НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ
кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

“До захисту допущено”
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА

“___”_____2024 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою
“Цифрові технології в енергетиці”
зі спеціальності *122 “Комп’ютерні науки”*

на тему: **Однокористувацька гра досягнення цілі на основі двигуна Unreal Engine**

Виконав (-ла):
студент (-ка) IV курсу, групи ТР-03
ШПАКОВСЬКИЙ Євген Олександрович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник:
асистент, Здор Констянтин Андрійович _____
(посада, науковий ступінь, вчене звання, прізвище, ім’я, по батькові) (підпис)

Рецензент: _____
_____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище, ім’я, по батькові) (підпис)

Нормоконтролер: _____ _____
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____
(підпис)

Київ – 2024

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

Рівень вищої освіти – перший (бакалаврський)
спеціальність 122 “Комп’ютерні науки”
Освітньо-професійна програма “Цифрові технології в енергетиці”

ЗАТВЕРДЖУЮ
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА_

«__»_____2024 р.

ЗАВДАННЯ
на дипломну роботу студенту

ШПАКОВСЬКОМУ Євгену Олександровичу

(прізвище, ім’я, по батькові)

1. Тема роботи **Однокористувацька гра досягнення цілі на основі рушія Unreal Engine**

керівник роботи ***Здор Констянтин Андрійович, асистент***

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 202__ р.
№_____

2. Термін подання роботи студентом **07.06.2024 року**

3. Вихідні дані до роботи мова програмування C++, рушій Unreal Engine

4. Зміст роботи: дослідження методів розробки рушія, розробка ігрових систем та механік, створення інтерфейсу, створення ігрових рівнів.

5. Орієнтований перелік ілюстративного матеріалу:_____

6. Дата видачі завдання «19» вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Затвердження теми роботи		
2.	Вивчення та аналіз задачі	17-20.04.24	
3.	Розробка архітектури та загальної структури системи	21-25.04.24	
4.	Розробка частин окремих підсистем	25.04-02.05.24	
5.	Програмна реалізація системи	03-07.05.24	
6.	Оформлення пояснювальної записки	08-12.05.24	
7.	Захист програмного продукту	13-17.05.23	
8.	Передзахист	03-06.06.23	
9.	Подання готової роботи на кафедру	07.06.2024	
10.	Захист	17-21.06.2024	

Студент _____ **Євген ШПАКОВСЬКИЙ**
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ **Констянтин Здор**
(підпис) (ім'я, ПРІЗВИЩЕ)

АНОТАЦІЯ

Дипломна робота виконана на 56 сторінках, містить 33 ілюстрації, 24 джерела в переліку посилань.

Мета роботи – створення однокористувацької гри в жанрі шутер від третьої особи на основі рушія Unreal Engine. Для цього було проведено аналіз жанрової класифікації ігор та вивчено сучасні зразки шутерів, а також різноманітні середовища розробки та ігрові рушії. На основі отриманих результатів було створено необхідні елементи ігрового процесу, зокрема дизайн рівня, запрограмована ігрова логіка, інтегровані 3D-моделі та анімації.

Методи та засоби: мова програмування C++, бібліотека Unreal Engine GameFramework, візуальна мова програмування Blueprints, рушій Unreal Engine, інтегроване середовище розробки Jetbrains Rider.

Результат – однокористувацька в жанрі шутер.

Ключові слова: UNREAL ENGINE, SINGLE-PLAYER, THIRD PERSON.

ABSTRACT

The thesis is executed on 56 pages, contains 32 illustrations, 24 sources in the list of references.

The purpose of the work is to create a single-player third-person shooter game based on the Unreal Engine. To do this, we analysed the genre classification of games and studied modern shooter examples, as well as various development environments and game engines. Based on the results, the necessary elements of the gameplay were created, including level design, programmed game logic, integrated 3D models and animations.

Methods and tools: C++ programming language, Unreal Engine GameFramework library, Blueprints visual programming language, Unreal Engine, Jetbrains Rider integrated development environment.

The result is a single-player game of goal achievement.

Keywords: UNREAL ENGINE, SINGLE-PLAYER, THIRD PERSON.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ ТА ТЕРМІНІВ

UE (Unreal Engine) – рушій Unreal Engine

DLC (Downloadable content) – завантажуваний контент

3D (3-Dimensional) – тривімірний

API (Application Programming Interface) – інтерфейс прикладного програмування

UI (User Interface) – інтерфейс користувача

BP (Blueprints) – вбудована мова візуального програмування рушія UE

TPS (Third-person shooter) – шутер від третьої особи

FPS (First-person shooter) – шутер від першої особи

ADS (Aim Down Sights) – прицілювання

SIMD (Single Instruction, Multiple Data) – одиночний потік інструкцій, множинний потік даних

HDR (High Dynamic Range) – високий динамічний діапазон

UAT (UnrealAutomationTool) – модуль рушія для автоматизації процесів розробки

Editor – редактор рушія UE

PickUp – ігрові бонуси, які можна підібрати

ВСТУП

У сучасному світі індустрія відеоігор стрімко розвивається, надаючи гравцям унікальні можливості занурюватися у віртуальні світи. Серед безлічі платформ та рушіїв для створення ігор одним із найпопулярніших є Unreal Engine, який завдяки своїй доступності та гнучкості дозволяє розробляти проєкти будь-якої складності та жанру. Незважаючи на численні переваги цього рушія, існують певні недоліки, з якими стикаються розробники під час створення ігор. Основною метою даної дипломної роботи є аналіз цих недоліків та розробка ефективних методів їх подолання на прикладі створення однокористувацької гри жанру шутер.

Одним із недоліків при розробці в Unreal Engine є складність оптимізації ігрового процесу та продуктивності, особливо для платформ з обмеженими ресурсами. Часто доводиться стикатися з проблемами ефективного управління пам’яттю, обробки великої кількості об’єктів у реальному часі та забезпечення стабільності гри.

Ще однією проблемою є високий поріг входу для новачків. Незважаючи на велику кількість навчальних матеріалів та документації, освоєння всіх аспектів Unreal Engine вимагає значного часу та зусиль. Це може бути особливо важким для людей, які не мають достатнього досвіду або ресурсів.

У цій дипломній роботі пропонуються методи, які дозволяють подолати зазначені недоліки, спираючись на поєднання існуючих інструментів та розробку нових алгоритмів. Зокрема, розглядаються питання оптимізації ігрового процесу, покращення продуктивності та ефективного управління ресурсами. Також особлива увага приділяється створенню інтуїтивно зрозумілого інтерфейсу для користувача, що дозволяє значно знизити поріг входу.

Дослідження та розробка проекту здійснювалися за допомогою сучасних методів та інструментів, що надаються Unreal Engine. Було проведено детальний аналіз архітектури та структури однокористувацької гри з метою розробки

оптимальних рішень для кожного з етапів розробки. Особлива увага приділялася питанням масштабованості та адаптивності проекту — це дозволяє розширювати гру у майбутньому без зайвих зусиль.

Результати роботи продемонстровано на прикладі створеної гри, яка демонструє високу продуктивність та стабільність. Реалізовані підходи можуть бути використані не тільки для розробки шутерів, але й для інших жанрів ігор, що робить запропоновану методологію універсальною. Таким чином, дана дипломна робота робить внесок у розвиток методів розробки ігор на Unreal Engine, дозволяючи подолати ключові недоліки рушія та полегшити розробку.

1 ГРА ЖАНРУ ШУТЕР ВІД ТРЕТЬОЇ ОСОБИ

Розробка гри є складним та багатогранним процесом, який включає кілька ключових стадій: концептуалізацію, передвиробництво, виробництво, тестування та пост-випускну підтримку. Кожна стадія відіграє важливу роль у створенні якісного ігрового продукту.

1.1 Підходи та методи розробки ігор

Першим етапом є – концептуалізація. На цьому етапі визначаються основні ідеї та концепції гри. Під час цього етапу проєктуються сюжет, ігрові механіки, стиль гри, та цільову аудиторію. Визначаються жанр гри, основні персонажі та візуальний стиль.

Наступним етапом є етап передвиробництва (Pre-Production) включає детальне планування проєкту. Детально описуються всі аспекти гри: від ігрових механік та систем до художнього оформлення та технічних вимог. Вибираються інструменти та платформи для розробки, формуються строки закінчення проєкту. Важливим кроком є створення прототипів ключових ігрових механік, щоб перевірити їх доцільність.

Після цього наступає найтриваліша та найскладніша фаза, а саме виробництво (Production). Вона включає активну розробку всіх елементів гри та реалізацію усіх планів, які були представлені на минулих етапах. Створюється графіка та анімації, пишеться код, для реалізації механік, або створення інструментів для спрощення розробки. Розробляються та наповнюються ігрові світи. Особлива увага приділяється розробці ігрового процесу, створенню штучного інтелекту ворогів та системи керування персонажем. Паралельно ведеться робота над звуковим оформленням гри, включаючи музичний супровід та звукові ефекти. Важливою частиною цього етапу є інтеграція всіх компонентів та забезпечення їхньої коректної взаємодії.

Після того, як була закінчена робота над основними механіками гри, починається всебічне тестування гри, щоб виявити та виправити помилки. Залучаються як внутрішні тестувальники, так і зовнішні гравці для проведення альфа- та бета-тестувань. Тестуються всі аспекти гри: геймплей, графіка, звуковий супровід, продуктивність на різних платформах та пристроях. Виправляються баги, оптимізується код та покращуються або додаються нові ігрові механіки на основі відгуків гравців.

Після релізу гра продовжує отримувати підтримку. Випускаються виправлення та оновлення, що виправляють виявлені помилки та додають новий контент. Переглядаються відгуки гравців для виправлення проблем, що виникають. Плануються додаткові оновлення, DLC (додатковий контент для завантаження) та інші способи підтримання інтересу до гри на тривалий час.

Кожна з цих стадій є критично важливою для успішної розробки гри. Від якісного планування на початкових етапах до ретельного тестування перед випуском, всі аспекти.

1.2 Жанр шутер від третьої особи

Шутер від третьої особи (TPS) – є піджанром 3D-шутерів, в яких ігровий процес складається переважно зі стрільби. Він тісно пов'язаний з шутерами від першої особи, але персонажа гравця видно на екрані під час гри. У той час як 2D-шутери також використовують перспективу від третьої особи, жанр TPS відрізняється тим, що гра представлена аватаром гравця, який є основним об'єктом зору камери (рисунок 1.1).

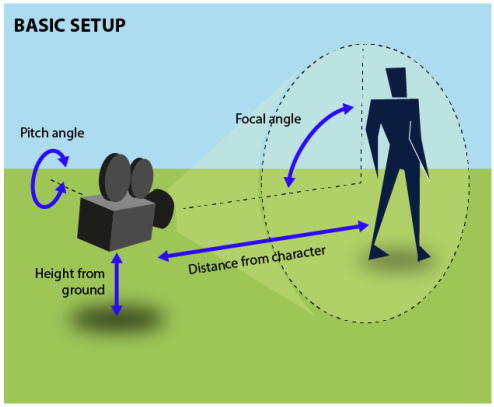

Рисунок 1.1 – Камера від третьої особи

Цей 3D-жанр набув популярності у 2000-х роках, особливо на ігрових консолях. Він містить елементи стрільби, іноді поєднуючи їх із елементами платформінгу, тобто ігровий процес базується навколо обходу перешкод через стрибки, головоломки та бойові сцени. Шутери від третьої особи часто включають функцію допомоги прицілювання, щоб компенсувати труднощі, пов'язані з прицілюванням з камери від третьої особи. Багато з цих ігор також мають режим вигляду від першої особи, що дозволяє точно стріляти і досліджувати деталі оточення, які можуть бути приховані від стандартної камери. У ранніх прикладах жанру гравцеві часто доводилося стояти на місці для використання вигляду від першої особи, але сучасні ігри дозволяють інтегрований режим гри як у FPS.

Ці ігри тісно пов'язані з шутерами від першої особи, які також прив'язують перспективу гравця до аватара, відрізняючись лише незначною зміною положення камери гравця. У той час як перспектива від першої особи дозволяє гравцям цілитися і стріляти без того, щоб їхній аватар закривав їм огляд, шутер від третьої особи показує головного героя з перспективи «постріл через плече» або «з-за спини». Таким чином, перспектива від третьої особи дозволяє гейм-дизайнеру створити більш яскраво виражений аватар і спрямовує увагу гравця, як при перегляді фільму. На противагу цьому, перспектива від першої особи забезпечує гравцеві більше занурення в ігровий всесвіт.

Шутери від третьої особи дозволяють гравцям чіткіше бачити місцевість навколо аватара. Ця точка зору сприяє більшій взаємодії між персонажем і

навколишнім середовищем. Таким чином, перспектива від третьої особи краще підходить для взаємодії з об'єктами в ігровому світі, наприклад, для стрибків на платформах, участі в ближньому бою або керування транспортним засобом. Однак перспектива від третьої особи може заважати ігровому процесу, коли від гравця вимагається більш точне прицілювання.

Межі між шутерами від третьої особи та від першої особи не завжди чіткі. Наприклад, багато шутерів від третьої особи дозволяють гравцеві використовувати точку зору від першої особи в деяких випадках, тоді як інші просто дозволяють гравцеві вільно перемикатися між перспективами від першої та третьої особи за бажанням.

Resident Evil 4 (2005) вплинула на переосмислення жанру шутерів від третьої особи завдяки використанню зміщених кутів камери «через плече», коли камера розміщується безпосередньо над правим плечем і тому не затуляє дію. (рисунок 1.2).

Рисунок 1.2 – Приклад роботи камери “через плече” у грі Resident Evil 4

Саме такий кут камери, був обраний для мого проєкту.

1.3 Існуючі проєкти схожого масштабу

Одним з прикладів схожих проєктів є проєкт з відкритим кодом від компанії Epic Games під назвою Lyra Starter Game (Рисунок 1.2). Проєкт був представлений разом з релізом версії рушія 5.0 для демонстрації нововведень, та так званих “Best-Practices” при роботі з ним. Проєкт є більшим, адже був розроблений окремою командою, яка займається створенням навчальних проєктів. Він має систему валідації для Blueprints, а також інших ассетів, які присутні у проєкті. Система введення даних, тобто Input, яка відповідає за мишку та клавіатуру була перероблена у новій версії рушія, і це якраз те, що використовується у проєкті Lyra Starter Game. Також гра має підтримку багатьох користувачів. І використовуючи покращення в рендер-системі двигуна, виглядає сучасно. Проте не реалізує засобів автоматизації розробки. Наприклад, не має скриптів для автоматичної генерації файлів проєкту та документації.

Рисунок 1.2 – Скріншот з проєкту Lyra Starter Game

Іншим прикладом, є Unreal Tournament 2014 (Рисунок 1.3). Гра, яку Epic Games розробляла та готувала до релізу, проте через те, що студії потрібно було фокусуватися на своєму амбітному проєкті Fortnite, вони відмовилися від релізу Unreal Tournament. Через деякий час код проєкту став доступним на Github.

Рисунок 1.3 – Скріншот з проєкту Unreal Tournament

Що цікаво, це система пікапів, яка була реалізована. Адже в цьому проєкті, підбирати можна не лише бонуси, але й зброю, яка випадає з убитих ворогів.

Але маючи цікаві рішення у своєму ігроладі, вона так само як і попередній проєкт не має засобів автоматизації процесів розробки.

2 РОЗРОБКА ІГРОВИХ МЕХАНІК

2.1. Проєктування ігрових механік

Ігрові механіки є основою, яка забезпечує захоплюючий і динамічний ігровий процес у грі. Вони повинні наповнювати ігровий процес та бути логічно поєднані між собою.

Механіка стрільби включає основні принципи та функціональність, які дозволяють гравцеві використовувати зброю в грі. Гравець повинен мати можливість обирати різні види зброї, стріляти, прицілюватися та перезаряджати обрану зброю. Важливими аспектами є точність, візуальна віддача та розсіювання куль, що роблять стрільбу більш реалістичною. Повна детальна розробка системи стрільби буде розглянута у наступному розділі.

Механіка ходьби та бігу є засобами навігації по ігровому світу. Гравець може переміщатися у двох режимах: повільний хід та біг. Ходьба забезпечує точне управління персонажем, особливо під час прицілювання та стрільби. Біг дозволяє швидко долати великі відстані, уникаючи ворожого вогню, але знижує точність при стрільбі.

Механіка пікапів (PickUp) дозволяє гравцю підбирати корисні предмети, розкидані по рівню. Основними видами пікапів є: PickUp здоров’я, він відновлює здоров'я гравця до максимального значення, дозволяючи продовжувати бій та рятуючи у напружених ситуаціях. Патрони, що доповнюють запас боєприпасів, необхідних для продовження стрільби. Пікапи повинні бути розміщені на карті таким чином аби не бути сконцентрованим в одному місці, аби не створювати точки в яких гравець може бути невразливим.

Механіка повернення гравця у гру після смерті (Respawn) визначає, яка затримка на появу гравця повинна бути, а також в якій точці на карті гравець з'являється. Важливо збалансувати час, щоб гравець не втрачав інтерес до гри. Оптимальна затримка має бути достатньо короткою, щоб гравець швидко повертався в гру, але не настільки, щоб втрачав відчуття покарання за помилку.

Вибір місця повернення повинен запобігати негайному влучанню у ворожий вогонь, що дозволяє гравцю підготуватися до продовження бою.

Механіка здоров'я визначає, скільки шкоди може витримати гравець, перш ніж загинути: Шкала здоров'я відображає поточний стан здоров'я гравця. Відновлення здоров'я може бути автоматичним з часом або вимагати пікапів здоров'я. Влучання у голову ворога завдає підвищену шкоду, тобто повинні бути множники шкоди. Влучання в голову завдає більше шкоди, ніж влучання в інші частини тіла, що заохочує гравця до високої точності при прицілюванні, а не простого клацання на мишку.

Для полегшення ідентифікації ворогів і союзників повинна існувати механіка розділення ворогів та союзників за кольором. Використовуються візуальні підказки: Кольорові маркери на моделях, завдяки яким вороги будуть позначатися кольором, який буде складно сплутати з кольором союзників.

2.2. Система стрільби

Система стрільби є однією з ключових механік у грі жанру шутер від третьої особи, оскільки саме від її якості та реалізації залежить значна частина ігрового досвіду. Ефективна та реалістична система стрільби забезпечує гравцям захоплюючий ігровий процес, високу динаміку та напруженість у бойових сценах, а також змушує їх навчатися, адже кожна зброя потребує свого підходу. Швидкість стрільби, перезарядки, поведінка випущеного снаряду, час його польоту, це все потребує від гравця спритності та досвіду, що стимулює його вкладати більше часу вивчення зброї, яка сподобалася йому найбільше та покращувати свої навички.

Система стрільби у грі складається з багатьох компонентів основним з яких звісно є зброя. Зброя поділена на два види, звичайна автоматична гвинтівка, та гранатомет. Базово зброю можна охарактеризувати декількома параметрами. Шкода, дальність, точність, швидкість перезарядки, ємність магазину та швидкість стрільби. (рисунок 2.1)

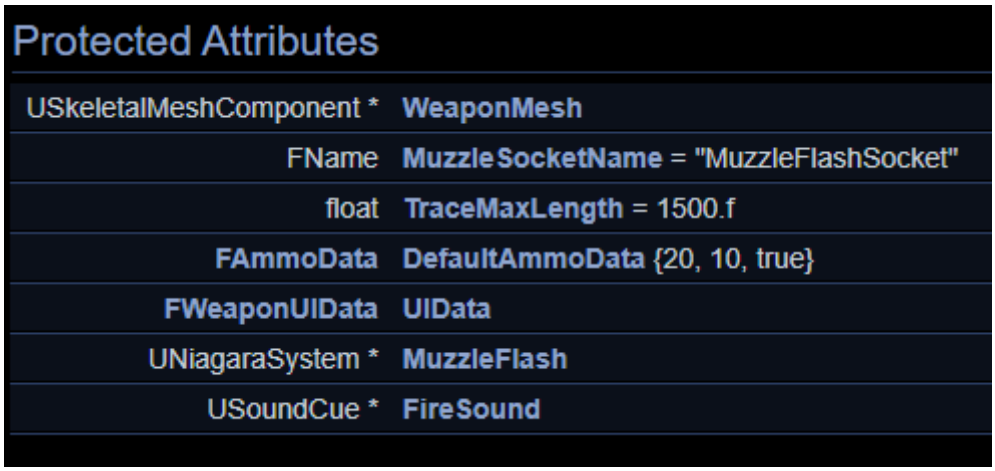

Рисунок 2.1 – Характеристики класу BaseWeapon

Отже базово дальність стрільби визначається 1500 єдиницями вимірювання Unreal Engine, або ж 1500 см, але це значення може легко бути змінено в редакторі. Структура AmmoData, яка була створена для зручного зберігання даних про боєприпаси зброї містить в собі інформацію про кількість магазинів, кількість патронів в одному магазині, а також чи є боєзапас скінченним. Вона допомагає у відстеженні кількості боєприпасів у магазині та інвентарі гравця.

Також вогонь зі зброї можна вести по-різному. Від стегна, або з домогою прицілювання через приціл (ADS - Aim Down Sights), що підвищує точність. Режими прицілювання є критично важливими для точності стрільби. Прицілювання (ADS) підвищує точність, але знижує швидкість руху персонажа. Стрільба від стегна дозволяє швидко реагувати на близькі загрози, але має вищий рівень розсіювання. Реалізація цих режимів включає створення відповідних анімацій та алгоритмів, що змінюють параметри зброї залежно від обраного режиму.

Візуальні індикатори прицілювання є необхідною частиною механіки, адже їх відсутність зробить стрільбу занадто складною і змусить гравці страждати через неможливість влучити у суперник. Кожен вид зброї має свій приціл. Як вже було розглянуто на рисунку 2.1, BaseWeapon має структуру WeaponUIData,

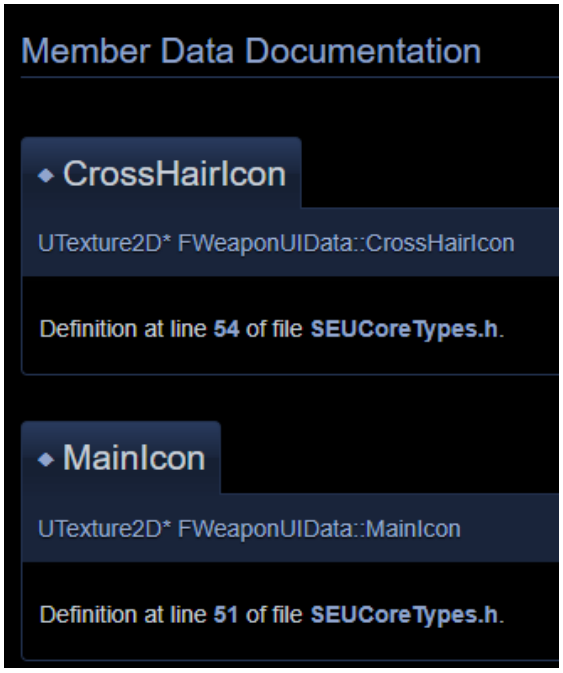

Рисунок 2.2 – Поля структури WeaponUIData

Ця структура зберігає текстуру для прицiла, а також головну іконку зброї, що відображається на екрані гравця під час матчу.

В системі стрільби гри, для кожної моделі зброї використовується сокет, який є іменованою трансформацією на скелеті. Сокет визначає положення та орієнтацію точки, з якої візуально вилітають кулі. Проте, фактичний початок траєкторії стрільби визначається з голови гравця, щоб забезпечити точнішу симуляцію прицілювання та пострілу. По-перше, потрібно отримати поточну позицію і орієнтацію камери гравця. Далі використовуючи наявні дані, та змінну що відповідає за кут розсіювання визначається напрямок пострілу. Для гвинтівки використовується ф-я VRandCone з математичної бібліотеки рушія, вона генерує випадковий вектор у межах конуса з центром у напрямку орієнтації персонажа. Ця функція використовується для моделювання розсіювання пострілів, надаючи певний рівень неточності, що відповідає реалістичному пострілу зі зброї. Функція VRandCone використовує принцип підбору точок на сфері (Sphere Point Picking). Цей принцип полягає у випадковому виборі точок на поверхні сфери, що забезпечує рівномірний розподіл напрямків. Спочатку генеруються випадкові координати для точок, які рівномірно розподілені по поверхні сфери. Ці координати конвертуються у сферичні координати для створення конуса навколо

центрального напрямку. Обрані точки використовуються для визначення напрямку пострілу, забезпечуючи певне розсіювання. Важливо, що ця функція використовує в обчисленнях SIMD інструкції, що підвищує швидкість обчислень. Проте це не основна причина використань, тим більше, що загальна складність обчислень стрільби не є надто високою і не потребує зайвих оптимізацій і як справедливо сказав комп'ютерний вчений Дональд Кнут: «Передчасна оптимізація - це корінь усього зла», що є важливим принципом в програмуванні, адже зміна коду для підвищення його продуктивності перед тим, як це стало реальною проблемою може забрати цінний час і при цьому не принести значних покращень.

Після виконання цих дій визначаються початкова та кінцеві точки трасування. Де початком буде положення голови гравця в просторі, а кінцем векторна сума початкової точки та напрямку стрільби, що помножена на максимальну дальність стрільби. Для гранатомета принцип такий самий, але при цьому не генеруються випадкові точки для розсіювання.

Обчислення влучень кулі в ціль відбувається за допомогою трасування, після того як були обчислені точки, засобам рушія випускається промінь, який повертає структуру, яка зберігає дані про влучання, якщо воно відбулося. З цієї структури ми можемо дізнатися, який актор повинен отримати шкоду, який її тип і з цього дізнатися, чи повинні ми застосувати якісь множники. Різні зони на тілі персонажа (голова, торс, кінцівки) мають різні множники шкоди. Значення цих множників можна легко налаштувати через редактор, вони зберігаються як мапа, ключем якої є фізичний матеріал на моделі персонажа (рисунок 2.3).

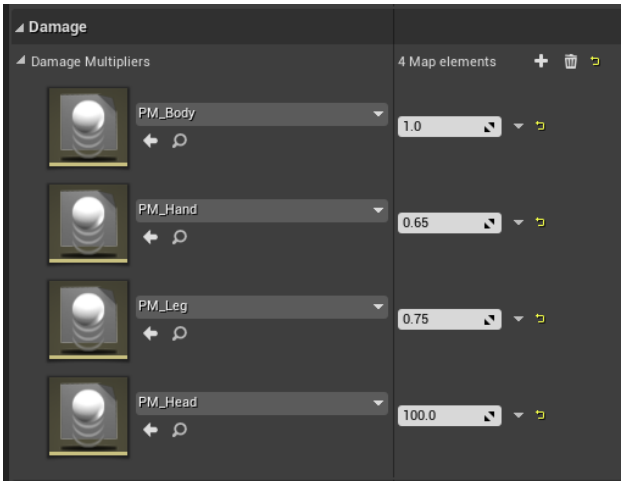

Рисунок 2.3 – Мапа множників шкоди

Звукові та візуальні ефекти є невід'ємною частиною для забезпечення іммерсивності пострілу. Після пострілу в локації сокету, створюється система часток, або ж Niagara System, також система створюється у місці влучання. Niagara є вбудованою системою часток в рушії. Програється звук пострілу.

2.3. Анімації

Розробка анімацій є одним із ключових етапів створення ігрового проекту, оскільки саме анімації забезпечують плавність руху персонажів, що підвищує загальну якість гри та занурення в ігровий світ.

Для реалізації анімацій були використані кілька ключових інструментів та технологій, включаючи Aim Offset, Animation Blueprints, State Machine та BlendSpace, які надає рушій Unreal Engine.

Aim Offset – це інструмент, який дозволяє плавно змінювати напрямок погляду персонажа в залежності від положення камери. Реалізація Aim Offset складається з використання спеціальних анімаційних поз, які об'єднуються та інтерполюються між собою в залежності від кута нахилу та повороту камери Завдяки цьому інструменту, анімації персонажа автоматично адаптуються до напрямку стрільби. (рисунок 2.4).

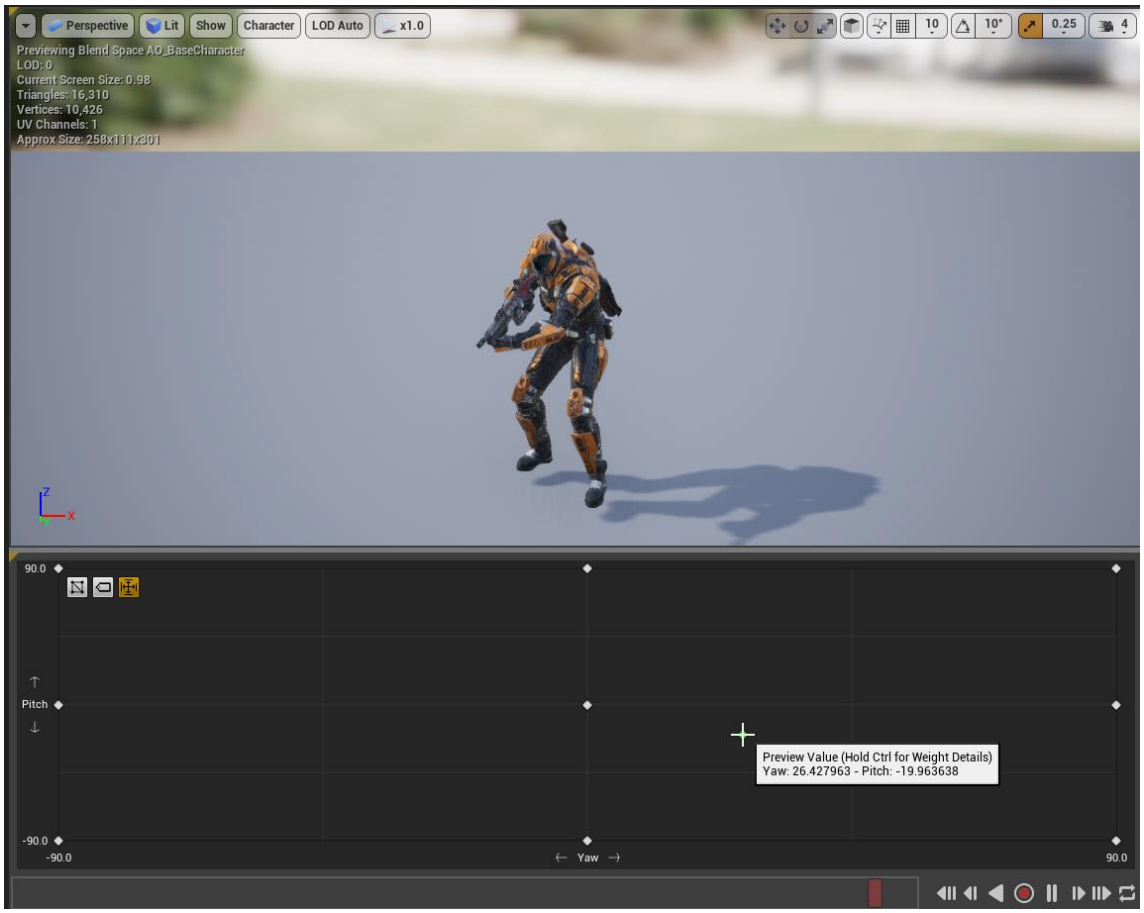

Рисунок 2.4 –Скріншот AimOffset

Animation Blueprints – це інструмент Unreal Engine, який дозволяє створювати складну логіку анімацій за допомогою візуального програмування.

В цьому проєкті Anim Blueprints були використані для визначення та контролю анімаційних станів персонажа. Цей інструмент відповідає за управління переходами між різними анімаціями, синхронізацію рухів та забезпечення плавності їх виконання. Наприклад, за допомогою Anim Blueprints було реалізовано перемикання між анімаціями ходьби, бігу та стрибків що забезпечує плавність рухів ігрового персонажа (рисунок 2.5).

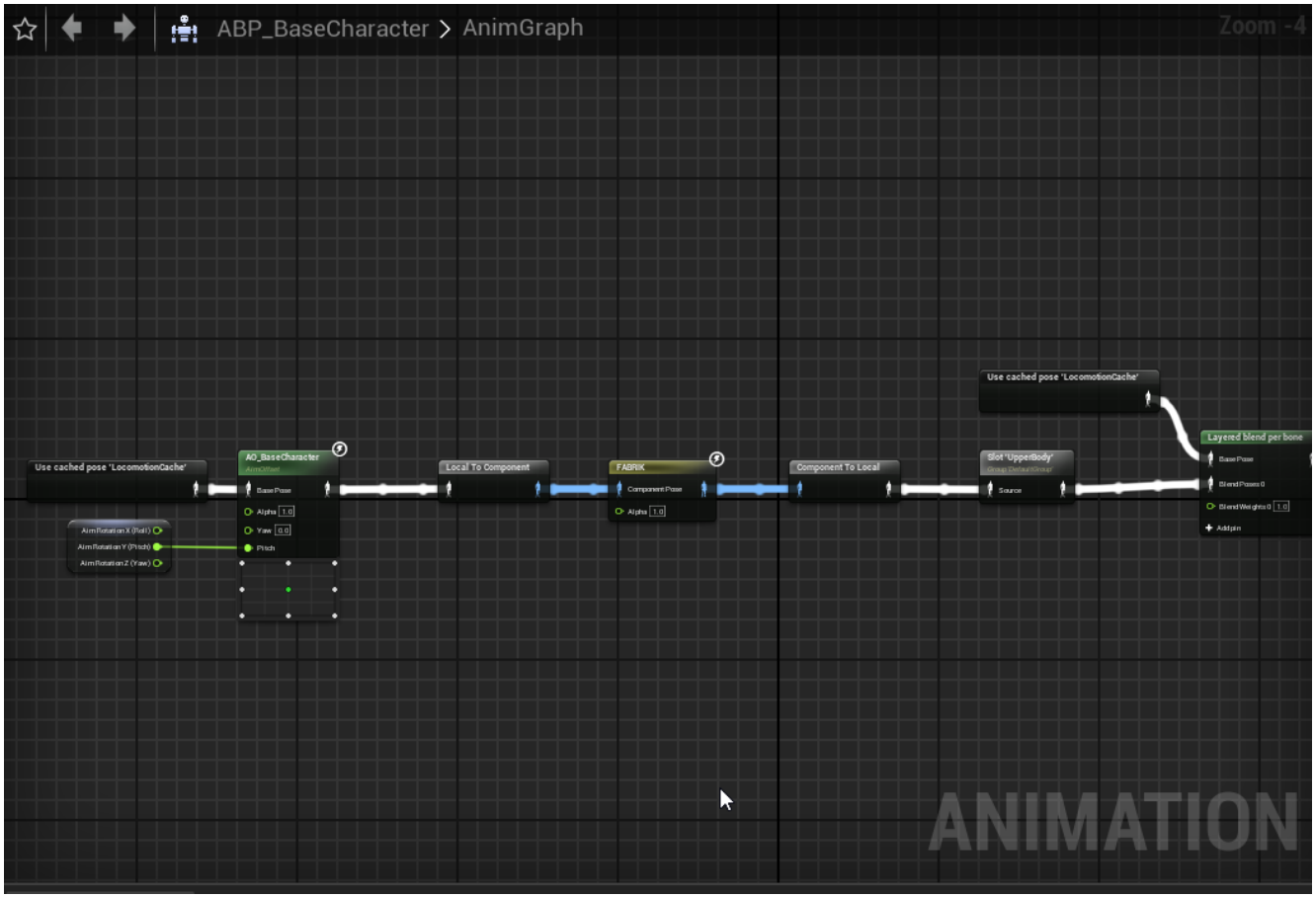

Рисунок 2.5 – Скріншот AnimationBlueprint персонажа

State Machine – це компонент Anim Blueprints, який дозволяє організовувати анімації у вигляді станів і визначати правила переходу між ними. State Machine була використана для управління основними станами персонажа, такими як стан покою, ходьба, біг, стрибки. Кожен стан мав свою власну анімаційну послідовність, яка активувалася в залежності від дій гравця та умов гри. Використання State Machine дозволило легко керувати складними анімаційними послідовностями та забезпечити їхню відповідність ігровим подіям.

(рисунок 2.6)

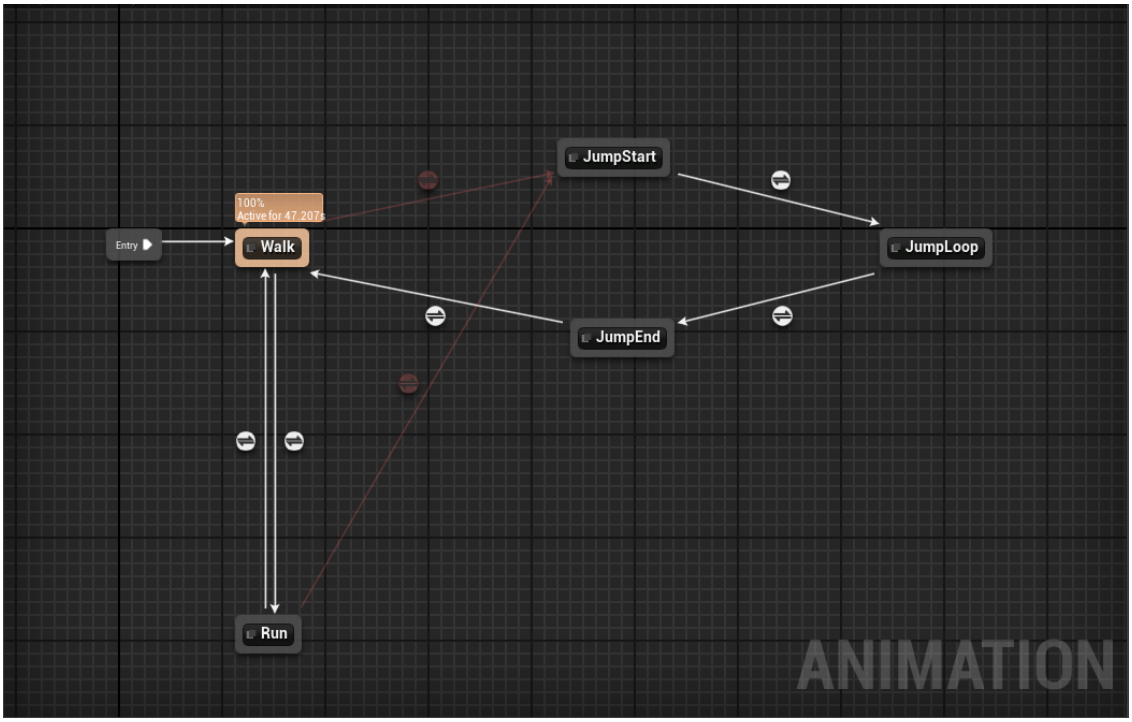

Рисунок 2.6 – StateMachine персонажа

BlendSpace – це інструмент, який дозволяє плавно переходити між різними анімаціями на основі певних параметрів, таких як швидкість руху персонажа чи напрямок. BlendSpace використовувався для створення плавних переходів між анімаціями ходьби та бігу. Це дозволило зробити рухи персонажа більш реалістичними і менш рваними. Наприклад, при прискоренні персонажа анімація плавно переходила від ходьби до бігу, а при уповільненні – у зворотному напрямку. Використання BlendSpace значно покращило візуальне сприйняття рухів (рисунок 2.7).

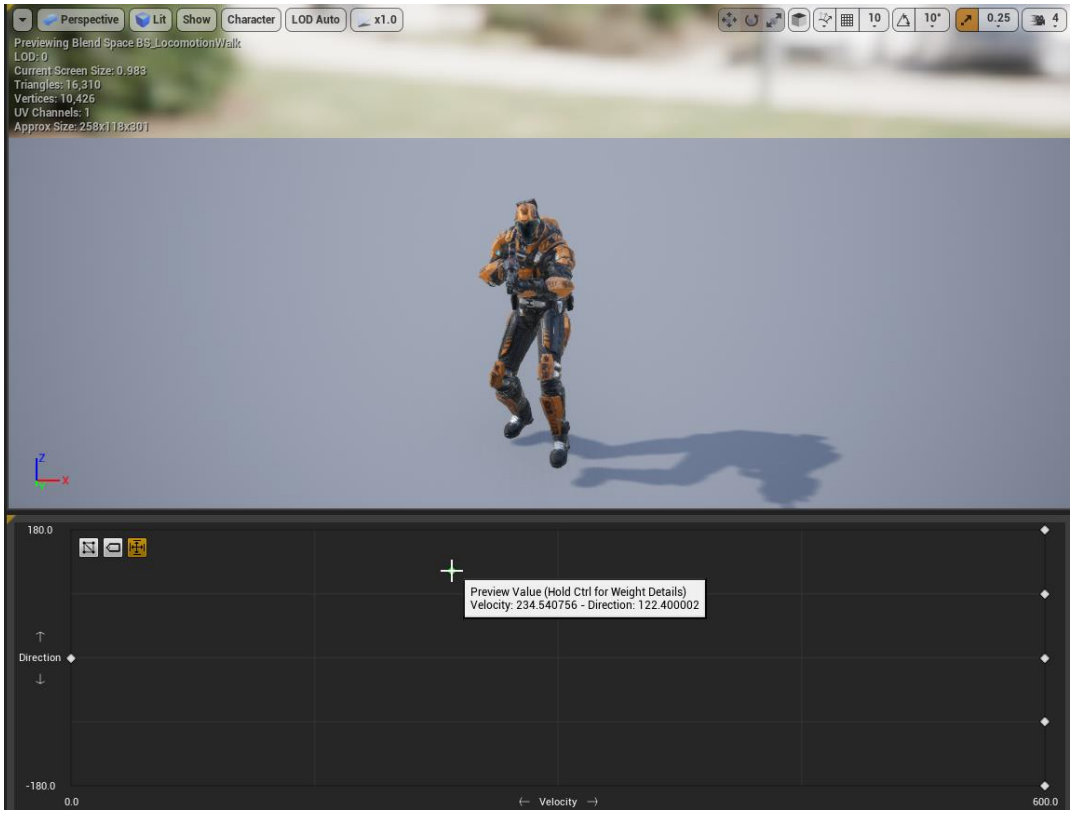

Рисунок 2.7 – Blendspace бігу

Реалізація анімаційного циклу в грі включала використання базових анімаційних кліпів для різних станів персонажа, таких як ходьба, біг, стрибки, стрільба та інші дії. Ці анімації були об'єднані в єдину систему за допомогою вищезазначених інструментів. Кліпи були імпортовані в Unreal Engine, де за допомогою вищезазначених інструментів було налаштовано їх переходи між ними.

2.4. Штучний інтелект ворогів

Unreal Engine надає широкий спектр можливостей для розробки інтелектуальних агентів, використовуючи такі компоненти, як дерева поведінки, сітка навігації, компонент сприйняття для штучного інтелекту, сервіси і декоратори, а також Environment Query System (EQS) та EQS Context.

Основним інструментом для управління поведінкою ботів у Unreal Engine є

Behavior Tree, або дерево поведінки. Дерево поведінки являє собою ієрархічну структуру, яка дозволяє описувати послідовність дій і умов, які визначають поведінку бота. Кожен вузол дерева може бути дією, умовою або комбінацією обох. Вузли умов перевіряють певні стани або тригери, тоді як вузли дій виконують конкретні команди. Це дозволяє створювати складні і гнучкі моделі поведінки, які можуть адаптуватися до змін у ігровому середовищі. Наприклад, бот має гілку дерева поведінки для патрулювання, а іншу — для атаки, причому вибір між ними залежить від наявності ворога у полі зору (рисунок 2.8).

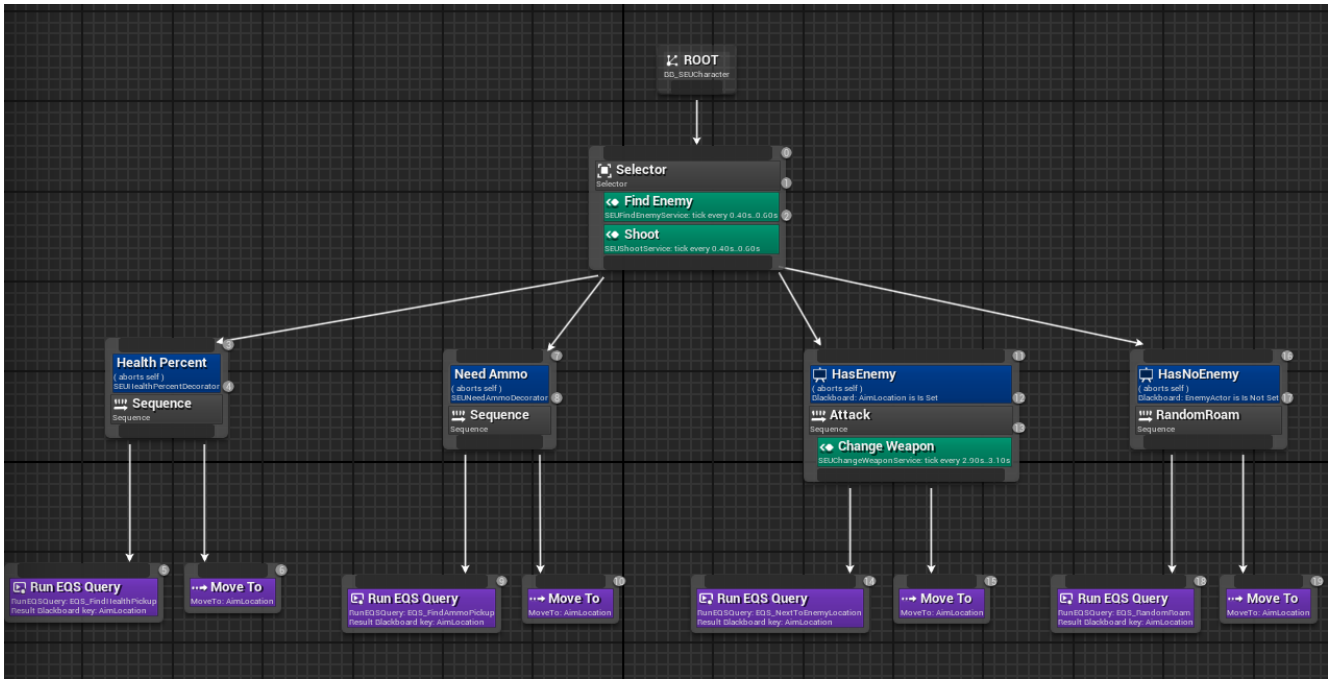

Рисунок 2.8 – Дерево поведінки неігрового персонажа

Navigation Mesh, або навігаційна сітка, є ключовим компонентом для забезпечення навігації ботів у складному ігровому середовищі. Навігаційна сітка створюється на основі геометрії ігрового рівня і визначає доступні шляхи для пересування ботів. Це дозволяє ботам обирати оптимальні маршрути для досягнення своїх цілей, уникати перешкод і знаходити укриття. За допомогою навігаційної сітки також реалізуються такі функції, як переслідування гравця.

Компонент AIPerceptionComponent забезпечує ботам можливість сприймати навколишнє середовище за допомогою різних сенсорів, таких як зір або слух. Цей

компонент дозволяє ботам виявляти гравців, інших ботів та об'єкти, а також реагувати на звуки і рухи. Наприклад, якщо бот побачить гравця він змінить свою поведінку та почне переслідування. AIPerceptionComponent також дозволяє визначати ступінь обізнаності бота про наявність ворогів у його зоні видимості, що робить поведінку ботів більш реалістичною і адаптивною.

Сервіс знаходження ворогів є спеціалізованим механізмом, що дозволяє ботам визначати і вибирати цілі для атаки. Цей сервіс аналізує інформацію, отриману від сенсорів AIPerceptionComponent, і оцінює загрозу від кожного потенційного ворога, враховуючи відстань до ворога. Сервіс знаходження ворогів допомагає ботам приймати рішення, які засновані на інформації з ігрового рівня, а не випадковостях.. Це дозволяє ботам діяти більш ефективно і агресивно, підвищуючи загальний рівень виклику для гравця.

Декоратори у дереві поведінки використовуються для додаткової обробки умов і модифікації поведінки ботів. Декоратори додають логічні перевірки до вузлів дерева, що дозволяє створювати більш складні умови для виконання дій. Наприклад, декоратор може перевіряти, чи знаходиться ворог у полі зору бота, перш ніж дозволити йому відкрити вогонь. Інший приклад — перевірка рівня здоров'я бота перед тим, як він вирішить вступити в бій або відступити. Використання декораторів дозволяє значно підвищити гнучкість і точність дерев поведінки, забезпечуючи більш реалістичну і відповідну поведінку ботів у різних ситуаціях. Для підвищення реалістичності поведінки ботів, був розроблений декоратор, за допомогою якого штучний інтеллект шукає найближчий PickUp здоров’я у випадку, якщо процент його здоров’я нижче заданого.

Сервіс стрільби є ще одним ключовим компонентом для реалізації бойових можливостей ботів. Він відповідає за управління всіма аспектами стрільби, включаючи вибір цілі, обчислення траєкторії пострілу і виконання самої стрільби. Цей сервіс враховувує різні параметри, такі як тип зброї, дистанція до цілі, наявність перешкод та інші фактори, що впливають на точність і ефективність стрільби. Наприклад, бот може використовувати автоматичну зброю для ближнього бою, або взяти у руки гранатомет, якщо у нього закінчилися боєприпаси змінюючи свою поведінку відповідно до ситуації. Сервіс стрільби

дозволяє ботам виконувати більш точні і стратегічні постріли, підвищуючи їхню ефективність у бою.

Environment Query System (EQS) є потужним інструментом для створення складних запитів до ігрового середовища. EQS дозволяє ботам аналізувати і оцінювати різні параметри навколишнього середовища для прийняття рішень. Наприклад, бот може використовує EQS для пошуку бонусів або визначення оптимальної позиції для атаки. Запити в EQS можуть враховувати різні фактори, такі як наявність перешкод, видимість, відстань до ворогів і союзників. Це дозволяє ботам приймати більш обґрунтовані і стратегічні рішення, покращуючи їхню взаємодію з ігровим середовищем.

3 ЗАСОБИ РОЗРОБКИ

3.1 Рушій Unreal Engine

Unreal Engine (UE) - це серія ігрових рушіїв для комп'ютерної 3D-графіки, розроблена компанією Epic Games, вперше продемонстрована у відеогрі-шутері від першої особи Unreal 1998 року. Спочатку розроблений для комп'ютерних шутерів від першої особи, він з тих пір використовується в різних жанрах ігор і був прийнятий в інших індустріях, особливо в кіно- і телеіндустрії. Unreal Engine написаний на C++ і відрізняється високим ступенем портативності, підтримуючи широкий спектр настільних, мобільних, консольних платформ і платформ віртуальної реальності. Тім Суїні, засновник Epic Games, почав створювати інструменти для редагування своїх перших ігор у 1991 році. Все почалося з пригодницької головоломки з дуже простою графікою під назвою ZZT (рисунок 3.1).

Рисунок 3.1 – Скріншот гри ZZT

У 90-х роках ігрова індустрія була сповнена потенціалу. Але навіть найбільші ігри робилися невеликими командами і створювалися з нуля за допомогою складного процесу. Суїні усвідомлював важливість ігрового рушія, який дозволяв користувачеві вносити значні зміни, що виходили за рамки простих редакторів мап того часу, але, що важливіше, цей підхід заклав концептуальну основу для Unreal Engine, який зробить революцію в ігровій індустрії.

У 1995 році Тім почав працювати над 1-м поколінням рушія Unreal. Рушій був використаний для розробки шутера від першої особи, відомого як Unreal. Дивно, але спочатку рушій покладався виключно на програмний рендеринг за допомогою центрального процесора, але з часом він також використовував переваги відеокарт, що дозволило створювати більш просунуті ігри та графіку. Гра Unreal була створена у 1998 році і стала однією з перших ігор, яка зробила акцент на текстурах та рівнях деталей. (Рисунок 3.2)

Рисунок 3.2 – Скріншот з гри Unreal

Гра мала шалений успіх, було продано понад 1,5 мільйона копій. GameSpot похвалив гру за використання 16-бітного кольору і додав, що рушій Unreal Engine підняв планку того, чого геймери очікують від майбутніх продуктів. Свою другу гру Epic випустила у 1999 році під назвою «Unreal Tournament». Хоча попередня гра мала шалений успіх, з рушієм все ще залишалися деякі проблеми. Наприклад, геймери скаржилися на високі системні вимоги для гри. Хоча були деякі проблеми і з онлайн складовою. (рисунок 3.3)

Рисунок 3.3 – Скріншот з гри Unreal Tournament (1999)

Так, розробники вдосконалили Unreal Engine для оптимізації продуктивності на машинах з низьким рівнем продуктивності, а також покращили багатокористувацькі функції. На додачу до цього була покращена графіка, що дозволило використовувати 24-бітні високоякісні текстури, які не вплинули на продуктивність. Всі зусилля розробників виправдали себе, адже Unreal Tournament часто вважають однією з найкращих відеоігор, коли-небудь створених. На Metacritic гра отримала вражаючу оцінку 92 бали, і було продано понад 2 000 000 копій.

Друга версія рушія Unreal Engine дебютувала у 2002 році з безкоштовним багатокористувацьким шутером America's Army. Армія США та ігрова індустрія працювали разом у 8-річному партнерстві над розробкою гри. Вона мала на меті інформувати, навчати та вербувати перспективних солдатів. Цікаво, що це був перший випадок, коли армія США використала широкомасштабну ігрову технологію для рекрутингу. Вона навіть отримала кілька нагород, в тому числі як найкраще використання податкових коштів за версією журналу «Computer Games» і як найбільший сюрприз року за версією IGN. До ігрового рушія було внесено численні вдосконалення, що дозволило реалізувати нові функції (рисунок 3.4).

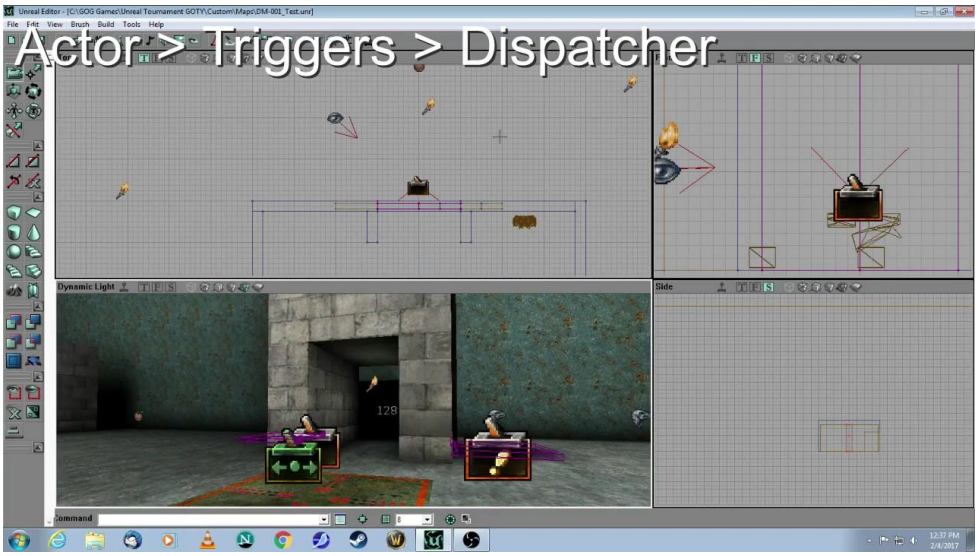

Рисунок 3.4 – Скріншот редактора Unreal

Unreal Engine 2 був заснований на своєму попереднику, але мав багато значних покращень. Він може бути майже в 100 разів більш деталізованим, ніж його попередник. Також було додано нові функції, зокрема інструмент кінематографічного редагування, систему частинок та систему скелетної анімації. У Unreal Tournament 2004 розробники вперше успішно реалізували ігровий процес на основі транспортних засобів.

У листопаді 2006 року вийшов Unreal Engine 3 або ж UDK (рисунок 3.5).

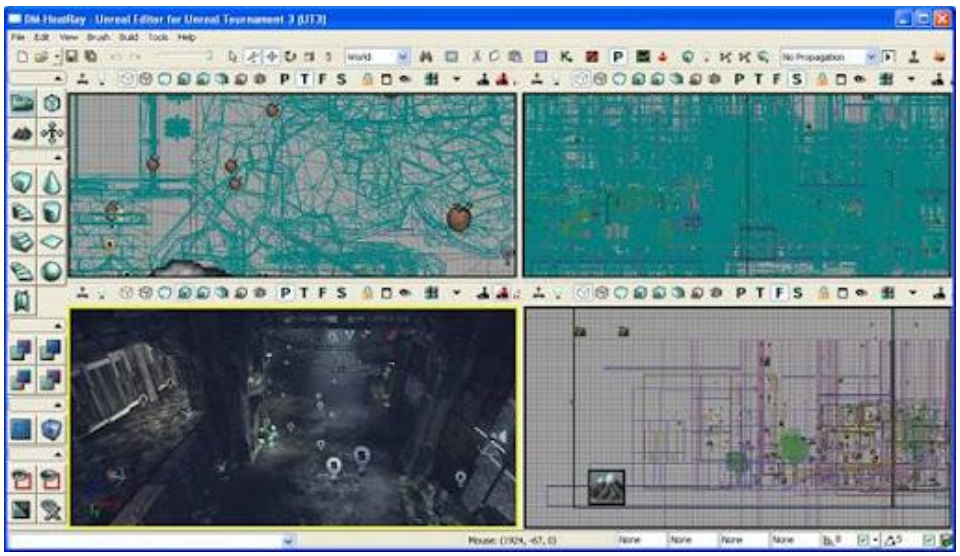

Рисунок 3.5 – Скріншот редактора UE3

Першою грою, яка використовувала рушій, стала Gears of War. На сайті Metacritic гра отримала вражаючу оцінку 94 бали. Gears of War отримала високу оцінку за концепцію, візуальну складову, презентацію та звук. IGN назвав її «найрозкішнішою грою на Xbox 360» (рисунок 3.6).

Рисунок 3.6 – Скріншот з гри Gears of War

Unreal Engine 3 прийшов з новими досягненнями в графіці, які допомогли зробити Gears of War успішною. Рушій базувався на першому поколінні, але до нього було додано багато нових функцій, включаючи систему фізики, звукову систему та інструменти, які стали значно потужнішими. Суттєвою зміною порівняно з Unreal Engine 2 стало те, що обчислення проводилися на піксель, а не на вершину. Це зробило графіку набагато кращою. Також було покращено рендеринг і додано підтримку рендерингу з високим динамічним діапазоном (HDR).

Протягом багатьох років розробники продовжували оновлювати рушій новими функціями, включаючи покращене руйнування середовища, симуляції великого натовпу, стереоскопічний 3D та м'яку динаміку тіла. Популярність рушія також значно зросла. Це призвело до ліцензійних угод з такими великими компаніями, як Atari, Activision, Capcom, Disney, Ubisoft, Electronic Arts та іншими; близько 300 ігор було розроблено з використанням Unreal Engine 3.

19 березня 2014 року вийшов рушій Unreal Engine 4. Тоді ж Epic відкрили вихідний код рушія для всіх бажаючих, зробивши його ще більш доступним.

Розробка почалася за 11 років до цього. На початку над ним працював лише сам Тім Суїні.

Одним з головних нововведень Unreal Engine 4 стало глобальне освітлення в реальному часі за допомогою воксельного трасування конусів, що додає більш реалістичного освітлення в 3D-сценах.

Ще однією значною зміною стало те, що художники та дизайнери тепер можуть розробляти елементи за допомогою системи візуальних скриптів Blueprint (рисунок 3.7).

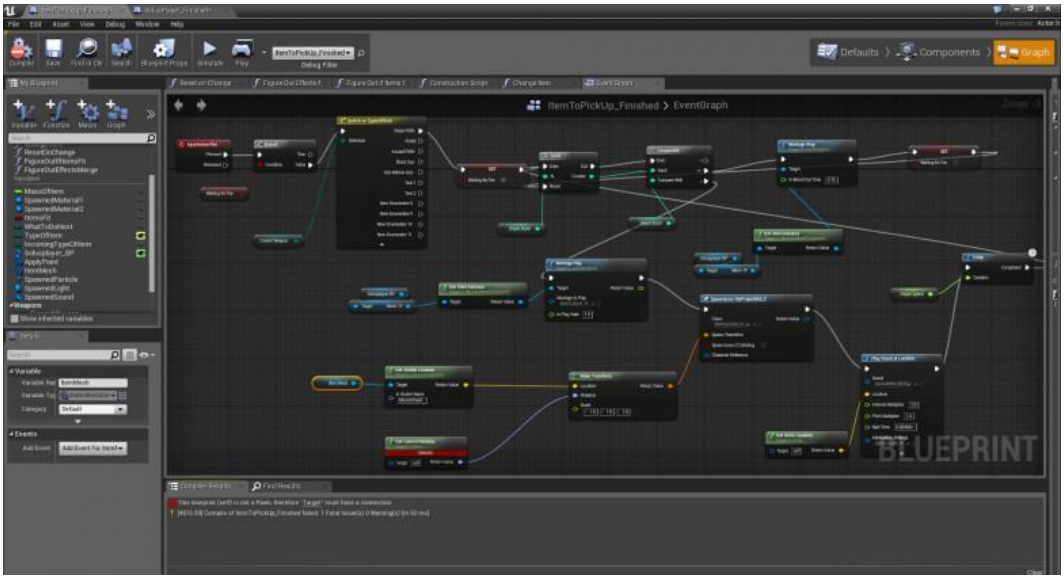

Рисунок 3.7 – Система візуального програмування Blueprints

Це дозволило їм створювати інтерактивні елементи у своєму світі без використання коду. Раніше потрібен був програміст, але тепер це міг зробити будь-хто, маючи відповідні ресурси. Також Unreal Engine 4 видаляє скрипт Unreal, який використовувався в попередніх поколіннях і починає використовувати мову C++. Причиною цього було те, що список речей, які потрібно було додати, ставав все більшим і більшим.

3.2 Обгрунтування вибору ігрового рушія

Вибір ігрового рушія є фундаментальним рішенням у процесі розробки гри, адже від обраної технології залежать майже всі аспекти кінцевого продукту. Він впливає на ефективність розробки, можливості масштабування, якість кінцевого продукту та навіть на маркетингові перспективи. Оскільки як можна було побачити на прикладі ситуації з Unity Technologies деякі рішення керуючого складу компаній можуть значно збільшити витрати студій на використання ігрового рушія.

Для цього проєкту був вибраний ігровий рушій Unreal Engine 4 (UE4) від компанії Epic Games, що є одним із найбільш популярних ігрових рушіїв, який використовується як великими студіями, так і інді-розробниками.

Одна з причин чому Unreal Engine 4 є привабливим вибором для початківців є його обширна та детальна документація, яка охоплює більшість аспектів роботи з рушієм. Його API Reference – це повний довідник по API для програмування на C++, який дозволяє з легкістю шукати потрібну інформацію з поясненнями, які були написані розробниками рушія. Велика кількість офіційних та сторонніх відеоуроків, які демонструють різні аспекти роботи з рушієм, від базових до просунутих технік. Незважаючи на величезну кількість інформації, яка доступна через документацію, деякі специфічні деталі, особливо ті, що стосуються систем низького рівня, можуть бути недостатньо висвітлені. Проте, завдяки відкритому коду рушія, завжди залишається можливість глибше зануритися у роботу рушія і самостійно розібратися з будь-якими питаннями.

Відкриття коду рушія було значущим кроком у розвитку індустрії розробки ігор, оскільки надало розробникам доступ до одного з найпотужніших ігрових двигунів, який до того часу був доступний лише за ліцензійними угодами. І відкритий код є однією з його найзначніших переваг. Тепер за бажанням можна вносити зміни в рушій для задоволення специфічних вимог власного проєкту. Відкриття коду, дає можливість компанії Epic Games використовувати користувачів продукту як розробників, адже через платформу GitHub вони можуть пропонувати свої зміни до коду рушія, що прискорює його розвиток.

Відкритий код також дозволяє уникнути проблем, пов'язаних з обмеженнями документації, оскільки розробники можуть самостійно досліджувати та адаптувати рушій.

Архітектура API рушія побудована на концепціях акторів (Actors) і компонентів (Components), що робить її гнучкою та легкою для розуміння і по суті представляє собою ECS (Entity Component System).

Актори є базовими ігровими об'єктами в Unreal Engine 4. Вони представляють будь-які об'єкти в грі, які можуть бути розміщені на сцені. Актори можуть мати різні функції, від персонажів і ворогів до предметів оточення та світлових джерел, все залежить від того, які компоненти вони мають.

Компоненти — це модульні частини, які можна додавати до акторів для розширення їх функціональності. Компоненти можуть бути візуальними (наприклад, сітки, матеріали), фізичними (колізії, фізичні властивості) та логічними (поведінка, числові властивості).

Архітектура компонентів дозволяє створювати складні об'єкти шляхом комбінування різних компонентів, що робить процес розробки більш гнучким і модульним. Це також спрощує тестування та налагодження, оскільки окремі компоненти можуть бути ізольовані та перевірені незалежно. Також така модульна структура полегшує подальше розширення системи, оскільки кожен компонент є незалежним і може поступово набувати нового функціоналу не спричиняючи при цьому зміни коду в інших місцях.

Проте Unreal Engine не є єдиним безкоштовним рушієм у світі, але на мою думку є кращим з них. Для порівняння Unity є одним із найпопулярніших ігрових рушіїв, який широко використовується для розробки ігор різних жанрів. Однак, він має кілька суттєвих відмінностей від UE4. Перша з яких - це закритий код. Вихідний код Unity недоступний для розробників, за винятком користувачів, які підписалися на найдорожчий тариф. Це обмежує можливості для модифікації рушія та глибокого розуміння його роботи. А також, хоча Unity і має модульну архітектуру, деякі розробники вважають її менш гнучкою порівняно з UE4. Використання компонентів у Unity іноді може призводити до складнощів у масштабуванні та підтримці. Також є Godot — це відносно новий рушій з

відкритим кодом, який набирає популярності серед починаючих розробників завдяки своїй гнучкості та простоті використання. Проте, саме через те, що Unreal Engine існує вже багато років і має велику кількість інструментів, наповнену документацію та широке коло розробників. Godot ще не досяг такого рівня, що обмежує можливості у вирішенні завдань.

3.3 Модуль UnrealAutomationTool

UnrealAutomationTool (UAT) є інструментом для автоматизації різних процесів у розробці ігор з використанням Unreal Engine. Його головна мета полягає в забезпеченні безперебійного та ефективного процесу створення, тестування, збирання та розгортання ігор. UAT є невід'ємною частиною екосистеми Unreal Engine, надаючи розробникам можливість зосередитися на вирішенні завдань, мінімізуючи рутину і мінімізуючи фактор людських помилок у часто повторюваних завданнях.

Однією з ключових особливостей UnrealAutomationTool є його здатність виконувати складні сценарії автоматизації. Це включає збирання проекту з вихідного коду, компіляцію, обробку контенту та створення пакунків для різних платформ. UAT підтримує широкий спектр платформ, таких як Windows, macOS, Linux, iOS, Android, PlayStation, Xbox та інші, що дозволяє розробникам гнучко налаштовувати свої робочі процеси під конкретні потреби проекту.

UAT використовує потужний і гнучкий механізм конфігурації, заснований на XML-файлах та командному рядку. Це дозволяє детально налаштовувати процеси збирання і розгортання, включаючи специфічні параметри для кожної платформи.

Важливою функцією UAT є можливість автоматизованого тестування. Він дозволяє створювати і запускати тести для перевірки різних аспектів гри. Це дозволяє виявляти помилки на ранніх етапах розробки, що значно скорочує час і зусилля, необхідні для їх виправлення. Одним із прикладів використання UAT є автоматизація збірок для різних конфігурацій і платформ. Наприклад, можна

налаштувати UAT для автоматичного збирання нічних збірок, які включають останні зміни в коді та контенті проекту. Це дозволяє отримувати актуальні версії гри без необхідності вручну виконувати кожен крок процесу збирання.

UAT має можливість інтеграції з системами безперервної інтеграції та доставки (CI/CD), такими як Jenkins, TeamCity чи Azure DevOps. Це дозволяє налаштувати автоматичне виконання різних завдань, таких як збірка, тестування та розгортання нових версій гри при кожному внесенні змін до основного репозиторію. Такі системи забезпечують безперервний цикл розробки, зменшуючи час простою та підвищуючи продуктивність команди.

Не менш важливим аспектом є розширюваність UAT. Відкритий вихідний код, а також вбудовані інструмент рушія дозволяють створювати власні скрипти та модулі для автоматизації специфічних завдань, які не охоплюються стандартним функціоналом UAT. Це дозволяє адаптувати інструмент під унікальні потреби проекту, забезпечуючи максимальну ефективність і гнучкість у роботі.

3.4 Мова програмування C++

Unreal Engine пропонує розробникам дві основні мови для створення ігрової логіки: C++ та Blueprints. Кожна з цих мов має свої переваги та недоліки, і вибір між ними залежить від конкретних потреб проекту. Я ж для свого проєкту обрав C++.

C++ є основною мовою програмування для Unreal Engine. Основною перевагою C++ у контексті Unreal Engine є продуктивність, адже головним пріорітетом UE була і залишається швидкість. C++ забезпечує високу швидкість виконання коду, що особливо важливо для ігор, оскільки виконання коду відбувається в реальному часі кожен кадр. Проте швидкість це не основна переварага C++ тим більше, що

Blueprints — це візуальна система сценаріїв в Unreal Engine 4, яка дозволяє створювати ігрову логіку без написання коду. Blueprints надають розробникам

інтуїтивно зрозумілий інтерфейс для створення взаємодій, поведінок та інших ігрових механік. Основною особливістю Blueprints є швидке прототипування. Вони дозволяють швидко створювати прототипи ігрових механік та тестувати ідеї без необхідності писати код. Також система нодів у Blueprints робить процес створення логіки більш візуальним і зрозумілим, що особливо корисно для дизайнерів та художників, які не мають досвіду програмування, проте можуть мати досвід з системами візуального програмування у таких програмах як Blender або Houdini. Blueprints легко інтегруються з іншими інструментами Unreal Engine, що спрощує процес розробки та налагодження.

Проте незважаючи на численні переваги, Blueprints мають свої недоліки. Основною причиною відмови є заплутаність системи. З часом логіка, створена за допомогою Blueprints, може стати дуже заплутаною та важкочитабельною. Це особливо в контексті майбутнього розширення систем. А також складність дебагу, адже система відстеження помилок у Blueprints реалізована незручно. В той час при розробці на C++ доступні потужні інструменти вбудованого середовища для відладки коду. Для складних та великих проєктів використання Blueprints буде обмеженням, оскільки зростання кількості нодів і взаємодій ускладнює підтримку та розвиток проекту, що значно ускладнить можливість легкого розуміння, а це є неприпустим для мене. Тому в цьому проєкті Blueprints використовуються лише для налаштування візуальної складової. Наприклад, дані, які повинні бути регульованими в редакторі винесені в Blueprints. Це дозволяє легко налаштовувати параметри без необхідності вносити зміни безпосередньо в коді, що прискорює процес тестування та налагодження.

3.5 Середовище розробки

У процесі розробки гри важливо обрати ефективне середовище розробки, яке сприятиме продуктивності та зручності роботи. Незважаючи на те, що Visual Studio є стандартом індустрії розробки ігор і широко використовується багатьма професіоналами, було обрано JetBrains Rider. Це рішення було прийнято на основі

кількох ключових аргументів, які роблять Rider більш привабливим варіантом для розробки на Unreal Engine.

Однією з головних причин вибору JetBrains Rider є його висока продуктивність і швидкість взаємодії. Rider забезпечує більш швидкий запуск та завантаження проектів порівняно з Visual Studio. Крім того, Rider використовує інтелектуальні методи кешування, що дозволяє значно скоротити час індексації проєктів. Завдяки цьому можна швидше отримувати доступ до необхідних файлів та класів, що підвищує ефективність роботи над проектом.

JetBrains Rider пропонує потужні інструменти для рефакторингу та аналізу коду, які дозволяють швидко та легко здійснювати зміни. Інструменти аналізу коду допомагають виявляти потенційні проблеми та оптимізувати код, що сприяє підвищенню його якості та стабільності.

Інтелектуальна система підказок (IntelliSense) у Visual Studio працює повільно та не завжди надає точні результати. У випадку з JetBrains Rider, підказки є точними та швидкими, що дозволяє зосередитися на написанні коду, а не на пошуку необхідних методів чи класів. Rider пропонує зручну навігацію по коду, що дозволяє швидко переходити між файлами, класами та методами.

JetBrains Rider надає широкі можливості для налаштування середовища розробки відповідно до індивідуальних потреб розробника. Інтерфейс Rider є більш гнучким та адаптивним, що дозволяє налаштовувати його під свої вподобання та покращувати зручність роботи.

4. ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Структура та компоненти системи

Структура проєкту побудована навколо ECS системи рушія Unreal Engine. Базовим класом є класс персонажа BaseCharacter, який є батьківським для класів PlayerCharacter і AICharacter. (Рисунок 4.1). Він реалізує поведінку, яка є спільною для всіх персонажів гри. Щодо компонентів, які зберігаються в ньому. HealthComponent – це компонент здоров'я, який є чисто логічним, він відповідає за зберігання поточного здоров'я та отримання шкоди від інших персонажів. WeaponComponent – контролює зброю персонажа, відповідає за її зміну, та постріли. Цей компонент зберігає в собі актора зброї, і напряму комунікує з ним.

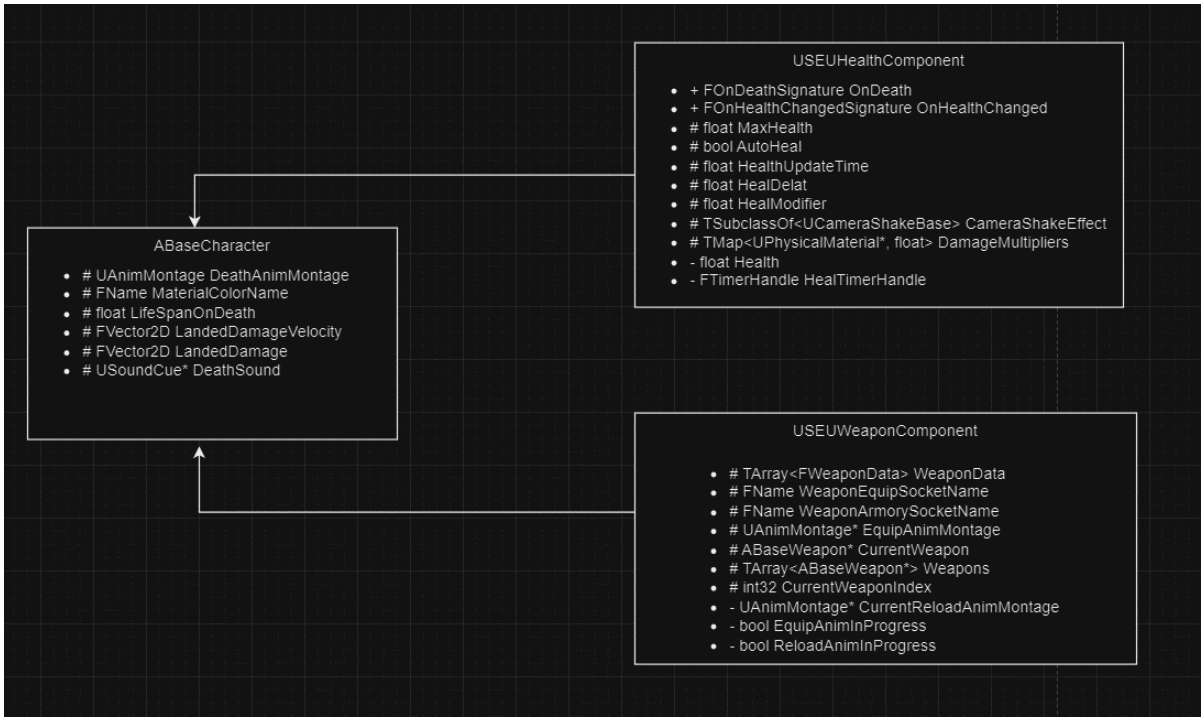

Рисунок 4.1 - Структура класу BaseCharacter

Його наслідник - PlayerCharacter представляє собою клас персонажа, який контролюється гравцем, окрім базових властивостей, він має ще декілька

компонентів, які потрібні для реалізації руху камери (Рисунок 4.2).

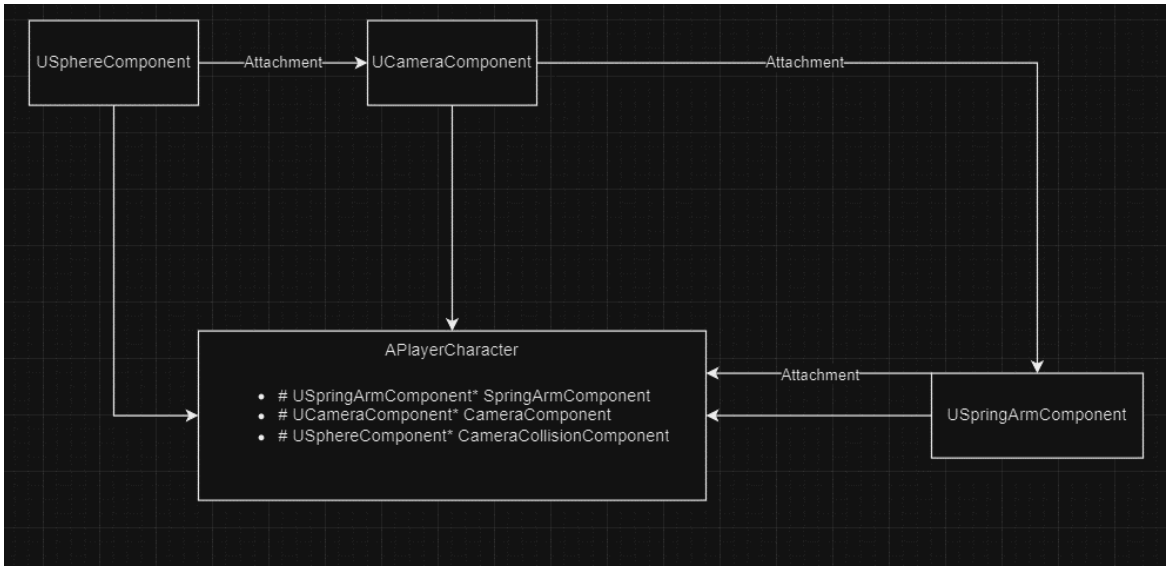

Рисунок 4.2 – Структура класу ігрового персонажа та схема з`єднань компонентів.

Щодо неігрових персонажів, то вони окрім базових властивостей, ще зберігають в собі компонент смуги здоров'я, яке відображається над ними, показуючи поточну кількість очків здоров'я. А також посилання на дерево поведінки, яке відповідає за рішення штучного інтелекту під час матчу. (Рисунок 4.3).

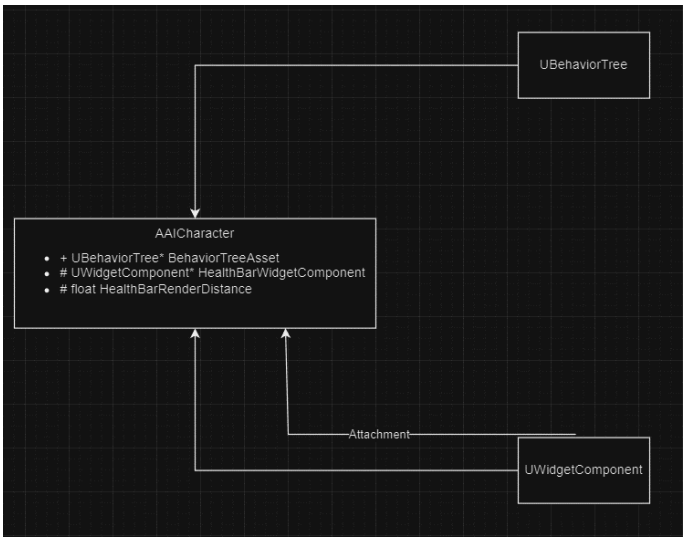

Рисунок 4.3 – Структура класу неігрового персонажа

Таке розділення на компоненти дозволяє досягти модульності та розширюваності. Також спрощує відладку систем адже ізолювати проблему стає легше, що є очевидним плюсом для загальної простоти системи.

4.2 Налаштування програми

З використанням Blueprints, також було створено нащадків для класів, для їх які мають властивості, що впливають на ігровий процес. Нащадки дозволяють візуально бачити значення, які налаштовуються, а також одразу відображати ефекти змін, що були внесені, у редакторі рушія. Приклад може слугувати класи персонажа та зброї (Рисунок 4.4 та Рисунок 4.5).

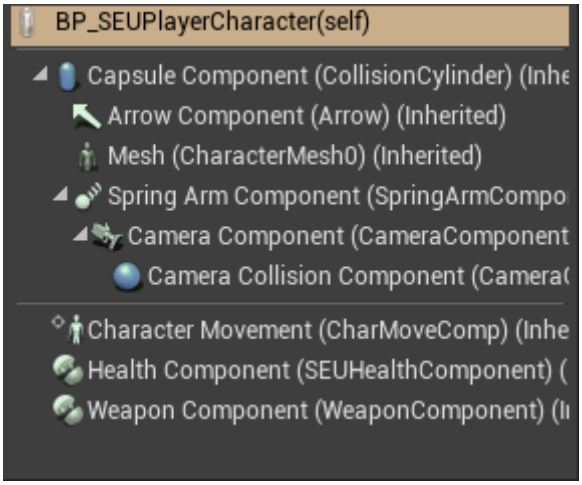

Рисунок 4.4 – Компоненти Blueprint класу ігрового персонажа в редакторі.

Рисунок 4.5 – Властивості Blueprint класу гвинтівки у редакторі.

Усі речі, які стосуються візуальних та геймплейних налаштувань винесені в редактор, для їх зручного редагування. На приклад, SpringArmComponent персонажа, який є штативом для камери, яка до нього прив’язана, має дуже гнучку систему налаштувань, в якій можна виставити затримку, при повороті камери, її відстань до персонажа, і все це можна легко налаштувати не чіпаючи код. Або ж структура FWeaponData, яка була згадана у розділі 2, також повністю сумісна з системою рефлексії рушія і через це, редагування властивостей зброї стає легшим. Це ще один пункт, який доводить, що розроблена система має низький поріг входу, та підтримувати проєкт досить легко навіть без навичок програмування або специфічних знань систем рушія.

4.3 Генерація документації за допомогою Doxygen

Генерація документації для проєкту на Unreal Engine (UE) з допомогою Doxygen є важливим кроком у забезпеченні зрозумілості та підтримуваності коду. Doxygen — це потужний інструмент для автоматичного створення документації з вихідного коду. Він підтримує різні мови програмування, включаючи C++ — основну мову Unreal Engine. Використання Doxygen дозволяє легко створювати і підтримувати актуальну документацію, що значно покращує якість коду та полегшує роботу.

Для початку роботи з Doxygen у проекті на Unreal Engine потрібно налаштувати конфігураційний файл Doxyfile, в якому визначаються параметри генерації документації. Цей файл включає в себе налаштування шляхів до вихідних файлів, формати вихідних документів (HTML, PDF, LaTeX та інші), а також різні опції для налаштування структури та вмісту документації.

Після налаштування Doxyfile, можна запустити процес генерації документації за допомогою Windows Batch файлу. Doxygen проаналізує вихідний код, згенерує документацію у вибраному форматі та створить структуровані файли, що містять всю необхідну інформацію. Документація, створена таким чином, буде доступна для перегляду через веб-браузер або інші засоби перегляду

документів (рисунок 4.1).

Рисунок 4.1 – Вигляд згенерованої документації

Використання Doxygen у проєкті сильно спрощує процес підтримки коду, оскільки актуальна документація завжди доступна, що дозволяє розбиратися у створених системах швидше.

4.4 Збірка проєкту. BuildCookRun

Збірка проєкту на Unreal Engine за допомогою інструменту Automation Tool BuildCookRun автоматизує кілька етапів створення гри. Цей інструмент забезпечує зручний спосіб інтеграції різних завдань, включаючи збирання, приготування контенту (cooking) та розгортання проекту на різних платформах. BuildCookRun спрощує ці процеси, об'єднуючи їх у єдину команду, що знижує ймовірність помилок, пов'язаних з ручним виконанням цих завдань.

Процес збірки починається зі створення версії проекту, яка включає компіляцію всіх необхідних файлів і створення виконавчих файлів для обраної платформи. Automation Tool використовує інформацію про проект, визначену в конфігураційних файлах, щоб зібрати всі необхідні залежності і ресурси. Після цього відбувається етап приготування контенту, під час якого всі ресурси проекту (моделі, текстури, звуки тощо) обробляються і готуються для кінцевої платформи.

Цей процес включає оптимізацію та конвертацію файлів у формат, що підходить для цільового пристрою, що забезпечує найкращу продуктивність гри.

Наступний етап — це розгортання проекту. BuildCookRun автоматично створює інсталяційні файли або пакети, готові для розгортання на обраній платформі, будь то ПК, консолі чи мобільні пристрої. Інструмент також підтримує створення debug-версій для тестування та виправлення помилок. Завдяки цьому, розробники можуть швидко і легко розгортати свої проекти на різних пристроях, забезпечуючи зручність та ефективність робочого процесу. Важливим аспектом використання BuildCookRun є його інтеграція з системами безперервної інтеграції (CI), такими як Jenkins або TeamCity. Це дозволяє автоматизувати процес збірки, приготування і розгортання проекту при кожному коміті змін до репозиторію. Така автоматизація значно підвищує продуктивність команди, дозволяючи розробникам зосередитися на вирішенні творчих завдань, а не на рутинних технічних аспектах.

4.5 Засоби автоматизації

Процес автоматизації збірки проєкту починається з налаштування Jenkins. Після встановлення Jenkins на сервері, що може бути локальним або виділеним, створюються відповідні джерела та завдання (jobs).

Наступним етапом є написання сценарію для збірки. У випадку Unreal Engine, це робиться за допомогою інструменту Unreal Automation Tool (UAT) і його команди BuildCookRun. Сценарій виглядає як командний файл, що запускає команду BuildCookRun з відповідними параметрами для цільової платформи. Після налаштування завдання збірки, Jenkins може автоматично виконувати його за розкладом, що забезпечує безперервну інтеграцію. Це дозволяє автоматично виявляти і виправляти помилки на ранніх стадіях розробки, значно скорочуючи час на виправлення і підвищуючи якість кінцевого продукту. Автоматизована збірка також дозволяє створювати різні версії продукту для тестування і розгортання.

Автоматизація генерації документації за допомогою Jenkins є наступним важливим аспектом, що доповнює процес розробки. Для мого проєкту генерація документації включає використання інструменту Doxygen, який автоматично створює документацію з коментарів у вихідному коді та назви класів, їх методів та полів. Цим забезпечується доступність технічної документації. Процес налаштування Jenkins для генерації документації схожий на налаштування для збірки. Спочатку створюється нове завдання. Сценарій для генерації є простим командним файлом, який запускає Doxygen з відповідними параметрами. Цей сценарій додається до завдання в Jenkins.

На мою думку, використання Jenkins для автоматизації збірки проекту та генерації документації має кілька ключових переваг. По-перше, це забезпечує стабільність і надійність процесу розробки, знижуючи ймовірність помилок. По-друге, це підвищує продуктивність, дозволяючи зосередитися на вирішенні завдань замість рутинних технічних аспектів. По-третє, це забезпечує актуальність і доступність документації, що є критично важливим для підтримки високої якості коду. Після налаштування сценаріїв їх можна запускати однією кнопкою (рисунок 4.2).

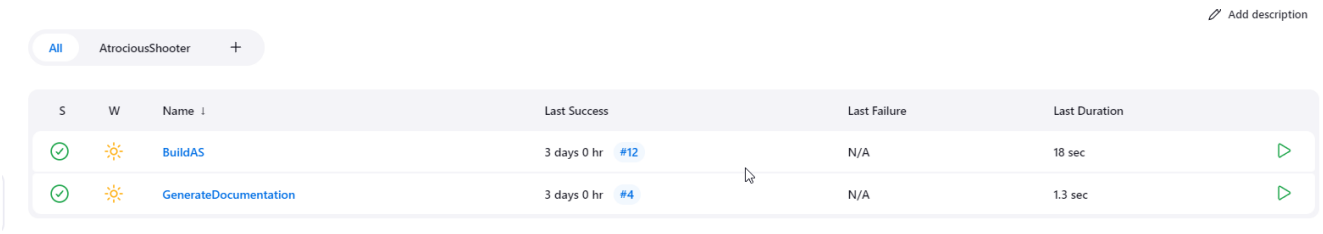

Рисунок 4.2 – Вигляд сценаріїв Jenkins

Таким чином, автоматизація збірки проекту та генерації документації за допомогою Jenkins є потужним інструментом, що дозволяє значно підвищити ефективність і якість процесу розробки. Використовуючи Jenkins, можна автоматизувати рутинні завдання, забезпечуючи безперервну інтеграцію і доставку, а також актуальність і доступність документації. Така оптимізація робочих процесів зводить до мінімуму складність інтеграції та розгортки проєкту.

Також автоматизація спрощує майбутню інтеграцію з API різних платформ дистрибуції, тобто за невеликого розширення, система дозволить збирати та доставляти проєкт в один клік миші.

5. РОБОТА КОРИСТУВАЧА З ПРОГРАМОЮ

5.1 Системні вимоги та інсталяція

Для успішного запуску проєкту достатньо після завантаження архіву запустити виконуваний файл, а також мишка та клавіатура, геймпади гра не підтримує. Для використання проєкту у редакторі потрібно мати ПК, який буде відповідати мінімальним системним вимогам рекомендованим компанією Epic Games для рушія, а саме. Процесор повинен мати 4 ядра та тактову частоту вище 2.5 Ггц. Мінімум 8 гігабайт оперативної пам’яті, а також встановлений DirectX 11 або 12.

5.2 Робочі сценарії

Взаємодія користувача починається з запуску гри через виконуваний файл. Після цього відкривається головне меню гри, якому представлені всі ігрові рівні (Рисунок 5.1).

Рисунок 5.1 – Головне меню гри

Коли користувач обирає рівень починається завантаження карти та відображається завантажувальний екран (Рисунок 5.2).

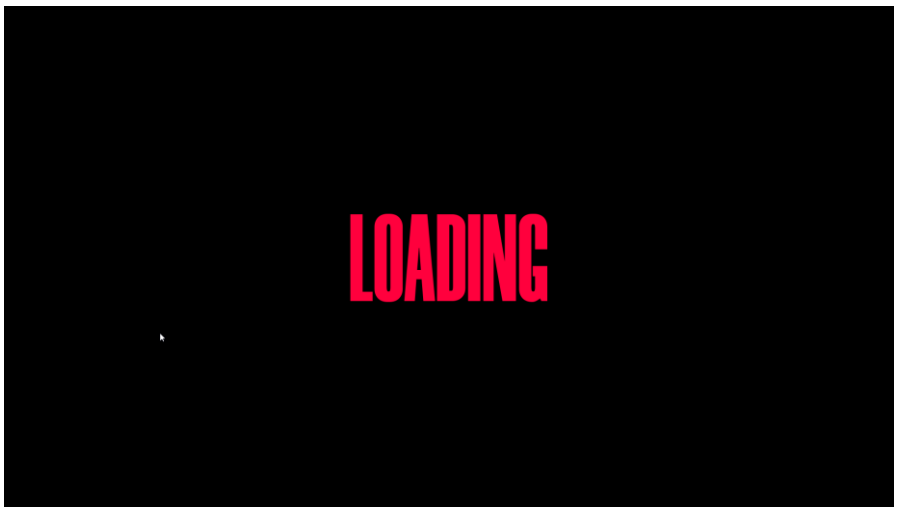

Рисунок 5.2 – Завантажувальний екран

Як тільки завантаження закінчиться починається сама гра і з’явиться інтерфейс HUD (Heads-up Display), на якому відображається поточна інформація, наприклад кількість здоров’я гравця та кількість його патронів (Рисунок 5.3).

Рисунок 5.3 – Heads-up Display

Під час гри відбувається бій, у якому персонажі можуть підбирати пікапа, аби поповнювати запас здоров’я або патронів. Також гравець може помирати, тоді

він побачить перед собою вікно смерті, та таймер зі зворотнім відліком до його повернення у гру (Рисунок 5.4).

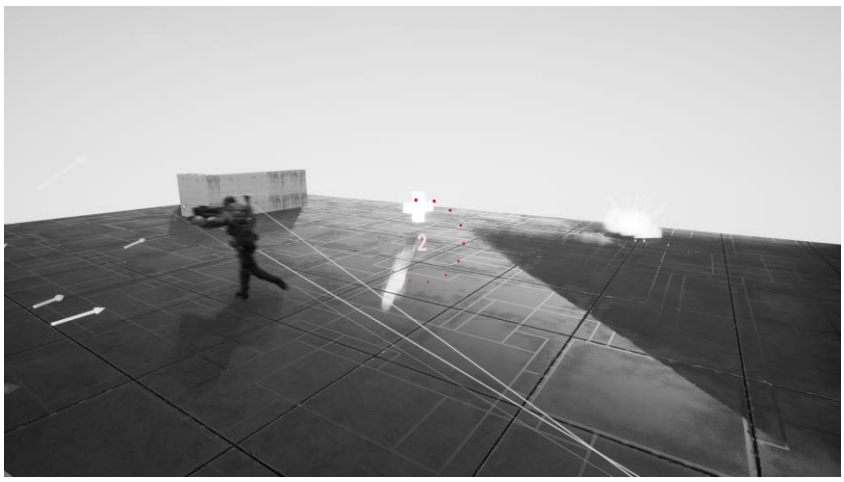

Рисунок 5.4 – Вікно смерті

За бажанням, гравець може поставити на паузу, тоді відкриється меню паузи, через яке можна або продовжити гру, або вийти у головне меню, або ж взагалі вийти з гри (Рисунок 5.5).

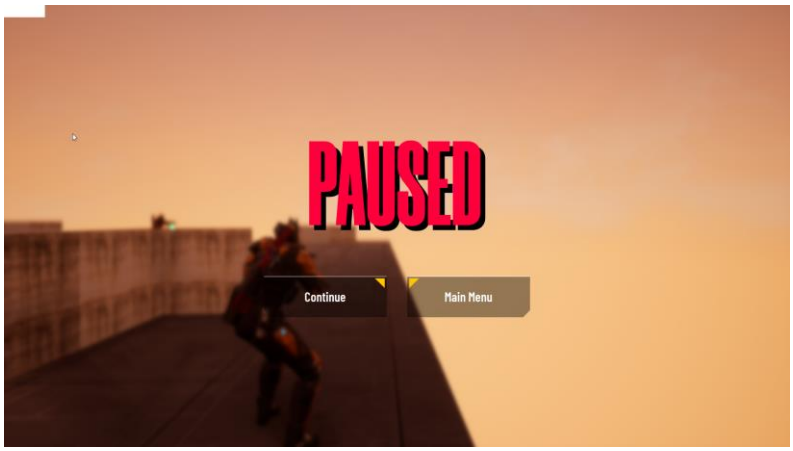

Рисунок 5.5 – Вікно паузи

Якщо є бажання, то за допомогою кнопки “M” можна повністю вимкнути звукове супроводження гри. Коли матч закінчується гравець бачить перед собою таблицю з фінальними результатами, тобто кількістю вбивств та смертей кожного

персонажа, з якого він може або повторити гру, або ж вийти у головне меню. (Рисунок 5.6)

Рисунок 5.6 – Вікно закінчення гри

Інтерфейс користувача є достатньо простим і ненавантаженим. Це є перевагою адже перевантажений та заплутаний інтерфейс може відлякувати потенціальних гравців.

5.3 Можливі покращення системи

Розробка гри — це процес, який постійно еволюціонує. Після завершення основної роботи над проектом завжди існують можливості для вдосконалення та розширення функціональності гри.

Перше з покращень, яке можливо імплементувати відносно легко – це інтеграція підтримки геймпада. Це є одним з важливих аспектів для покращення користувацького досвіду є додав, адже багато гравців надають перевагу використанню конролера для більшої зручності та точності під час гри. Комфорт гравця є важливим, адже гра в першу чергу для гравців, а для багатьох з них геймпад є більш зручним інструментом порівняно з клавіатурою та мишею, що підвищує задоволення від гри. З цього випливає ширше охоплення аудиторії. Також, підтримка геймпада робить гру доступнішою для гравців, які звикли до

консолей.

Наступне це додавання нового контенту, такого як нові рівні та нові типи зброї. Різноманітність арсеналу зброї є ключовим елементом для урізноманітнення геймплею. Додавання нових видів зброї з різними характеристиками та механіками може значно підвищити динаміку гри. Наприклад, снайперські гвинтівки з високою точністю та великою дистанцією пострілу. Автоматичні пістолети, швидкострільні, проте з меншою шкодою на дальніх дистанціях.

Достатньо складним процесом буде впровадження мультиплеєрного режиму. Проте це одне з найскладніших, але у той самий час і найважливіших покращень, яке може значно розширити функціональність гри та збільшити її популярність. Це привнесе у гру соціальну взаємодію, адже можливість грати з друзями або іншими гравцями по всьому світу додає соціальний аспект до гри. Це дасть можливість запровадити нові ігрові режими: командні бої, дуелі, кооперативні місії та інші можуть значно урізноманітнити геймплей. В той самий час, завдяки більшій іммерсивності збільшується життєвий цикл гри. Мультиплеєр підтримує постійний інтерес з боку гравців, бо як ніщо інше додає змагальний аспект у гру. Для реалізації мультиплеєра необхідно врахувати кілька важливих аспектів такі як серверна інфраструктура. Балансування гри під мультиплеєр, це зовсім інша задача на відміну від одиночного режиму. Забезпечення рівних можливостей для всіх гравців, незалежно від їх рівня є обов'язковою, тому що без цього початківці будуть відчувати фрустрацію, а не задоволення, якщо вони не матимуть шансів впоратися із суперником. Захист від нечесної гри, адже як тільки з'являється можливість бути краще ніж інші, почнуть з'являтися люди, які захочуть зробити це використовуючи вразливості гри та сторонні програми.

Покращення системи стрільби може суттєво вплинути на загальний ігровий досвід. Наприклад, варто звернути увагу на систему стрільби в CS:GO, де вона не є випадковою, а базується на шаблонах. Очевидною перевагою шаблонної системи стрільби є передбачуваність, а якщо система передбачувана, значить з досвідом та часом її можна повністю засвоїти, це буде тримати гравців.

Користувачі, які витрачають більше часу та сил на вивчення шаблонів стрільби, отримують винагороду у вигляді кращої точності та ефективності, а від цього більше перемог у змагальних режимах.

Покращення системи автоматизації збірки проєкту на Unreal Engine через Jenkins може значно підвищити ефективність і надійність процесу розробки. Одним із ключових аспектів такого покращення є переведення з локального на виділений сервер. Використання виділеного комп'ютера для Jenkins забезпечує ряд переваг, що сприяють оптимізації робочого процесу.

Найочевидніше поліпшення полягає у збільшенні продуктивності. Локальний хост, який одночасно використовується для розробки та автоматизації збірки, може бути перевантажений, що впливає на швидкість і стабільність як розробницьких завдань, так і процесів збірки. Виділений сервер дозволяє розділити ці обов'язки, що забезпечує більш стабільне і швидке виконання збірок, оскільки всі ресурси машини будуть зосереджені на виконанні завдань Jenkins. Крім того, використання виділеного сервера підвищує надійність і безпеку системи. На локальному хості існує ризик випадкових перезавантажень, оновлень або інших подій, які можуть перервати процес автоматизації. Виділений сервер може бути налаштований для мінімізації таких ризиків, забезпечуючи безперервну роботу Jenkins без збоїв.

ВИСНОВКИ

У ході дипломної роботи було створено однокористувацьку гру жанру TPS з використанням рушія Unreal Engine. Основна мета проекту полягала в розробці захоплюючого ігрового досвіду, який у той самий час буде достатньо складним для гравця, аби втримувати його.

Під час розробки було проведено аналіз існуючих ігор цього жанру, що дозволило виділити ключові механіки та особливості, необхідні для створення якісного продукту. На основі цього аналізу було розроблено концептуальний дизайн гри, включаючи ігрові механіки. Особлива увага приділялася забезпеченню збалансованості ігрового процесу та елементів, що сприяють утриманню уваги гравця, а також засобам автоматизації процесів розробки.

Реалізація гри в середовищі Unreal Engine дозволила використати сучасні технології та інструменти для створення та підтримки проєкту, а також досягти високого рівня продуктивності, що важливо для гравців, які мають ПК, що технічно застаріли. Було успішно інтегровано різноманітні механіки, такі як система стрільби, штучний інтелект ворогів, плавні управління персонажем та анімації.

Важливою особливістю реалізованих систем є їхня модульність і розширюваність, що досягається завдяки використанню компонентів. Це дозволяє легко адаптувати та доповнювати гру новими функціональними можливостями в майбутньому.

Крім того, було розроблено засоби автоматизації для збірки та генерації документації проєкту з використанням Jenkins. Це значно спростило процес розробки, дозволяючи автоматизувати рутинні задачі, забезпечуючи швидку та надійну збірку гри, а також актуальність технічної документації.

Таким чином, поставлені задачі були успішно виконані, і результатом роботи стала повноцінна гра жанру TPS.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Robert N. Game Programming Patterns. Genever Benning, 2014. 354 с.
2. Eric L. Foundations of Game Engine Development, Volume 1: Mathematics 1st Edition. Terathon Software LLC, 2016. 200 с.
3. Scott M. Effective Modern C++ 42 Specific Ways to Improve Your Use of C++11 and C++14. O'Reilly Media, 2014. 354 с.
4. Stuart B., Tom O. Game Development Patterns with Unreal Engine 5. Packt Publishing, 2024. 254 с.
5. Sufian bin U. Mastering Unreal Engine: A Beginner's Guide Taylor & Francis 2022. 453 с.
6. Marcos R., Brenden S., Luis C. Blueprints Visual Scripting for Unreal Engine 5. Pack Publishing 2022. 568 с.
7. Bjarne S. The C++ Programming Language 4th Edition. Addison-Wesley, 2013. 1366 с.
8. Bjarne S. Tour of C++ 2nd Editiron. Addison-Wesley, 2018. 254 с.
9. Joanna L. Learning Unreal Engine Game Development, 2016. 249 с.
10. The Seven Stages of Game Development. GameMaker. URL: https://gamemaker.io/en/blog/stages-of-game-development
11. Third person shooters. Microsoft. URL: https://news.xbox.com/en-us/2015/10/09/games-know-your-genres-third-person-shooters/
12. Game Development Process. Perforce. URL: https://www.perforce.com/resources/game-development-process
13. Gaming is Booming. WeForum. URL: https://www.weforum.org/agenda/2022/07/gaming-pandemic-lockdowns-pwc-growth/

14. How a Pricing Change Led to a Revolt by Unity's Video Game Developers. The New York Times. URL: https://www.nytimes.com/2023/10/02/technology/how-a-pricing-change-led-to-a-revolt-by-unitys-video-game-developers.html
15. Blueprints Overview. Epic Games. URL: https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/Overview/
16. Know Your Genres: Third-Person Shooters URL: https://news.xbox.com/en-us/2015/10/09/games-know-your-genres-third-person-shooters/
17. https://books.google.com.ua/books?id=NUFIBQAAQBAJ&pg=PA114&redir_esc=y#v=onepage&q&f=false
18. https://externlabs.com/blogs/unreal-engine-and-its-evolution/
19. https://www.pcmag.com/news/25-years-later-the-history-of-unreal-and-an-epic-dynasty
20. Gameplay Architecture. Epic Games. URL: https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/GameplayArchitecture/Classes/
21. Artificial Intelligence URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/artificial-intelligence-in-unreal-engine
22. Components. Epic Games. URL: https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/ProgrammingWithCPP/UnrealArchitecture/Actors/Components/
23. Unreal Automation Tool URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-automation-tool-for-unreal-engine
24. UE4 Unreal Automation Tool URL: https://github.com/botman99/ue4-unreal-automation-tool