіні та освітлення, додаючи глибини і деталізації в ігрові сцени.

Chaos Physics and Destruction

Нова система фізики, яка дозволяє створювати реалістичні симуляції та руйнування об'єктів, що додає глибини і динаміки ігровим сценам (див. рис. 2.9).

Рисунок 2.9 – Приклад можливостей функії Chaos Physics and Destruction

World Partition

Технологія, яка автоматично ділить ігровий світ на підрозділи, оптимізуючи завантаження ресурсів і забезпечуючи плавний досвід для гравців у великих відкритих світах (див. рис. 2.10).

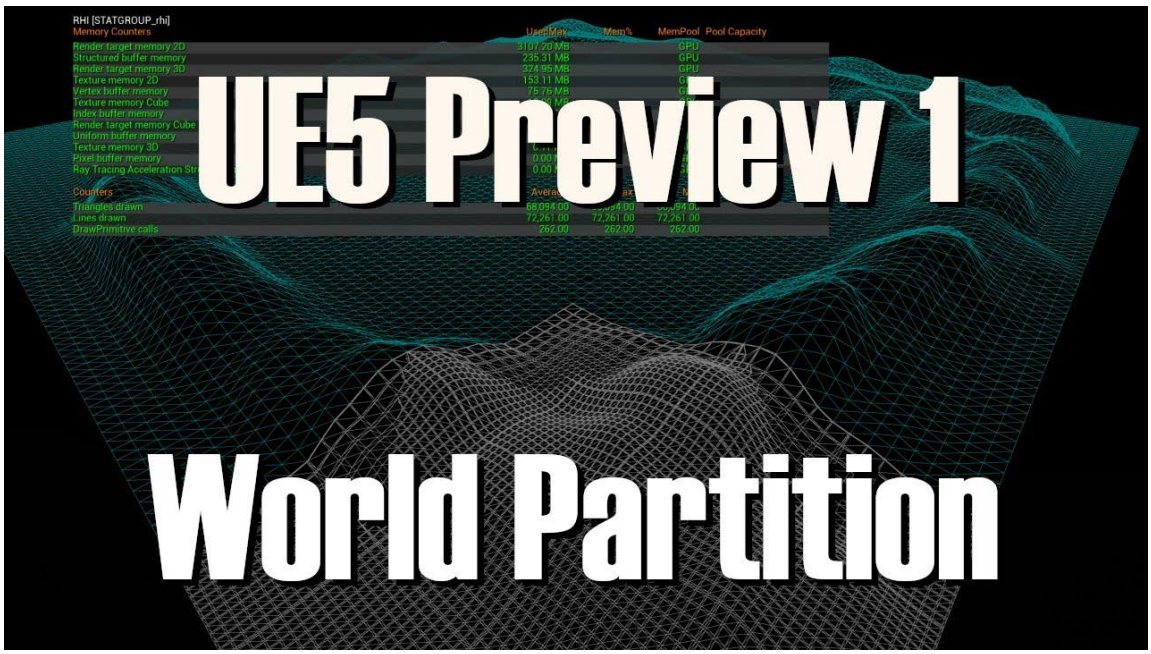

Рисунок 2.10 – Приклад роботи технології World Partition

3 РОЗРОБКА 2D ПЛАТФОРМЕРА НА РУШІЇ UNREAL ENGINE 5

Розробка 2D платформера на Unreal Engine 5 (UE5) включає кілька ключових етапів, які охоплюють всі аспекти створення гри — від встановлення рушія до налаштування ігрових механік та остаточного тестування.

3.1 Встановлення Unreal Engine 5 та створення проекту

Процес розробки почався зі встановлення необхідного програмного забезпечення. Завантаження та встановлення Epic Games Launcher стало першим кроком. Це програмне забезпечення було отримано з офіційного сайту Epic Games. Після встановлення лаунчера було створено обліковий запис для доступу до усіх функцій та ресурсів Unreal Engine. Ввійшовши в Epic Games Launcher, ми перейшли до розділу «Unreal Engine», де в бібліотеці було вибрано та встановлено останню версію Unreal Engine 5. Після цього було вибрано розділ “магазин”, де було завантажено безплатний плагін PaperZD для більш зручного способу розробки.

Коли підготовка була завершена, ми можемо перейти до створення проекту. У версії Unreal Engine 4 був варіант створення проекту підготовленного до розробки 2D ігор. На жаль, у оновленій версії рушія цей шаблон було видалено, тому створюємо пустий шаблон Blank.

При створенні обираємо основу створення блупрінт, через це можна не турбуватись, бо можемо додати C++ код у будь-яку мить. Далі обираємо платформу для якої будемо розробляти гру, в нашому випадку Desktop. Проект називаємо відповідно до його тематики та зберегаємо в обраній директорії (див. рис. 3.1).

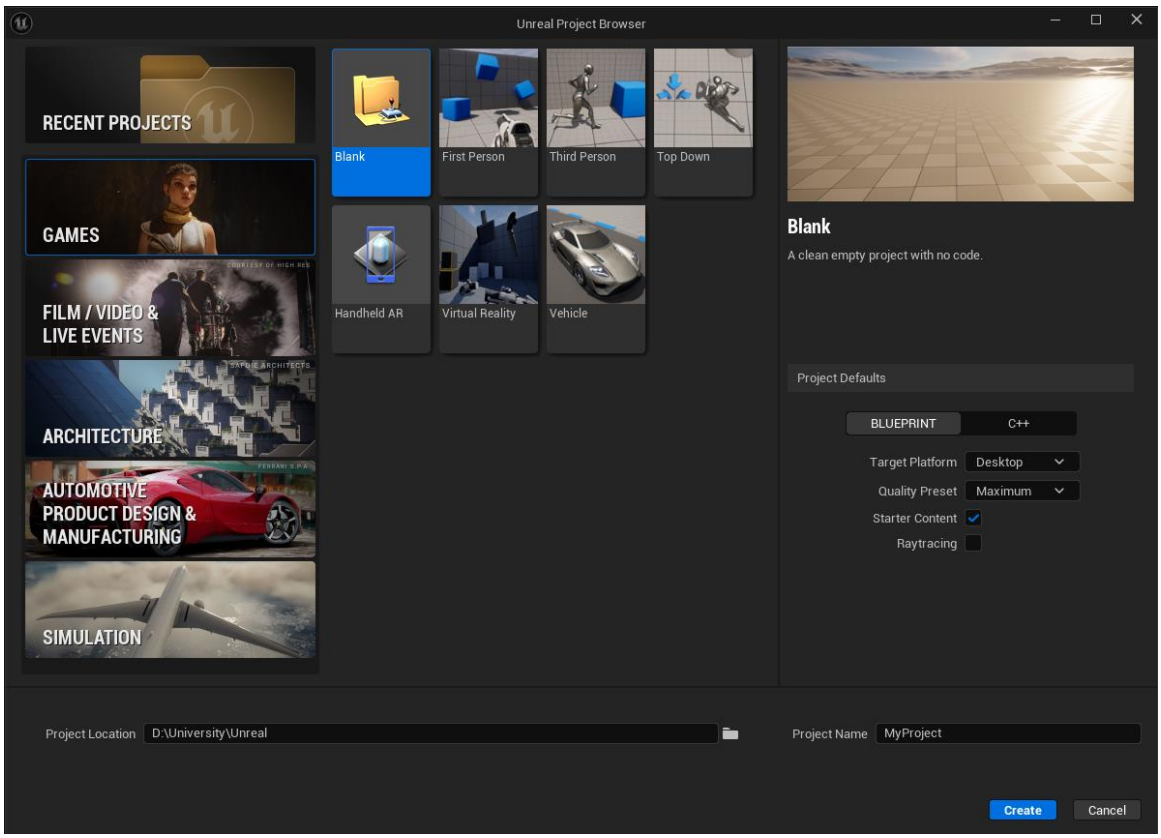

Рисунок 3.1 – Створення проекту

3.2 Створення основних елементів гри

3.2.1 Створення ігрового персонажа

Основною складовою будь-якого 2D платформера є ігровий персонаж. Для його створення було використано Blueprint Class на основі Paper Character, названий «BP_Player». Використання класу Paper Character дозволило швидко налаштувати базові функції та взаємодії персонажа з оточенням, забезпечивши основу для подальшого налаштування та розширення функціональності.

Використання Blueprint Class на основі Paper Character.

Paper Character — це спеціалізований клас, створений для роботи з 2D персонажами у Unreal Engine 5. Він має вбудовані компоненти та функції, які спрощують розробку 2D ігор.

Цей клас містить всі необхідні базові компоненти для 2D персонажа, такі як компоненти для відображення спрайтів і обробки колізій (див. рис. 3.2).

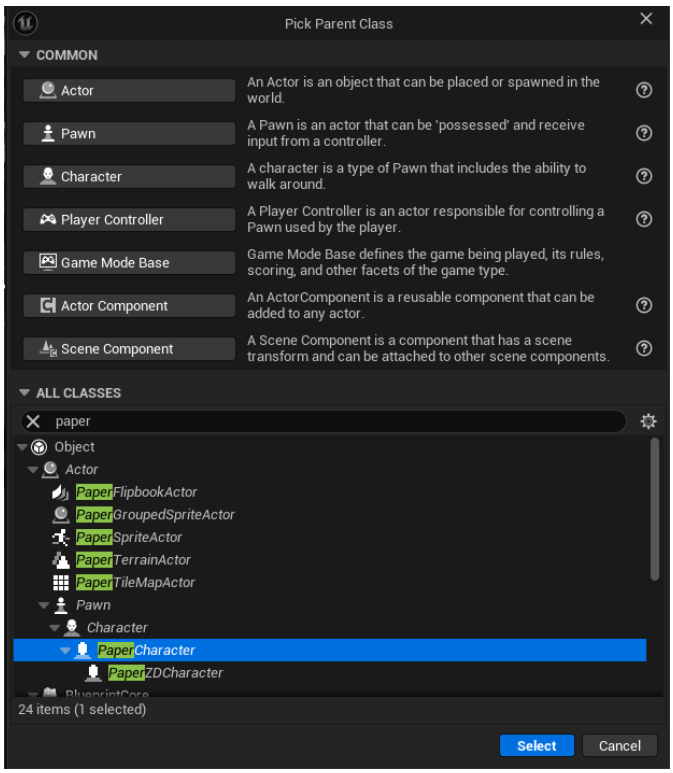

Рисунок 3.2 – Створення блупринту персонажа

Для створення персонажу було використано безкоштовні ассети з інтернету. Відібравши та завантаживши найкращі моделі, які б підійшли за стилістикою до ідеї гри, імпортуємо їх до проекту (див. рис. 3.3).

Рисунок 3.3 – Імпортовані текстури персонажа

Наступним кроком у роботі буде розбиття цих текстур на окремі спрайти, які у подальшому будуть використані для створення фліпбуків. Для цього використовуємо на них функцію Extract Sprites та обираємо автоматичне розбиття на спрайти (див. рис. 3.4).

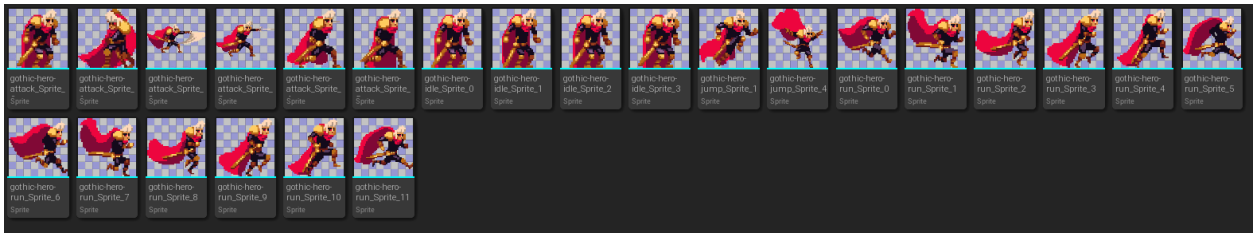

Рисунок 3.4 – Готові спрайти персонажа

Після цього об’єднуємо спрайти у фліпбуки з готовими анімаціями, використовуючи функцію. Після цього ми маємо готовий набір анімацій який можемо використовувати у подальшій роботі. Також треба налаштувати їм швидкість з якою буде відбуватись анімація, у нашому випадку нам вистачить 6-8 кадрів у секунду (див. рис. 3.5).

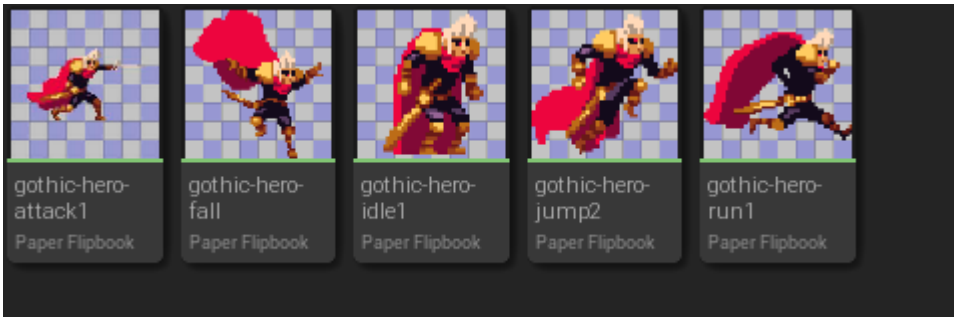

Рисунок 3.5 – Готові анімації персонажа

Налаштування базового вигляду та функцій персонажа відбувається при роботі з блупринтом BP_Player. Відкриваємо блупринт та додаємо фліпбук який буде використовуватись персонажом у стані спокою. У детальних налаштуваннях обираємо вкладку Sprite та додаємо потрібну анімацію. Також додаємо до моделі компонент Box Collision та даємо йому назву BoxHit. Це буде далі використано для функції отримування шкоди від ворогів.

Налаштовуємо BoxHit та капсулу персонажа під його розмір. Реалізація представлена у додатку А.

Також одразу додаємо камеру та налаштовуємо її. Для цього у блупринті BP_Player створюємо компонент SpringArm який буде тримати камеру завжди навпроти персонажа. Потім додаємо об’єкт камеру , та прив’язуємо її до SpringArm (див. рис. 3.6).

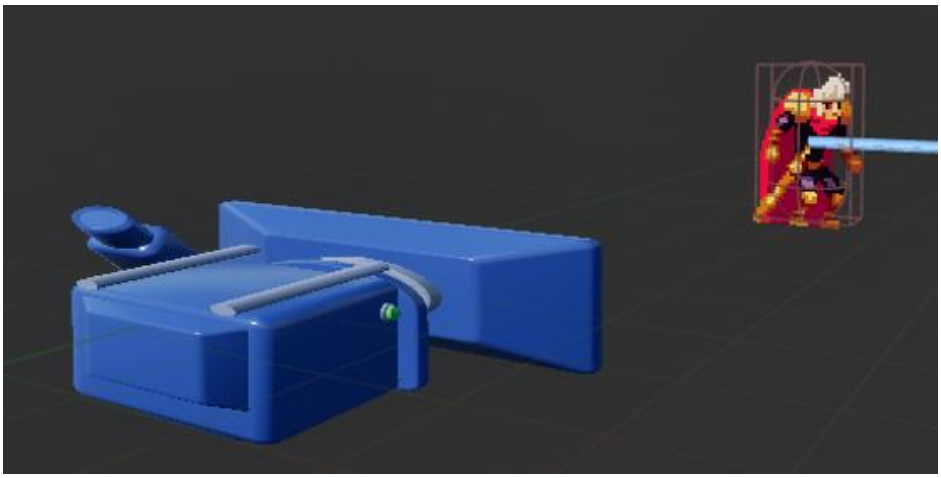

Рисунок 3.6 – Підготована модель з колізією та камерою

3.2.2 Створення початкової локації

Для подальшого налаштування персонажа , треба підготувати локацію на якій вже можна буде тестувати можливості руху. Набір текстур для гри було взято у відкритому доступі в інтернеті. Після завантаження набір було імпортовано в проект. Наступним кроком є трансформація звичайних текстур у тайлсети за допомогою функції Create Tile Set (див. рис. 3.7).

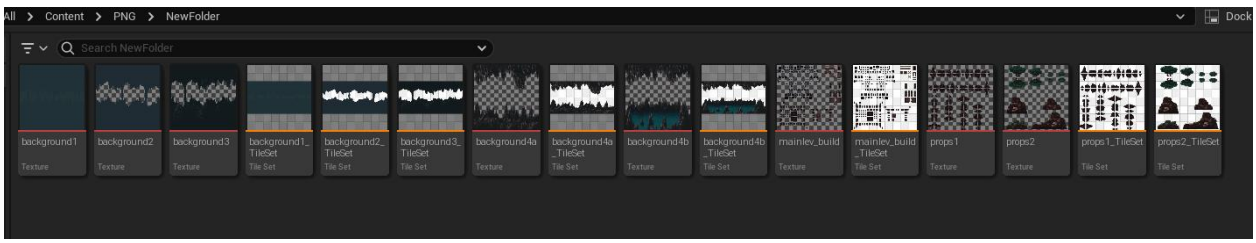

Рисунок 3.7 – Підготовлені тайл сети

Tile Set — це набір плиток, які використовуються для побудови рівнів у 2D-іграх. У Unreal Engine, Tile Set є невід'ємною частиною процесу створення 2D-ігор, особливо платформерів. Використання Tile Set дозволяє легко створювати складні ігрові світи з повторюваних елементів, забезпечуючи ефективність та зручність у розробці.

Наступним етапом є налаштування тайл сетів , шляхом додавання колізій на потрібних ділянках. Для цього відкриваємо редактор тайл сетів, виділяємо потрібні ділянки та використовуємо на них функцію Add Box (див. рис. 3.8).

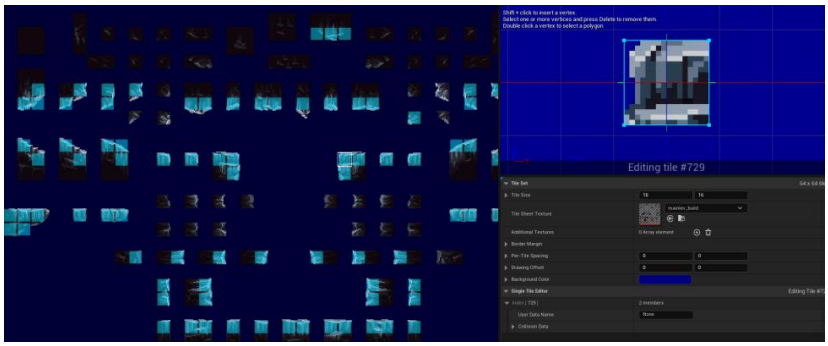

Рисунок 3.8 – Додавання колізій на потрібних текстурах

Після цього відбувається створення Tile Map та розміщення на ній текстур. Відкривши редактор Tile Map треба вибрати потрібний Tile Set та виділивши потрібні текстури , розмістити їх на Tile Map.

Провівши аналіз побудови рівней у 2D іграх, було створено власний рівень (див. рис. 3.9).

Рисунок 3.9 – Готова Tile Map

3.2.3 Реалізація руху головного героя

Для забезпечення руху головного героя у нашому 2D платформері було налаштовано систему введення (Input). Це включало створення Input Actions для обробки користувацького введення з клавіатури.

У меню налаштувань проекту було додано нові Input Actions. Зокрема, ми створили два окремих інпути: один для горизонтального руху в сторони, інший для стрибка. Це дозволило більш гнучко обробляти дії гравця і забезпечити зручне керування персонажем.

У секції "Action Mappings" ми задали відповідні клавіші для кожного Input Action. Для руху ліворуч і праворуч використовувалися клавіші A і D, а для стрибка — клавіша пробілу, або W. Це стандартне налаштування, яке є інтуїтивно зрозумілим для гравців і забезпечує комфортне керування (див. рис. 3.10).

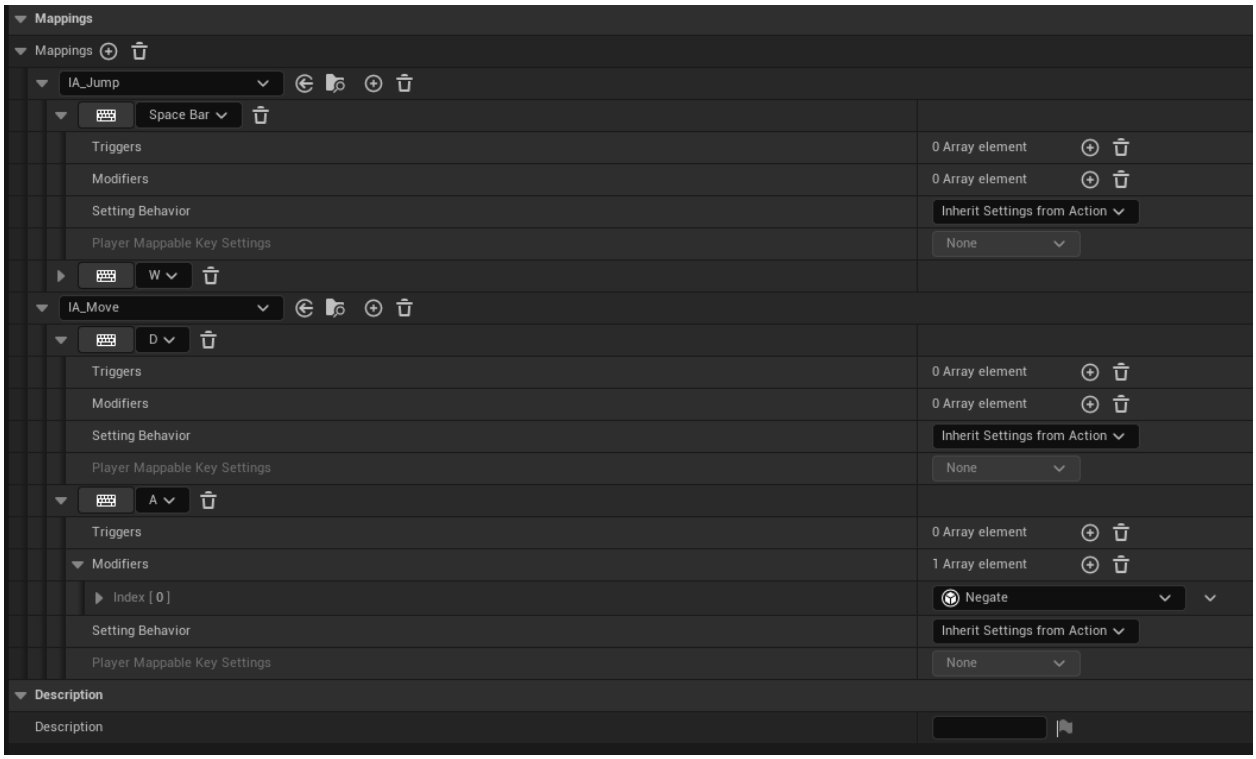

Рисунок 3.10 – Прив’язка руху до кнопок

Після налаштування системи вводу переходимо до створення логіки руху персонажа у Blueprint класі.

В Blueprint персонажа (названий «BP_Player») було додано Event Tick, який викликається кожен кадр гри. Створено функцію "Update Animation", яка постійно оновлює анімацію персонажа, залежно від його стану та дій гравця. Це дозволило забезпечити плавний перехід між анімаціями різних дій, таких як ходьба, стрибок або очікування (див. рис. 3.11).

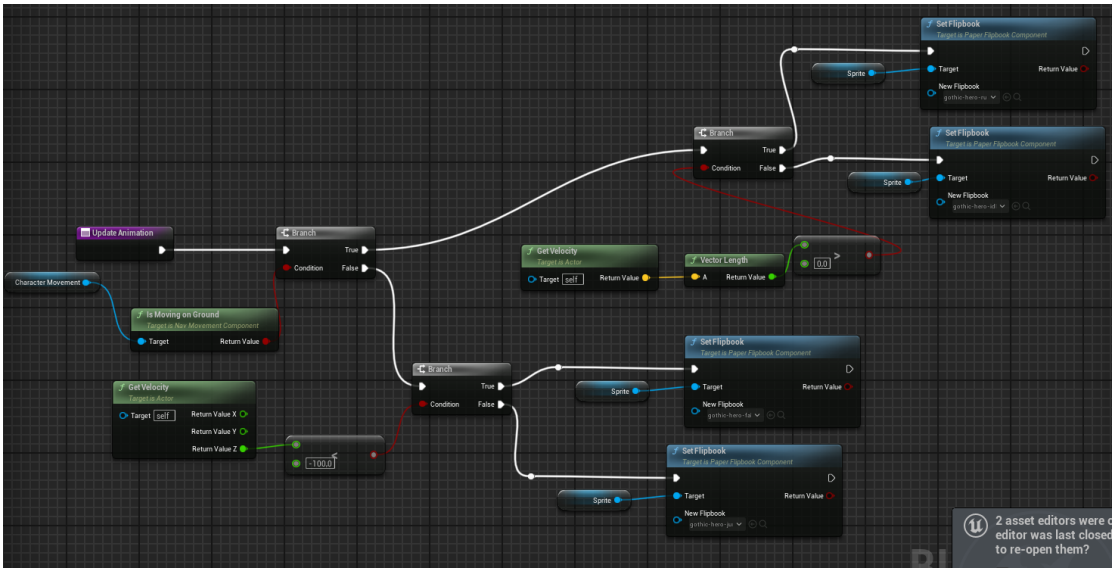

Рисунок 3.11 – Логіка функції Update Animation

Створення логіки для:

– переміщення по осі X;
– початок та завершення стрибка;
– розворот моделі персонажа в залежності від напрямку руху.

Було додано обробку Input Actions для руху ліворуч і праворуч. За допомогою функції "Add Movement Input" персонаж отримував можливість рухатися по осі X, реагуючи на натискання відповідних клавіш.

Для створення логіки стрибка було використано Enhanced Input Action для обробки користувацького введення, а також функції Jump і Stop Jumping, які забезпечують основну функціональність стрибка. Логіка оновлення

анімації стрибка забезпечила плавний перехід між стрибками та іншими діями персонажа.

Для коректного відображення руху було додано логіку для розвороту персонажа залежно від напрямку руху. Зміна масштабування (scale) персонажа по осі X дозволила забезпечити правильний напрямок його погляду під час руху ліворуч або праворуч.

Логіка повороту персонажа реалізована у Blueprint класі за допомогою використання Enhanced Input Action для обробки руху, а також функцій для зміни напрямку контролера (Controller) персонажа. Реалізація цих функцій представлена у додатках Б, В, Г.

Після налаштування руху персонажа ми розмістили його на ігровій мапі. Це було здійснено за допомогою функції PlayerStart, яка забезпечила появу персонажа у визначеній точці на мапі при запуску гри. Місцезнаходження PlayerStart було налаштовано відповідно до дизайну рівня, що забезпечило правильне розташування персонажа на початку гри.

Після завершення налаштувань ми запустили проект для перевірки роботи системи вводу та логіки руху персонажа. Під час тестування було перевірено, чи коректно реагує персонаж на введення з клавіатури для руху в сторони та стрибку. Гравець міг натискати клавіші для руху (A та D), і персонаж йшов та змінював напрямок погляду залежно від обраного напрямку руху. Гравець міг натискати пробіл для стрибка, і персонаж виконував стрибок з відповідною анімацією. Перевірка включала також завершення стрибка: коли персонаж досягував максимальної висоти стрибку, він плавно приземлявся, що забезпечувалося викликом функції Stop Jumping (див. рис. 3.12 – 3.16).

Рисунок 3.12 – Персонаж дивиться вправо

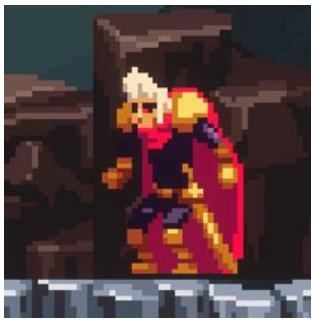

Рисунок 3.13 – Персонаж дивиться вліво

Рисунок 3.14 – Стрибок персонажа

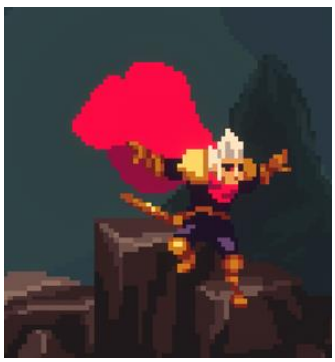

Рисунок 3.15 – Падіння персонажа

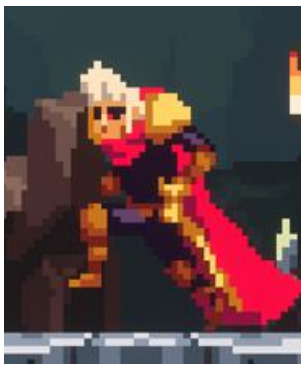

Рисунок 3.16 – Рух персонажа

Після проведених налаштувань та перевірок ми впевнилися, що основна механіка руху персонажа працює стабільно та коректно. Це забезпечило фундамент для подальшої розробки гри, включаючи додавання нових механік, елементів рівня та покращення взаємодії з оточенням.

3.3 Створення прототипу ворога

Наступним кроком у розробці 2D платформера є створення ворогів, які будуть взаємодіяти з головним героєм та створювати виклики для гравця.

3.3.1 Створення Базового Фундаменту

Нового персонажа було вирішено зробити іншим способом. Для цього було вирішено використати плагін PaperZD, який забезпечує додаткові можливості для роботи з 2D анімаціями та спрощує процес інтеграції анімацій у гру. Плагін PaperZD був обраний завдяки його зручному інтерфейсу та потужним функціям, які значно спрощують роботу з анімаціями у 2D проектах.

Початок роботи відбувається так само :

– пошук та завантаження текстур;
– імпортування текстур до проекту;

– розробка спрайтів;

– створення фліпбуків.

Після цього підхід змінюється, створюється компонент Animation Source, куди підключаються потрібні нам анімації. Потім створюємо компонент AnimBP де ми задамо логіку змін анімацій (див. рис. 3.17 – 3.18).

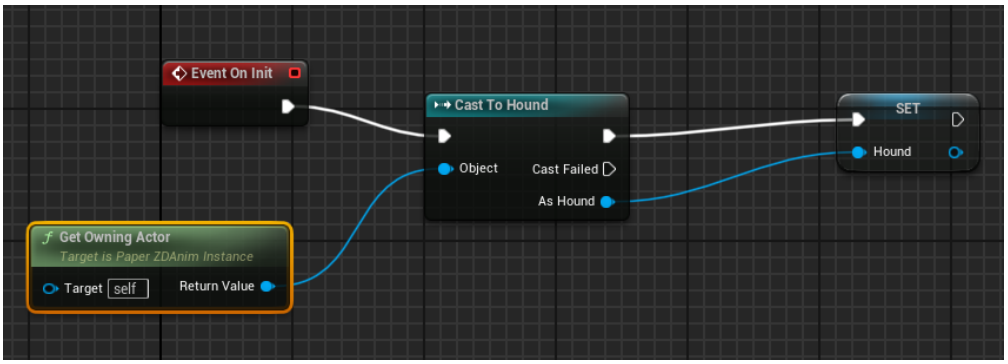

Рисунок 3.17 – Вміст Event Graph

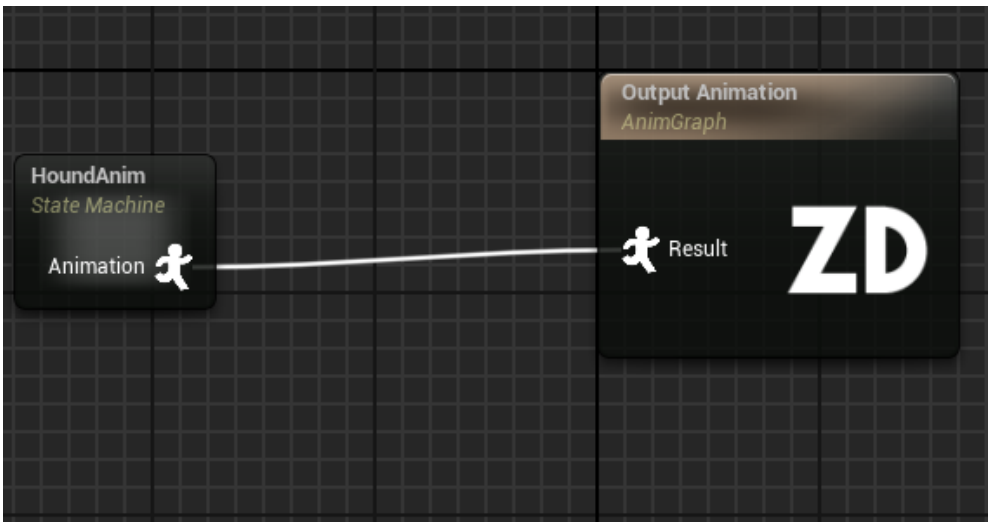

Рисунок 3.18 – Вміст Anim Graph

Відкрив State Machine ми прописуємо логіку зміни анімацій (див. рис. 3.19 – 3.21).

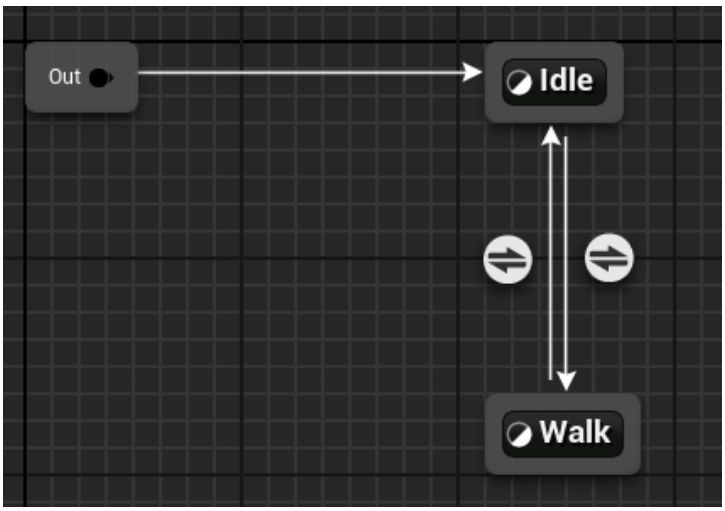

Рисунок 3.19 – Вміст State Machine

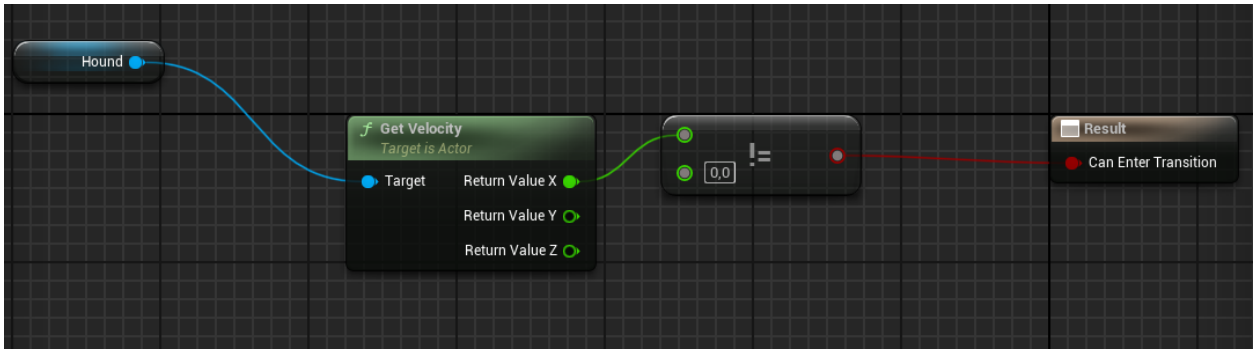

Рисунок 3.20 – Логіка переходу від Idle до Walk

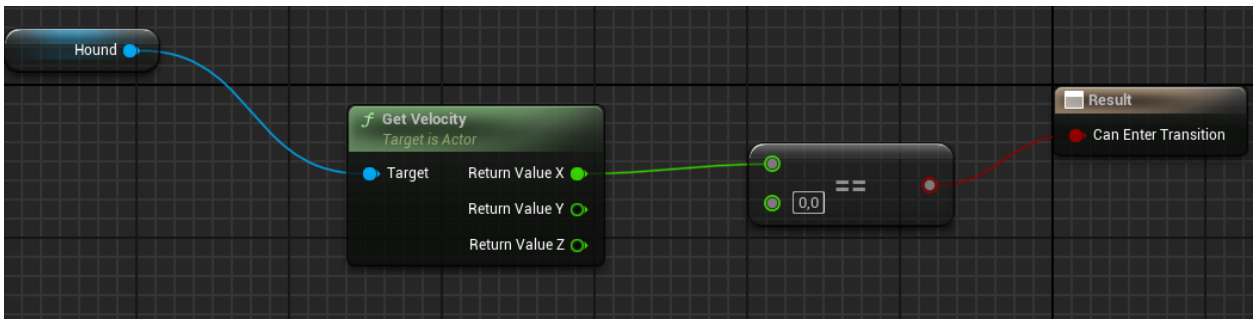

Рисунок 3.21 – Логіка переходу від Walk до Idle

Створюємо блупринт Hound , який і буде виконувати роль ворога. Повторюємо додавання та налаштування BoxHit та капсули персонажа по аналогії з головним героєм (див. рис. 3.22).

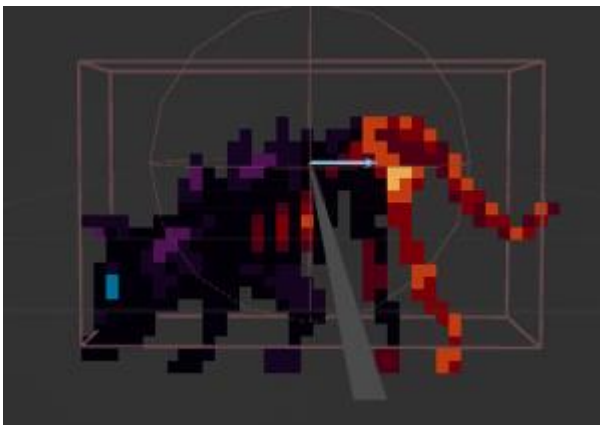

Рисунок 3.22 – Підготована модель ворога

Після цього треба прописати логіку його руху. В нашому випадку ворог буде патрулювати окрему ділянку мапи. Реалізація цих дій буде відбуватись через 2 створені перемінні S та S1.

Гонча має пройти цю відстань по осі X за допомогою функції Add Movement Input. Тому буде використано Event Tick, своєрідний нескінченний цикл, який повторює одні й ті самі події раз за разом.

Спочатку ворог отримує початкову позицію та значення змінної S, яке визначає відстань патрулювання. За допомогою функції Add Movement Input ворог переміщується на невелику відстань у кожному кадрі. Напрямок руху визначається поточним значенням змінної S. Змінна S1 постійно оновлюється, відображаючи відстань, яку ворог вже пройшов. Коли значення S1 зрівнюється з S, ворог досягає кінцевої точки свого маршруту. Це означає, що він повинен змінити напрямок. Коли ворог досягає кінцевої точки, значення змінної S1 скидається на 0, а значення змінної S змінюється на протилежне. Це забезпечує зміну напрямку руху ворога, і він починає рухатися у зворотному напрямку. Реалізація представлена у додатку Н.

Реалізував цю логіку у блупринті, розташовуємо персонажа на мапі для перевірки. Було проведено декілька перевірок, щоб упевнитися у правильності реалізації патрулювання:

– ворог коректно рухався між двома точками, змінюючи напрямок при досягненні кінцевих точок;

– значення змінних S та S1 оновлювались відповідно до очікувань;
– логіка зміни напрямку працювала без збоїв, забезпечуючи плавний рух ворога;
– анімації працюють корректно та плавно.

3.3.2 Реалізація Взаємодії Персонажів

На даному етапі розробки буде реалізовано механіку взаємодії між головним героєм та ворогом у 2D платформері. Основною метою було забезпечити, щоб при перетині хітбоксів цих персонажів герой отримував шкоду, а обидва персонажі відкидалися один від одного. Це додало до гри елемент боротьби та взаємодії з ворогами.

Коли хітбокси головного героя та ворога перетинаються, тригериться подія, яка завдає герою шкоди. Одночасно обидва персонажі відкидаються у протилежні сторони, що створює ефект удару та розділення після зіткнення.

Використовується система колізій, яка визначає, коли хітбокси персонажів перетинаються. Подія OnComponentBeginOverlap виявляє момент, коли хітбокси головного героя та ворога починають перетинатися.

Створюється функція для зменшення здоров'я героя при зіткненні з ворогом. Використовується змінна Health, яка відображає поточний рівень здоров'я героя. Реалізація представлена у додатках Д, М.

Використовується функція Launch Character для відкидання персонажів у протилежні сторони після зіткнення. Визначається напрямок та сила відкидання на основі поточної позиції героїв.

Також герою та гончій було додано рівень здоров’я, яке знижувалось після отримування шкоди. Після спускання здоров’я до нуля вони помирають, а герой з’являється на початку рівня. Реалізація представлена у додатку И.

Систему здоров’я було реалізовано створеними компонентами Health та IsDead. Компонент Health це і є здоров’я, а IsDead є перевіркою чи є воно у героя або ворога. Коли перемінна Health спускається до нуля, тригериться

перемінна IsDead.Коли здоров’я ворога дорівнює нулю, він помирає та зникає. Реалізація представлена у додатку К. Коли герой з’являється на початку рівня, його здоров’я відновлюється.

Після реалізації логіки у Blueprint взаємодія між героями була перевірена на практиці:

– перевірено, що подія OnOverlapBegin коректно визначає перетин хітбоксів персонажів;
– зменшення здоров'я героя при зіткненні з ворогом відбувалося коректно;
– відкидання героїв у протилежні сторони відбувалося відповідно до очікувань.

Наступним етапом було додавання можливості наносити удари головному герою. Вона була реалізована додаванням герою нової колізії з назвою Attack Collision. Вона знаходиться у вимкненому стані, доки гравець на натисне клавішу удару. Це стригерить анімацію удару та появу коллізії, яка при перетині з HitBox ворога нанесе йому шкоду. Реалізація представлена у додатках Е, Ж, К.

Реалізація взаємодії між головним героєм та ворогом у 2D платформері була успішно завершена. Використання колізій для визначення перетину хітбоксів, завдання шкоди персонажам та відкидання персонажів додало грі динамічності та взаємодії. Це значно покращило ігровий процес, зробивши його більш захоплюючим та викликовим для гравця (див. рис. 3.23).

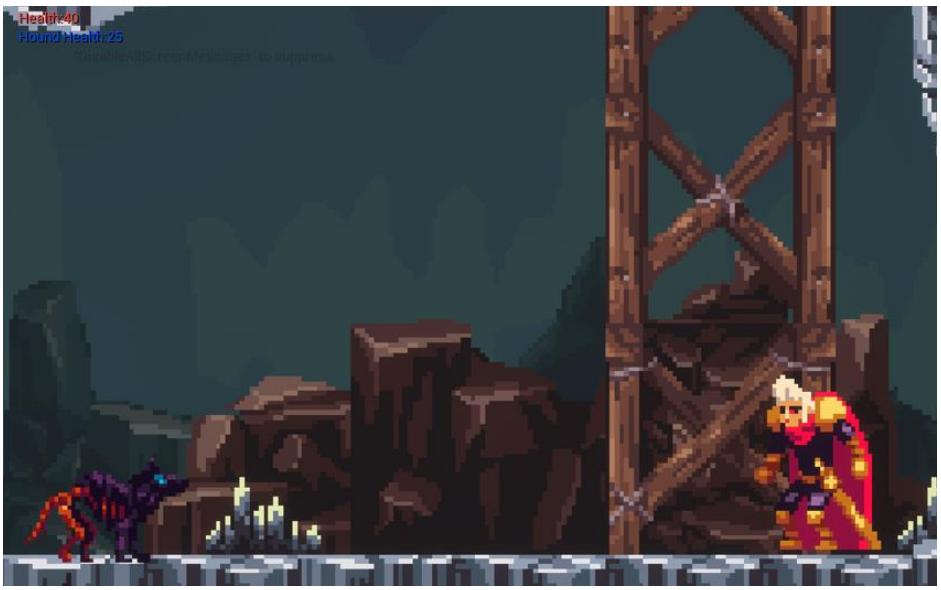

Рисунок 3.23 – Демонстрація отримування шкоди

3.3.3 Реалізація відображання поточного здоров’я

Важливим етапом розробки є також реалізація користувацького інтерфейсу. В нашому випадку це індикатор здоров’я, який буде показувати поточний стан героя.

Для цього створюється окремий Віджет блупринт. Відкрив його ми можемо побачити що від розділений на 2 вкладки: Designer та Graph.

Починаємо роботу з вкладки Designe, у ній ми спочатку додаємо компонент Canvas. Це поле яке показує собою як буде розташовано потрібний нам віджет на екрані. Підібрав потрібну нам роздільну здатність екрану, створюємо компонент Progress Bar, який ми адаптуємо під Health Bar. Змінюємо текстури на завантажені нами, правильно підставляючи пустий та повний індикатор здоров’я. Розташовуємо його у потрібному нам місці та переходимо до налаштування його у Graph (див. рис. 3.24 – 3.27).

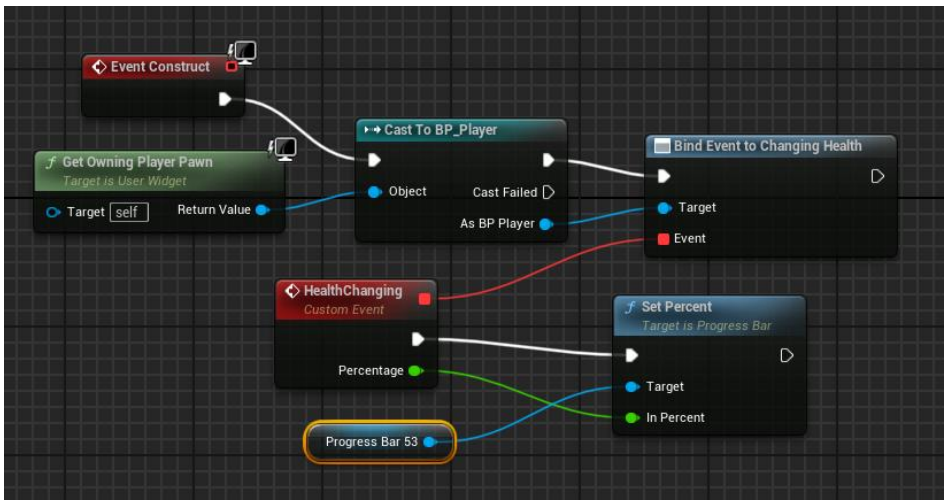

Рисунок 3.24 – Реалізація прив’язки здоров’я до HealthBar

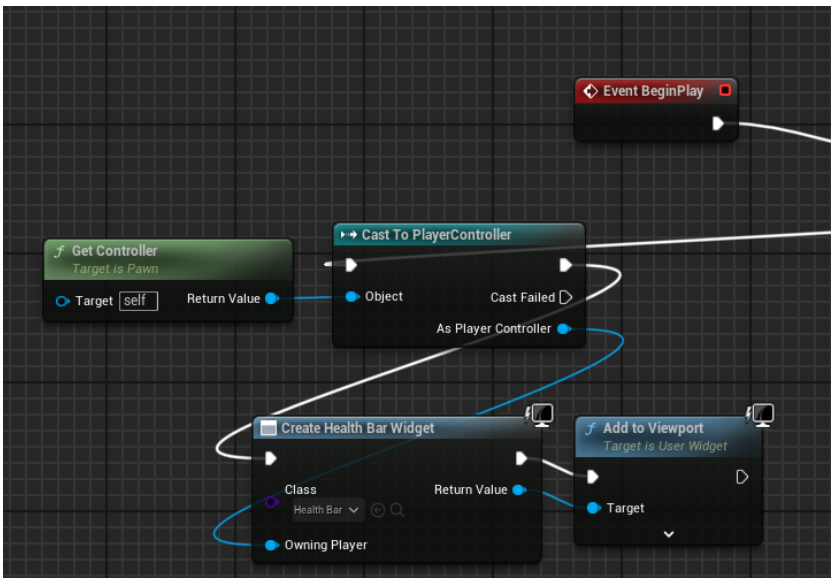

Рисунок 3.25 – Логіка прив’язки HealthBar до блупринта героя

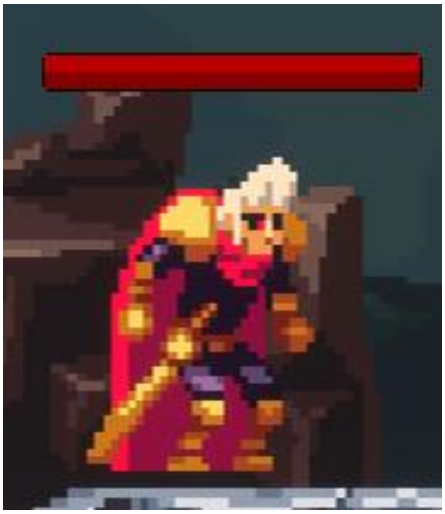

Рисунок 3.26 – Герой з повним здоров’ям

Рисунок 3.27 – Герой отримав шкоду

Провівши тестування ми переконались в тому що індикатор здоров’я працює так як було задумано:

– здоров’я відображається правильно;
– втрата здоров’я інформується одразу ж;
– після респавну персонажа індикатор оновлюється чинним образом.

ВИСНОВКИ

Розробка 2D платформера на рушії Unreal Engine 5 є результатом комплексної роботи, що включала кілька ключових етапів. Спершу було проведено аналіз існуючих популярних RPG ігор, що дозволило виділити основні елементи, які роблять ігри цього жанру успішними та привабливими для гравців. Цей аналіз став фундаментом для розробки власного проекту, оскільки він допоміг уникнути розповсюджених помилок та визначити найкращі практики у створенні захоплюючих ігрових світів.

У ході роботи над проектом було досліджено можливості та особливості Unreal Engine 5. Основною частиною проекту стала безпосередня розробка 2D платформера. Було створено ігрового персонажа з використанням Blueprint Class на основі Paper Character, налаштовано його анімацію та реалізовано логіку руху. Впровадження механіки стрибків та взаємодії з оточенням додало динаміки та реалістичності до геймплею.

Особлива увага була приділена розробці ворогів, для чого використовувався плагін PaperZD. Реалізовано їхню поведінку та патрулювання ділянок мапи, а також взаємодію з головним героєм. Логіка руху ворогів, яка передбачала їхнє патрулювання, була реалізована за допомогою Blueprint, що додало глибини та інтерактивності до гри.

На завершальному етапі була впроваджена механіка взаємодії між головним героєм та ворогами. При зіткненні з ворогом герой отримував шкоду та відкидався у протилежний бік, але й сам герой міг нанести атаку, що підвищувало реалістичність ігрового процесу та додавало йому динаміки. Усі аспекти гри були ретельно протестовані та налаштовані для забезпечення стабільної роботи та якісного геймплею.

Варто зазначити що це не готова на сто відсотків гра, а лише реалізація декотрих можливостей цього рушія. У майбутньому буде реалізовано багато нових функцій, таких як переходи на нові рівні,

різноманітність ворогів та їх штучного інтелекту, нові взаємодії з оточуючим середовищем та багато іншого.

Проект підтвердив доцільність використання Unreal Engine 5 для розробки 2D ігор, продемонструвавши його широкі можливості та ефективність. Використання сучасних технологій та інструментів дозволило реалізувати всі поставлені завдання та досягти високого рівня якості проекту. Набуті знання та досвід стануть міцною основою для подальшої роботи у сфері розробки ігор та створення нових, ще більш складних та цікавих проектів.

Успішна реалізація всіх етапів проекту засвідчила, що навіть складні задачі можна вирішити за допомогою правильного підходу та наполегливої праці. Цей проект став значним кроком вперед у розвитку навичок розробки ігор та відкрив нові перспективи для створення якісних та захоплюючих ігрових продуктів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Unreal Engine 5.3 Documentation | Unreal Engine 5.3 Documentation | Epic Developer Community (epicgames.com). URL : https://dev.epicgames.com/documentation/ru-ru/unreal-engine/unreal-engine-5-3-documentation?application_version=5.3 (дата звернення : 12.03.2024).

2. Overview Of Blueprints Visual Scripting In Unreal Engine | Unreal Engine 5.4 Documentation | Epic Developer Community (epicgames.com). URL : https://dev.epicgames.com/documentation/en-us/unreal-engine/overview-of-blueprints-visual-scripting-in-unreal-engine (дата звернення : 28.04.2024).

3. Paper2 D | Unreal Engine 5.4 Documentation | Epic Developer Community (epicgames.com). URL : https://dev.epicgames.com/documentatio/enus/unrealengine/API/Plugins/Paper2Dapplication_version=5.0 (дата звернення : 11.05.2024).

4. Official Unreal Engine YouTube channel. URL : https://www.youtube.com/user/UnrealDevelopmentKit (дата звернення : 15.05.2024).

ДОДАТОК А

Блупринт Update Animations героя

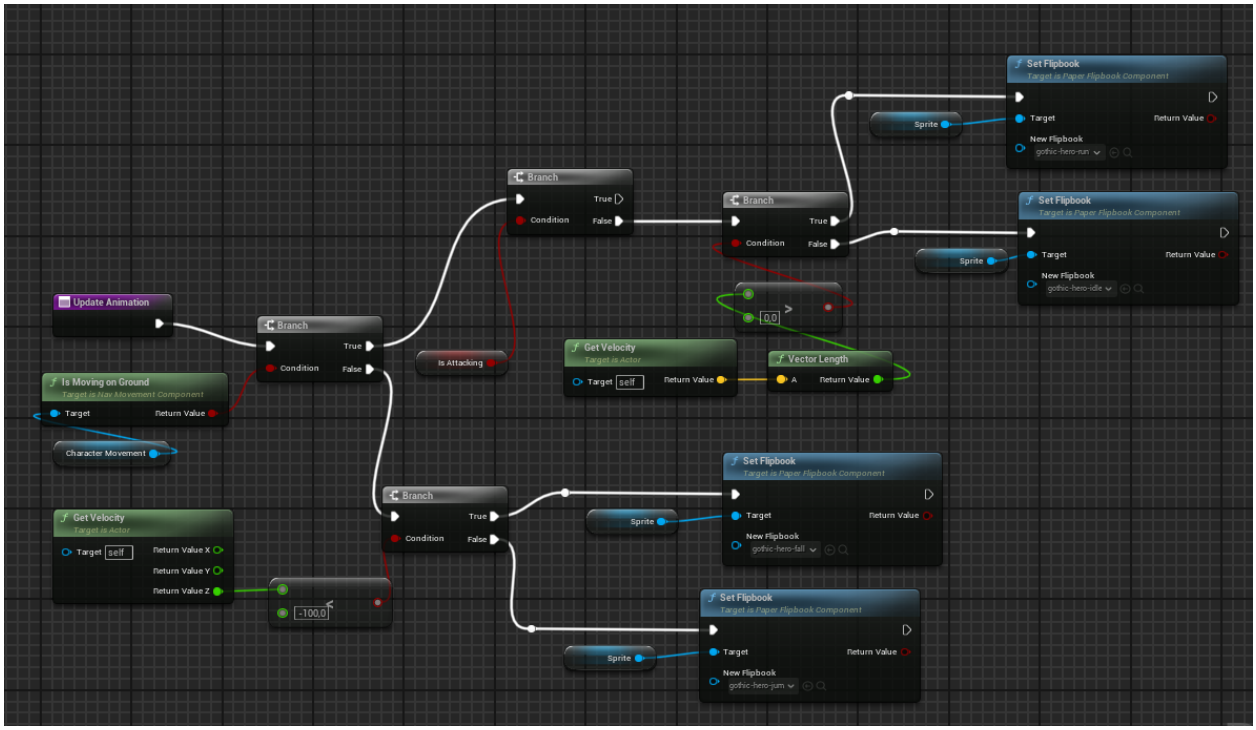

Рисунок А.1

ДОДАТОК Б

Список компонентів, що залежать від Event BeginPlay у блупринті героя

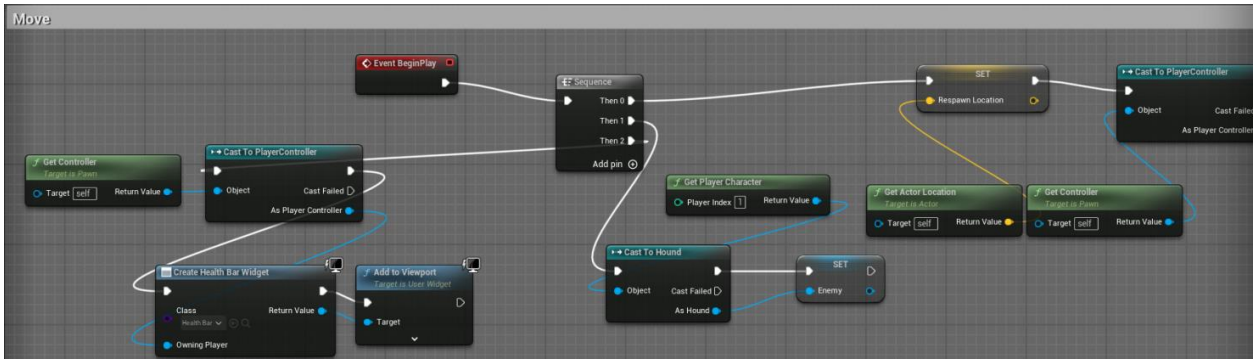

Рисунок Б.1

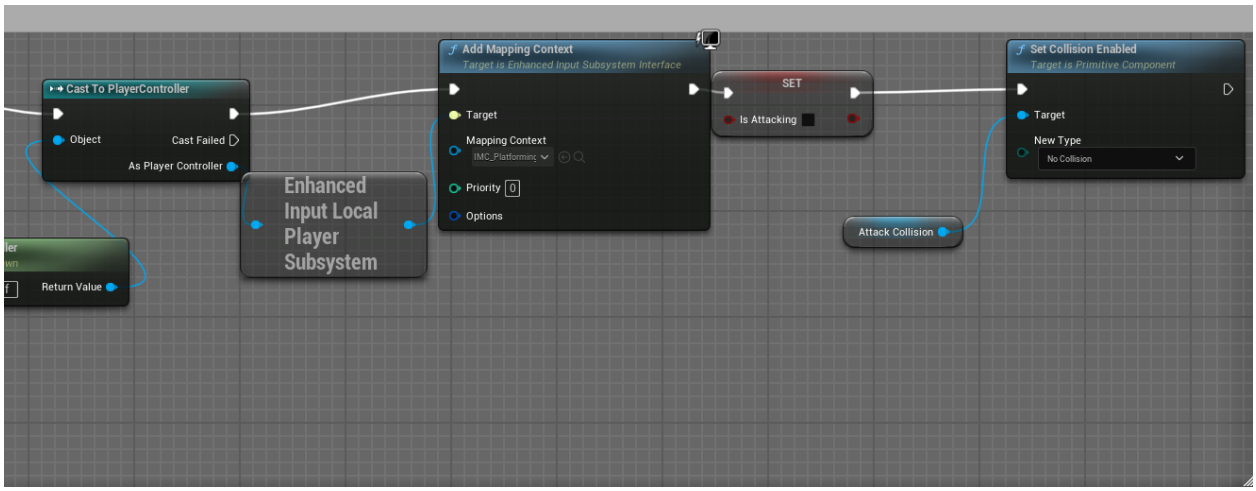

Рисунок Б.2

ДОДАТОК В

Блупринт стрибку героя

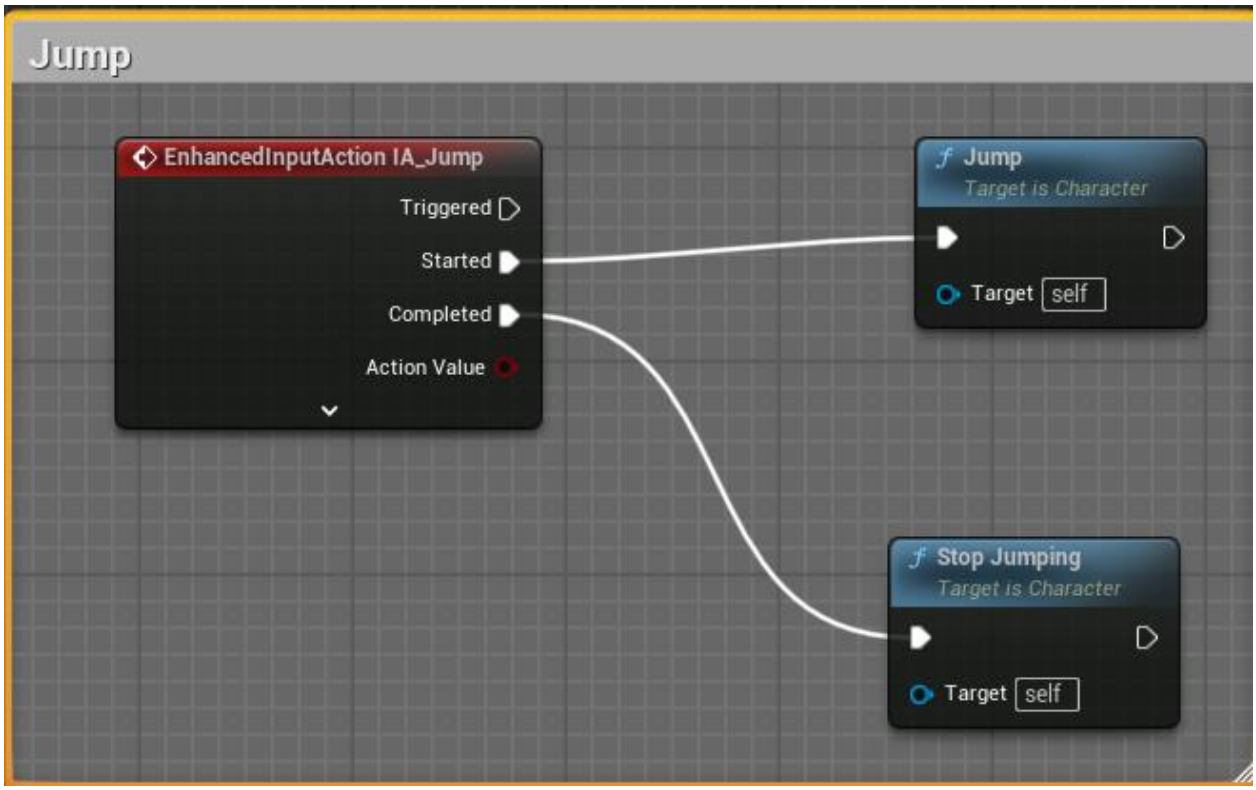

Рисунок В.1

ДОДАТОК Г

Блупринт повороту героя

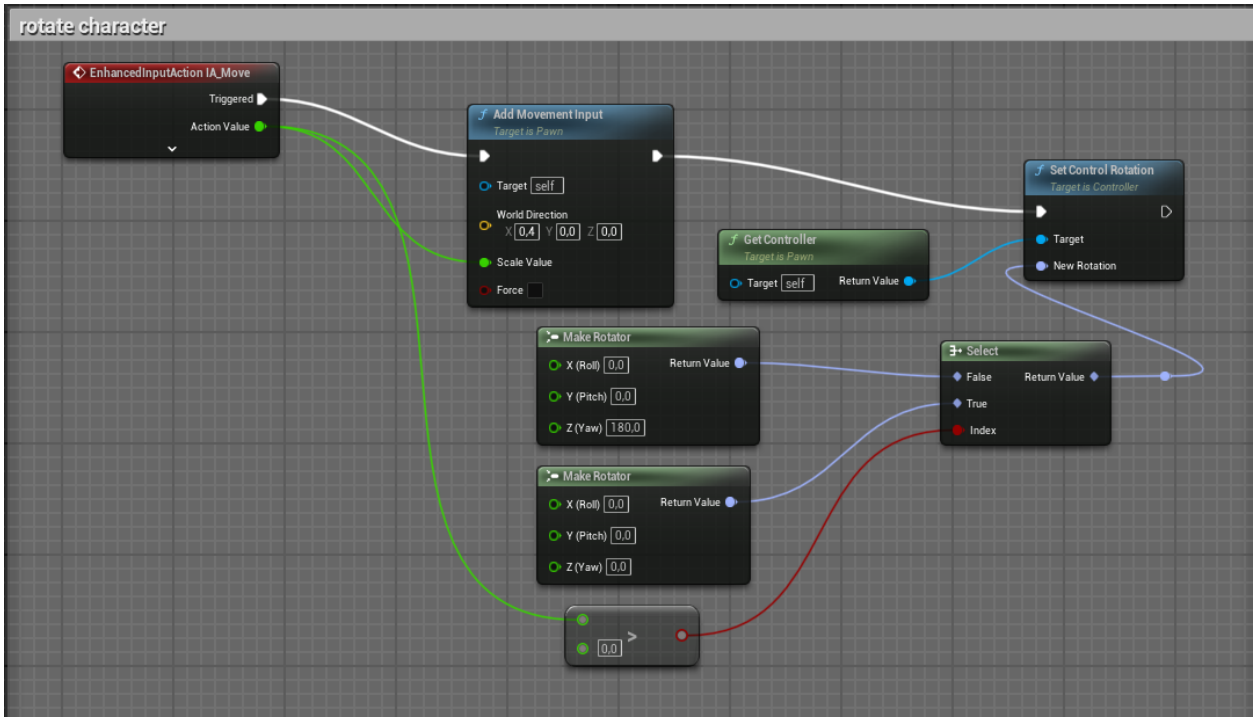

Рисунок Г.1

ДОДАТОК Д

Блупринт отримання шкоди героєм

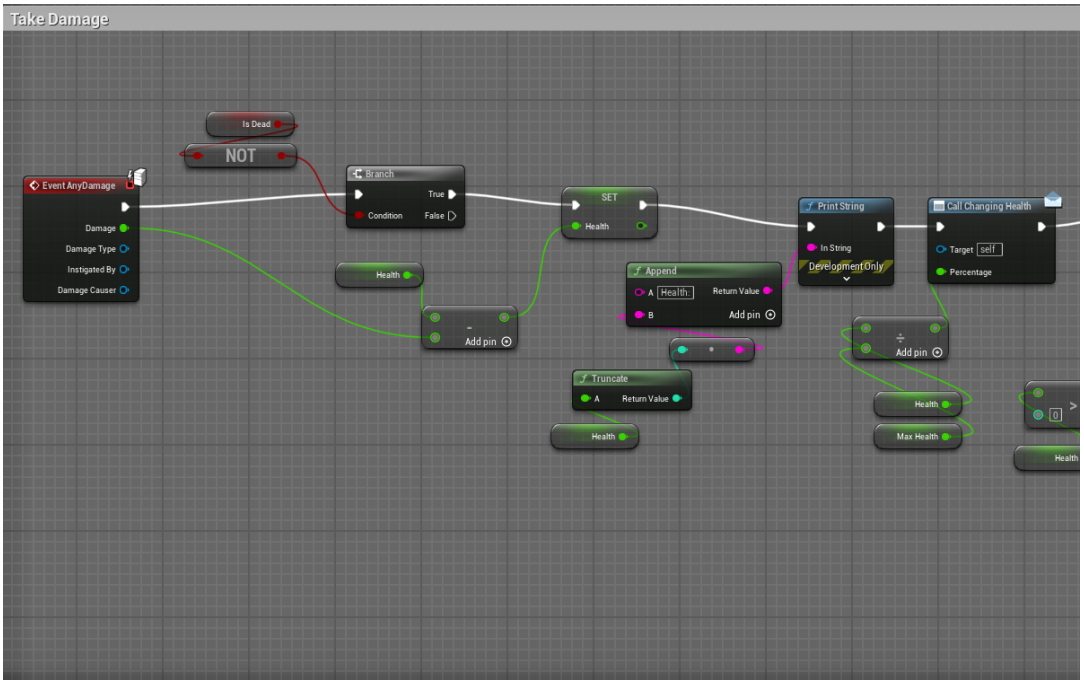

Рисунок Д.1

ДОДАТОК Е

Блупринт реалізації атаки героя

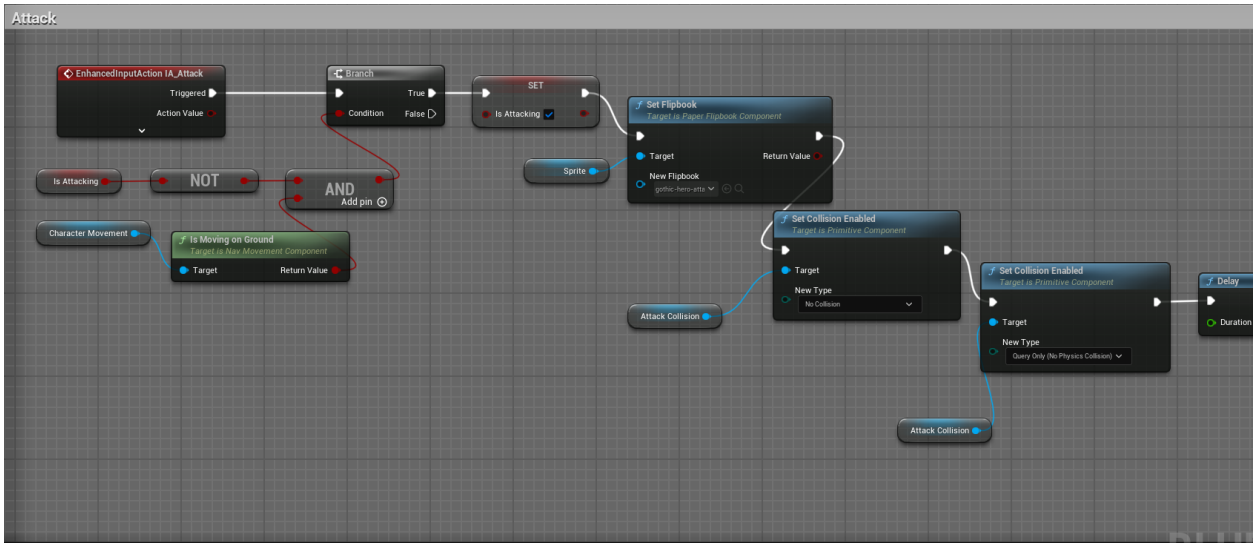

Рисунок Е.1

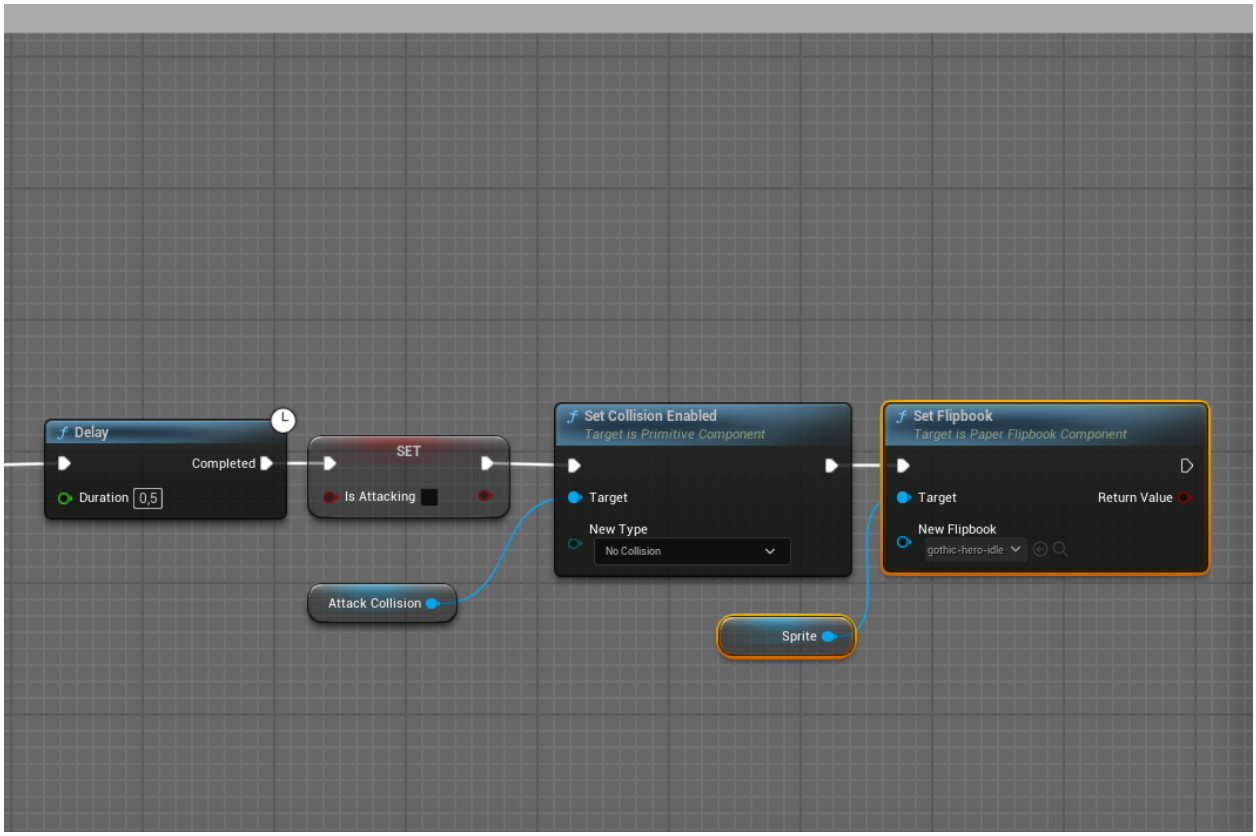

Рисунок Е.2

ДОДАТОК Ж

Блупринт нанесення шкоди героєм

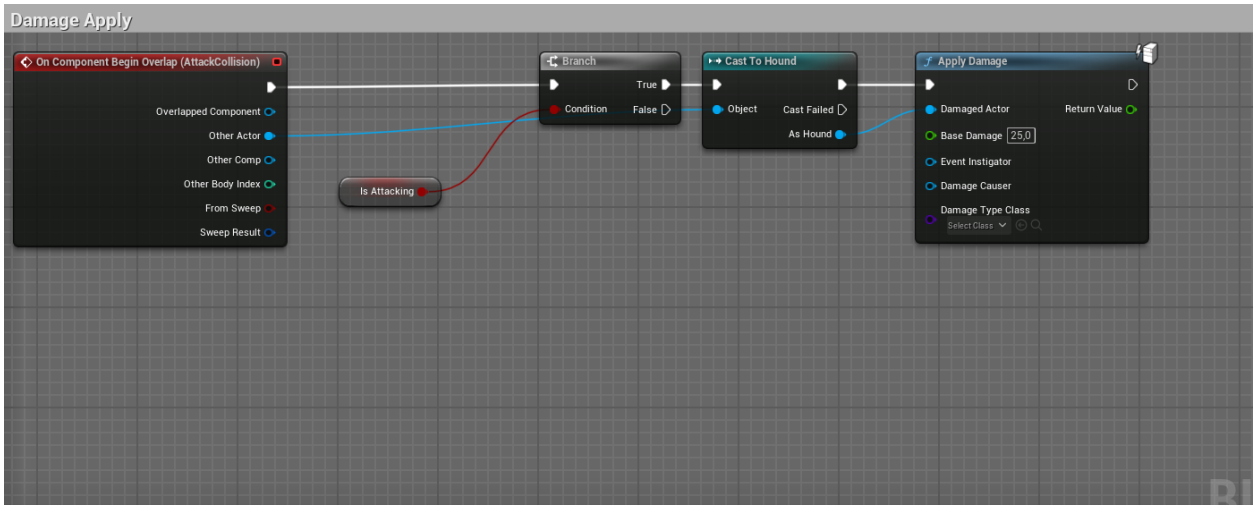

Рисунок Ж.1

ДОДАТОК И

Блупринт респавну героя

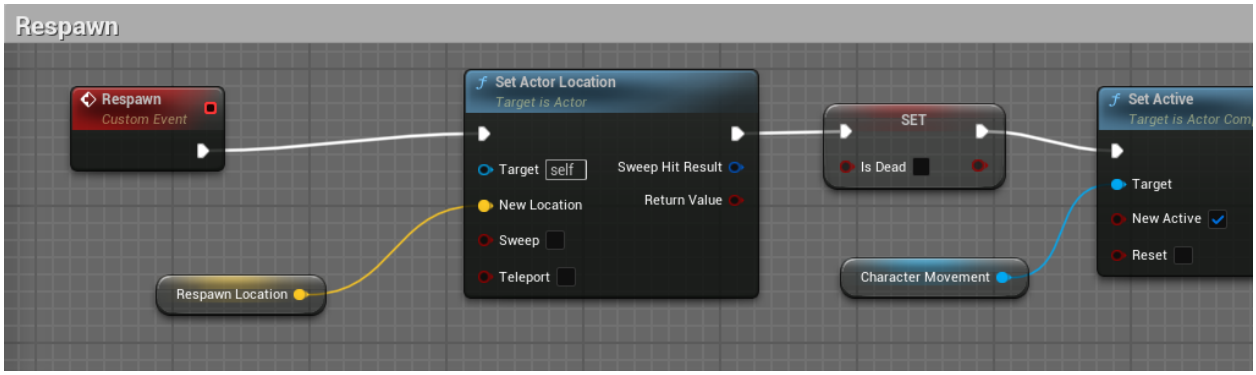

Рисунок И.1

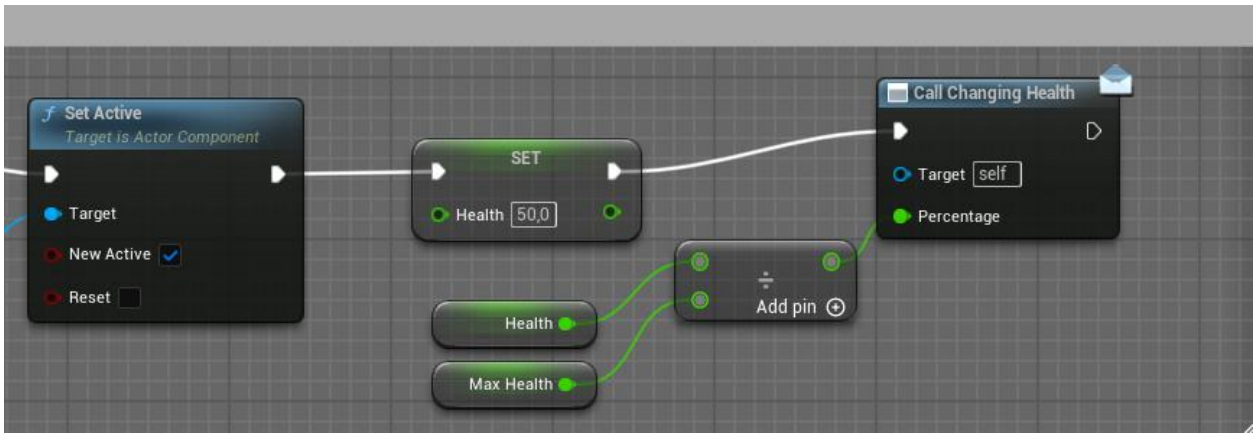

Рисунок И.2

ДОДАТОК К

Блупринт отримання шкоди ворогом

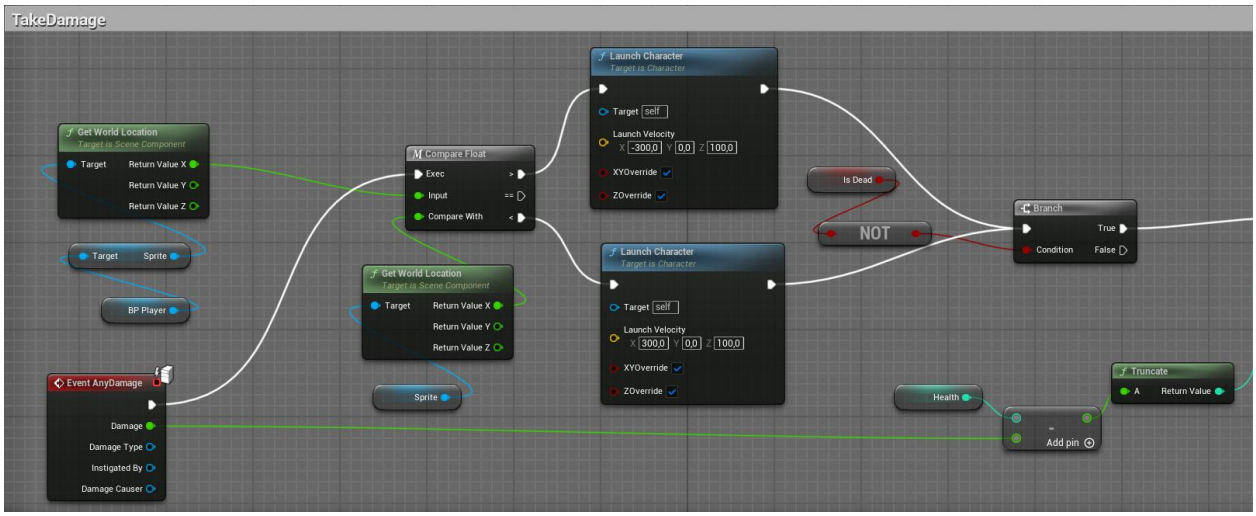

Рисунок К.1

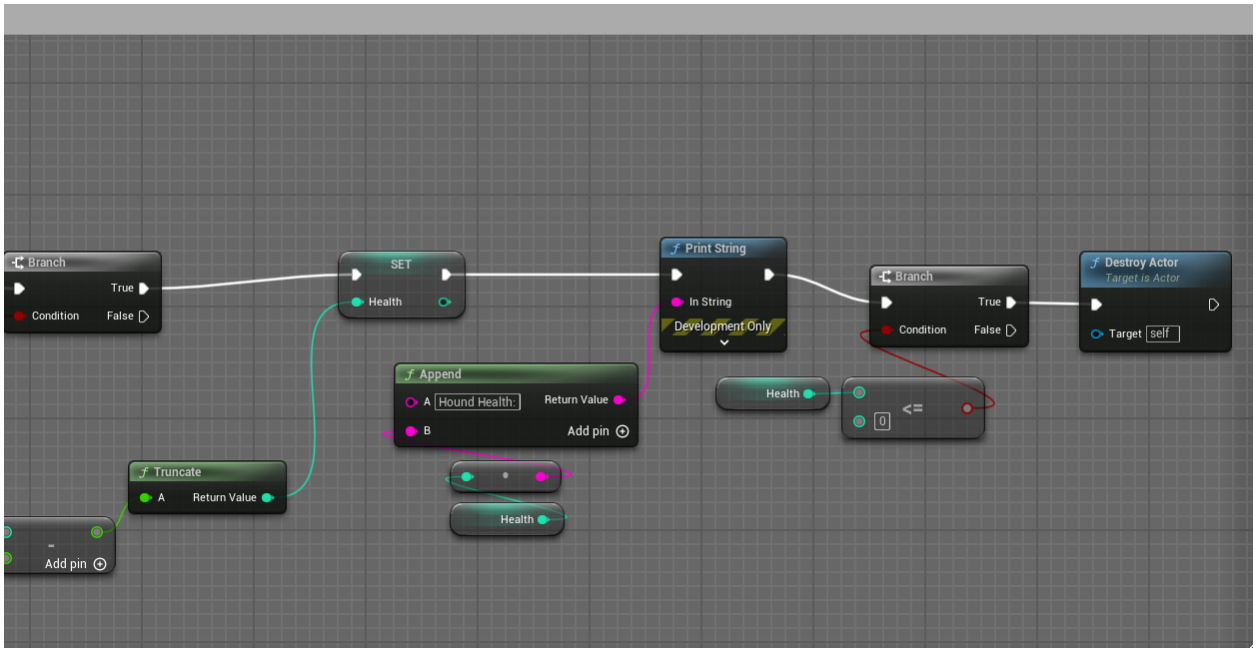

Рисунок К.2

ДОДАТОК Л

Допоміжні функції у блупринті ворога

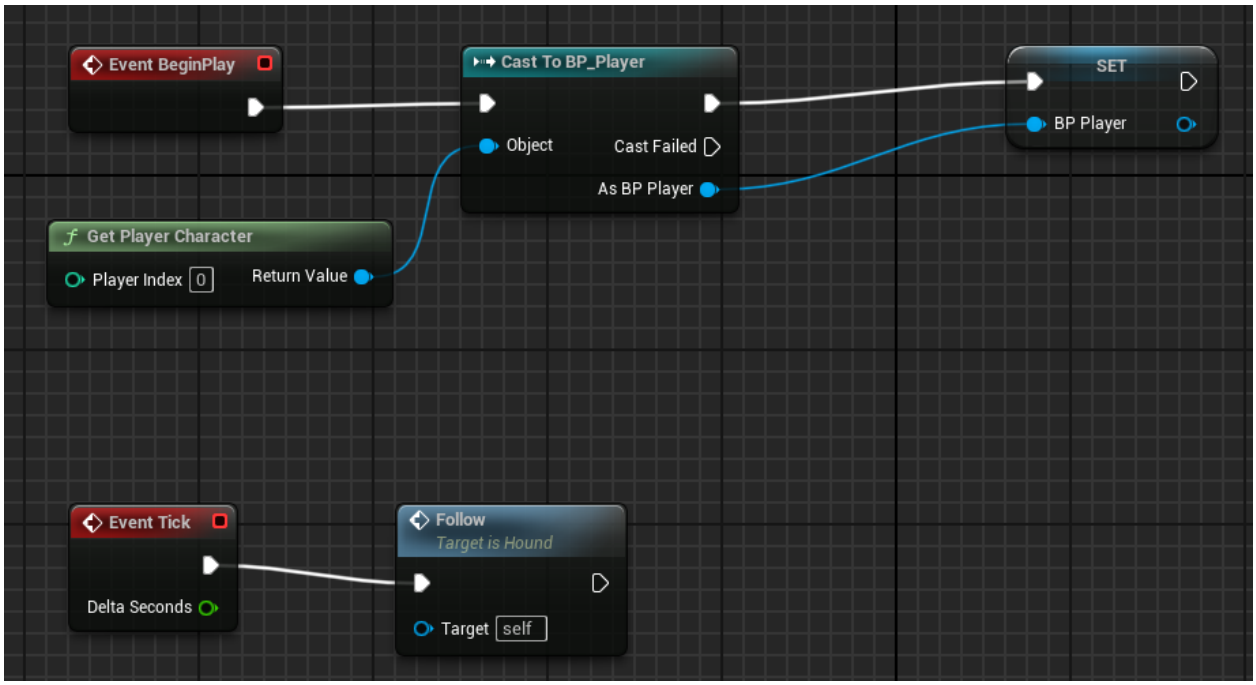

Рисунок Л.1

ДОДАТОК М

Блупринт нанесення шкоди ворогом

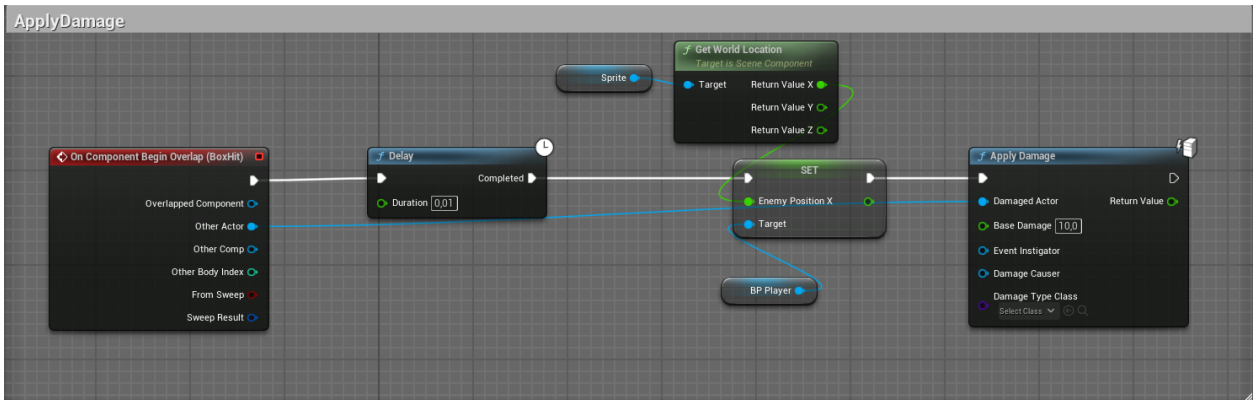

Рисунок М.1

ДОДАТОК Н

Блупринт патрулювання ворога

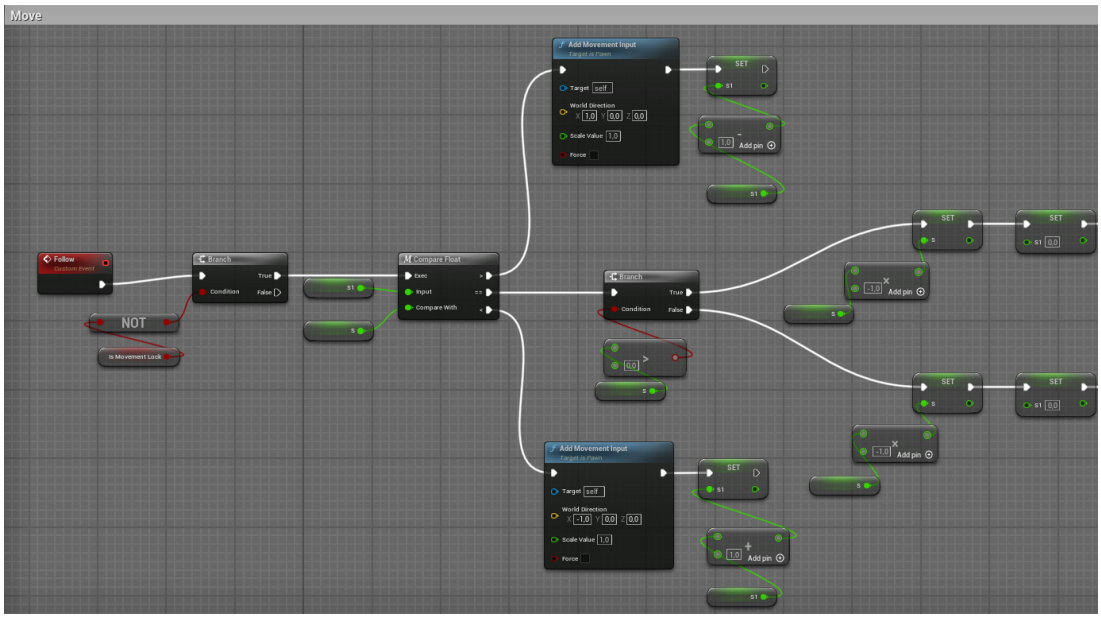

Рисунок Н.1

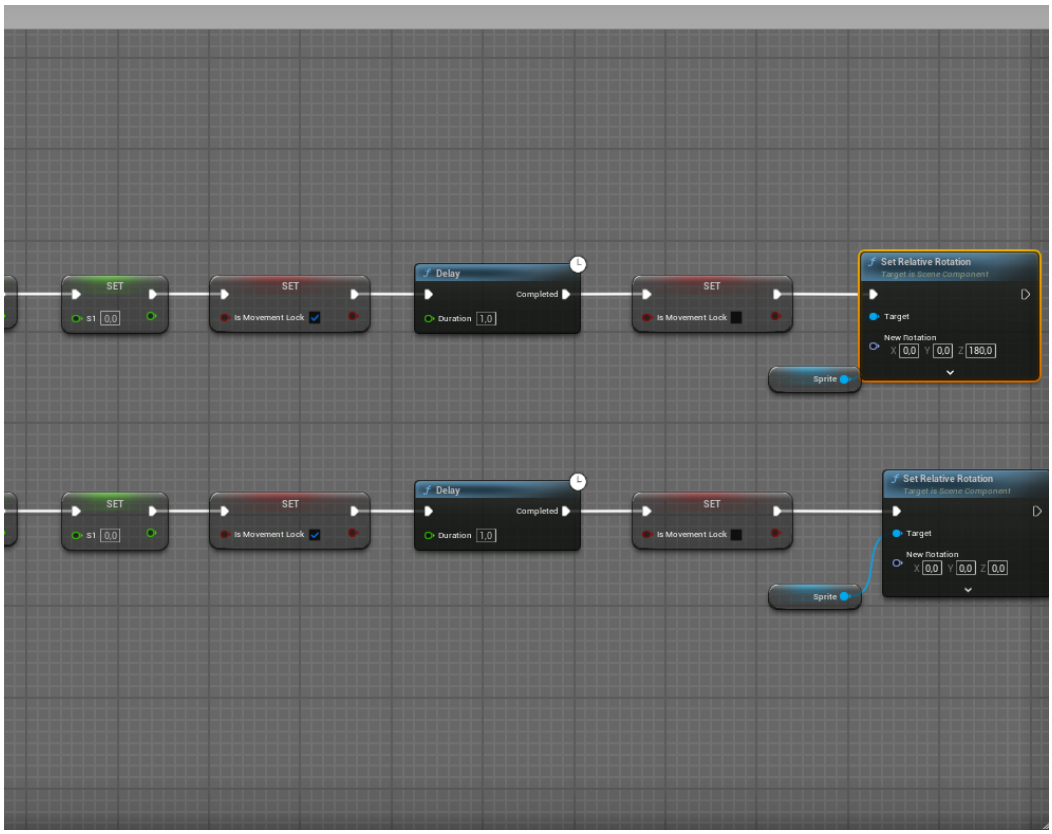

Рисунок Н.2

Декларація

академічної доброчесності

здобувача ступеня вищої освіти ЗНУ

Я, _____

студент __4__ курсу, ___денної____ форми навчання, математичного факультету, спеціальності __122 комп'ютерні науки_____, адреса електронної пошти ___ _____ , – підтверджую, що написана мною кваліфікаційна робота бакалавра на тему «_____» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст. 42 Закону України «Про освіту», зі змістом яких ознайомлений/ознайомлена;

– заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

– згоден/згодна на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Студент

_____	_____	_____
(дата)	(підпис)	(прізвище, ініціали)

Науковий керівник

_____	_____	_____
(дата)	(підпис)	(прізвище, ініціали)