МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ УПРАВЛІННЯ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____Литвиненко О.Є.
"_____"_____ 2022 р.

ДИПЛОМНИЙ ПРОЕКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ЗА НАПРЯМОМ ПІДГОТОВКИ 123 «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ»

Тема: «Інтерактивний додаток для здійснення ігрових маніпуляцій на основі Unreal Engine»

Виконавець: Долбарев Є.А.
(прізвище, ім'я, по-батькові)

Керівник: д.т.н., проф. Мислович М.В.
(науковий ступінь, вчене звання, прізвище, ім'я, по-батькові)

Нормоконтролер: _____ Тупота Є.В.
(підпис)

КИЇВ 2022

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, ком'ютерної та програмної інженерії
Кафедра *комп'ютеризованих систем управління*.
Напрям (спеціальність) 123 *«Комп'ютера інженерія»*.
(шифр, найменування)

ЗАТВЕРДЖУЮ
Завідувач кафедри
О.Є. Литвиненко
«_____» _____ 2022 р.

ЗАВДАННЯ
на виконання дипломного проекту
Долбарева Єгора Андрійовича
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема дипломного проекту «**Інтерактивний додаток для здійснення ігрових маніпуляцій на основі Unreal Engine**» затверджена наказом ректора від « 15 » лютого 2022 р. № 251/ст

2. Термін виконання проекту: з 16 травня 2022 р. по 19 червня 2022 р.

3. Вихідні дані до проекту: рушій для створення інтерактивних додатків, технологія створення додатків, структура роботи рушія

4. Зміст пояснювальної записки:

1) Рушій для розробки ігор - Unreal Engine

2) Процес розробки інтерактивного додатку для здійснення ігрових маніпуляцій на основі Unreal Engine 5

3)Демонстрація створеного додатку та тестування

5. Перелік обов'язкового графічного (ілюстративного) матеріалу:

1) Процес запуску рушія(схема алгоритму)

2) Візуалізація трикутників віртуалізованої геометрії Nanite

3) Структура ігрового режиму додатка

4) Приклад роботи інтерактивного додатку

6. Календарний план-графік

№ з/п	Завдання	Термін виконання	Підпис керівника
1.	Аналіз джерел за темою дослідження. Розроблення плану дипломного проекту	16.05.2022-18.05.2022	
2.	Затвердження плану дипломного проекту та проведення консультацій з науковим керівником, щодо наповнення пояснювальної записки	19.05.2022-22.05.2022	
3.	Дослідження існуючих проектів на рушії та технології розробки	23.05.2022-26.05.2022	
4.	Розробка ідеї інтерактивного додатку	27.05.2022-31.05.2022	
5.	Реалізація проекту інтерактивного додатку у рушії	01.06.2022-04.06.2022	
6.	Написання пояснювальної записки	05.06.2022-06.06.2022	
7.	Підготовка демонстраційного матеріалу	07.06.2022 09.06.2022	

7. Дата видачі завдання: «_6_» _лютого_ 2022 р.

Керівник дипломного проекту _____ д.т.н., проф. Мислович М.В.
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Долбарев Є.А
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту «**Інтерактивний додаток для здійснення ігрових маніпуляцій на основі Unreal Engine**»: 80 с., 59 рис., 2 табл., 6 літературних джерел.

ІНТЕРАКТИВНИЙ ДОДАТОК, СИСТЕМА СТВОРЕННЯ ІНТЕРАКТИВНИХ ДОДАТКІВ, РУШІЙ UNREAL ENGINE.

Об’єктом дослідження даного дипломного проекту є процес розробки інтерактивного додатку на базі рушія Unreal Engine 5.

Предметом дослідження є рушій Unreal Engine 5.

Метою даного дипломного проекту є розробка інтерактивного додатку для здійснення ігрових маніпуляцій.

Методи дослідження – пошук інформації про методи створення додатків у джерелах, порівняння існуючих проектів, дослідження функціоналу рушія, методи об’єктно-орієнтованого прогшрамування.

Здійснено огляд теорії структури роботи рушія; здійснено порівняльний аналіз функціональних можливостей вже створених проектів; проаналізовано структуру створення інтерактивних додатків; здійснено огляд принципів тестування інтерактивних додатків; реалізовано інтерактивний додаток за допомогою рушія Unreal Engine 5.

Матеріали дипломного проекту рекомендується використовувати при проведенні досліджень з теорії створення додатків, у навчальному процесі фахівців з системного програмування, а також у сферах, пов’язаних зі створенням інтерактивних додатків.

Прогнозні припущення про розвиток об’єкту та предмету дослідження – застосування в якості навчальної платформи, а також як додаток у сфері розваг.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

VFX – візуальні ефекти

LOD – рівні деталізації об’єктів в залежності від відстані до спостерігача

SSAO – технологія розсіянного непрямого освітлення поверхонь у реальному часі

RTX – лінійка відеокарт від компанії NVIDIA, яка відрізняється технологією відслідковування променів у реальному часі

Low poly – означає низькополігональную модель об’єкта

2D – система кординат рівня, при якій на рівні використовуються дві осі, X та Y

3D – система кординат рівня, при якій на рівні використовуються три осі, X, Y та Z

ВСТУП

Актуальність. У наш час розвиток технологій досягнув того, що якщо раніше для якщо раніше для дозвілу читали газети або книжки, то сьогодні використовують комп'ютер або смартфон. Обчислювальна машина зараз об'єднує речі, які раніше потрібно було придбати окремо, це камера, блокнот, газета, курси університету, відеопрогравач, аудіопрогравач, телевізор та ін. Таким чином, розробники вирішили використати потужності комп'ютера для того, щоб запускати з нього ігри. Однією з перших комп'ютерних ігор була Tennis for Two, яка була створена у 1958 році фізиком Уільямом Хігінботамом у Брукхейвенській національній лабораторії. Це був м'яч на тенісному корті, який рухався від одного гравця до іншого, це було на осцилографі. З тих пір технології створення інтерактивних додатків лише удосконалювались, створювались комп'ютери, на яких можна було написати код, після виконання якого запускалась програма. Суспільності сподобалась ідея ігор у комп'ютерах, тому створювались команди розробників, які їх розроблювали. Спочатку створювались 2D ігри, а теперішні технології дозволяють створювати і 3D ігри. Якщо раніше були прості ігри по типу Маріо, то потім вимоги до додатків стали набагато складніше, а амбіції розробників все більші, тому було принято рішення створити рушій для додатків, в якому будуть усі інструменти для створення ігор нового покоління. Таким рушієм був Unreal Engine 1996 року, зроблений розробником Epic Games. С тих пір, роль комп'ютерних ігор тільки збільшувалась у житті людей, а у світі з'явилась індустрія ігор, тому створення інтерактивного додатку сьогодні буде як ніколи актуальне.

Об'єктом дослідження даного дипломного проекту є процес розробки інтерактивного додатку на базі рушія Unreal Engine 5.

Предметом дослідження є рушій Unreal Engine 5.

Метою даного дипломного проекту є розробка інтерактивного додатку для здійснення ігрових маніпуляцій, спрямований на удосконалення функцій керування головним героєм.

Методи дослідження дипломного проекту: дослідження джерел інформації, порівняльний аналіз, структура рушія, створення інтерактивних додатків. За допомогою порівняльного аналізу було досліджено сучасні інтерактивні додатки, оглянуто їх переваги та недоліки, їх можливості. З огляду на існуючі проекти був виділений основний функціонал майбутнього додатку. За допомогою рушія Unreal Engine 5 було реалізовано інтерактивний 3D додаток з можливістю керування головного героя.

Новизна дипломного проекту полягає у застосуванні найновішої версії рушія Unreal Engine 5 для створення високого рівня гри зі складним освітленням та просунутою технологією режиму гри.

Практичне значення отриманих результатів. У дипломному проекті створено інтерактивний 3D додаток, в якому можна керувати головним героєм, взаємодіяти з іншими об’єктами та виконувати умови гри для перемоги гравця.

Особистий внесок випускника. Всі результати, представлені у дипломній роботі, отримані випускником особисто. Особливої уваги заслуговує проектування ігрової концепції шутера.

Практичні значення отриманих результатів в дипломному проекті дають змогу використовувати отримані дані при створенні нових кращих додатків, можливість отримати знання структури сучасних рушіїв для створення своїх та використання у відповідних сферах

Апробація отриманих результатів. Міжнародна науково-технічна конференція «Інтелектуальні технології лінгвістичного аналізу:», 20 – 21 жовтня 2020 р., м. Київ

Публікації. Долбарев Є.А. Носовська В.І. Метод остаточного навчання глибоких нейронних мереж/ Долбарев Є.А. Носовська В.І. // Інтелектуальні

технології лінгвістичного аналізу: міжнародна науково-технічна конференція, 20 – 21 жовтня 2020 р.: тези доп. – К., 2020. – С. 19.

Прогнозні припущення про розвиток об’єкту та предмету дослідження – застосування в якості платформи для навчання та есперементів нових методів та розвиток проекту до повноцінної гри для дозвілля.

Інтерактивний додаток для здійснення ігрових маніпуляцій на основі Unreal Engine

1. Рушій для розробки ігор - Unreal Engine

1.1. Загальні поняття та визначення

Unreal Engine — ігровий рушій, розроблюваний та підтримуваний компаніей Epic Games. Першою ігрою на цьому рушії був шутер від першого обличчя Unreal, випущений у 1998 році. Хоча рушій був призначений для розробки шутерів від першого обличчя, його наступні версії успішно використовувались в іграх різних жанрів, в тому числі стелс-ігри, файтинги та массові багатокористувацькі рольові онлайн ігри.

Для спрощення портування рушій використовує модульну систему залежних компонентів; підтримує різні системи рендерингу (Direct3D, OpenGL, Pixomatic; у ранніх версіях: Glide, S3, PowerVR), відтворення звуку (EAX, OpenAL, DirectSound3D; раніше: A3D), засоби голосового відтворення тексту, розпізнавання мови, модулі для роботи з мережею та підтримки різних пристроїв введення.

Для гри в мережі підтримуються технології Windows Live, Xbox Live, GameSpy та інші, включаючи до 64 гравців (клієнтів) одночасно. Таким чином, рушій адаптували і для застосування в іграх жанру MMORPG (один із прикладів: Lineage II).

Написаний на мові C++ рушій допомагає створювати ігри для більшості платформ: Microsoft Windows, Linux, Mac OS и Mac OS X; консолей Xbox, Xbox 360, Xbox One, PlayStation 2, PlayStation 3, PlayStation 4, PSP, PS Vita, Wii, Dreamcast, GameCube та ін., а також на різних портативних пристроях, наприклад пристроях Apple (iPad, iPhone), під управлінням системи IOS та інших. Вперше робота з IOS була представлена в 2009 році, в 2010 продемонстрована робота двіжка на пристрої з системою webOS.

Переваги Unreal Engine:

- Якість графіки Unreal Engine дивовижна і здійсненна. Юзабіліті програми дуже затребувана.
- Інтерфейс користувача Unreal Engine постійно оновлюється з використанням новітніх інструментів та опцій
- Він має прості коди та використовує вузли, так звані Blueprints. Ці вузли допомагають користувачам створювати відеоігри та інші високоякісні ігри без написання скриптів та кодів.

Для використання Unreal Engine потрібні знання ігрового дизайну, основи розробки ігор, а також знання мови програмування C++.

Функціонал Unreal Engine:

- Фотореалістичний рендеринг у реальному часі
- Надійна багатокористувацька платформа
- VFX та моделювання частинок
- Просунутий ШІ
- Unreal Audio Engine
- Безмежна розширюваність

Розповсюджується рушій по безоплатній моделі, при цьому якщо дохід з проекту більше 1000000$, то виплачується 5% розробникам Unreal Engine. Раніше за двіжок потрібно було платити місячною підпискою.

Усі елементи представлені у вигляді об’єктів, які мають свої характеристики і класу, який визначає доступні їм характеристики. У свою чергу, кожний клас є дочірнім до класа object. Основні класа рушія такі:

Актор (actor) (жаргонне висловлювання серед розробників, цим словом пояснюють «об’єкт» або «суб’єкт», тобто той, хто діє) – батьківський клас, який містить усі «об’єкти», які мають відношення до ігрового процесу та мають свої просторові координати.

Пішак (pawn) – один з основних класів усіх акторів, якими може керувати гравець або штучний інтелект. Це клас не лише фізичної моделі актора, а й як він взаємодіє зі світом через колізії з іншими акторами та взаємодією між собою. Метод керування пішаком описаний у об'єкті Контролер. Контролер задає лише поведінку для штучного інтелекту, наприклад на якій відстані пішак помічає гравця, а характеристики по типу кількості здоров'я(характеристика, після якої пішак перестає існувати) закладаються у самому об'єкті окремо.

Всесвіт, рівень(world, game level) – об'єкт, в якому закладаються характеристики рівня гри, тобто силу тяжіння або туман та режим гри, для якого призначений рівень.

Увесь двійковий простір у рушії поділений на «заповнене» та «порожнє», це працює, як правило, з простими нерухомими елементами ігрового процесу(наприклад стіна). У «порожній» частині простору можуть вільно знаходитись усі об'єкти та «точка відображення» сцени. Можна помістити один об'єкт у інший, наприклад м'яч у підлогу, тобто помістити «заповнене» у «заповнене», але тоді не виключені помилки при прорахуванні фізичної взаємодії об'єктів або помилки відображення сцени, якщо помістити «точку відображення» у «заповнене». Усі пішаки, які попадають у «заповнений» простір, одразу «помирають».

Поверхня (surface) є основним елементом двійкового дерева «заповненого» та «порожнього». Ці елементи створюються на межі цих просторів, а група поверхонь називається нодом (node, укр. вузол). Цей термін застосовується у контексті кількості нодів на сцені – node count. Кількість нодів прямо впливає на швидкодію додатка. Для пришвидшення швидкодії, рушій не оброблює нод, якщо він перекривається цілком іншим нодом або знаходиться за «точкою відображення». Ділення простору на групи нодів називається зонуванням.

Іноді для цього використовують портали, невидимі поверхні, якими власноруч ділять один великий нод на два поменше(Також з версії Unreal

Engine 3 була введена підтримка аддитивної геометрії, завдяки якій можна було відмовитись від цієї технології). Також можливе використання антипорталів.

Для опису «заповненого» та «порожнього» простору використовують інструмент під назвою браш (brush, укр. пензлик). Він заповнює простір замкнутими непересічними поверхнями, цей принцип називається конструктивною суцільною геометрією. Браш може як «аддитивним»(додавати «заповнений» простір у «порожній»), так і субстрактивною(видаляти простір у «заповненого»).

Браші поділяються на три основних типи:

- Суцільні (solid) – використовуються для додавання та видалення простору
  - Аддитивні (additive) – заповнюють «порожній» простір.
  - Субтсрактивні (substractive) – видаляють простір у «заповненого»
- Полу-суцільні (semi-solid) – не впливають напряму на двійкове дерево «заповненого» та «порожнього», лише на фізичну модель, тобто можуть створювати невидимі перешкоди або знижувати кількість полігонів та нодів.
- Порожні (non-solid) – тільки створюють поверхні, на впливають на двійкове дерево. Потрібні для створення об’ємів(volume), частини простору, де знаходяться інші властивості, на відміну від звичного рівня, наприклад інший рівень гравітації, туман, фізичні властивості. Об’єм має більший пріоритет, ніж рівень, тому властивості об’єма впливають на всіх акторів, які знаходяться всередині. З версії Unreal Engine 2 використовуються для створення водного простору(але не поверхні).

1.2. Функціональні можливості рушія для розробки ігор Unreal Engine 5

Цей зручний рушій дозволяє розробникам ігор втілювати у реалність всі їх амбіції в реальному часі з великою свободою дій та точністю ніж коли-небудь раніше.

Unreal Engine 5 має усі інструменти, щоб створювати дійсно великі світи для гравців або інших осіб, які можуть досліджувати світ через масштабуючий контент, а нові революційні технології Nanite та Lumen зможуть зробити світи набагато динамічнішими та реалістичнішими.

Нові інструменти для модельєрів з розширеним переліком функцій значно зменшують кількість ітерацій, пришвидчуючи творчий процес. Користувацький інтерфейс став більш інтуітивно зрозумілий та підведений до сучасних стандартів дизайну. Тепер можна швидко викликати браузер акторів і закріплювати вкладки на бічних панелях, за допомогою чого потрібні характеристики об'єкту знаходяться е швидше. За допомогою сервісу Quixel Bridge можна перетягувати об'єкти з бібліотеки Megascans прямо до сцени.

Тепер за допомогою технологій віртуалізованної системи геометрії мікрополігонів Nanite та Virtual Shadow Map можна створювати дійсно великі світи з неперевершенною деталізацією. Завдяки цим інноваційним технологіям, рушій може відрисовувати полігони об'єктів без видимих втрат точності у реальному часі. Ці інтелектуальні системи передають лише ті деталі, які можна побачити, позитивно впливаючи на швидкодію всього додатка.

Переваги технології:

- Збільшення складності геометрії на кілька порядків — більша кількість трикутників та об'єктів, що обробляються у режимі реального часу, ніж це було можливо раніше(Рис.1.1).
- Тепер бюджети на кадр більше не обмежені кількістю полігонів, викликами на відрисовку та використанням пам'яті для геометрії.
- Рівень Деталізації обробляються автоматично і більше не потребують ручного налаштування LOD'ів для окремо взятих мешей.
- Втрати практично повністю відсутні, особливо це стосується переходів між LOD'ами.

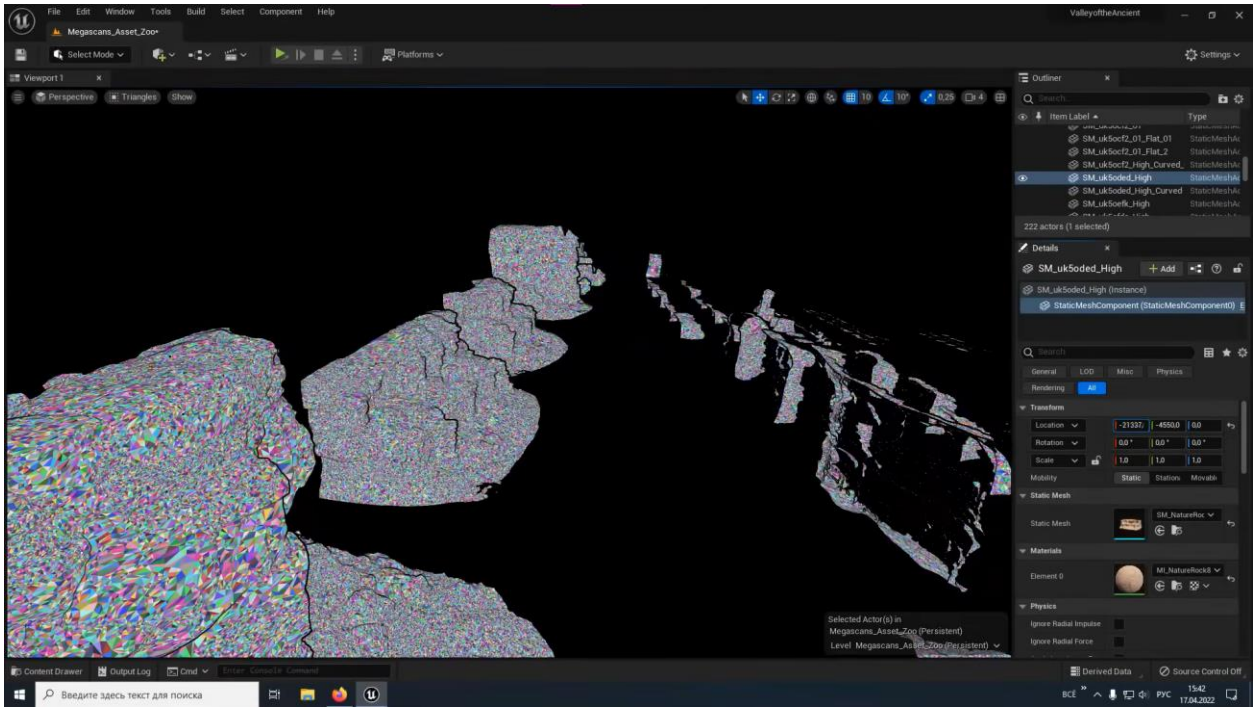

Рис.1.1 Вигляд тих самих трикутників. При зближенні об'єкта до наглядача, вони роздроблюються, а при віддаленні склеюватися, при цьому модель буде виглядати однаково деталізовано.

За своєю суттю, Nanite Mesh, як і раніше, є сіткою, що складається з трикутників, але з високим рівнем деталізації і високим рівнем стиснення, що застосовується до її даних. На додаток до всього, Nanite використовує абсолютно нову систему для надзвичайно ефективного рендерингу цього формату даних.

Процес створення контенту для системи Nanite нічим не відрізняється від створення традиційних моделей, за винятком того, що Nanite може обробляти на порядки більше трикутників і екземплярів, ніж це можливо для геометрії, що візуалізується традиційним способом. Перемістіть камеру досить близько, і Nanite відобразить всі оригінальні трикутники в їхньому вихідному вигляді, в якому вони були імпортовані в редактор.

Під час імпорту: меши аналізуються та розбиваються на ієрархічні кластери, що складаються з груп трикутників (clusters of triangle groups).

Під час рендерингу: на різних рівнях деталізації залежно від положення камери ці кластери змінюються (swapped) прямо на льоту, і ідеально без жодних швів з’єднуються з сусідніми кластерами, всередині одного і того ж об’єкта. Стрімінг (потокова передача) даних виконується за запитом, тому пам'яті повинні бути лише видимі деталі. Система Nanite виконується у своєму проході рендеринга (rendering pass), який повністю обходить традиційні виклики відрисовки (draw calls).

Система Nanite, як правило, має бути активована скрізь, де це можливо. Будь-які Статичні Меші для яких вона активована будуть рендеруватись швидше, а також займатимуть менше пам'яті та місця на диску.

Динамічне глобальне освітлення здійснюється технологією Lumen. Ця технологія повного динамічного освітлення допомагає непрямому освітленню адаптуватися у реальному часі, збільшуючи реалістичність сцени. В залежності від часу доби або інших подій(наприклад відкрите вікно, Lumen відтворює відповідні промені світла і відзеркалює від поверхні.

З Lumen більше не потрібно створювати UV-мапи, робити захват відзеркалювання поверхонь, і чекати на рендеринг усієї сцени, бо те, що можна побачити на сцені у реальному часі, те і буде у зарендереній сцені; те, що бачимо на картинці Unreal Editor, те і буде насправді. Lumen – нова система глобального освітлення та відображень у Unreal Engine 5. Вона створює повністю динамічне непряме освітлення з огляду на параметри геометрії, матеріалів та світла на сцені. Зазвичай глобальне освітлення вимагає багато обчислень, особливо в реальному часі, але Lumen робить це простіше.

Розтікання кольору(Рис.1.2) (Colour Bleeding). Оптичний ефект, при якому забарвлення найближчих поверхонь змінюється при відображенні світла від навколишніх об'єктів. Він майже непомітний, але з ним сцена виглядає реалістичнішою. Функція працює тільки тоді, коли об'єкт, від якого відскакує світло, перебуває у видимості камери. Якщо об'єкт зникає із зони видимості, відсвічування пропадає.

Рис.1.2 Ледь помітне розтікання кольору на колонні

М’які непрямі тіні(Рис.1.3) (Soft Indirect Shadows). Технологія створює розмиті тіні у місцях без спрямованого світла. Lumen повністю замінює SSAO (Screen Space Ambient Occlusion) у Unreal Engine і створює точні та реалістичні м'які тіні.

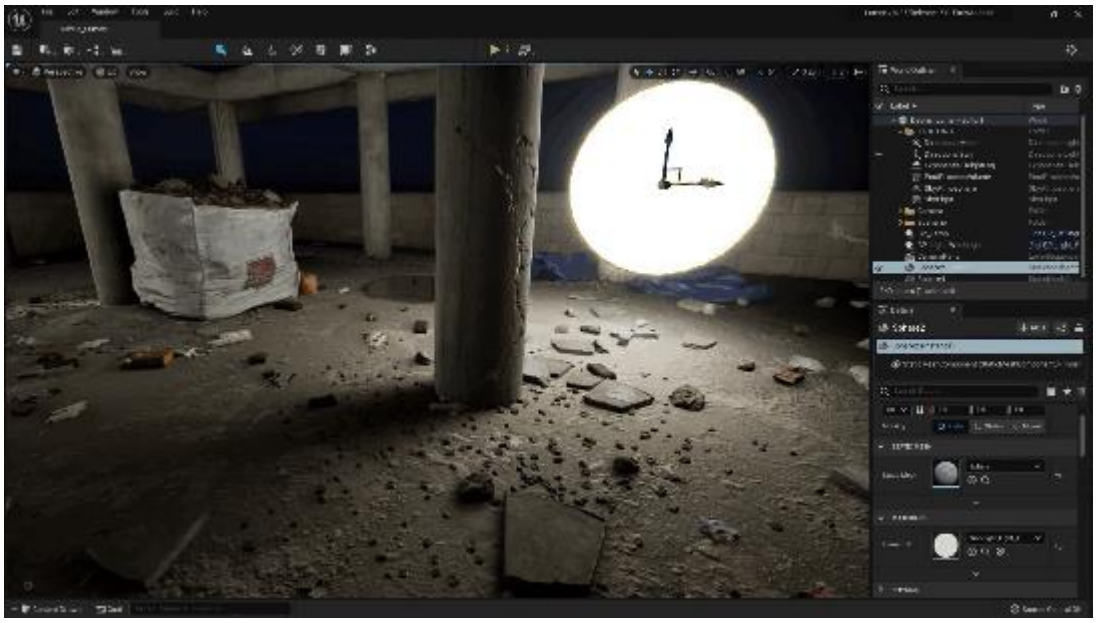

Рис.1.3 Реалізація м’яких непрямих тіней від джерела світла

Lumen підтримує Sky Light. Він враховує колір віддалених частин рівня та додає непряме світло на об'єкти. Наприклад, коли у грі зміниться час доби, відображення інших об'єктах рівня теж зміняться(Рис.1.4).

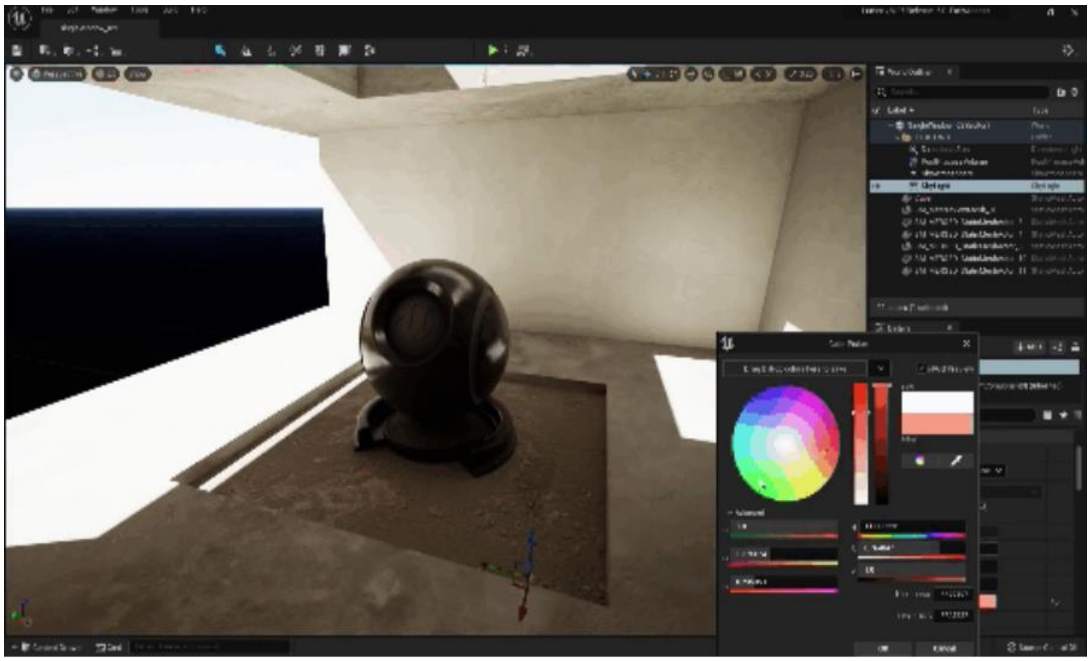

Рис.1.4 Приклад віддзеркалення червоного кольору в реальному часі

Тобто перевага Lumen перед звичайним трасуваням променів в тому, що воно займає менше ресурсів комп’ютера, даючи при цьому картинку однакового фотореалістичного рівня, не потребуючи при цьому відеокарту лінійки RTX.

Очікується, що у додатках для консолей наступного покоління, має відображатися не менше 60 кадрів за секунду, що звісно складно зробити при зростаючому рівні графіці. Саме тому, розробники рушію реалізували систему динамічного рендерингу Temporal Super Resolution(TSR). Суть у тому, що ця високоякісна система, незалежно від платформи, на якій відображається додаток, дозволяє рушію виконувати рендеринг з меншим розширенням, видаючи таку ж саму кількість пікселів на виході, як при рендерингу з більшим розширенням.

У Unreal Engine 5 також з’явилась система World Partition, яка ділить увесь рівень на осередки та передає на рендеринг лише ті, які задіяні у даний

момент. Також можна вільно створювати різні варіанти одного рівня, наприклад нічна версія та денна, у вигляді слоїв, які існують у одному просторі.

Blueprints дозволяють створювати свої поведінки для об’єктів(Рис.1.5). Об’єкт може бути чимось фізичним (типу поворотного столу) або абстрактним, наприклад, системою здоров’я.

Хочете створити автомобіль, що рухається? Використовуйте Blueprints. А як щодо літаючої свинки? Використовуйте Blueprints. А якщо потрібен котик, що вибухає при торканні? Blueprints.

В Blueprints використовується система на основі нодів. Це означає, що достатньо створити ноди та з’єднати їх – жодного коду не потрібно.

Blueprints прості у використанні, проте не такі швидкі, як код на C++. Тобто якщо вам потрібно створити щось «важке» з погляду обчислень, наприклад, складний алгоритм, краще скористатися C++.

Але навіть якщо ви віддаєте перевагу C++, то бувають випадки, коли оптимально використовувати Blueprints. Ось деякі з переваг Blueprints:

- Зазвичай розробка на Blueprints швидше, ніж C++.
- Простота впорядкування. Можна розділяти ноди на різні області, наприклад, на функції та графи.
- Якщо ви працюєте з людьми, які не знають програмування, то зміна Blueprint простіше завдяки їхній наочності та інтуїтивній зрозумілості.

Хорошим підходом буде створення об’єктів за допомогою Blueprints. А коли потрібні додаткові можливості, перетворення їх на C++.

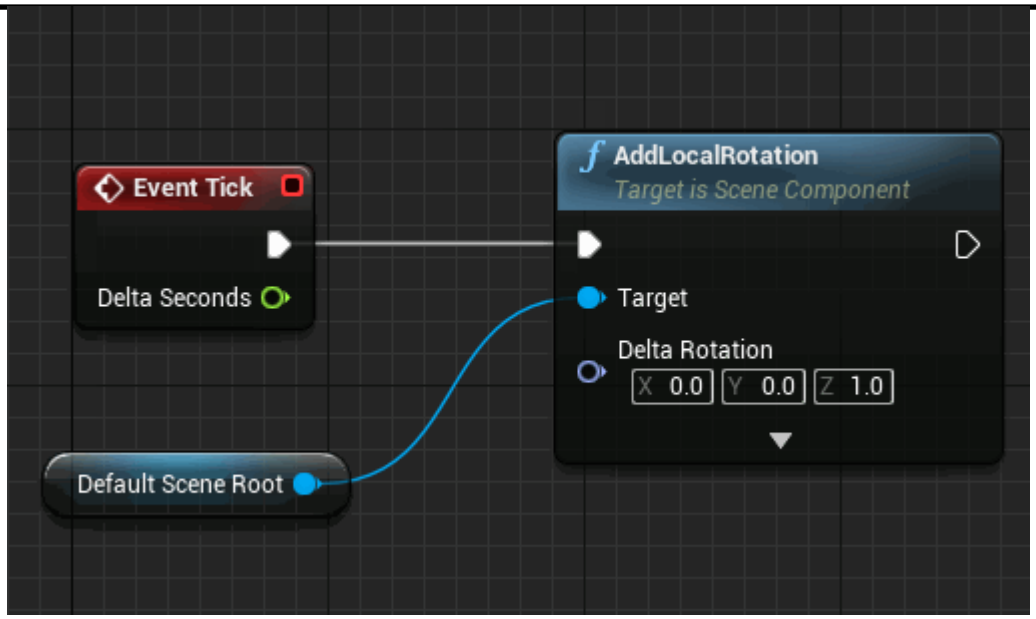

Рис.1.5 Blueprints, на якому показана функція, яка обертає об’єкт

Для тільки почавших навчання з роботою Unreal Engine 5, розробники зробили велику кількість документації до рушія, відеоуроків, де розказано як працювати з рушієм, керівництва по процедурній генерації світу, робота з фізикою транспорту та багато іншого, а якщо щось не зрозуміло, то завжди можна спитати у великої спільноти рушія.

Щоб спростити навчання та адаптування до Unreal Engine 5, розробники створили Unreal Marketplace – майданчик, де можна знайти готові приклади проектів на Unreal Engine 5, підтримуються популярні формати асетів, в т.ч. 3D моделей стандартів FBX та USD.

Одним з плюсів Unreal Engine 5 є обернена сумісність з проектами більш старих версій, проте треба перевірити плагіни перед портуванням на нову версію, бо деякі з них більше не підтримуються.

1.3. Огляд популярних продуктів на основі Unreal Engine

Найбільш популярні компанії, які використовували цей двіжок – Nintendo , Корпорація Valve, Rockstar Games, Sony Computer Entertainment, Activision Blizzard.

Одним з найяскравіших приміром буде комп’ютерна гра Chorus(2021) від розробника Fishlabs та видавця Deep Silver. Це космічний шутер с видом від третього обличчя. Головна героїня Нара має протистояти ворогам на своєму потужному кораблі з ракетами та кулеметами прямо посередь галактики(рис.1.6)

Рис.1.6 Скріншот з гри Chorus

Саме двіжок Unreal Engine допоміг зробити таку якісну гру з великою кількістю змінних, об’єктів та ефектів, реалізувавши при цьому максимальний потенціал комп’ютера. При цьому, що найголовніше, двіжок якісно використовує ресурси обчислювальної машини. Поки ігри на інших двіжках видають занадто низьку частоту кадрів, Chorus видає мінімум 60 кадрів на секунду. Тобто ця гра показує, що двіжок дійсно вміє працювати з великою кількістю об’єктів та ефектів, не втрачаючи швидкодію.

Іншим вдалим прикладом можна вважати ігру Valorant від розробника Riot Games. Це багатокористувацький шутер від першого обличчя, у якому гравці стикаються між собою у різних режимах гри та сперечаються між собою хто краще грає(рис.1.7).

Рис.1.7 Скріншот з гри Valorant

Це вдалий приклад використання двіжку Unreal Engine тому, що окрім великої кількості гравців на арені, у них ще є суперздатності, наприклад построїти стіну в реальному часі, яка буде мати фізичну модель, а також зброя, кулі від якої потрібно відображати одночасно у всіх гравців і при цьому об’єднати одним мережевим кодом. Цю гру хвалять як раз за гарну швидкодію на комп’ютерах різної потужності, за що вона і отримала гарну оцінку.

Іншим вдалим прикладом можна вважати гоночний симулятор Assetto Corsa Competizione(Рис.1.8) від розробника Kunos Simulazioni.

Рис.1.8 Зображення з самої гри з інтерфейсом

Особливість цієї гри в тому, що розробники спочатку хотіли застосувати ігровий двигун своєї розробки, але помітили, що він не зможе реалізувати усі їх ідеї, тому вони прийшли до Unreal Engine. Новий двигун зміг зробити реалістичні погодні умови, нічні гонки, оновлену систему моделей шин, аеродинамічну фізику, анімацію захвату руху, лазерне сканування усіх трас у грі. Завдяки цим технологіям, розробники досягли більш високого стандарту реалізму та більшого погруження у гру.

Ще однією особливістю в цій грі є технологія трасування променів на базі відеокарт серії RTX. Ігровий двигун може впоратися за таким складним методом глобального освітлення, що без сумнівів гарно впливає на погруження в ігровий процес.

Наступним прикладом використання ігрового рушія я хочу привести Hellblade: Senua’s Sacrifice(Рис.1.9) від компанії Ninja Theory.

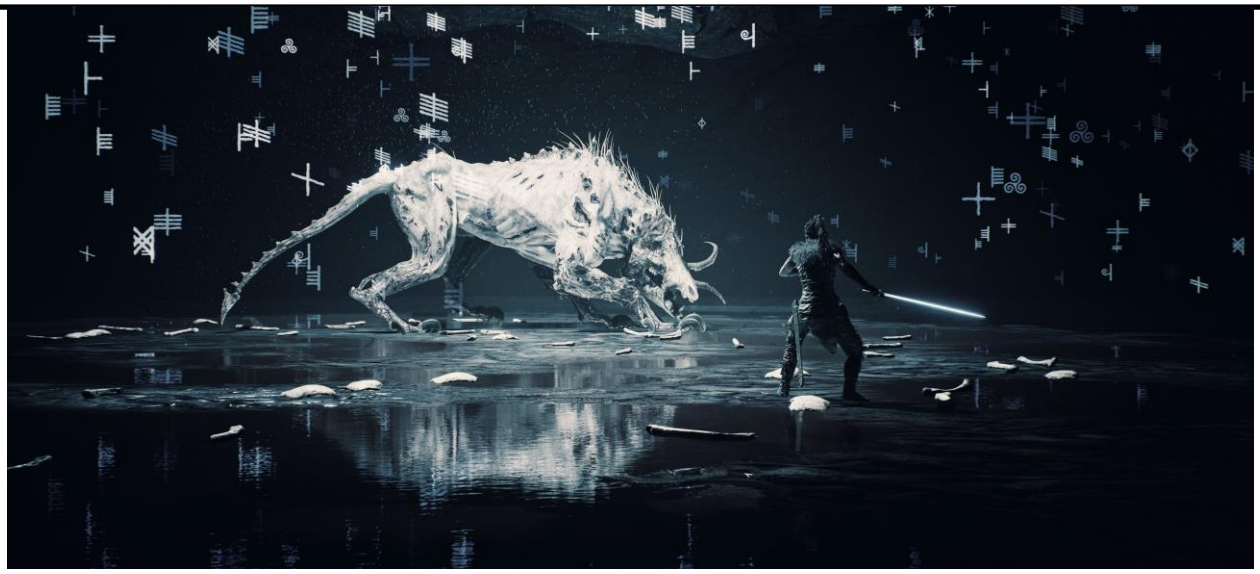

Рис.1.9 Скріншот з гри

Це сюжетна гра з елементами екшену погружає гравця у досередньовічний світ психічнохворої героїні Сенуї. Однію з особливостей гри є відмовлення від стандартного інтерфейсу. Кількість здоров'я показується «почервонінням» екрану, а елементи головоломок – символами у грі. Завдяки тому, що гра не перегружена гемплейно, розробники вклали багато сил на створення реалістичної графіки і саме тому здається, ніби і сам знаходишся у грі. Тут застосовані технології трасування променів, захвату руху та бінауральний запис звуку. Захват руху тут використовувався для реалістичної міміки головної героїні, тому коли наступають кат-сцени, то здається, ніби дивишся кіно, а бінауральний запис звуку для того, щоб гри грі у навушниках, голоси в голові Сенуї ніби говорили не їй, а прямо гравцю, тобто і сам асоціюєш себе з героїнею.

Усі ігри запускались на різних налаштуваннях графіки від низької до високої, в таблиці(Табл.1.1) вказана середня кількість кадрів за секунду часу на процесорі Intel Core i5 6300HQ, оперативною пам'яттю на 16 гігабайт, жорсткому диску з ємніст'ю 1 терабайт та відеокартою Nvidia GTX 960M.

Табл.1.1

Назва гри	Низька графіка, к/с	Середня, к/с	Висока, к/с
Chorus	60 к/с	31 к/с	26 к/с
Valorant	160 к/с	100 к/с	40 к/с
Assetto Corsa	70 к/с	35 к/с	31 к/с
Senua	124 к/с	60 к/с	39 к/с

Як ми можемо побачити, в багатокористувацьких та сюжетних іграх, ігровий двигун справляється краще, ніж в навантаженних об’єктами або змінними, по типу галактик у Chorus та фізикою великою кількості машин у Assetto Corsa.

Зважаючи на виділені переваги та недоліки існуючих ігор, що функціонують на основі рушія Unreal Engine 5, у дипломному проєкті вирішено створити **інтерактивний додаток** з великою кількістю об’єктів та потужною фізикою. Для цього у дипломному проєкті необхідно виконати наступні завдання:

- ознайомитися з існуючими рушіями для створення сучасних ігор
- дослідити можливості найпопулярніших продуктів, створених на основі рушія Unreal Engine 5;
- вивчити функціонал рушія Unreal Engine 5;
- вивчити технології розробки гри як інтерактивного додатку для роботи з великою кількістю об’єктів та потужною фізикою;
- створити інтерактивний додаток для здійснення ігрових маніпуляцій на основі Unreal Engine5;
- оцінити кількісні показники роботи розробленого інтерактивного додатку.

2. Процес розробки інтерактивного додатку для здійснення ігрових маніпуляцій на основі Unreal Engine 5

2.1. Запуск проекту

На початку розробки потрібно створити новий проект у головному меню рушія. На вибір є шутер від першого обличчя, пустий рівень від третього обличчя, рівень з видом зверху, гонка з готовою машиною, рівень для альтернативної реальності для телефону та віртуальна реальність. Обираємо проект First Person, тобто шутер від першого обличчя.(Рис.2.1)

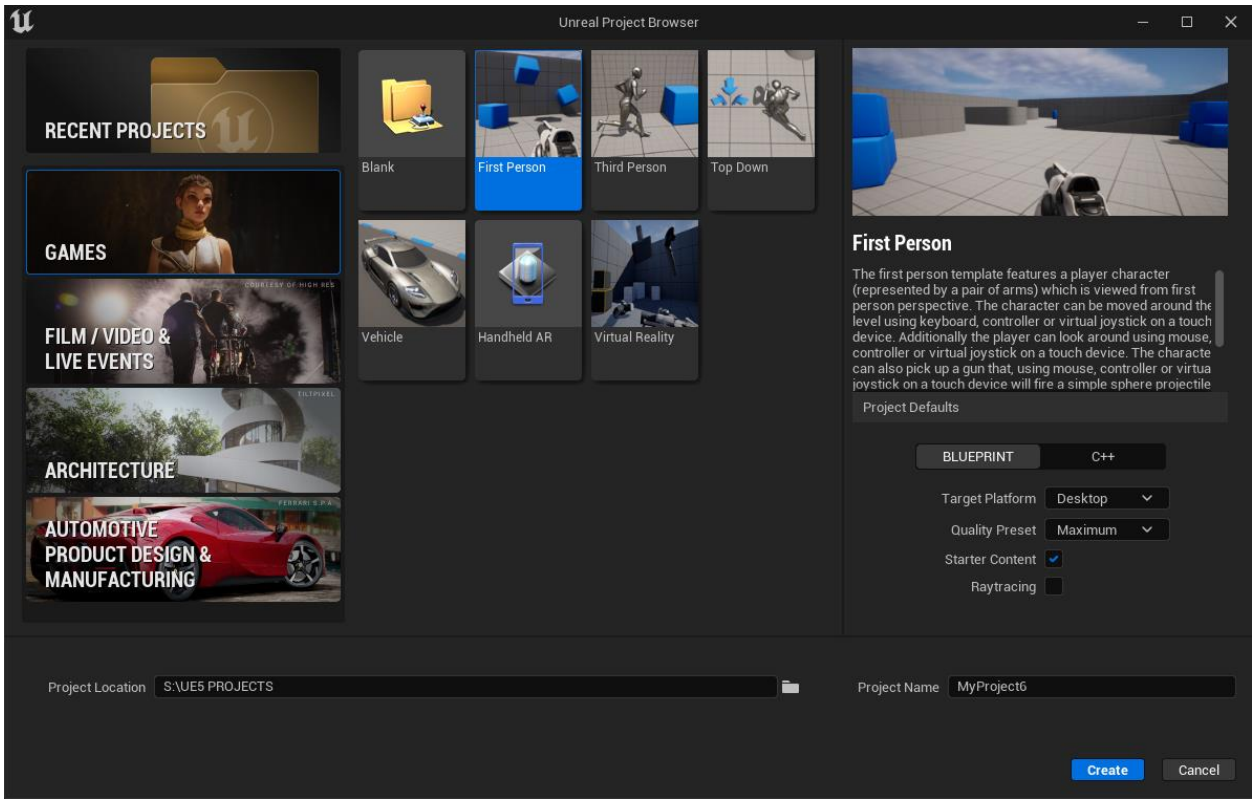

Рис.2.1 Вибір проекту на головному меню рушія

Після завантаження рівня отримаємо готовий рівень з базовою ідеєю шутера(Рис.2.2), тобто пішак бере(точніше стикається з ним) об’єкт гвинтівки і отримує можливість стріляти з неї, бігати по рівню та експерементувати з фізикою кубів.

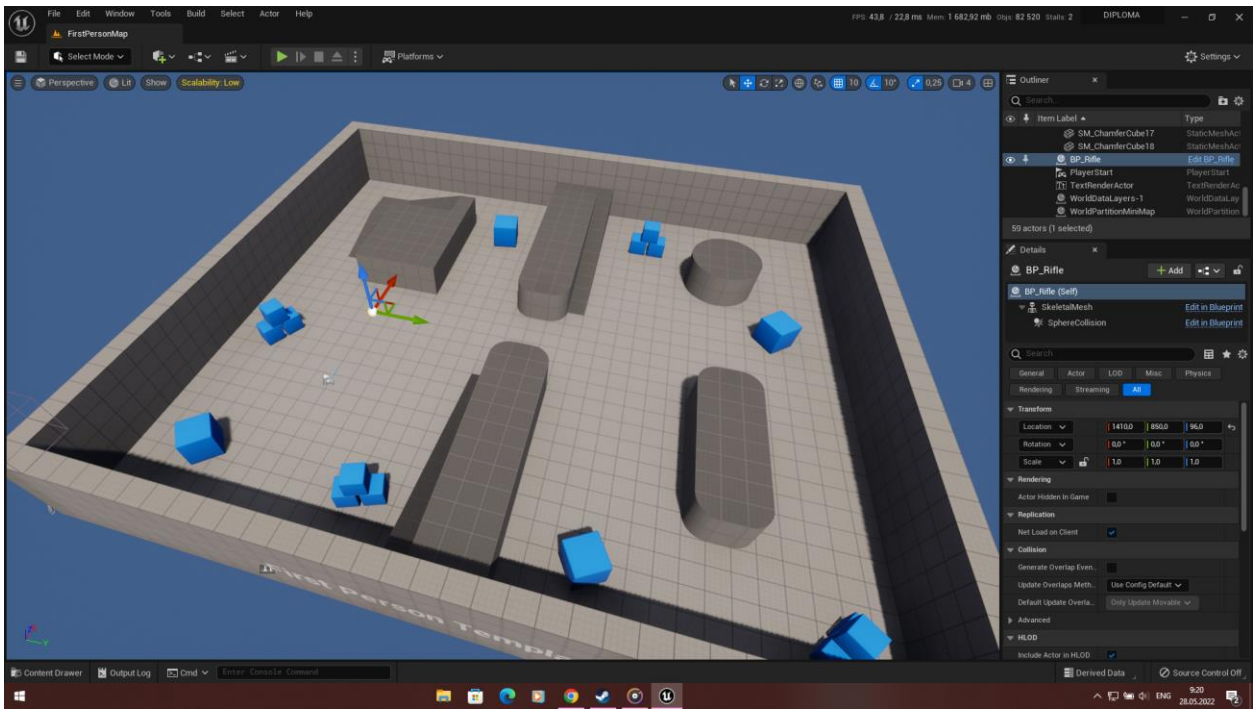

Рис.2.2 Створений рівень гри

Сам процес взяття гвинтівки зроблений у технології Blueprints таким чином:

1. Спочатку перевіряться умова чи перетиналась модель пішака зі сферою колізії гвинтівки(Рис.2.3)

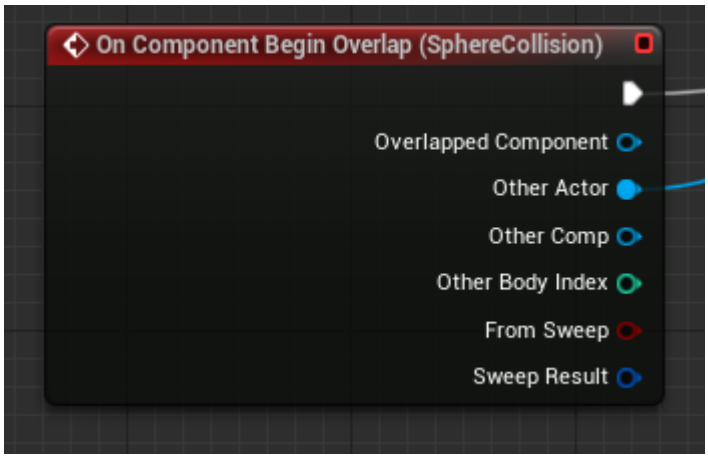

Рис.2.3 Перевірка на перетинання моделей

2. Якщо так, то процес іде далі, якщо ні, вистрілити з гвинтівки не можна.
3. Рушій звертається до пішака, бере точку, куди потрібно приєднати гвинтівку(Рис.2.4)

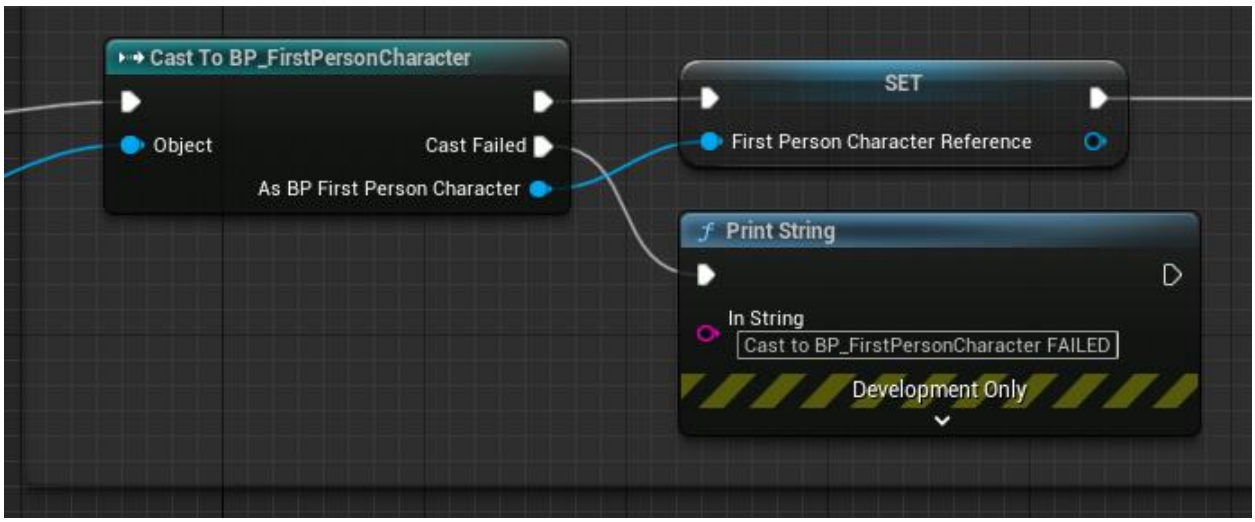

Рис.2.4 Звертання до пішака

4. Приєднує гвинтівку (self) до потрібного місця (grip point) пішака (First person mesh)(Рис.2.5)

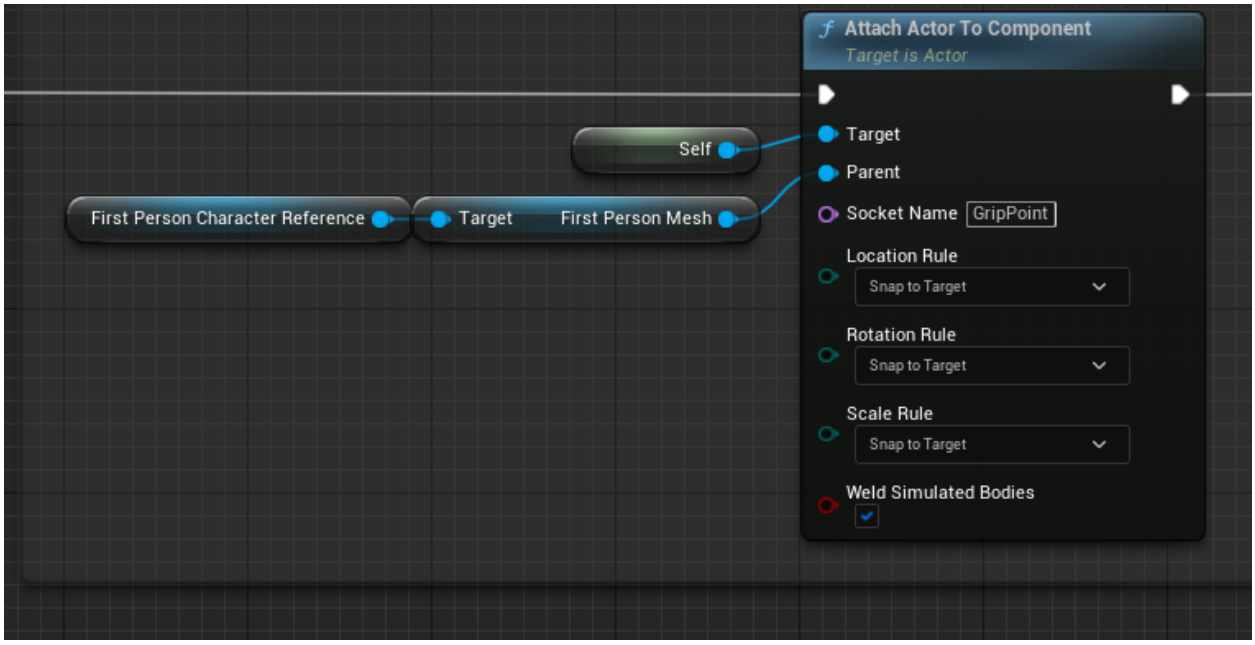

Рис.2.5 Приєднання гвинтівки до пішака

5. Після цього включається метод, який дозволяє пішаку стріляти з гвинтівки(Рис.15)

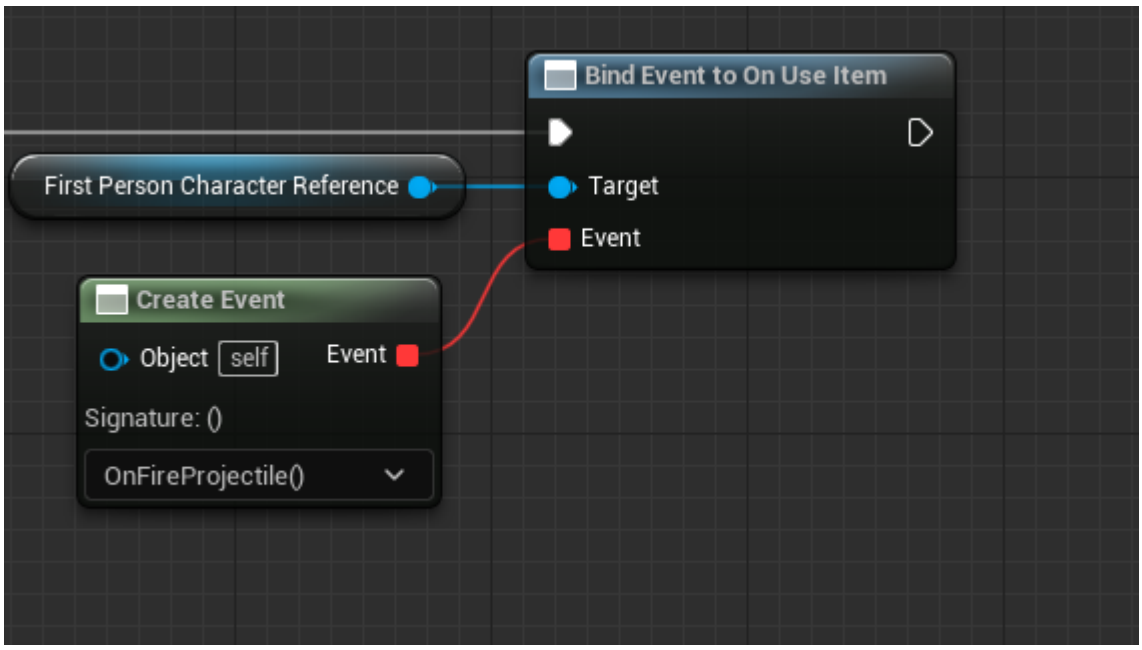

Рис.2.6 Метод On Use Item, який дозволяє пішаку стріляти

Зі сторони пішака процес виглядає так: Гравець натискає кнопку вогонь, якщо метод On Use Item спрацював(тобто гвинтівка підібрана), то запускається подія OnFireProjectile(), тобто запуск кулі(Рис.2.7)

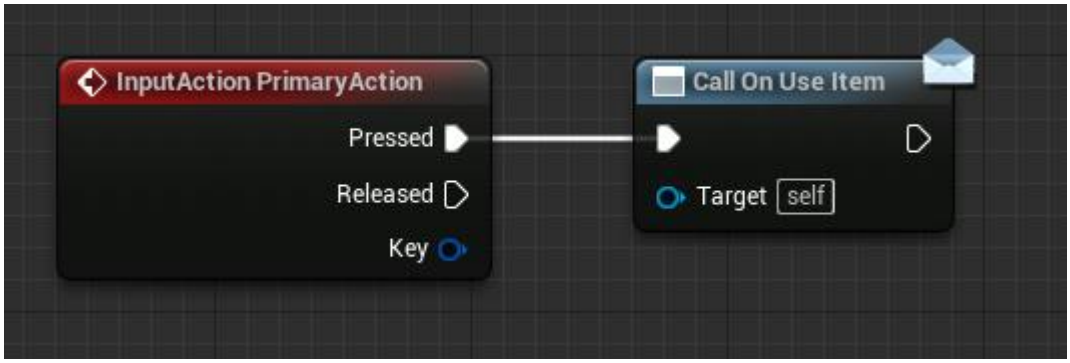

Рис.2.7 Після натискання кнопки вогонь іде перевірка спрацьовував On Use Item чи ні

Ще одним важливим алгоритмом є запуск гри з нуля. Знання цього алгоритму допомагає уникнути помилок, коли наприклад розробник викликає мережевий код, коли навіть не створений екземпляр гри.

Схема алгоритму створення інтерактивного додатку для здійснення ігрових маніпуляцій на основі Unreal Engine 5 відображена на рис.2.8.

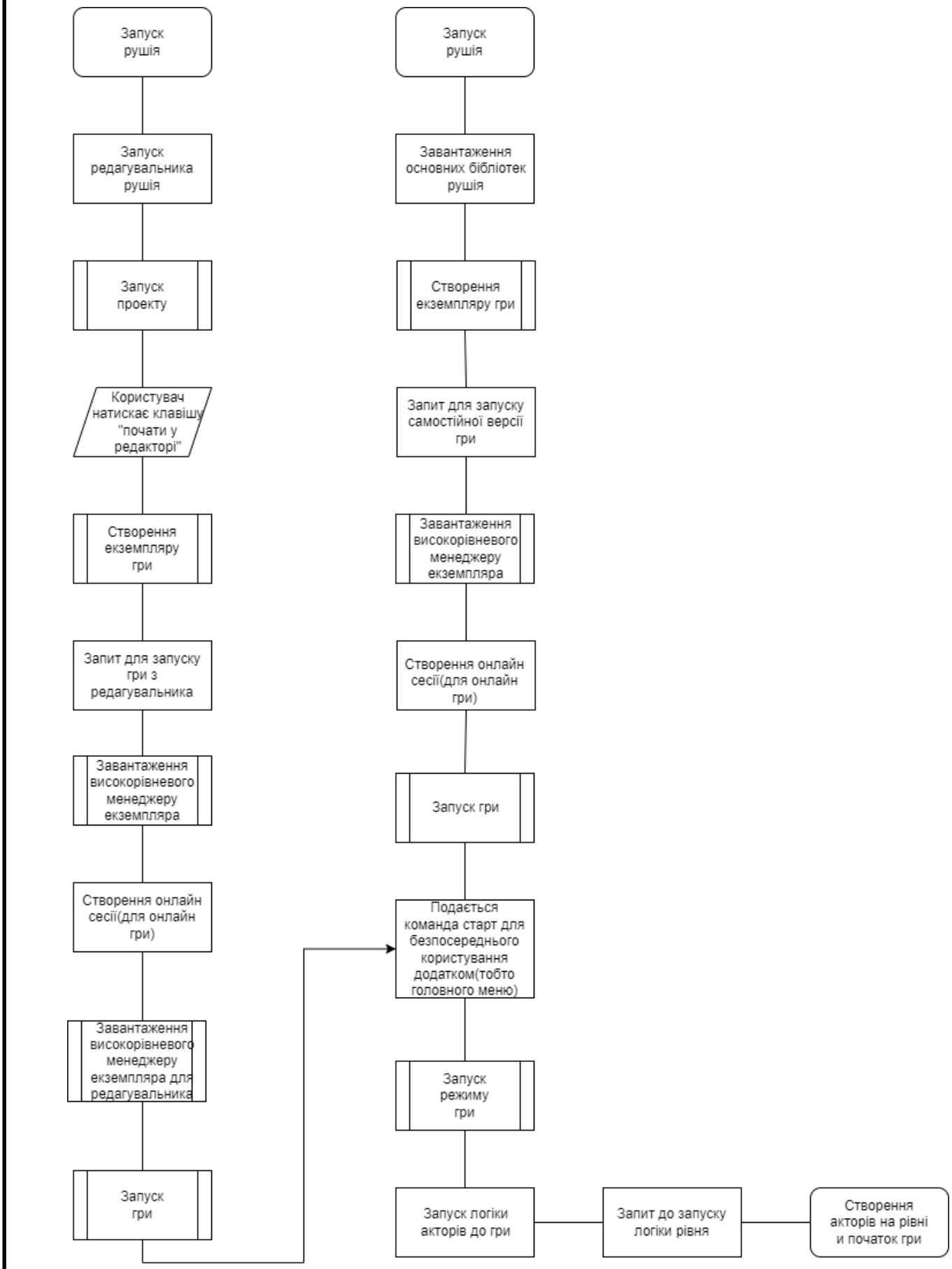
Запуск рушія
Запуск редагувальника рушія
Запуск проекту
Користувач натискає клавішу "почати у редакторі"
Створення екземпляру гри
Запит для запуску гри з редагувальника
Завантаження високорівневого менеджеру екземпляра
Створення онлайн сесії(для онлайн гри)
Завантаження високорівневого менеджеру екземпляра для редагувальника
Запуск гри
Запуск рушія
Завантаження основних бібліотек рушія
Створення екземпляру гри
Запит для запуску самостійної версії гри
Завантаження високорівневого менеджеру екземпляра
Створення онлайн сесії(для онлайн гри)
Запуск гри
Подається команда старт для безпосереднього користування додатком(тобто головного меню)
Запуск режиму гри
Запуск логіки акторів до гри
Запит до запуску логіки рівня
Створення акторів на рівні и початок гри

Рис.2.8 Схема алгоритму запуску рівня багатокористувацького додатку

У цілях економії ресурсів комп’ютера, рушій не завантажує акторів та рівні усієї гри у оперативну пам’ять перед запуском головного меню.

2.2. Створення мішеней

Надалі я планую зробити мішені, тобто акторів, при зіткненні куль з якими, вони будуть падати. Для цього потрібно імпортувати моделі мішеней до проекту. Потрібно завантажити проект з акторами та через функцію «migrate» перемістити до нашого проекту(Рис.2.9).

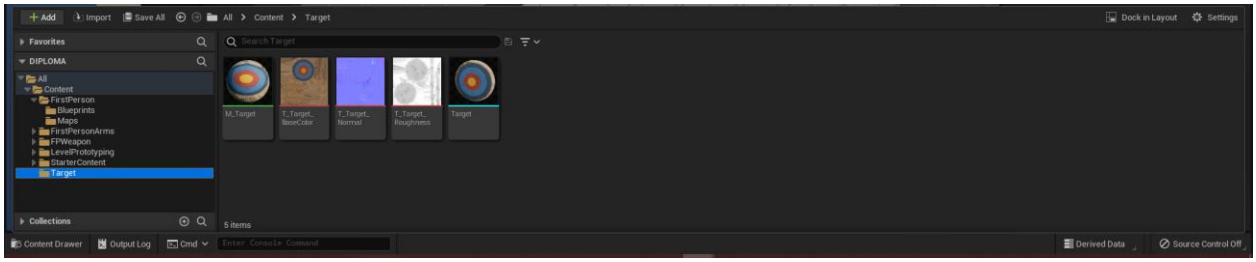

Рис.2.9 Моделі успішно переміщені

Тепер потрібно створити клас Blueprints для того, щоб створювати логіку актора та помістити до нього модель(Рис.2.10).

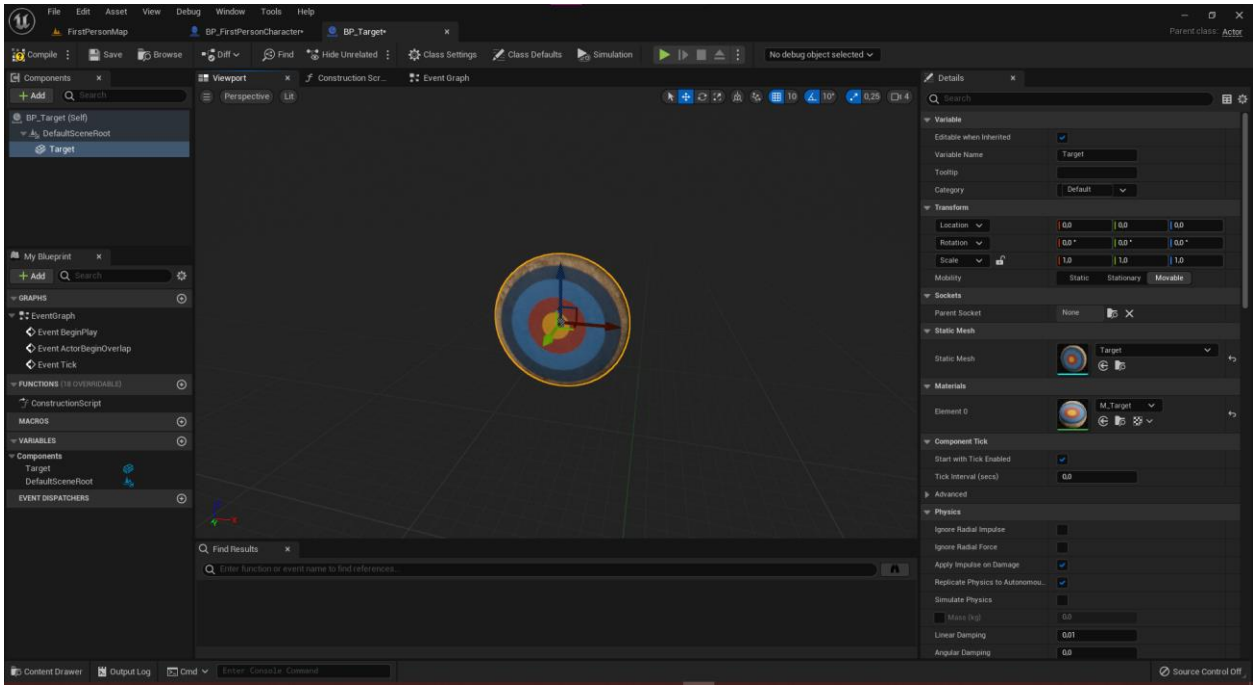

Рис.2.10 Створений клас Blueprints для мішеней

Тепер у додатку є актор, якого можна помістити на сцену(Рис.2.11).

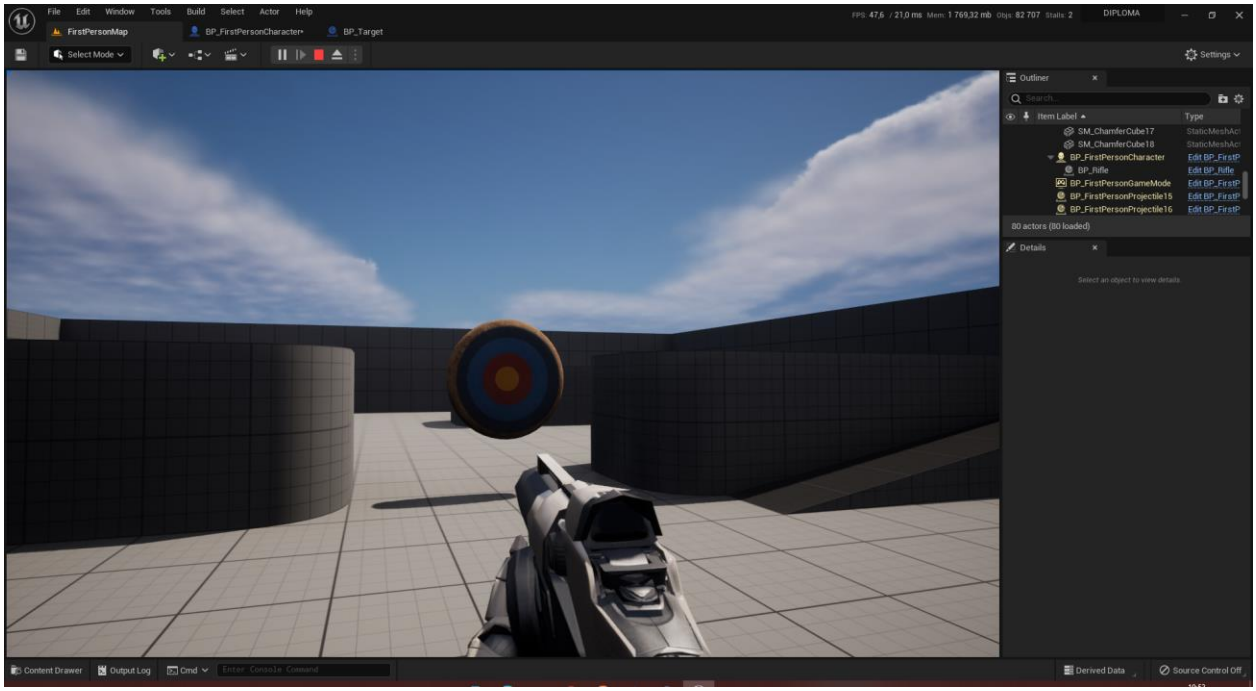

Рис.2.11 Мішень поміщена до сцени

Тепер потрібно додати режим гри. Що я хочу зробити: При певній кількості попадань по мішеням, висвітлювався текст «Ви перемогли». Тобто є наприклад три мішені, якщо по ним усім попасти, то я виконав умови перемоги.

Для цього потрібно додати до мішені функцію, яка фіксує перетинання фізичних моделей кулі та мішені(Рис.2.12).

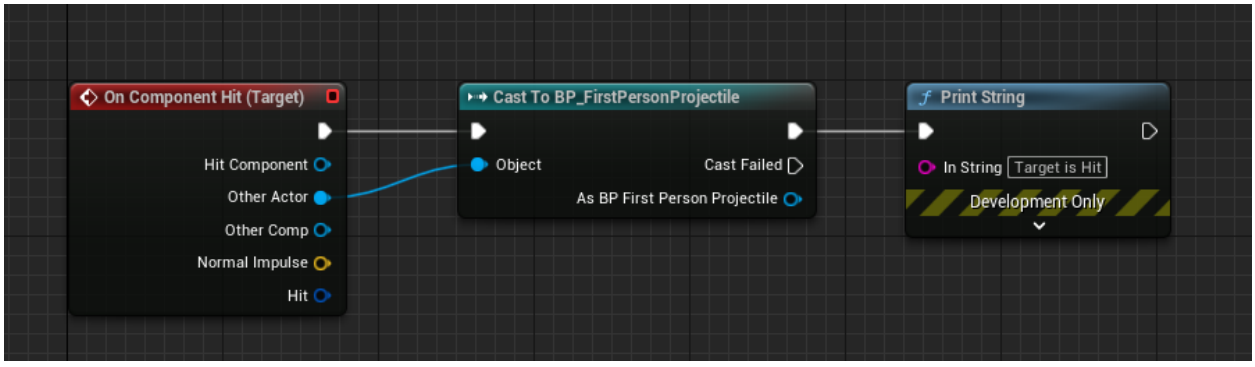

Рис.2.12 Створюється подія «On Component Hit», якщо фізичні моделі актора та кулі перетинаються, то створюється надпис «Target is Hit»

У результаті ми маємо побачити надпис «Target is Hit»(Рис.2.13).

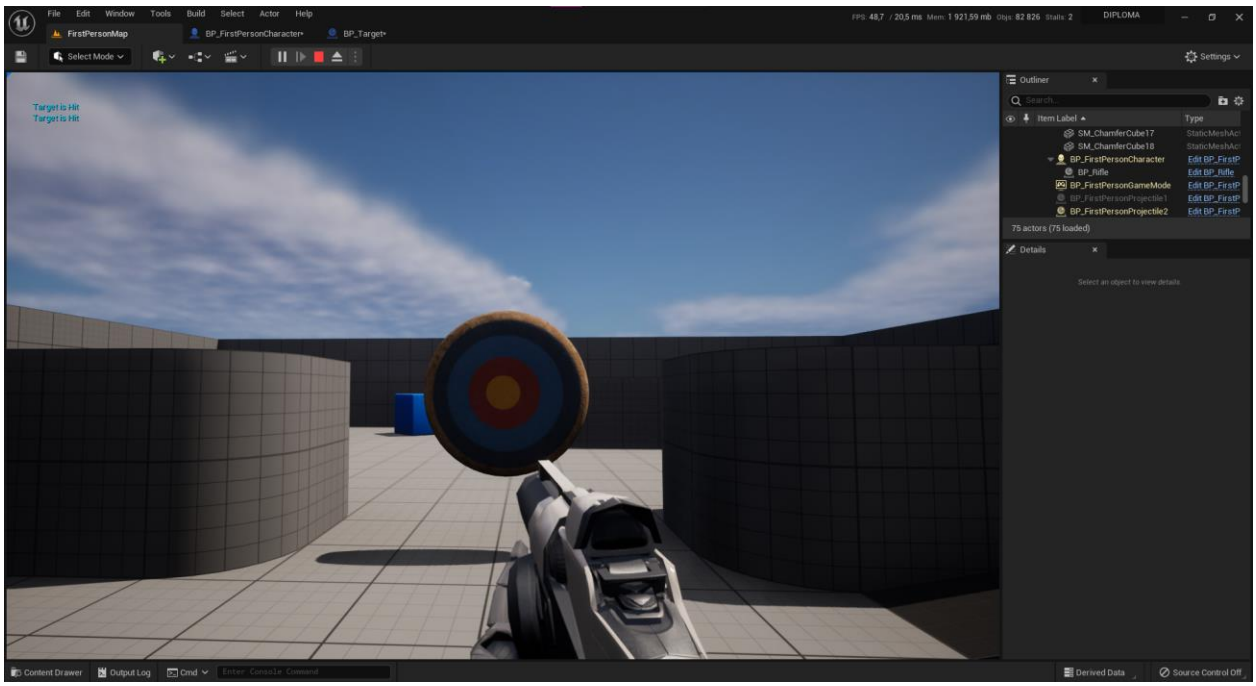

Рис.2.13 Схема Blueprints працює, надпис з’явився

2.3. Створення Режиму гри

Тепер потрібно створити режим гри, в якій прописана логіка завершення гри, для цього створюємо Blueprint клас Game Mode Base(Рис.2.14).

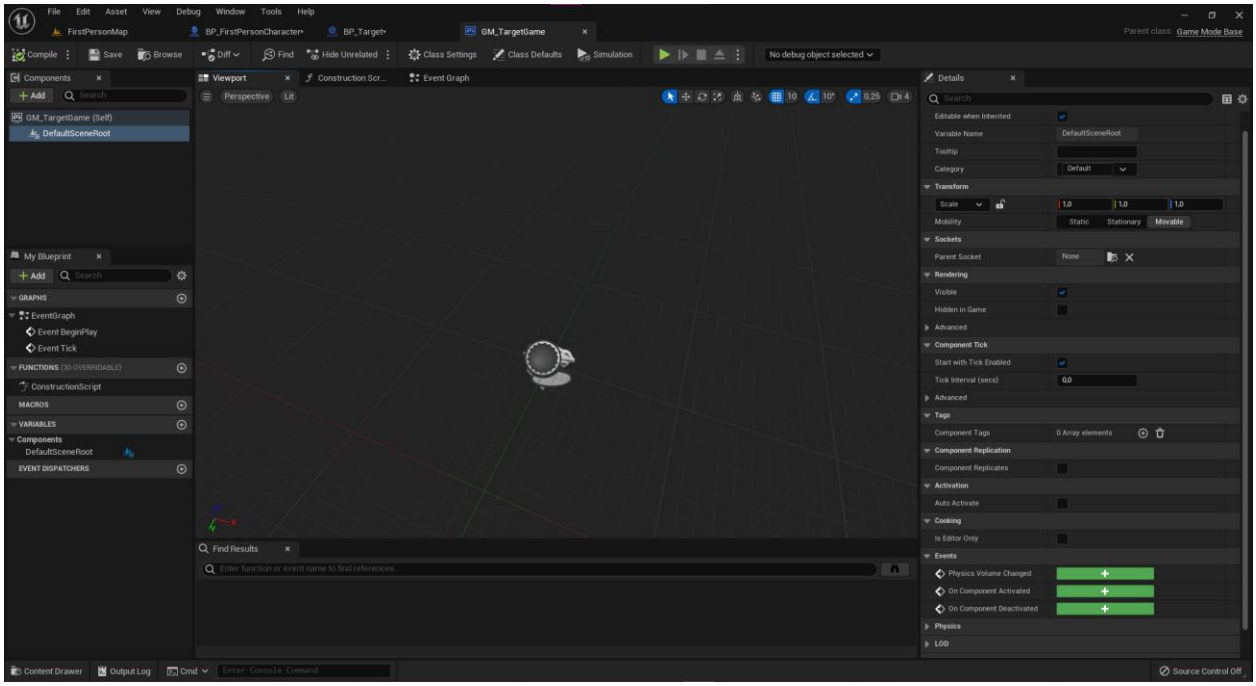

Рис.2.14 Клас GM_TargetGame

Тепер змінюємо і сам ігровий режим у додатку(Рис.2.15)

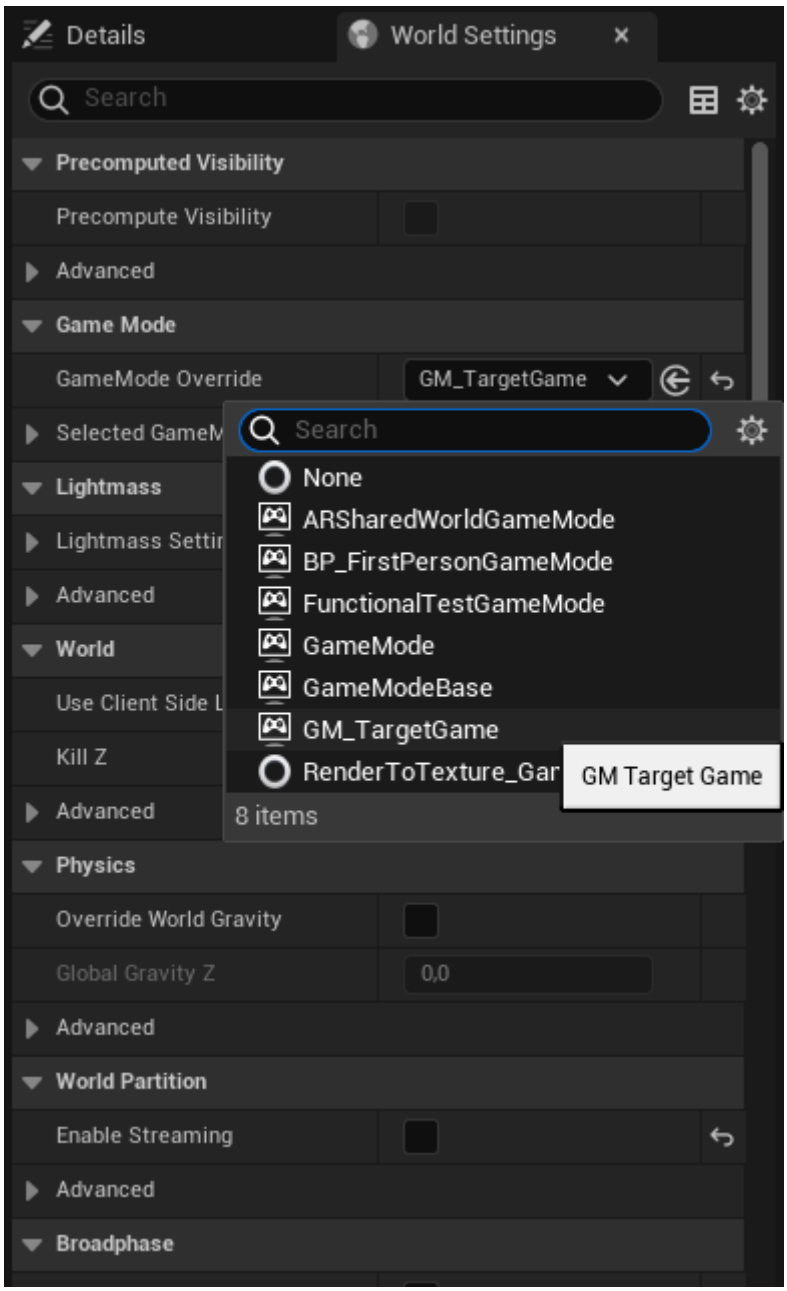

Рис.2.15 У налаштуваннях рівня змінюємо ігровий режим на GM_TargetGame

Тепер потрібно додати лічильник, який відслідковує кількість попадань по цілям. Для цього у класі режиму гри створюємо подію «AddScore». Тобто коли функція буде задіяна, то спрацює лічильник. Цим лічильником буде змінна «CurrentScore» типу Integer. Після спрацювання функції змінна збільшиться на одиницю та зробить надпис поточного значення на екран(Рис.2.16).

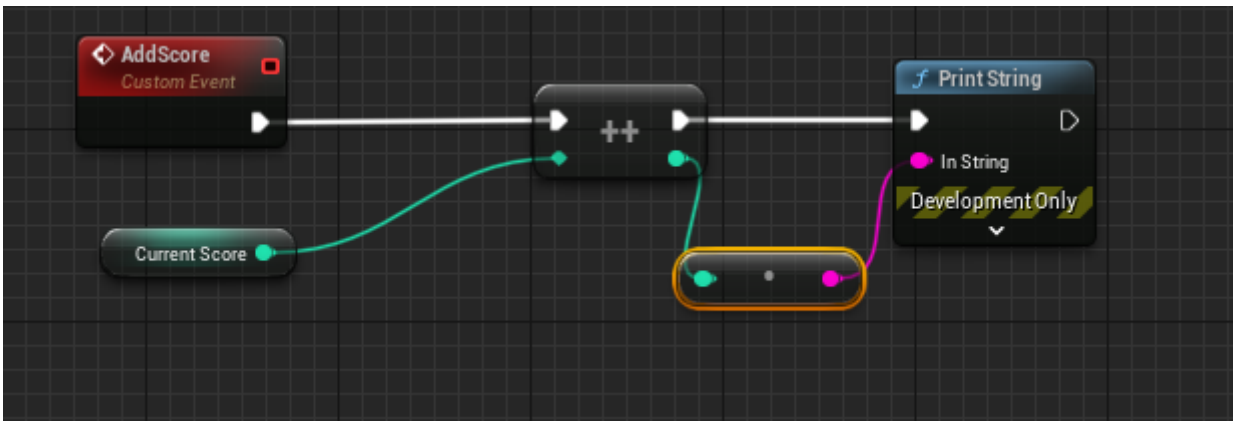

Рис.2.16 Вигляд функції лічильника попадань по цілям

Тепер потрібно додати до мішеней механіку, що при попаданнях по акторам, буде спрацьовувати функція AddScore. Для цього редагуємо Blueprint таким чином, що зв'язуємо мішень та режим гри через вказівник на останній. Таким чином «мішень» зможе визивати функції у режимі гри(Рис.2.17).

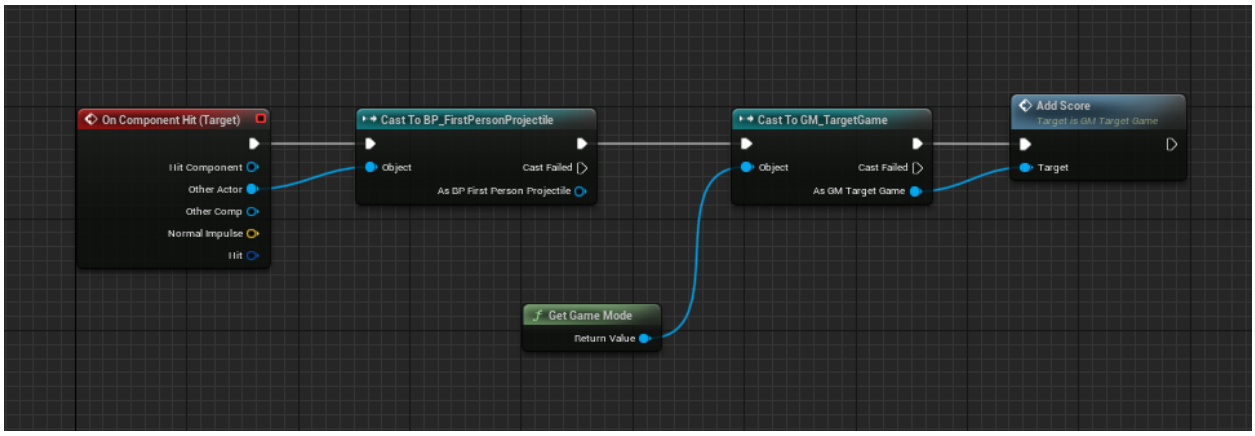

Рис.2.17 Оновлений вигляд Blueprint мішені

Таким чином, якщо куля перетинає мішень, то остання звертається до функції AddScore режиму гри, після чого з'являється надпис про поточне значення змінної CurrentScore.

Можна помітити, що таким чином можна в одну ціль попадати декілька разів і це буде зараховуватись, саме тому потрібно прибрати цю можливість через змінну типу Boolean, яка фіксує чи були вже попадання по мішені. Коли куля попадає по мішені, то одразу потрібно поставити функцію SET, яка змінює значення змінної «IsHit?» на True, щоб попадання більше не зараховувались. Тобто якщо значення змінної «IsHit?» False(тобто попадань

ще не було), то функція буде виконуватись, а якщо True, то вже виконуватись не буде(Рис.2.18).

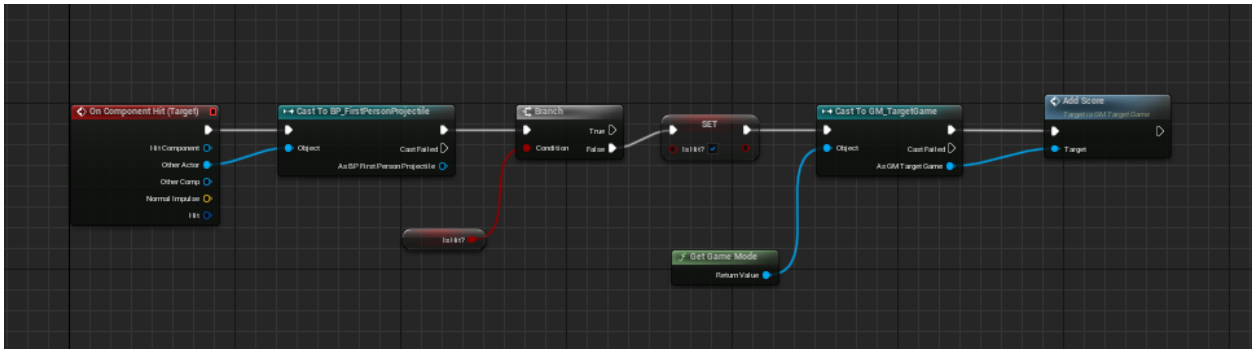

Рис.2.18 Вигляд BP_Target з вирішеною проблемою

2.4. Створення ігрового інтерфейсу

Тепер якщо подивитися на скріншот з гри, то можна побачити, що гравцю недостатньо інформації скільки він вже попав у мішені, да і виглядає воно не сучасно, тому потрібно встановити новий гарний інтерфейс(Рис.2.19).

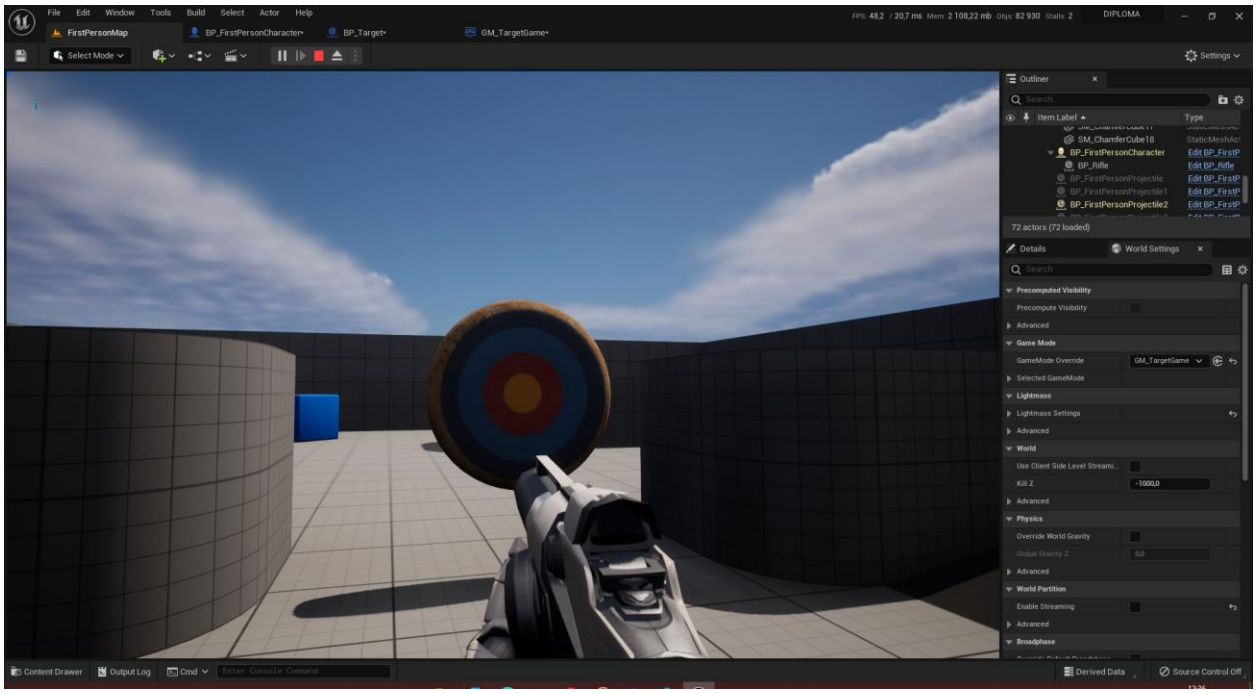

Рис.2.19 Інтерфейс гри – лише цифра угорі скільки разів потрапив у мішені, при цьому вона ще й швидко пропадає

Саме тому, треба створити клас інтерфейсу, де можна налаштовувати інформацію про гру. Для цього створюється текстовий блок усередині площадки для розміщення інформації, нехай цей блок буде називатися Score.

В цьому класі можна налаштовувати як сам вигляд інтерфейсу, так і його логіку.

У Blueprint інтерфейсу потрібно прописати подію «Event Constuct», що інтерфейс звертається до Режиму гри. Ця подія спрацьовує на початку гри та лише один раз. Далі потрібно створити подію UpdateScore. При кожному оновленні кількості попадань по мішені, буде створюватись ця подія і вона буде оновлювати кількість попадань на екрані. Вона надсилає сигнал відобразити текст на екрані. Для цього береться вказівник на блок тексту інтерфейса та вказівник на режим гри, де знаходиться змінна Current Score, яка відповідає за кількість попадань у мішень(Рис.2.20).

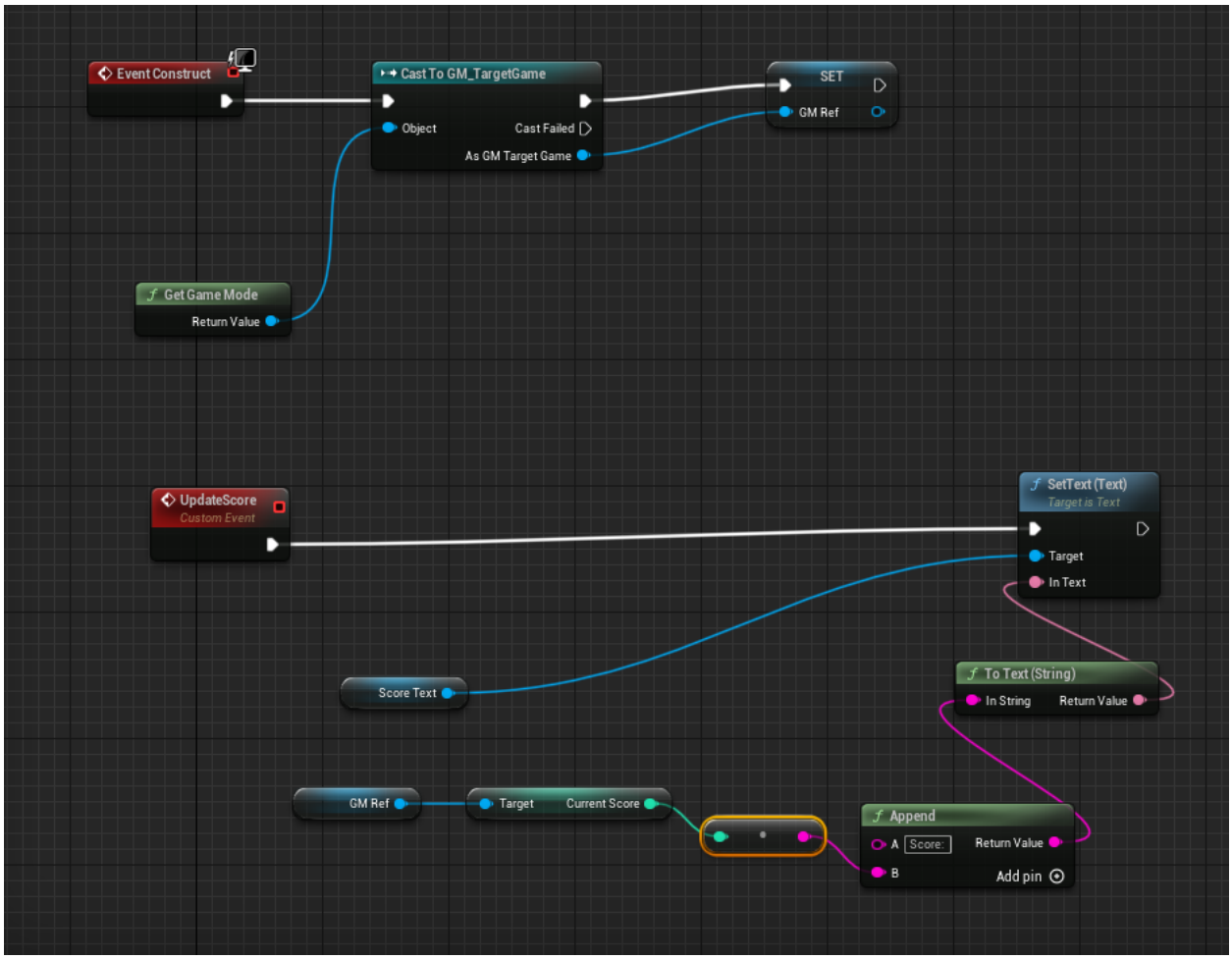

Рис.2.20 Blueprint інтерфейсу додатка

Тепер потрібно, щоб цей інтерфейс показувався на екрані. Для цього у Режимі гри GM_TargetGame треба створити подію «Event Begin Play». Вона працює так само, як і подія «Event Construct», тобто запускається один раз лише на початку гри. Ця подія буде відповідати за те, щоб запустити

звернутися до інтерфейсу, відобразити на екрані гравця та включити подію Update Score, щоб вже з початку гри на екрані кількість попадань була рівна нулю(Рис.2.21).

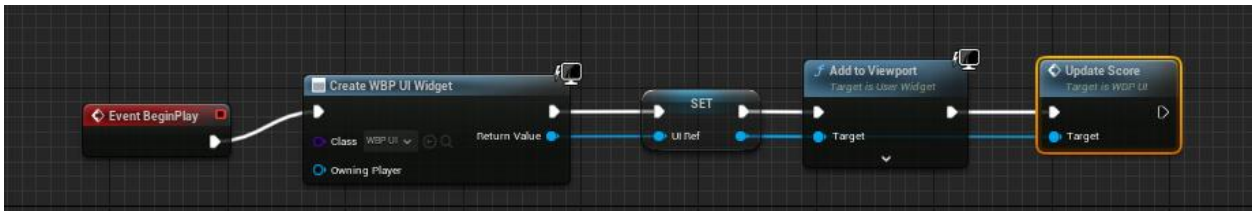

Рис.2.21 Вигляд події для відображення інтерфейсу

Також потрібно змінити функцію оновлення кількості попадань у режимі гри. Тепер коли фіксується попадання по мішені, має викликатись функція Update Score, тому прибираємо функцію Print String і ставимо Update Score(Рис.2.22).

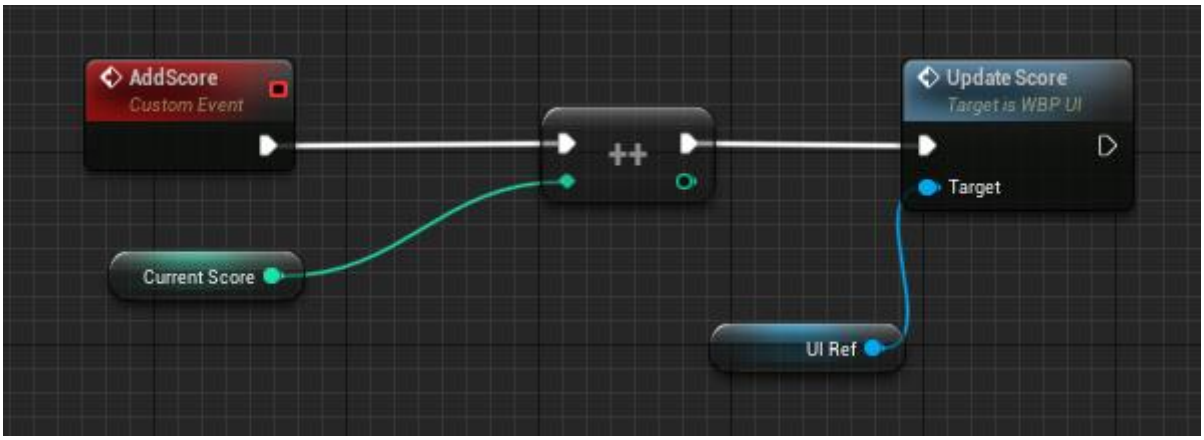

Рис.2.22 Вигляд нової функції оновлення кількості попадань по мішені у Blueprint Режиму гри

Тепер потрібно додати умову виграшу, тобто 3 попадання по мішеням. Це проміжний варіант для тестування каркасу ігрового режиму. Для цього у Blueprint ігрового режиму беруться усі актори певного типу(тобто мішені зі сцени), зчитується їх кількість та записується у спеціальну змінну Max Score типу Integer. Умовою виграшу буде чи співпадає кількість влучених мішеней з кількістю мішеней(Рис.2.23).

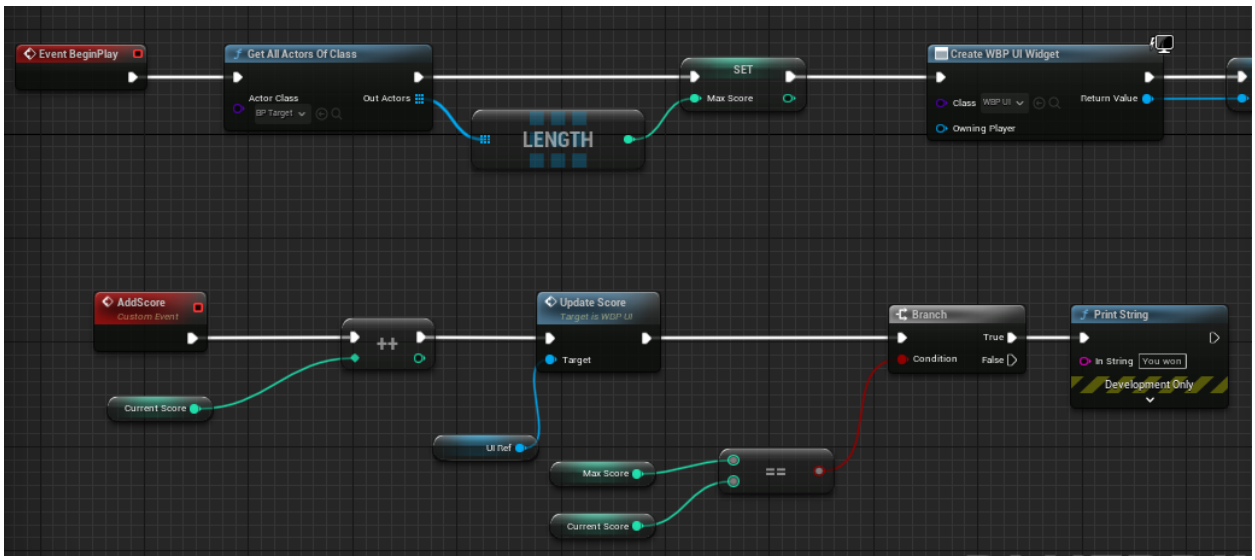

Рис.2.23 На скріншоті показано як встановлення змінної MaxScore від кількості мішеней, так і оператор Branch, який перевіряє співпадіння MaxScore та Current Score

Тепер потрібно зафіксувати це у інтерфейсі. Коли умова виконається, то з’явиться окремий інтерфейс поверх існуючого, котрий покаже інформацію, що гравець виграв.

Коли умова виграшу виконується, то спрацьовує спеціальна функція, яка викликає інтерфейс закінченої гри, а маніпуляції над пішаком стають заборонені(Рис.2.24).

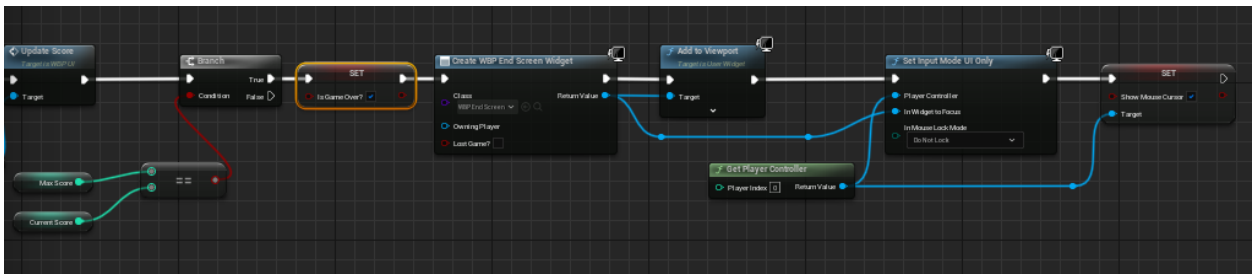

Рис.2.24 Зображення функції виграної гри

Після доробок інтерфейсу та додання функції перезапуску рівня і виходу з гри, отримаємо таке зображення(Рис.2.25).

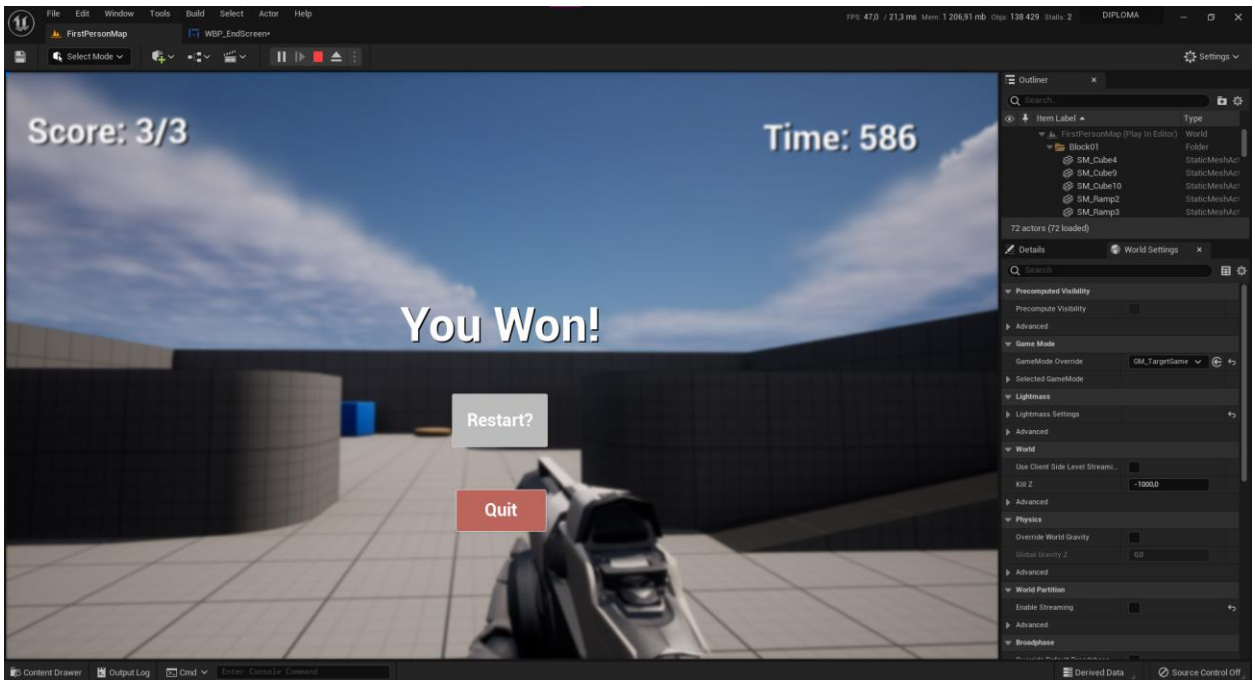

Рис.2.25 Готовий варіант інтерфейсу виграної гри

2.5. Додавання умови лічильника

На даній ітерації, є робоча версія гри, тепер можна додати нові умови виграшу, наприклад поцілити мішені за певний час, додати інтерфейс програної гри, якщо не встигнути поцілити усі мішені, додати фізику попадань на мішені та зробити гарну локацію.

Спочатку я вирішив зробити Таймер. Створюється змінна StartTime, в якій фіксується початковий час. Далі кожну 0,01 секунду значення зменшується на 1. Коли значення зменшується до 0, то з’являться інтерфейс програної гри з надписом «You Lost»(Рис.2.26).

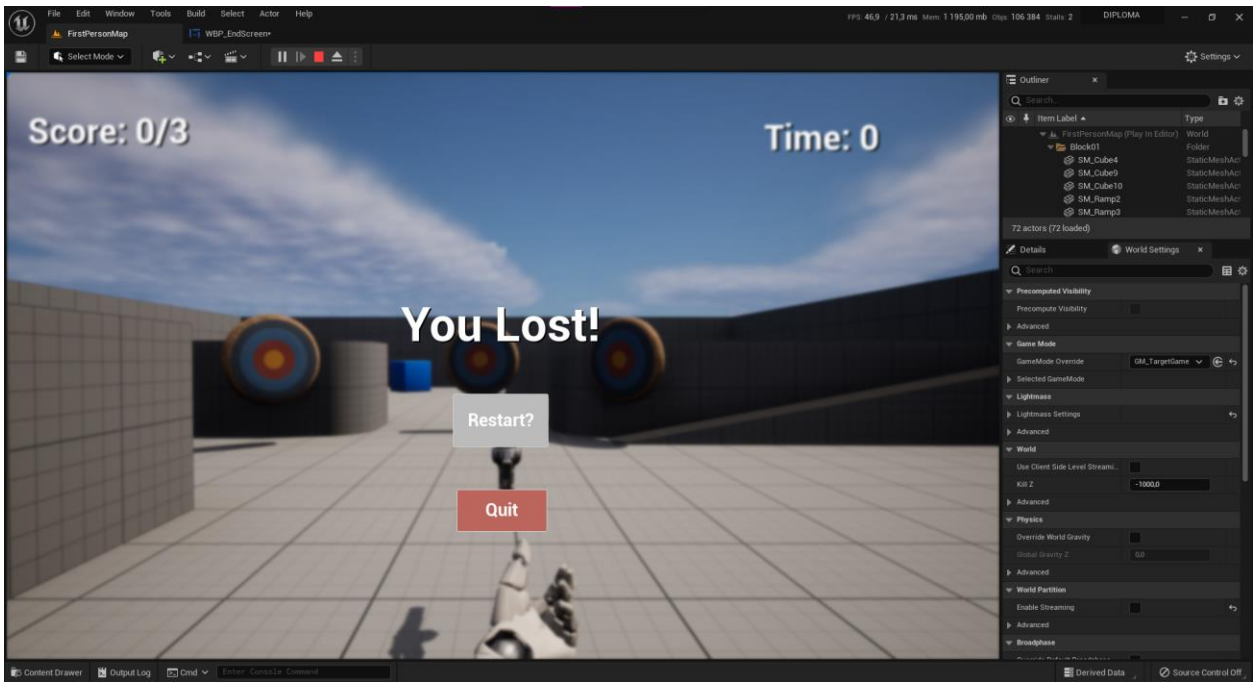

Рис.2.26 Якщо гравець не встигає влучити у цілі – він отримує надпис, що він програв та отримує можливість запустити рівень спочатку

Механіка роботи інтерфейсу в тому, що створюється один клас інтерфейсу для виграної гри та програної і один текстовий блок, який може одночасно використовуватись для обох сценаріїв гри. Працює це так, що якщо гравець виграє, то з’являється інтерфейс кінця гри, в якому за умовчуванням стоїть надпис «You Won!», а якщо гравець програє, то надпис змінюється на «You Lost!», тобто можна зберегти ресурси електронної обчислювальної машини і замість двох різних інтерфейсів кінця гри використовувати один.

Реалізовано це у режимі гри(Рис.2.27), та у інтерфейсі кінця гри(Рис.2.28)

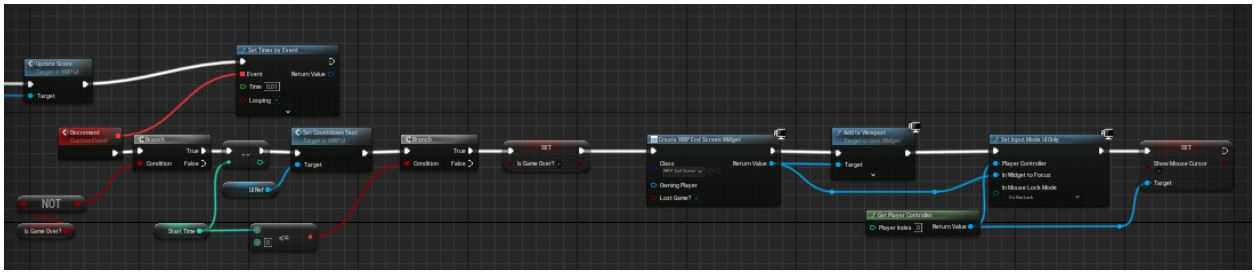

Рис.2.27 Змінна StartTime швидко зменшується і коли вона стає рівна нулю, то з’являється інтерфейс зі змінною «LostGame?» з надписом «You Lost!»

Саме Змінна «LostGame?» відповідає за те, який надпис буде у кінці гри.

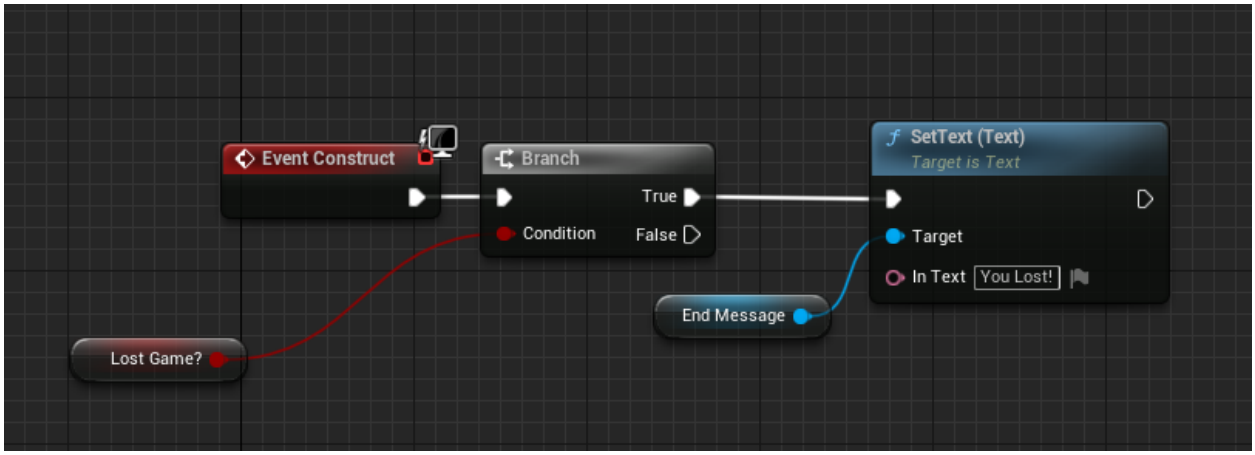

Рис.2.28 Якщо змінна «LostGame?» зі значенням True, то буде надпис «You Lost!», якщо False, то буде надпис за замовчуванням, тобто «You Won!»

2.6. Додавання фізики до рівня

Тепер можна додати фізику до мішеней, на які попадають кулі, щоб вони одразу падали під силою тяжіння. Для цього потрібно перейти до Blueprint мішені(Рис.2.29).

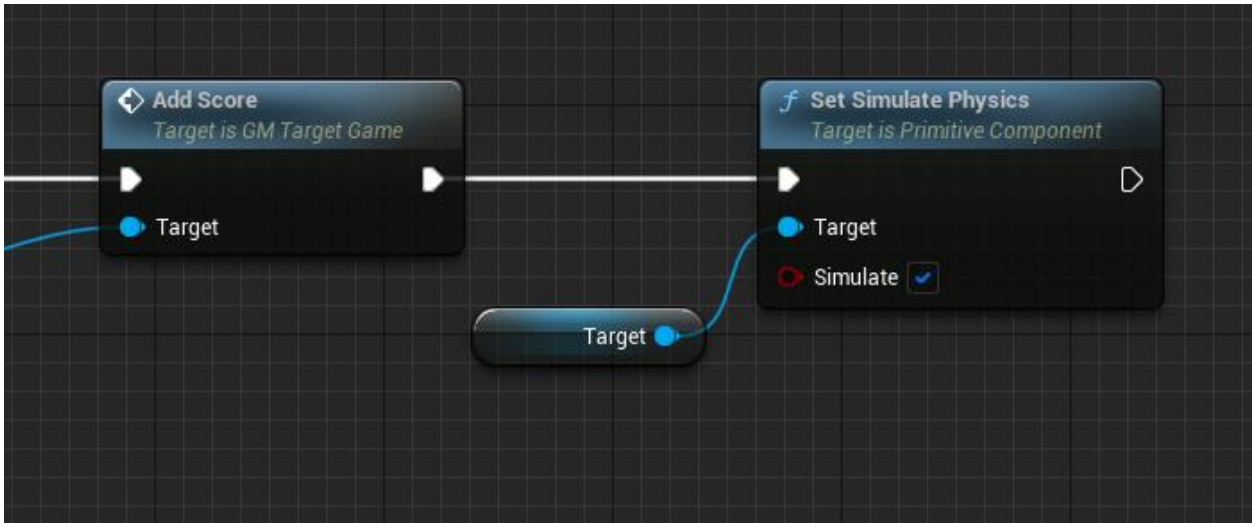

Рис.2.29 Додаємо симуляцію фізики до мішені

Тепер коли фізичні моделі кулі та мішені перетинаються, то включається фізика останнього і під силою тяжіння просто падає. Щоб зробити попадання цікавішим, потрібно додати імпульсу до мішені. Це робиться за допомогою спеціальної функції Add Impulse, яка бере імпульс від кулі та додає до мішені, але ефект недостатньо сильний. Тоді можна

примножити силу за допомогою функції «Multiply». Для більшого враження від гри можна створити ще додаткові прокручення мішені від попадання кулі за допомогою функції Add Torque(Рис.2.30).

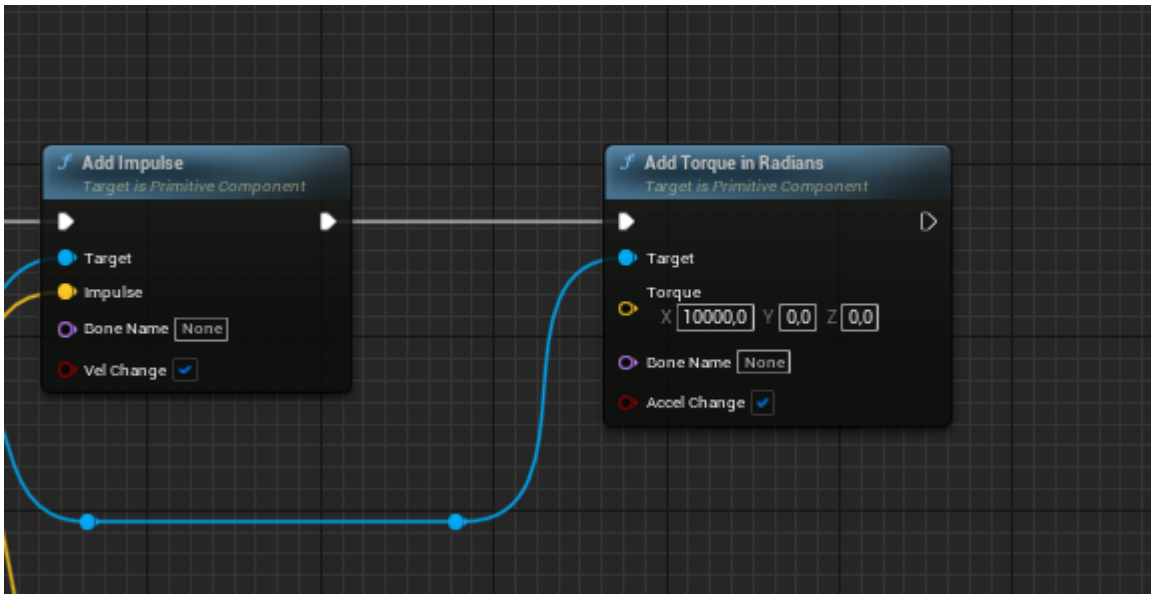

Рис.2.30 Реалізована функція прокручення мішені

Тепер проблема у тому, що ця функція повертає мішень по осі рівня, а не об’єкта, тому замість того, щоб повернутись по осі X, воно буде повертатися як колесо. Щоб це вирішити, потрібно взяти координату об’єкту по якій він буде обертатися, збільшити її силу та надати до функції Add Torque(Рис.2.31).

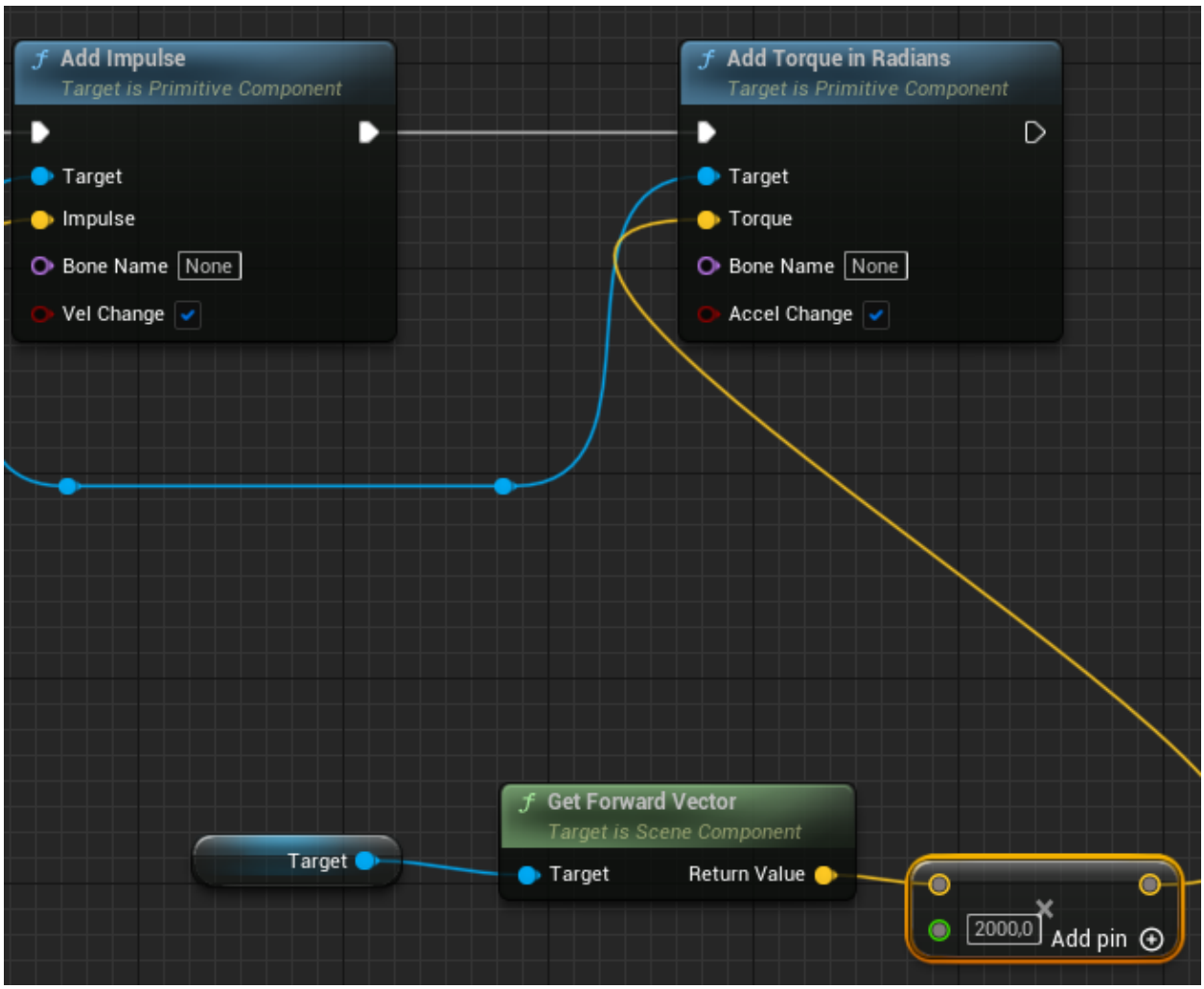

Рис.2.31 Реалізована фізика для мішені.

2.7. Міграція створеного проекту на нову локацію

Тепер усе працює як треба, логіка гри прописана. Зараз можна зробити оточення, арену для гравця. Для цього я завантажу локацію місцевого парку та буду розташовувати там мішені для пострілів. Щоб перемістити усі об’єкти з нашого тестового майданчику, потрібно використати функцію Migrate, яка переносить усі асети з одного проекту до іншого. Після цього потрібно увімкнути наш режим гри у налаштуваннях рівня у новій локації(Рис.2.32).

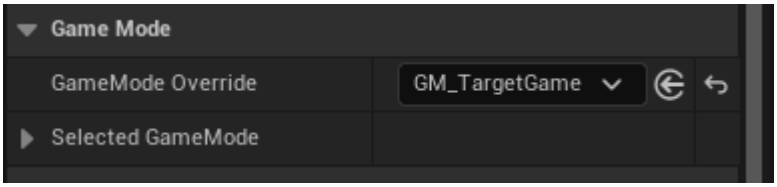

Рис.2.32 Зміна ігрового режиму на рівні

Тепер потрібно розставити мішені, стартову позицію та гвинтівку, без якої не можна стріляти. Я вирішив розмістити їх таким чином(Рис.2.33).

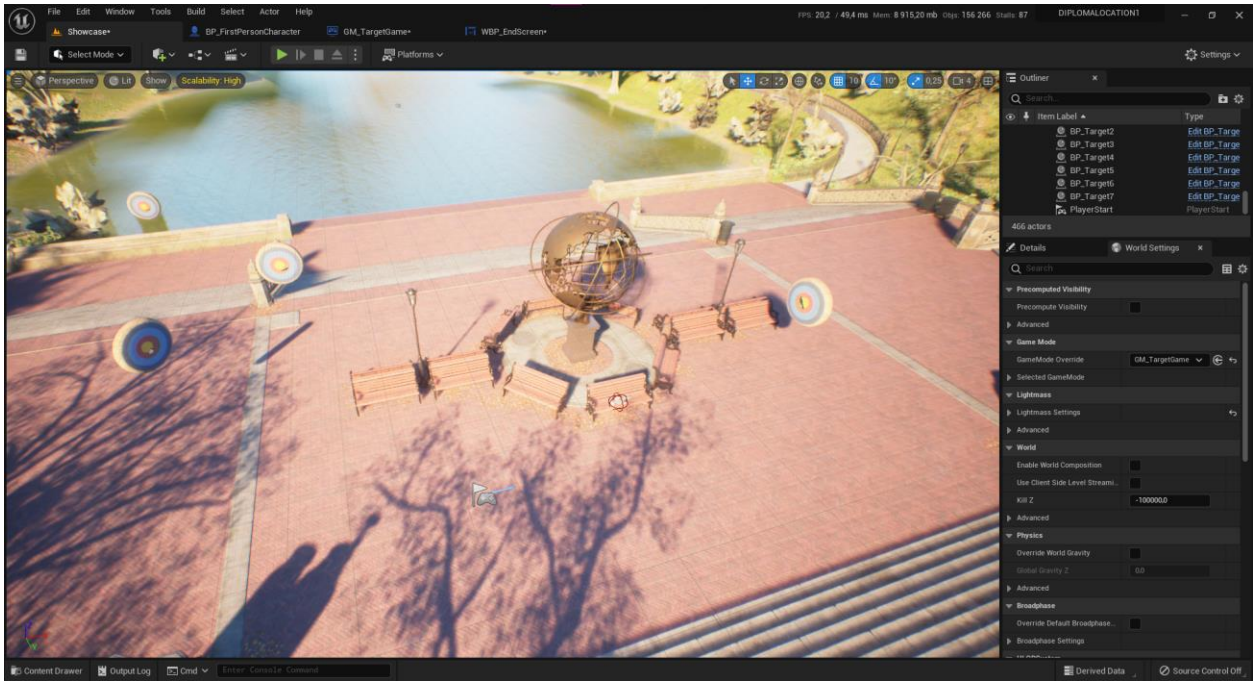

Рис.2.33 Розміщення ігрових акторів на сцені, для перемоги потрібно збити усі мішені

Висновки: було створено інтерактивний додаток, у якому є умова виграшу, можливість керування пішаком та велика просторна локація.

3. Демонстрація створеного додатку та тестування

3.1. Обґрунтування способу запуску додатку

Тепер можна опрацювати готовий додаток. Для цього потрібно запустити додаток. Запустити можна трьома шляхами, це якщо запускати безпосередньо у ігровому рушію, запускати проект рушія з поміткою «Запуск гри», або через підготовку окремого самостійного проекту від рушія, який може запускатися сам в незалежності від встановлення рушія на комп'ютері. Якщо запускати через ігровий рушій, то запуск займає не більше 10-ти секунд, оскільки усі бібліотеки та рівень вже прогружені у пам'ять, потрібно лише подати команду до запуску. Якщо запускати додаток через проект рушія, то запускається додаток, то беруться бібліотеки рушія для запуску гри та програма проекту, вони об'єднуються і запускається ніби самостійна версія гри. Можна також створити окрему самостійну версію, яка не залежить від встановлення рушія на комп'ютері, але кожний раз, коли проходять зміни у проекту гри, то потрібно знову створювати самостійну версію. Рушій гри тоді проходить багато ітерацій завантаження налаштувань для кожного рівня, щоб підготувати освітлення, обробку пост-ефектів та іншого перед тим, як зробити самостійну версію. Останній варіант займає не менше 3 годин реального часу, а версія запуску рівня безпосередньо у рушії проходить через інші етапи перед запуском самої гри, тому я обрав варіант запуску через проект додатку з поміткою «Запуск гри». Він буде і швидше, і буде показувати версію гри, яка нібито самостійно запускається. Також перед запуском гри я встановив спеціальні вузли у Blueprint «GM_TargetGame». Ці вузли відповідальні за те, щоб регулювати налаштування рівня графіки у додатку, щоб можна було протестувати гру на різних налаштуваннях, вони запускаються одразу із запуском цього Blueprint(Рис.3.1).

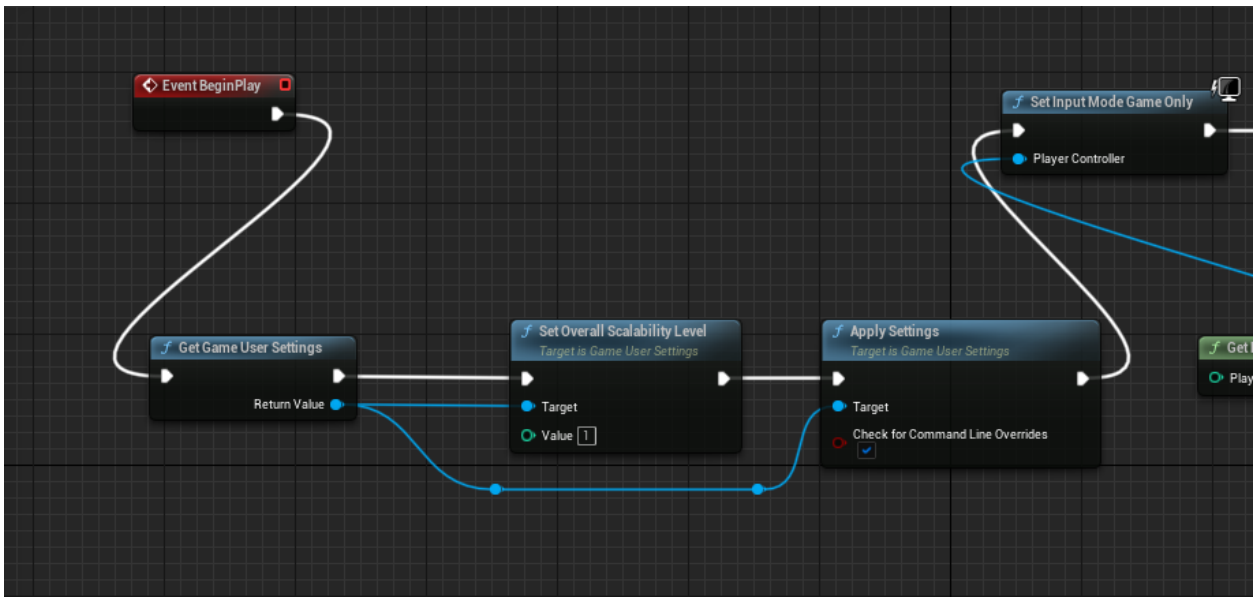

Рис.3.1 Разом з початком завантаження логіки режиму гри, завантажуються доступні налаштування гри, ставиться налаштування графіки від низької до кінематографічної, застосовуються, а потім ідуть інші налаштування рівня

3.2. Тестування різних рівнів графіки

Першим кроком я вирішив протестувати налаштування низької графіки(Рис.3.2)

Рис.3.2 Налаштування низької графіки

Як я помітив, максимум було 65 кадрів за секунду, мінімум 50, тіні майже відсутні, а текстури і полігони були найменшої якості.

Тепер я зміню налаштування графіки на середню(Рис.3.3).

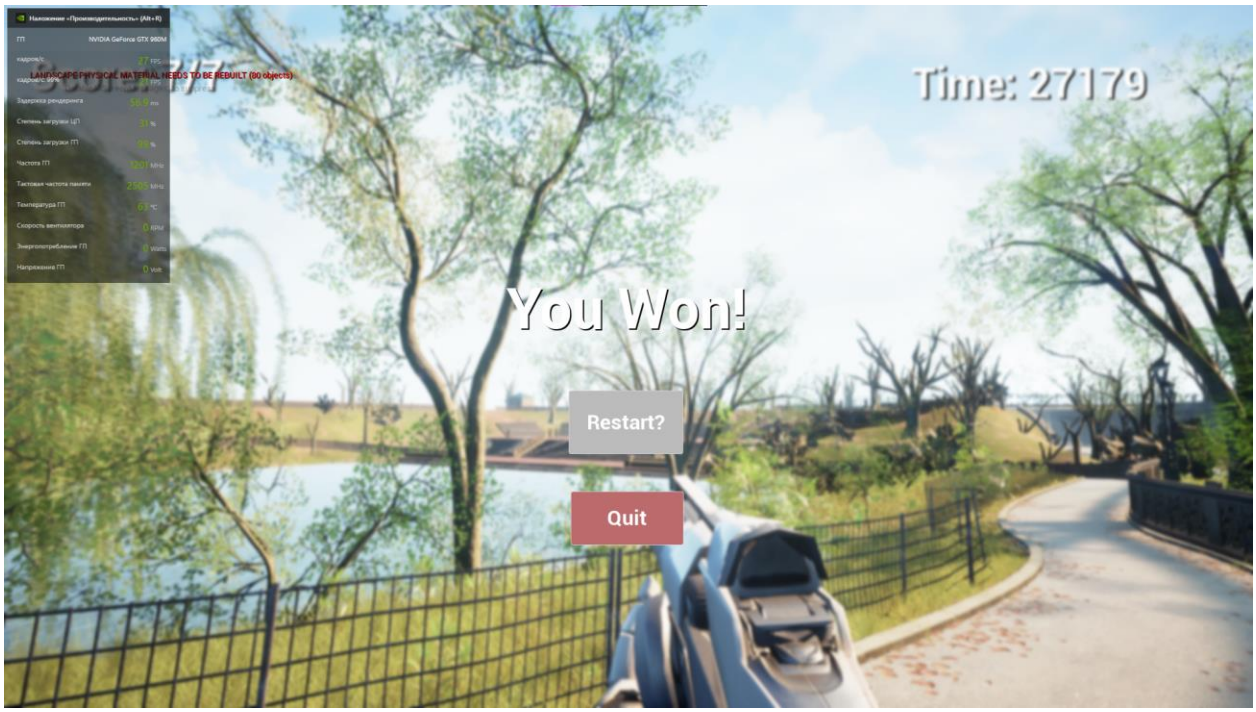

Рис.3.3 Різниця помітна одразу

Мною було помічено, що максимум кадрів за секунду часу було 32, мінімум 25. Ці налаштування знаходяться на порогу комфортної гри на мій погляд, тобто десь 30 кадрів на секунду і при цьому з гарними налаштуваннями графіки.

Наступним кроком, я зроблю високі налаштування графіки(Рис.3.4).

Рис.3.4 Високі налаштування графіки вражають

На високому рівні графіки можна помітити, що гра стала дуже реалістичною, а тіні більш оброблені та деталізовані, також стало краще освітлення, із мінусів – максимальна кількість кадрів на секунду 19, мінімальна кількість 14, тобто грати у швидкі ігри по типу шутерів вже не можна.

Наступним кроком я вирішив протестувати епічний рівень графіки(Рис.3.5)

Рис.3.5 Віддзеркалення стали набагато краще

Ключові особливості цих налаштувань у реалістичності зображення, більш оброблених тінях та променях а також віддзеркаленнях озера. У вдалечині можна помітити Low poly моделі акторів, це також регулюється в налаштуваннях рушія з метою оптимізації ресурсів, кількість кадрів на секунду часу коливалась від 9 до 6.

Тепер запускаю останні, кінематографічні налаштування графіки(Рис.3.6).

Рис.3.6 Останній, кінематографічний рівень графіки

Найкращий рівень графіки, який може обробити мій комп'ютер, реалістичність зображення максимальна, з'являються відчуття, ніби сам тримаєш гвинтівку та стріляєш по мішеням. Максимум комп'ютер оброблював 4 кадра за секунду, мінімум 2 кадри.

Виходячи з цих отриманих даних, можна зробити таблицю порівняння графіки з середньою кількістю оброблених кадрів за секунду часу(Табл.3.1).

Табл. 3.1

Низька графіка, к/с.	Середня графіка, к/с.	Висока графіка, к/с.	Епічна графіка, к/с.	Кінематографічна графіка, к/с.
57 к/с	30 к/с	16 к/с	8 к/с	3 к/с

Можна зробити висновки, що цей додаток найкраще себе показує на середніх налаштуваннях графіки, тобто на 30 кадрах за секунду. Цю кількість кадрів за секунду я вважаю найменш можливим для комфортної гри у будь-якому жанрі, в тому числі і у шутерах. На низькій графіці додаток показує найбільшу кількість кадрів за секунду, але в той же час це досягнуто за рахунок відсутніх тіней, полігонів об'єктів та віддзеркалень, що я вважаю не комфортним. Графіка з налаштуваннями високої та вище виглядає дуже

якісно і насичено, але тоді не вистачає кадрів за секунду, що для шутерів найбільш критично.

Чи можливі ще більш якісні налаштування графіки, ніж Кінематографічні? Так, наприклад з технологією оброблення променів з відеокартою NVIDIA 2000 серії та вище. Ця технологія допомагає оброблювати промені, ніби як у реальному світі різної складності. Ще більшої реалістичності може додати технологія Nanite, яка прибирає «рівні обробки деталей» (LOD) та самостійно зменшує кількість полігонів, не зменшуючи візуально якість деталізації на різних відстанях від спостерігача.

Наступним кроком я запущу проект на кінематографічних налаштуваннях графіки та вимкну функцію глобального освітлення Lumen, перевірю швидкодію та різницю у зображеннях(Рис.3.7).

Рис.3.7 Різниця у швидкодії помітна одразу

Без технології Lumen, рушій видає від 7 до 12 кадрів за секунду часу, тобто приріст швидкодії у 3 рази. Найбільша різниця помітна у експозиції, за замовчуванням Lumen, як ми бачимо, робить занадто велику величину, через що тіні пропадають, а світлі ділянки стають ще світліше. На щастя, це регулюється у налаштуваннях.

3.3. Тестування технології Nanite

Наступну технологію, яку я хочу протестувати, буде Nanite. Потрібно подивитися наскільки зміниться швидкодія та якість зображення. Тестування будуть проходити на кінематографічних налаштуваннях графіки без технології Lumen(Рис.3.8).

Рис.3.8 Як можна побачити, різниця невелика

Якщо дивитися на поточному скріншоті, то різниця між до включення технології і після не дуже помітна, але це тільки на перший погляд, адже коли рушій без технології просто понижує деталізацію до низької, а потім і зовсім видаляє об’єкт, то з технологією у великих містах дійсно видно безшовність переходу моделі здалека до близької відстані.

3.4. Тестування функцій додатку

Тепер потрібно перевірити чи працюють усі функції додатку. Спочатку подивимось на процес взяття гвинтівки(Рис.3.9).

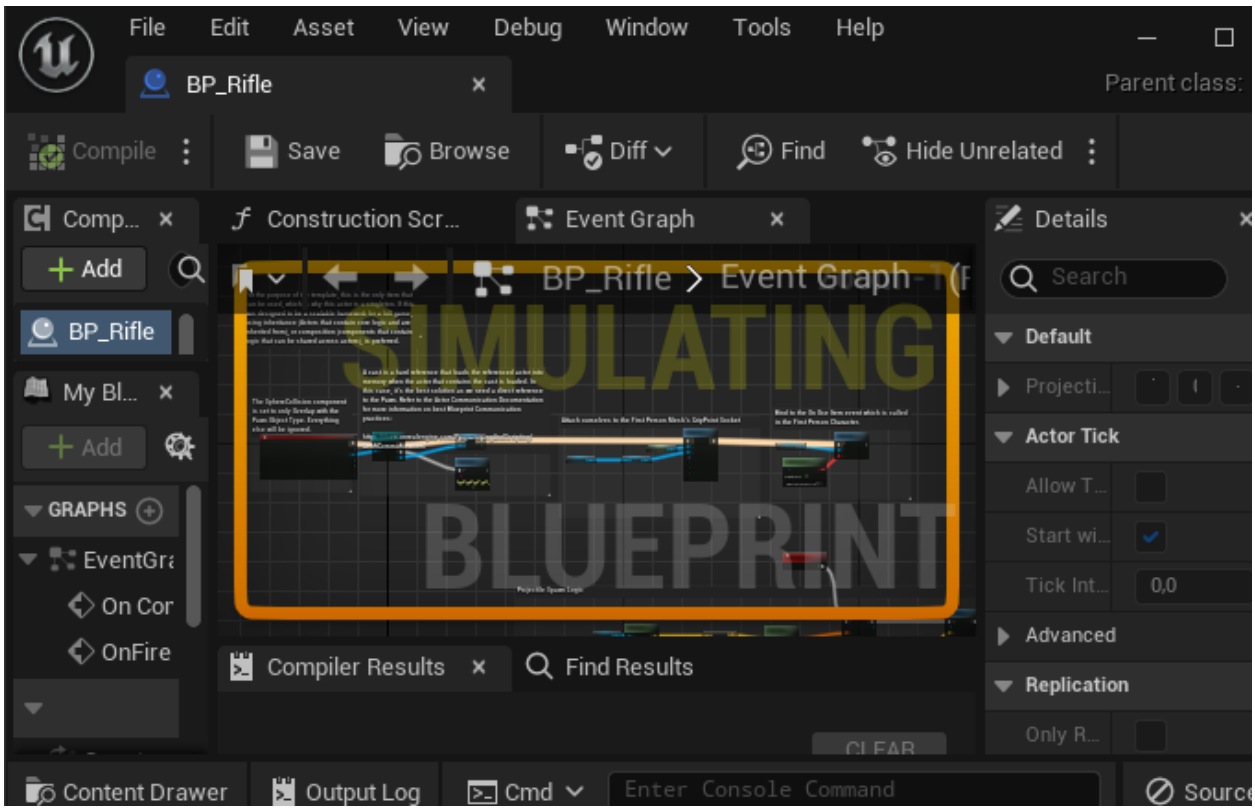

Рис.3.9 Через встроєнний дебагер можна побачити усю інформацію у реальному часі

Як ми бачимо, спрацювала функція підняття гвинтівки, тобто коли модель гравця перетинає модель гвинтівки, то спрацьовує функція, при якій гвинтівка стає на певне місце і дозволяє гравцю стріляти з неї(Рис.3.10)

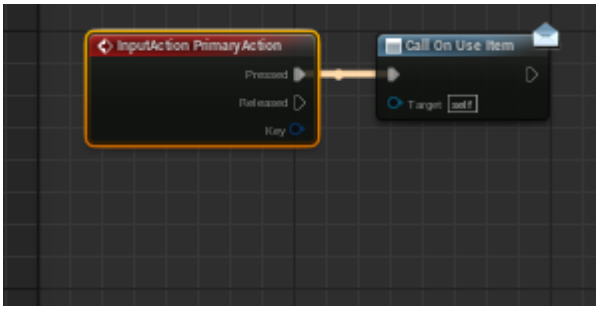

Рис.3.10 Система дійсно фіксує натискання кнопки вогонь, але поки не спрацює функція взяття гвинтівки, вона не спрацює

Тепер потрібно перевірити чи працює реєстрація попадань по мішені. Якщо я стрільну з гвинтівки по мішені, то має спрацювати функція, яка відповідає за реєстрацію попадань, додати один до очок влучених мішеней і

перевірити чи співпадає кількість влучених мішеней з кількістю мішеней на рівні(Рис.3.11)

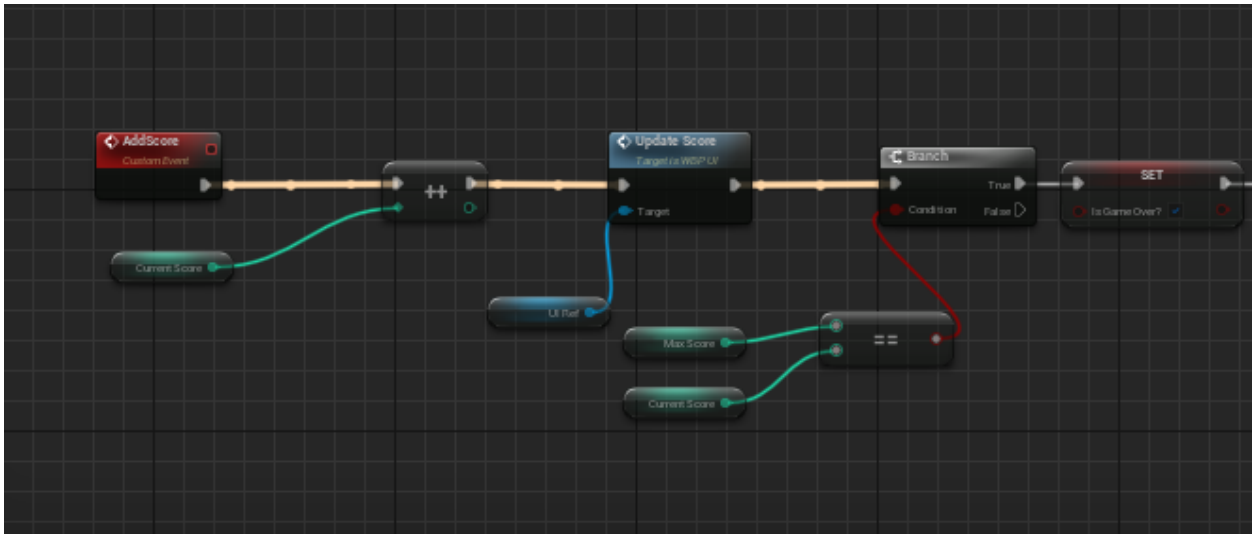

Рис.3.11 Рушій дійсно зареєстрував попадання, оновив кількість зарахованих очок гравця, передав дані про поточну кількість влучених мішеней та перевірив чи співпадає загальна кількість мішеней з вже влученою

Тепер потрібно перевірити чи працює система, якщо гравець влучить у всі мішені(Рис.3.12).

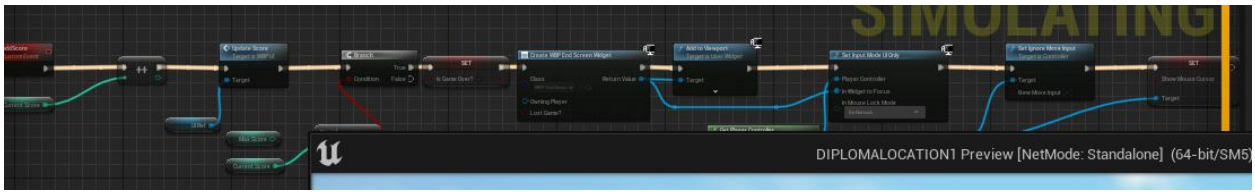

Рис.3.12 Умова виграшу зафіксована

Для тесту, я зробив умовою виграшу влучення у одну мішень, а не усі 7 на рівні. Як ми бачимо, система зареєструвала влучення у мішень і перевірила, що влучена мішень співпадає кількості необхідних мішеней для виграшу, тобто одна. Після цього, рушій змінив значення змінної «IsGameOver?», яка відповідає за те чи закінчилась гра, на 1, щоб таймер більше не зменшувався, бо якщо цього не зробити, то гравець вже виграв, буде надпис «You Won!», але через деякий час буде надпис «You Lost!», бо час таймеру закінчився. Далі рушій посилає сигнал на створення інтерфейсу закінченної гри, відображення його на екрані, відключає керування пішаком і

передає на курсор миши. Якщо до події виграшу пішак рухався, то функція «SetIgnoreMoveInput» відключає команди керування пішаком і таким чином пішак перестає рухатися. Остання функція відповідає за те, щоб показати курсор миші на екрані(Рис.3.13).

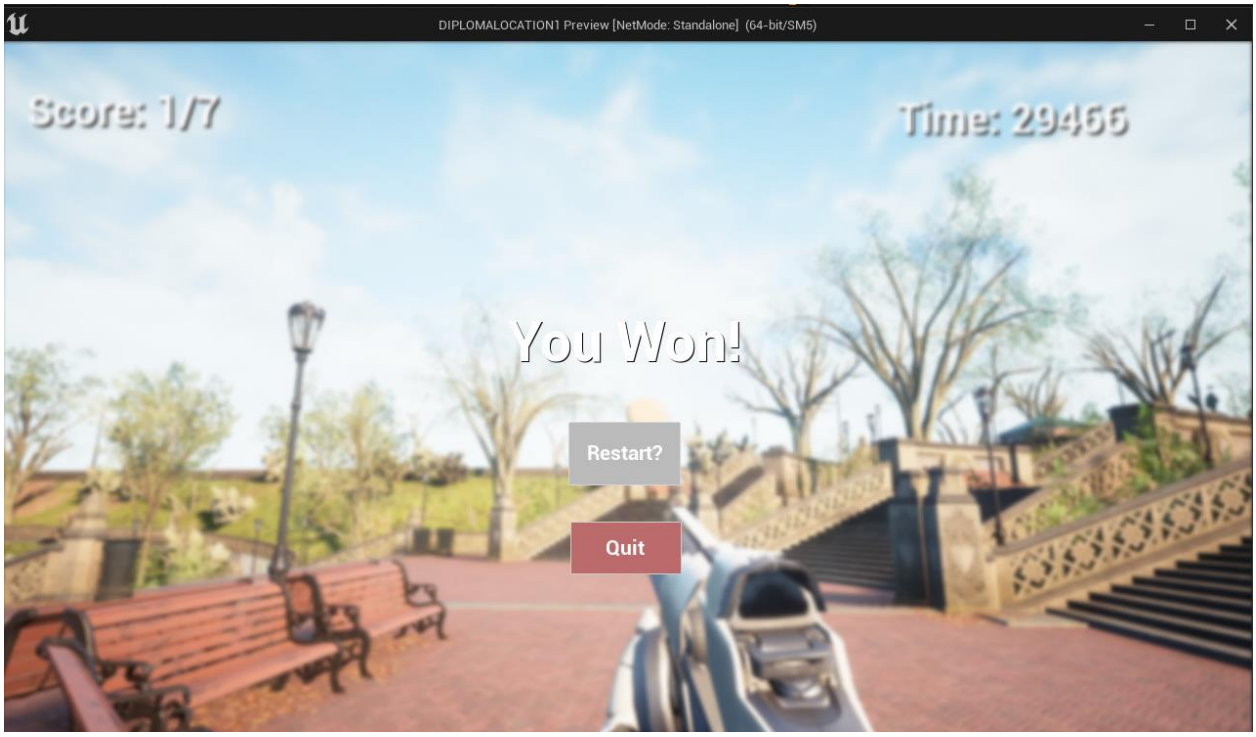

Рис.3.13 Функція виграшу працює, інтерфейс виграшу відобразив надпис «You Won!» разом з кнопками рестарту рівня та вихід з гри

Тепер потрібно протестувати чи працює функція програшу так, як задумано для програми. Для цього потрібно НЕ влучити в усі цілі та просто дочекатись кінця таймеру. Для зручності, я зменшу його до менших значень(Рис.3.14).

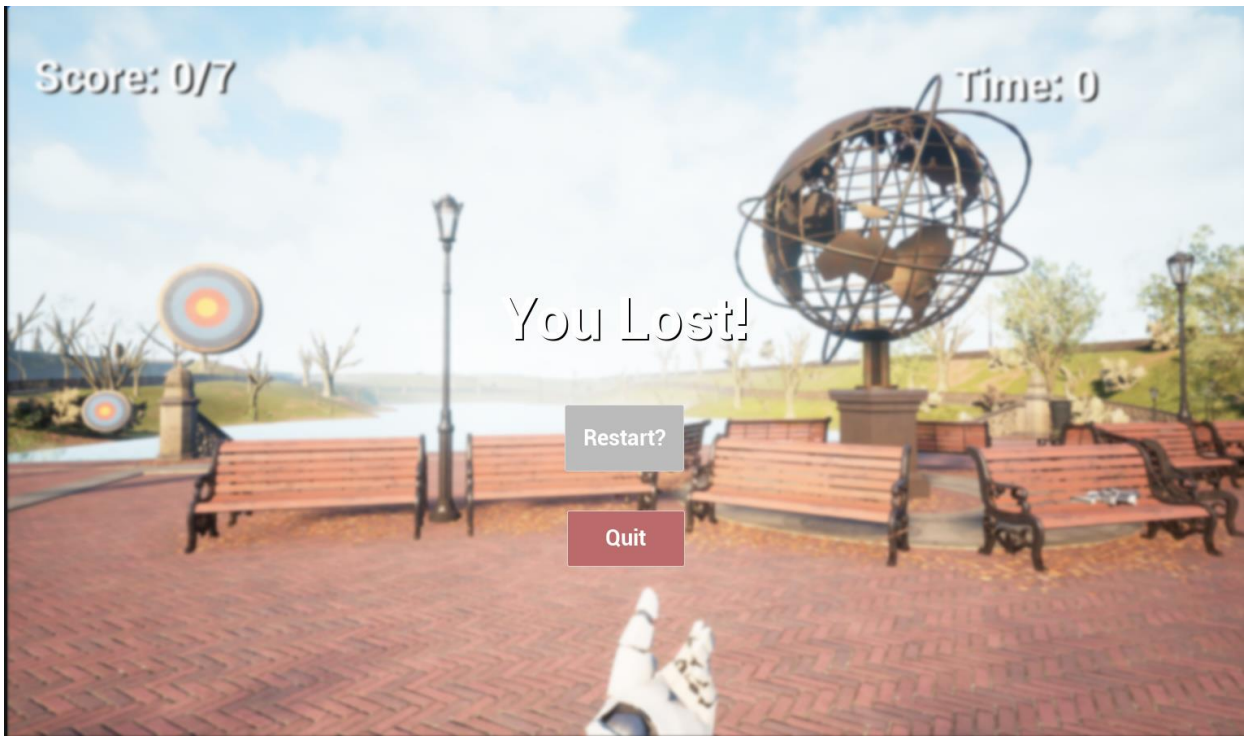

Рис.3.14 Функція програшу спрацювала як потрібно

Ще одна важлива функція – вихід з гри. Вона з’являється тільки коли гравець виграв чи програв, коли він грає, то визвати цю функцію він не може. Як задумувалось, при натисненні на кнопку «Quit», гра закривається, а усі процеси припиняються.

Тепер можна подивитися які зараз баги присутні у грі.

1. Гравець може вільно пройти у паркан(Рис.3.15)

Рис.3.15 Гравець точно не має тут бути

2. Гравець може вільно пересуватися по мапі навіть до країв і бачити пустий простір. Потрібно або створити невидимий бар'єр, або зменшити розмір мапи та налаштувати зовнішній вигляд кордонів(Рис.3.16).

Рис.3.16 Рівень має чимось закінчуватись, або хоча б мати обгрунтування чому немає кордонів

3. Маленькі дерева та кущі мають прострілюватись, а не бути перешкодою для куль гвинтівки. Як мінімум, мною не задумувались кулестійкі кущі(Рис.3.17).

Рис.3.17 Невеликі дерева та кущі точно мають пробиватися

4. Дуже велике значення експозиції від технології Lumen

3.5. Плани на розвиток проекту

Тепер коли створенний додаток досліджен, то можна подивитися на те як розвинути його далі. У планах у мене є декілька ідей:

1. Можна збільшити мапу для використання, а не обмежуватись тільки невеликим шматком, бо зараз прогружається у пам`ять комп`ютера усе, навіть якщо воно не знаходиться у полі зору.
2. Разом зі збільшенною мапою, можна додати транспорт до гри. Зробити наприклад так, що щоб виграти, потрібно влучити у цілі не тільки на цій локації, а і ще на декількох місцях.
3. Зробити мішені такими, щоб вони ще й рухалися

Висновки: у цьому розділі було досліджено створений додаток, протестовано функціональні можливості як рушія, так і частини гейм-дизайну, розроблено план подальших дій на проект.