КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота
на здобуття освітнього рівня бакалавра
за спеціальністю 121 Інженерія програмного забезпечення
на тему:

ДОСЛІДЖЕННЯ ПРОБЛЕМ ПОРТУВАННЯ ІГРОВИХ ЗАСТОСУНКІВ НА МОБІЛЬНІ ПЛАТФОРМИ НА ПРИКЛАДІ UNREAL ENGINE

Виконав студент 4-го курсу
Степан ФЕНДЬО

_____
(підпис)

Науковий керівник:
канд. фіз.-мат. наук, доцент
Євгеній ІВАНОВ

_____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Роботу розглянуто й допущено до захисту
на засіданні кафедри інтелектуальних програмних систем
« 29 » травня 2023 р.,
протокол № 11
Завідувач кафедри
Олександр ПРОВОТАР _____
(підпис)

Київ – 2023

РЕФЕРАТ

Обсяг роботи 49 сторінок, 23 ілюстрації, 25 джерел посилань.

ІГРОВИЙ ЗАСТОСУНОК, ІГРОВИЙ РУШІЙ, ВИДИ ІГРОВИХ РУШІЇВ, МОБІЛЬНІ ПЛАТФОРМИ, ПОРТУВАННЯ ІГРОВИХ ЗАСТОСУНКІВ, UNREAL ENGINE, ОПТИМІЗАЦІЯ ІГРОВОГО ЗАСТОСУНКУ, ПРОФАЙЛІНГ.

Об’єктом роботи є аналіз проблем портування ігрових застосунків на мобільні платформи. Предметом роботи є розробка рекомендацій щодо ефективного портування ігор на мобільні девайси з використанням рушія Unreal Engine.

Метою роботи є дослідження проблематики портування ігрових застосунків на мобільні платформи.

За методами розробки робота виконувалася сумісно з роботами у сфері гейм-девелопменту, які пов’язані з портуванням ігрового застосунку на мобільні платформи з використанням Unreal Engine.

Інструменти розроблення: безкоштовний, вільно поширюваний рушій UnrealEngine, мова програмування C++, середовище розробки JetBrains Rider, Android Studio.

Результати роботи: визначено сутність, функції та можливості ігрового рушія; сформульовано переваги та недоліки різних видів ігрових рушіїв; проаналізовано переваги та недоліки використання Unreal Engine для портування ігрових застосунків на мобільні платформи; здійснено процес портування ігрового застосунку на мобільні платформи з використанням Unreal Engine; проведено оптимізацію ігрового застосунку під мобільні платформи; розроблено рекомендацій щодо ефективного портування ігор на мобільні девайси.

Розроблені рекомендації щодо ефективного портування ігрових застосунків на мобільні платформи з використанням Unreal Engine можуть використовуватись гейм-девелоперами для вирішення подібного кола проблем.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

ADB – Android Debug Bridge;

API – Application Programming Interface;

HUD – англ. heads-up display, користувацький інтерфейс, що знаходиться на екрані під час самого процесу гри;

JDK – Java Development Kit;

NDK – Native Development Kit;

Rendering – процес обробки та виведення зображення на екран;

SDK – Software Development Kit;

UI/UX – UI (англ. User Interface – користувацький інтерфейс) та UX (англ. User Experience – користувацький досвід), дизайн та взаємодія користувача з продуктом, яке спрямоване на створення зручного та задоволеного користувацького досвіду;

ОС – операційна система;

ПК – персональний комп’ютер;

Логи – англ. Logs, список системних та програмних повідомлень, що повідомляють розробника про теперішній стан застосунку та операційної системи;

Шейдер – програма що виконується на GPU та відіграє роль у відображенні графічних об’єктів, ефектів та матеріалів, а також відповідає обчислення моделі освітлення.

ВСТУП

Оцінка сучасного стану об’єкта розробки. У сучасному світі мобільні пристрої, зокрема смартфони, планшети, смартгодинники є невід'ємною частиною повсякденного життя мільйонів людей. З кожним роком популярність цифрових гаджетів невпинно зростає, що обумовлено безперешкодним і необмеженим доступом до мережі Інтернет, багатством віртуальної реальності, поліформатністю комунікацій, а також безліччю інших можливостей, які надають ці пристрої для здійснення навчальної та пізнавальної діяльності, розвитку гнучкості та креативності мислення користувачів тощо.

В окрему групу варто виокремити ігрову індустрію та кіберспорт, де мобільні девайси на операційних системах iOS, Android, Windows Phone, Blackberry широко застосовуються користувачами для аматорських або професійних змагань, залежно від рівня ігрового досвіду. В розрізі цього, впродовж останніх років активно розвивається сфера гейм-девелопменту, яка визначає процес розробки ігрових застосунків під конкретну мобільну ігрову платформу.

Актуальність роботи та підстави для її виконання. Одним із ключових аспектів успішного впровадження ігрових застосунків на ринку ігрової індустрії є наявність версії для мобільних платформ. Саме тому портування існуючих ігор на мобільні платформи стає актуальним завданням для розробників.

Портування ігрових застосунків на мобільні платформи означає адаптацію ігрового контенту та механік для оптимальної роботи на пристроях з обмеженими обчислювальними ресурсами, такими як нижчі ніж в ПК потужність процесора, оперативна пам'ять та графічні можливості. Це вимагає не лише змін у програмному коді гри, але і врахування особливостей взаємодії з користувачем на сенсорних екранах, адаптації управління та оптимізації продуктивності.

Портування ігрових застосунків на мобільні платформи має кілька вагомих переваг. По-перше, це розширення цільової аудиторії гравців. Мобільні

пристрої є постійно доступними та мобільними, що робить ігрові застосунки більш доступними для широкого кола користувачів. Крім того, мобільні гравці відрізняються своїми уподобаннями та геймінговими звичками, що відкриває нові можливості для створення та монетизації ігор.

По-друге, портування ігрових застосунків на мобільні платформи сприяє розвитку індустрії гейм-девелопменту. Завдяки швидкому росту ринку мобільних ігор з'являється потреба у талановитих програмістах та гейм-дизайнерах, що стимулює розширення команди розробників та залучення нових фахівців до галузі.

Однак, портування ігрових застосунків на мобільні платформи ставить перед гейм-розробниками низку викликів та обмежень, які обов’язково потрібно враховувати для успішної реалізації задуму. Технічні обмеження мобільних пристроїв можуть вимагати компромісів у якості графіки та фізики гри, а також призвести до складності ігрових механік. Крім того, розробники ігрових застосунків повинні враховувати особливості мобільних платформ, серед яких: різні розміри екранів, різні способи вводу та взаємодії з грою.

Аналіз існуючих проблем та викликів, з якими стикаються гейм-розробники під час портування ігор, дозволить визначити стратегії та методи для успішної розробки ігрових застосунків під конкретну мобільну ігрову платформу. Також в процесі дослідження буде проаналізовано вплив портування на індустрію (гейм-девелопменту) та розглянуто перспективи розвитку мобільного геймінгу. Саме тому дослідження проблем портування ігрових застосунків на мобільні платформи є актуальним.

Мета й завдання роботи. Метою кваліфікаційної роботи є дослідження проблематики портування ігрових застосунків на мобільні платформи з використанням Unreal Engine. Для досягнення поставленої мети необхідно виконати наступні завдання:

1. Здійснити науковий аналіз змісту поняття ігровий рушій та дослідити його роль у створенні ігрових застосунків;

2. Проаналізувати використання Unreal Engine для портування ігрових застосунків на мобільні платформи;

3. Дослідити процес портування ігрового застосунку на мобільні платформи з використанням Unreal Engine;

4. Обґрунтувати інструменти оптимізації та профайлінгу ігрового застосунку під мобільні платформи.

Об'єкт, методи й засоби розроблення. Об’єктом дослідження є аналіз проблем портування ігрових застосунків на мобільні платформи.

За методами розроблення робота виконувалася сумісно з роботами у сфері гейм-девелопменту, які пов’язані з портуванням ігрового застосунку на мобільні платформи з використанням Unreal Engine.

Інструменти розроблення: безкоштовний, вільно поширюваний рушій UnrealEngine, мова програмування C++, середовище розробки JetBrains Rider, Android Studio.

Можливі сфери застосування. Розроблені рекомендації щодо ефективного портування ігрових застосунків на мобільні платформи з використанням Unreal Engine можуть використовуватись гейм-девелоперами для вирішення подібного кола проблем.

Взаємозв’язок з іншими роботами. Розроблені рекомендації щодо ефективного портування ігрових застосунків на мобільні платформи з використанням Unreal Engine можуть використовуватись гейм-девелоперами для вирішення подібного кола проблем.

РОЗДІЛ 1. ІГРОВИЙ РУШІЙ ТА ЙОГО РОЛЬ У РОЗРОБЦІ ІГРОВИХ ЗАСТОСУНКІВ

1.1 Визначення ігрового рушія

Ігровий рушій є ключовим компонентом при розробці ігрових застосунків і виконує низку важливих функцій, що впливають на процес створення та функціонування ігрового додатку. У науковій літературі ігровий рушій визначається як «програмне забезпечення, яке надає набір інструментів та функцій для розробки ігор» [1-3].

Використання ігрового рушія має декілька вагомих переваг. По-перше, він дозволяє гейм-девелоперам зосередитися на самому процесі створення гри, що суттєво заощаджує час на розробку базового двигуна. Ігровий рушій пропонує готову архітектуру, яка включає в себе такі основні компоненти гри, як фізика, графіка, штучний інтелект, звукові ефекти тощо.

По-друге, ігровий рушій спрощує розробку ігрових застосунків шляхом надання інструментів і ресурсів, що дозволяють створювати та налаштовувати графіку, анімацію, фізичну модель та інші складові гри. Це дозволяє гейм-девелоперам швидко прототипувати та впроваджувати нові ідеї, зосереджуючись на творчому процесі, без необхідності вирішувати технічні деталі з нуля.

Слід зазначити, що на сьогодні на ринку геймдеву існує безліч ігрових рушіїв, кожен з яких має свої переваги та недоліки. Найпопулярніші серед них – Unity, Unreal Engine, CryEngine, GameMaker та інші. Кожен рушій має свої особливості, такі як масштабованість, гнучкість, легкість використання, доступність ресурсів та підтримка платформ.

Одним з рушіїв, який викликає особливий інтерес та широке застосування в індустрії геймдеву, є Unreal Engine. Unreal Engine від Epic Games [4, 5] є потужним рушієм, який володіє великою кількістю інструментів для розробки ігор, а також широким спектром підтримуваних платформ, включаючи мобільні

пристрої. Обираючи Unreal Engine для нашої роботи, ми бачимо його переваги з точки зору функціональності, гнучкості, спільноти розробників та широкої підтримки платформ, що дозволить нам вивчати проблеми портування ігор на мобільні платформи з використанням цього рушія.

1.2 Функції та можливості ігрового рушія

Основна мета ігрового рушія полягає у спрощенні та оптимізації процесу розробки ігрових застосунків. Він надає засоби для створення ігрових об'єктів, обробки графіки та звуку, фізичної симуляції, реалізації штучного інтелекту, управління ресурсами та багато інших функціональностей, які є важливими для розробки ігор.

Проаналізуємо найважливіші функції ігрового рушія:

1. Забезпечення реалізації геймплею: ігровий рушій надає розробникам інструменти для створення ігрових об'єктів, логіки гри, фізики та інших елементів геймплею. Він дозволяє реалізувати інтерактивність та управління грою, створюючи основні механіки та правила.

2. Візуалізація та графіка: ігровий рушій надає інструменти для створення та відображення графічного контенту в грі. Він дозволяє розробникам створювати реалістичну графіку, використовуючи різні технології, такі як 2D або 3D графіка, освітлення, ефекти частинок та інші візуальні ефекти.

3. Звук та музика: ігровий рушій надає інструменти для відтворення звуків, музики та звукових ефектів у грі. Він дозволяє розробникам створювати атмосферу та іммерсію за допомогою аудіо компонентів, таких як звуки оточення, діалоги персонажів та музичний супровід.

4. Управління ресурсами: ігровий рушій відповідає за управління ресурсами, такими як пам'ять, текстури, моделі персонажів та інші активи гри. Він дозволяє ефективно використовувати обмежені ресурси пристроїв та забезпечує оптимізацію продуктивності гри.

Слід зазначити, що ігровий рушій є важливою складовою під час розробки ігрових застосунків, оскільки надає розробникам потужні інструменти та функціональність для створення ігрового контенту. Таким чином, він полегшує процес розробки ігрового застосунку, дозволяючи розробникам фокусуватися на творчому процесі та геймплеї, забезпечуючи при цьому високу якість графіки, звуку та продуктивності гри.

1.3 Види ігрових рушіїв: огляд, переваги та недоліки

На сьогодні існує велика кількість різноманітних ігрових рушіїв і одразу обрати той, який підійде найкраще, буває досить складно. Адже різні мови програмування, платформи, функції, які надаються, суттєво ускладнюють вибір. І хоча базовий перелік функцій у всіх спільний, все ж конкретні реалізації сильно відрізняються між собою. Деякі є більш універсальними, однак менш продуктивними, інші ж створені для певних жанрів ігор, але можливо їх інструментарій є набагато вужчим.

Зараз на ринку ігрових рушіїв домінуючими є Unity, Unreal Engine та CryEngine. Проаналізуємо більш детальніше кожен з ігрових рушіїв.

Unity [6-8] – один з найпопулярніших ігрових рушіїв, відомий своєю доступністю та простотою використання. Він підтримує широкий спектр платформ, включаючи мобільні пристрої, і має велику спільноту розробників. Основні переваги Unity полягають у швидкості розробки, кросплатформеності і можливості легко втілити різноманітні ідеї. Проте, недоліками Unity є обмежені можливості графічного двигуна та проблеми з продуктивністю при реалізації великих проектів.

Unreal Engine [9, 10] – ігровий рушій, розроблений компанією Epic Games. Цей рушій володіє потужними графічними можливостями, забезпечує високу якість візуальних ефектів та фотореалістичність. Unreal Engine також підтримує мобільні платформи, що робить його привабливим для портування ігор. Перевагами Unreal Engine є потужний графічний двигун, розширені можливості

програмування, широка функціональність та підтримка віртуальної реальності. Однак, розробка ігрових застосунків у Unreal Engine може бути складнішою та вимагати більше часу в порівнянні з іншими рушіями.

CryEngine [11, 12] – ігровий рушій, створений компанією Crytek. Цей рушій відомий своїми високоякісними графічними можливостями та реалістичною фізикою. Він також підтримує мобільні платформи, що дозволяє розробляти ігрові застосунки для мобільних пристроїв. Основними перевагами CryEngine є відмінна графіка, потужна система розсіяного освітлення та фотореалістичність. Проте, недоліками є складність використання та високі вимоги до апаратного забезпечення.

Кожен ігровий рушій має свої переваги та недоліки, і вибір підходящого рушія залежить від специфіки проекту та вимог розробки. Для реалізації завдань нашого дослідження ми обрали Unreal Engine з урахуванням його потужних графічних можливостей, широкої підтримки мобільних платформ та доступної функціональності, що дозволить дослідити проблеми портування ігрових застосунків на мобільні пристрої максимально ефективно.

РОЗДІЛ 2. ВИБІР РУШІЯ ДЛЯ ПОРТУВАННЯ ІГРОВОГО ЗАСТОСУНКУ НА МОБІЛЬНІ ПЛАТФОРМИ

2.1 Аналіз Unreal Engine та його особливостей

У цьому розділі розглянуто Unreal Engine – ігровий рушій, обраний для портування ігрових застосунків на мобільні платформи. Проаналізуємо його основні особливості та можливості, які роблять його популярним серед розробників.

Unreal Engine є потужним ігровим рушієм, розробленим компанією Epic Games. Він володіє широким спектром функцій та інструментів, які дозволяють розробникам створювати вражаючі ігри для різних платформ, включаючи мобільні пристрої.

Однією з основних переваг Unreal Engine є його потужний графічний двигун. Він надає розробникам доступ до високоякісних графічних ефектів, фотореалістичності та деталізації. Багатошарова система освітлення дозволяє створювати реалістичні світи зі світлом, його відображенням від різних типів поверхонь, а також тінями. Крім того, Unreal Engine підтримує використання візуалізації на основі фізичних матеріалів, що дозволяє реалістично відтворювати матеріали з різними властивостями поверхні, такими як метал, скло, дерево і т.д.

Unreal Engine також має розширені можливості програмування. Розробники можуть використовувати мову програмування C++ для створення складних логічних систем, штучного інтелекту, фізики та багатьох інших ігрових функцій. Unreal Engine надає зручне середовище для візуального програмування, що дозволяє розробникам створювати скрипти та взаємодіяти з різними об'єктами в грі без прямого кодування.

Особливо зручною є також мультиплатформена підтримка Unreal Engine. Розробники можуть створювати ігри, які працюють на різних платформах, включаючи Windows, macOS, Linux, консолі Xbox, PlayStation, а також мобільні

пристрої з ОС Android та iOS. Це значно спрощує розробку ігрових застосунків із множинними цільовими платформами.

Загалом, Unreal Engine є потужним ігровим рушієм з розширеними можливостями програмування та вражаючим графічним двигуном. Він забезпечує розробникам все необхідне для створення високоякісних ігрових застосунків на мобільних платформах. Використання Unreal Engine для портування ігор на мобільні пристрої може значно спростити процес розробки та дозволить досягти вражаючих результатів.

2.2 Переваги використання Unreal Engine для портування ігрових застосунків на мобільні платформи

Unreal Engine, як потужний ігровий рушій, надає розробникам ряд інструментів та можливостей, які полегшують та прискорюють процес портування, а також допомагають досягти високої якості гри [13, 14].

Однією з найбільших переваг Unreal Engine є його кросплатформенність. Розробники можуть створювати гру один раз і запускати її на різних платформах, включаючи мобільні пристрої з ОС Android та ОС iOS. Оскільки в кожної платформи є своя ОС та свій набір апаратних можливостей, то до кожної з них необхідний окремий підхід, який вимагав би великої кількості часу для адаптації програмного коду та графічних ресурсів. Можливості Unreal Engine суттєво заощаджують час та зусилля гейм-девелоперів, оскільки в процесі використання не потрібують розробки окремих версій гри для кожної платформи. Крім того, Unreal Engine забезпечує зручні інструменти для адаптації ігрових застосунків під різні екрани та пристрої, забезпечуючи оптимальний геймплей та візуальний досвід на різних мобільних пристроях.

Ще однією перевагою Unreal Engine є його потужний графічний двигун. Завдяки високоякісній графіці та деталізації, Unreal Engine дозволяє створювати вражаючі візуальні ефекти, реалістичні світи та віртуальну реальність, враховуючи при цьому апаратні обмеження кожної з платформ та пропонуючи

інструменти для первинної оптимізації графічного контенту. Це особливо важливо для мобільних ігрових застосунків, оскільки вони суттєво залежать від графічного враження, яке справляють на геймерів, та завдяки відмінному візуальному досвіду можуть залучити більше користувачів. Разом з тим, варто зазначити, що при всій своїй зручності та доступності у використанні, мобільні ігрові застосунки не володіють такою потужністю, як наприклад настільні ПК чи ігрові консолі, що може суттєво обмежувати їх застосування для складних та масштабних ігор.

Unreal Engine також має широкий спектр вбудованих інструментів та функцій, що полегшують розробку ігрових застосунків для мобільних платформ. Наприклад, він надає доступ до багатьох готових компонентів, таких як система фізики, штучний інтелект, анімація персонажів та багато іншого. Це дозволяє розробникам ефективно використовувати готові рішення і скорочує час, потрібний для розробки гри.

Крім того, Unreal Engine має активну спільноту розробників, що забезпечує підтримку та обмін знаннями. Розробники можуть звертатись до форумів, документації та ресурсів Unreal Engine, щоб отримати допомогу та поради від досвідчених колег. Це стимулює продуктивну співпрацю та колаборацію, дозволяє розробникам вирішувати проблеми, які виникають під час портування ігрових застосунків на мобільні платформи.

Також варто зазначити, що Unreal Engine надає розробникам зручні інструменти для тестування та налагодження ігрових застосунків на мобільних пристроях. Це дозволяє ефективно перевіряти функціональність ігри, а також оптимізувати його під різні мобільні платформи.

Загалом, використання Unreal Engine для портування ігрових застосунків на мобільні платформи має значні переваги. Кросплатформенність, потужний графічний двигун, вбудовані інструменти та активна спільнота розробників роблять Unreal Engine привабливим для розробки якісних ігрових проектів для мобільних пристроїв [15].

Використання Unreal Engine дозволяє гейм-девелоперам зосередитись на творчому процесі та створенні захоплюючого ігрового досвіду для мобільних користувачів. Саме з цих причин для виконання завдань кваліфікаційної роботи був обраний цей рушій.

РОЗДІЛ 3. ПРОЦЕС ПОРТУВАННЯ ІГРОВОГО ЗАСТОСУНКУ НА МОБІЛЬНІ ПЛАТФОРМИ З ВИКОРИСТАННЯМ UNREAL ENGINE

3.1 Налаштування мобільних пристроїв з ОС Android для використання в режимі розробника

У цьому розділі детально проаналізовано процес налаштування пристроїв з ОС Android для розробки ігрових застосунків у середовищі Unreal Engine.

Для успішної роботи з мобільним пристроєм, перше, що необхідно зробити, це увімкнути режим розробника, увімкнути USB-налагодження, встановити Android Debug Bridge (ADB) на персональному комп'ютері та переконатись, що пристрій вдалося підключити до ADB [16].

Існує безліч різноманітних моделей та виробників мобільних пристроїв, тому конкретні назви меню чи їх місцезнаходження можуть відрізнятися, однак у даній інструкції описано процес у загальних термінах, які повинні бути схожими на всіх смартфонах.

1. Увімкнення режиму розробника:

– перейдіть до «Налаштування» (Settings) на вашому Android пристрої;

– прокрутіть вниз і знайдіть «Про телефон» (About Phone) або «Про пристрій» (About Device);

– у цьому розділі знайдіть «Номер збудови» (Build Number) і кілька разів швидко натисніть на нього;

– після декількох натискань з’явиться повідомлення, що режим розробника увімкнено (рисунок 3.1);

2. Увімкнення USB-налагодження:

– після ввімкнення режиму розробника поверніться до головного меню «Налаштування»;

– знайдіть і виберіть «Опції розробника» (Developer Options) або «Системні налаштування» (System Settings > Developer Options);

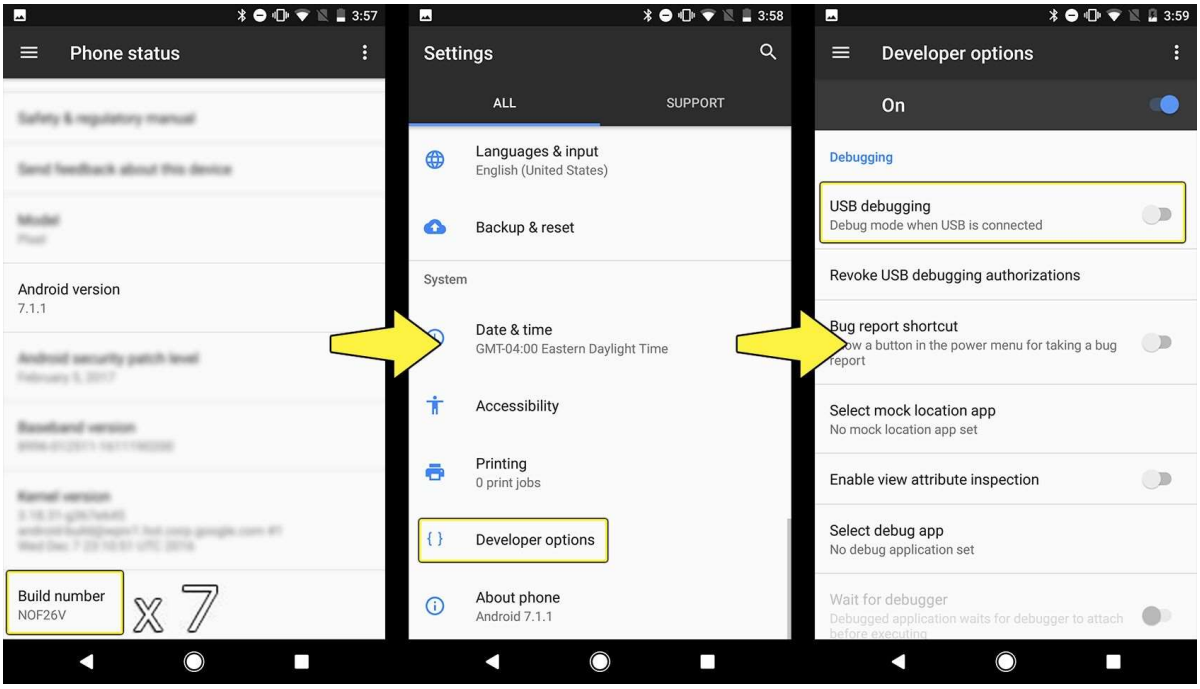

Рисунок 3.1 – Процес увімкнення режиму розробника

– у цьому розділі знайдіть «USB-налагодження» (USB Debugging) і встановіть його в положення «Увімкнено» (Enabled);

– при підключенні пристрою до комп'ютера через USB кабель, буде надано запит на дозвіл налагодження через USB, який необхідно підтвердити;

3. Встановлення Android Debug Bridge (ADB):

– завантажте Android SDK Platform Tools з офіційного сайту розробників Android (URL: https://developer.android.com/studio/releases/platform-tools);

– розархівуйте завантажений архів на вашому персональному комп'ютері в зручне місце;

– додайте шлях до розархівованої папки з ADB до змінної середовища PATH вашої ОС. Це дозволить викликати команду ADB з будь-якої папки у командному рядку;

4. Перевірка підключення пристрою до ADB:

– підключіть свій Android пристрій до персонального комп'ютера за допомогою USB кабелю;

– відкрийте командний рядок (Command Prompt) на вашому персональному комп'ютері (рисунок 3.2);

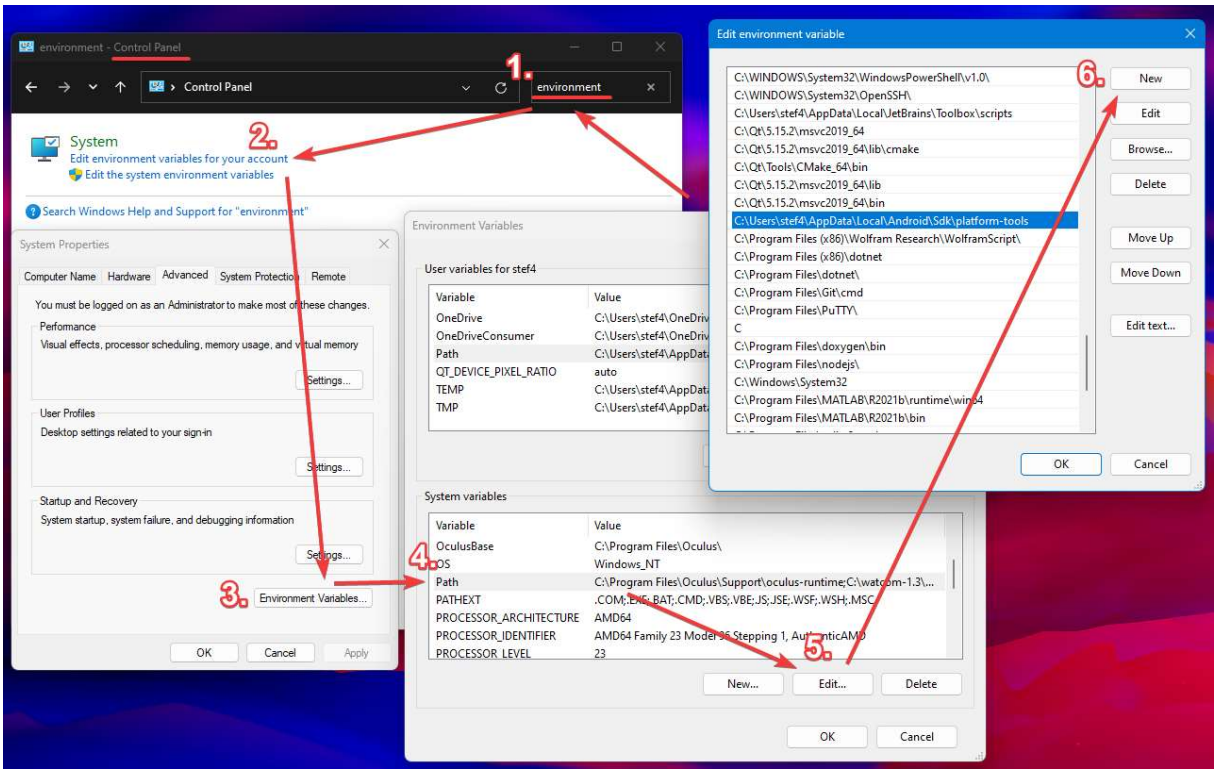

Рисунок 3.2 – Процес додавання інструментів ADB до системної змінної PATH

– введіть команду «Adb Devices» і натисніть Enter (рисунок 3.3);

– якщо все пройшло успішно, повинен з’явитися список підключених пристроїв, включаючи ваш Android пристрій.

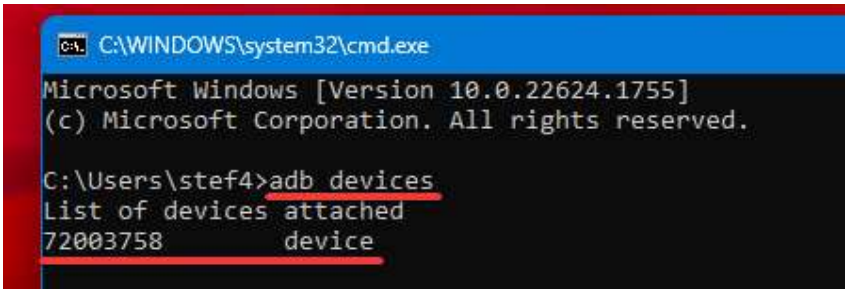

Рисунок 3.3 – Приклад відображення правильно налаштованого пристрою

Завдяки налаштуванню режиму розробника, USB-налагодження та встановленню ADB на конкретному персональному комп'ютері, користувач зможе інсталювати свій ігровий застосунок на пристрій, не публікуючи його в

Google Play Store, а також мати можливість дебажити (англ. Debugging – пошук несправностей в програмі) свій додаток, мати доступ до статистики та логів. Всі вищезазначені кроки необхідні кожному розробнику, який створює ігрові застосунки для мобільних пристроїв на ОС Android.

3.2 Налаштування середовища Unreal Engine для розробки під ОС Android

Для ефективної розробки ігрових застосунків для мобільних пристроїв з ОС Android з використанням середовища Unreal Engine, необхідно провести налаштування шляхів до трьох основних компонентів: Software Development Kit (SDK), Native Development Kit (NDK) та Java Development Kit (JDK). Кожен з цих компонентів має свою функціональність та важливий внесок у процес розробки ігрових застосунків під ОС Android.

1. SDK (Software Development Kit) [17, 18]: є набором інструментів, бібліотек та ресурсів, які надаються компанією Google для розробки програмного забезпечення під ОС Android. Він включає в себе різні API (Application Programming Interface), документацію та засоби для створення, тестування та відлагодження програм під платформу Android. SDK надає розробникам доступ до функціональних можливостей пристроїв, таких як екран, камера, сенсори та багато іншого. Встановлення шляху до Android SDK у середовищі Unreal Engine дозволяє розробникам використовувати всі можливості Android-платформи при створенні ігрових застосунків.

2. NDK (Native Development Kit) [18, 19]: є набором інструментів, які дозволяють розробникам використовувати мову C++ для написання частини програмного забезпечення під ОС Android. У розробці ігрових застосунків це особливо корисно, оскільки деякі частини гри можуть бути оптимізовані шляхом написання високоефективного C++ коду. NDK забезпечує можливість використовувати низькорівневі функції, взаємодіяти зі сторонніми бібліотеками та виконувати інші операції, які можуть бути складніші або недоступні в мові

Java або Kotlin. Встановлення шляху до Android NDK у середовищі Unreal Engine дозволяє розробникам використовувати мову C++ для створення оптимізованого коду та забезпечити високу продуктивність ігрових застосунків.

3. JDK (Java Development Kit) [17-19]: є набором інструментів для розробки програмного забезпечення на мові Java. Він містить Java Runtime Environment (JRE), компілятор Java, бібліотеки та інші утиліти, необхідні для розробки і виконання програм на мові Java. JDK використовується в розробці під ОС Android для написання коду на мові Java або Kotlin, розробки користувацького інтерфейсу та взаємодії з різними функціями платформи Android. Встановлення шляху до JDK у середовищі Unreal Engine дозволяє розробникам використовувати мову Java або Kotlin для розробки функціональності ігрових застосунків та взаємодії з Android API.

Налаштування шляхів до SDK, NDK та JDK у середовищі Unreal Engine дозволяє забезпечити правильну інтеграцію з платформою Android та використовувати всі її функціональні можливості під час розробки ігор. Розробники отримують доступ до різноманітних API, можливість писати оптимізований код на мові C++ та використовувати різноманітні функції Java або Kotlin для реалізації потужного та вражаючого досвіду гри на мобільних пристроях під управлінням ОС Android.

Розглянемо основні кроки, які необхідно виконати для налаштування середовища Unreal Engine для розробки під ОС Android:

1. Встановлення Android Studio:

– завантажте Android Studio з офіційного веб-сайту розробників Android;

– запустіть інсталяційний файл та дотримуйтесь інструкцій для встановлення;

– під час процесу встановлення переконайтеся, що встановлюєте Android Studio за шляхом за замовчуванням (рисунок 3.4). Це дозволить Unreal Engine автоматично знайти розташування необхідних файлів;

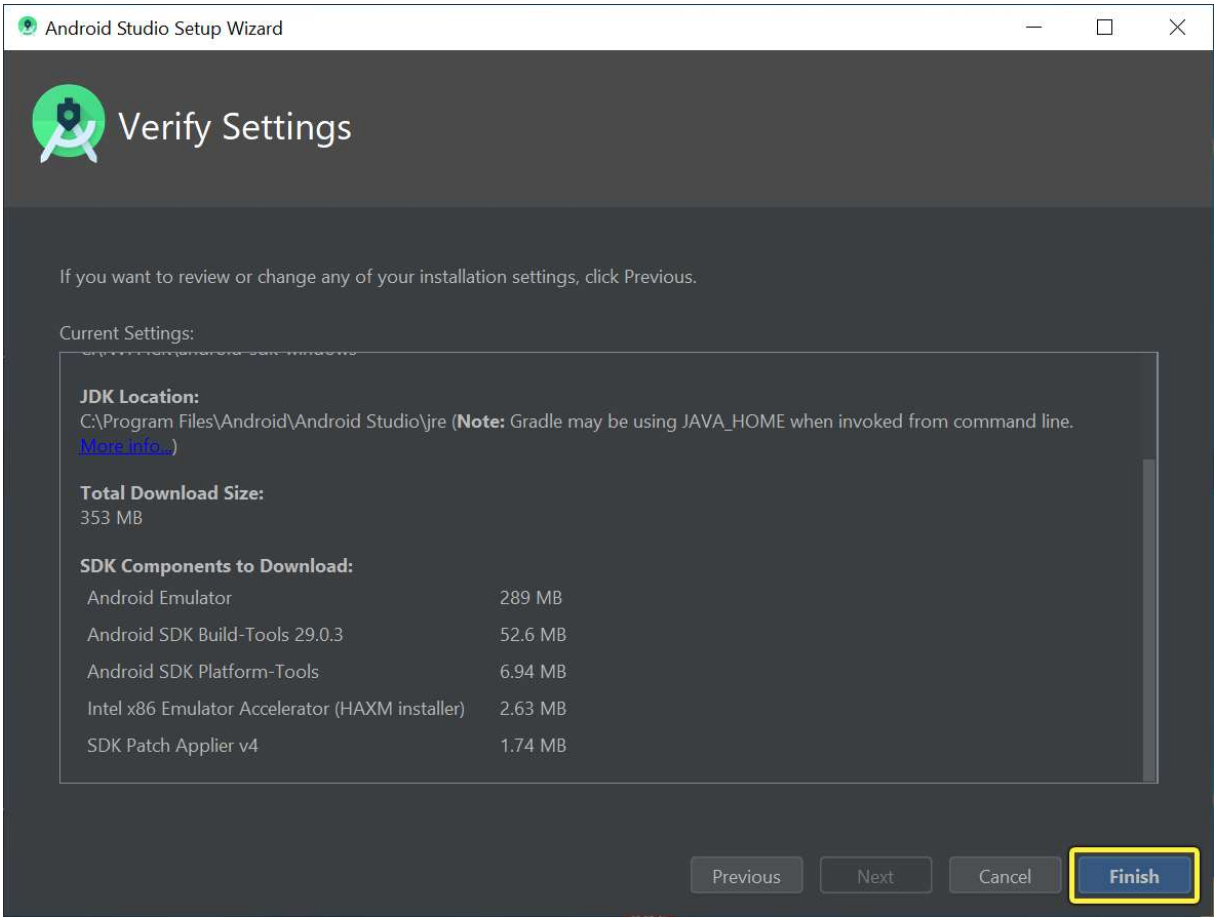

Рисунок 3.4 – Вікно завершення інсталяції, де необхідно перевірити що шляхи обрані за замовчуванням

Встановлення Android SDK Command-line Tools:

– після успішної установки Android Studio необхідно запустити його та перейти до меню налаштувань (Settings);

– у меню налаштувань знайдіть пункт SDK Manager;

– у вікні SDK Manager перейдіть на вкладку «SDK Tools»;

– переконайтеся, що «Android SDK Command-line Tools» вибрано для установки (рисунок 3.5);

– натисніть «Apply» або «OK», щоб почати процес завантаження та встановлення Command-line Tools.

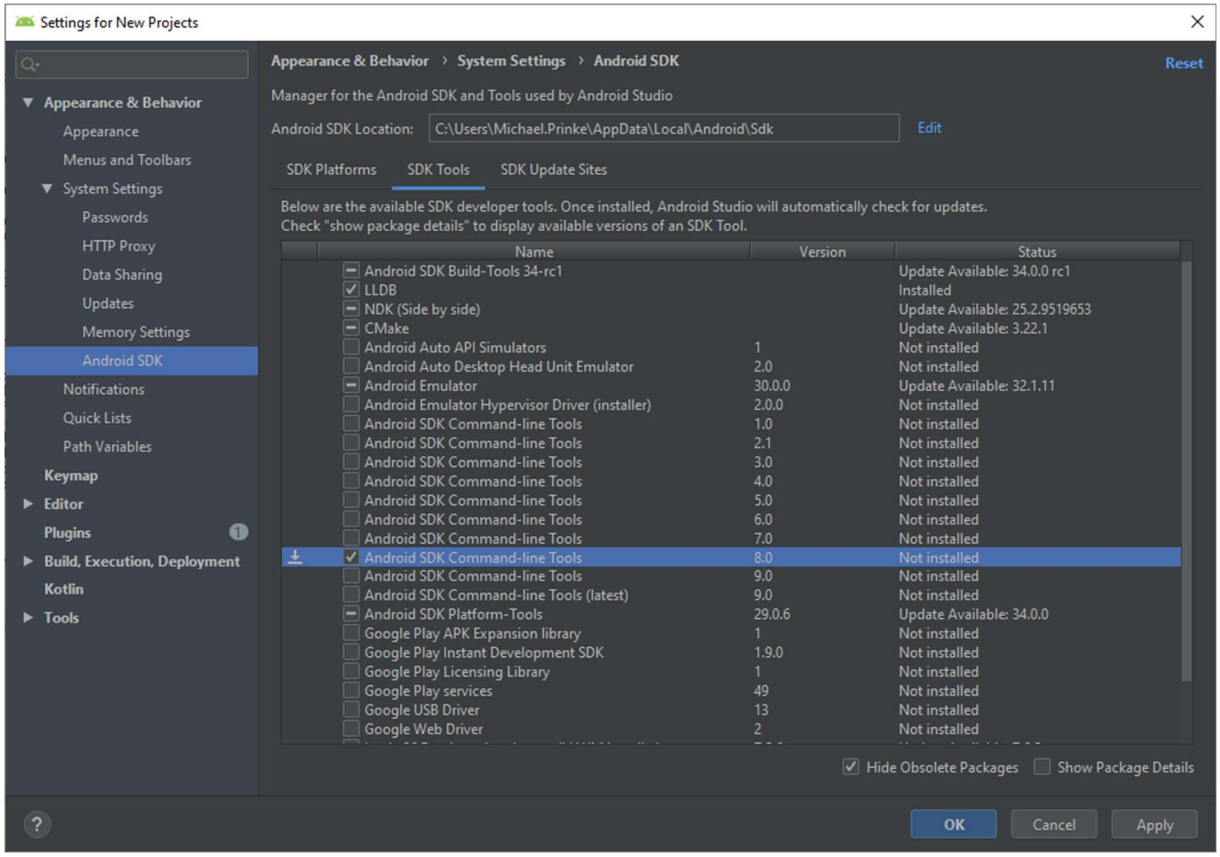

Рисунок 3.5 – Приклад правильно обраних інструментів SDK

Налаштування Unreal Engine:

– зайдіть до папки інсталяції Unreal Engine на вашому комп'ютері;

– відкрийте папку «Engine/Extras/Android»;

– залежно від операційної системи, виконайте файл «SetupAndroid.bat» (для ОС Windows), «SetupAndroid.command» (для ОС macOS) або «SetupAndroid.sh» (для ОС Linux);

– цей скрипт запустить процес налаштування Android середовища для Unreal Engine;

2. Налаштування шляхів до SDK, NDK та JDK в Unreal Engine:

– запустіть Unreal Engine на вашому комп'ютері;

– у верхньому меню виберіть «Edit» (Редагувати) та перейдіть до «Project Settings» (Налаштування проекту);

– у вікні «Project Settings» виберіть «Platforms» (Платформи), а потім «Android SDK»;

– встановіть шлях до встановленого Android SDK, NDK та JDK. Android SDK вказує на папку, де встановлено Android SDK (наприклад, «C:/Android/sdk»). NDK та JDK вказують на папки, де встановлено Android NDK та Java Development Kit відповідно;

– збережіть зміни.

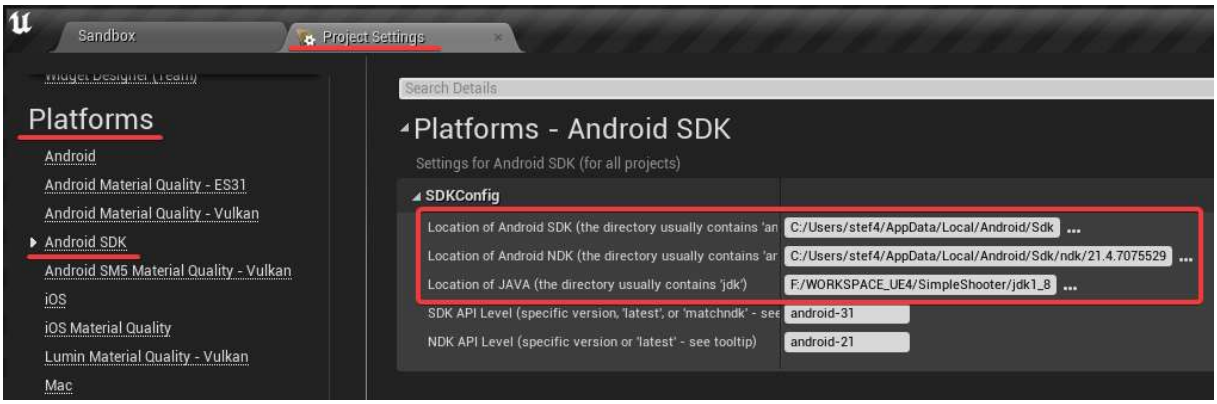

Рисунок 3.6 – Приклад правильно заповнених налаштувань SDK, NDK, JDK рушія

Налаштування середовища Unreal Engine для розробки під ОС Android включає в себе встановлення Android Studio, завантаження Android SDK Command-line Tools та налаштування шляхів до SDK, NDK та JDK. Це дозволяє Unreal Engine коректно взаємодіяти з Android-платформою та забезпечує можливість розробки ігрових застосунків для мобільних пристроїв.

Зазначимо, що точні кроки налаштування можуть певним чином відрізнятися в залежності від версії Android Studio, Unreal Engine та операційної системи.

Рекомендується використовувати офіційну документацію та посібники, надані розробниками Android та Unreal Engine, для отримання докладних інструкцій.

3.3 Запуск ігрового застосунку та аналіз наявних проблем

Перший запуск ігрового застосунку вдався не одразу, і після усіх налаштувань середовища та рушія, при спробі інсталяції було отримано помилку, зображену на рисунку 3.7. Це пов’язано з недавніми оновленнями ОС

```
F:\WORKSPACE_UE4\SimpleShooter\Builds\Android_ETC2>C:/Users/stef4/AppData/Local/Android/Sdk\platfo
rm-tools\adb.exe  install SimpleShooter-arm64.apk
Performing Streamed Install
adb: failed to install SimpleShooter-arm64.apk: Failure [INSTALL_PARSE_FAILED_MANIFEST_MALFORMED:
Failed parse during installPackageLI: /data/app/vmdl233278478.tmp/base.apk (at Binary XML file lin
e #52): com.epicgames.ue4.SplashActivity: Targeting S+ (version 31 and above) requires that an exp
licit value for android:exported be defined when intent filters are present]
```

Рисунок 3.7 – Помилка інсталяції застосунку одразу після завершення налаштувань

Android, оскільки починаючи з 11 версії Android в Manifest File для кожного тегу <Activity> необхідно прописати атрибут android:exported=«true» (рисунок 3.8).

```
ShooterCharacter.cpp | ActorComponent.h | AndroidToolChain.cs | UEDeployAndroid.cs | Obj.cpp | ShooterPlayerController.h | ShooterPlayerController.cpp | ShooterAIControll
        if (!bIsForDistribution && SDKLevelInt >= 29 && !bRequestedLegacyExternalStorage)
        {
            // work around scoped storage for non-distribution for SDK 29; add to ExtraApplicationNodeTags if you need it for distribution
            Text.AppendLine("\t            android:requestLegacyExternalStorage=\"true\"");
        }
        Text.AppendLine("\t            android:hasCode=\"true\">");
        if (bShowLaunchImage)
        {
            // normal application settings
            Text.AppendLine("\t\t<activity android:name=\"com.epicgames.ue4.SplashActivity\"");
            Text.AppendLine("\t\t          android:exported=\"true\"");
            Text.AppendLine("\t\t          android:label=\"@string/app_name\"");
            Text.AppendLine("\t\t          android:theme=\"@style/UE4SplashTheme\"");
            Text.AppendLine("\t\t          android:launchMode=\"singleTask\"");
            Text.AppendLine(string.Format("\t\t          android:screenOrientation=\"{0}\"", Orientation));
            Text.AppendLine(string.Format("\t\t          android:debuggable=\"{0}\">", bIsForDistribution ? "false" : "true"));
            Text.AppendLine("\t\t\t<intent-filter>");
            Text.AppendLine("\t\t\t\t<action android:name=\"android.intent.action.MAIN\" />");
            Text.AppendLine(string.Format("\t\t\t\t<category android:name=\"android.intent.category.LAUNCHER\" />"));
            Text.AppendLine("\t\t\t</intent-filter>");
            Text.AppendLine("\t\t</activity>");
            Text.AppendLine("\t\t<activity android:name=\"com.epicgames.ue4.GameActivity\"");
            Text.AppendLine("\t\t          android:exported=\"true\"");
            Text.AppendLine("\t\t          android:label=\"@string/app_name\"");
            Text.AppendLine("\t\t          android:theme=\"@style/UE4SplashTheme\"");
            Text.AppendLine(bAddDensity ? "\t\t          android:configChanges=\"mcc|mnc|uiMode|density|screenSize|smallestScreenSize|scree
                                        : "\t\t          android:configChanges=\"mcc|mnc|uiMode|screenSize|smallestScreenSize|screenLayout|
        }
        else
        {
            Text.AppendLine("\t\t<activity android:name=\"com.epicgames.ue4.GameActivity\"");
            Text.AppendLine("\t\t          android:exported=\"true\"");
            Text.AppendLine("\t\t          android:label=\"@string/app_name\"");
            Text.AppendLine("\t\t          android:theme=\"@android:style/Theme.Black.NoTitleBar.Fullscreen\"");
            Text.AppendLine(bAddDensity ? "\t\t          android:configChanges=\"mcc|mnc|uiMode|density|screenSize|smallestScreenSize|scree
```

Рисунок 3.8 – Зміни у вихідному коді рушія для можливості інсталяції додатку

На таку помилку не розраховувала компанія Epic Games, тому ніяких програмних рішень для вирішення цієї проблеми для користувача не передбачалось, відтак необхідно було відредагувати вихідний код рушія. А саме в файлі UEDeployAndroid.cs, в методі GenerateManifest. Після цього ігровий застосунок встановився та почав завантажуватись.

Під час першого запуску ігрового застосунку в середовищі Unreal Engine було помічено дуже низьку частоту кадрів, а саме близько 15 FPS (англ. кадр за секунду), що може негативно вплинути на плавність та відчуття гри (рисунок 3.10). Для отримання точних вимірювань частоти кадрів була використана консольна команда «stat FPS» (рисунок 3.9). Ця команда дозволяє відображати інформацію про частоту кадрів (Frames Per Second) прямо на екрані гри. Щоб активувати цю команду, необхідно клікнути по екрану одразу чотирма пальцями.

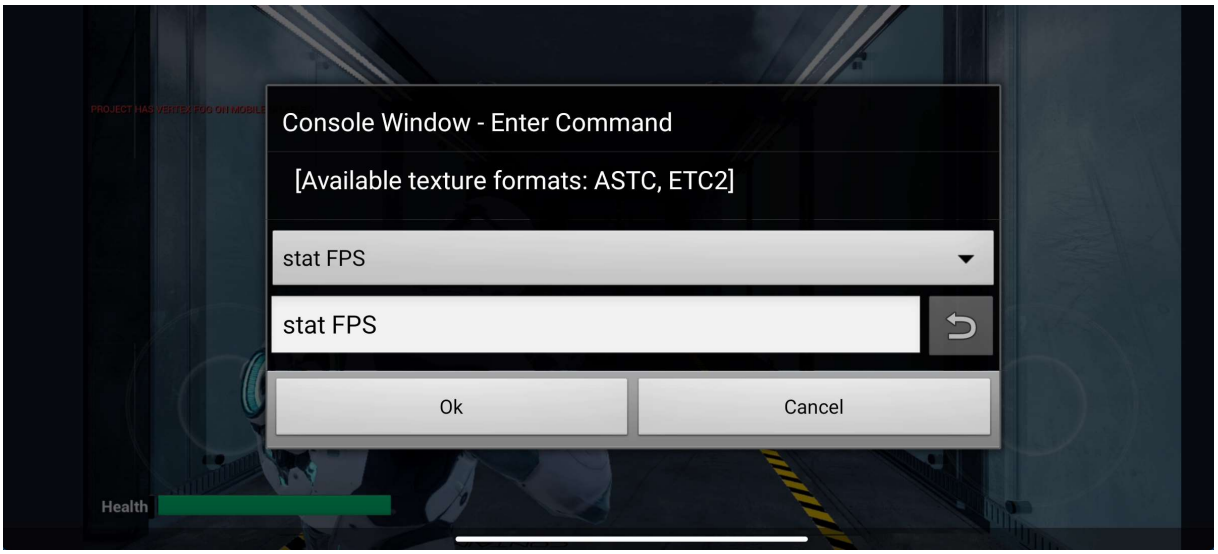

Рисунок 3.9 – Демонстрація відображення вікна введення консольних команд

Під час першої спроби запуску ігрового застосунку було помічено, що головний персонаж не може стріляти (рисунок 3.10). Ця проблема виникла через те, що інтерфейс за замовчуванням не передбачав можливості стріляти, а лише надавав джойстики для керування камерою та переміщенням персонажу.

Для вирішення цієї проблеми необхідно внести зміни в інтерфейс гри та додати елемент керування стрільбою. Наприклад, можна розмістити кнопку «Стріляти» на екрані ігрового застосунку, яка буде активувати відповідну функцію у головного персонажа.

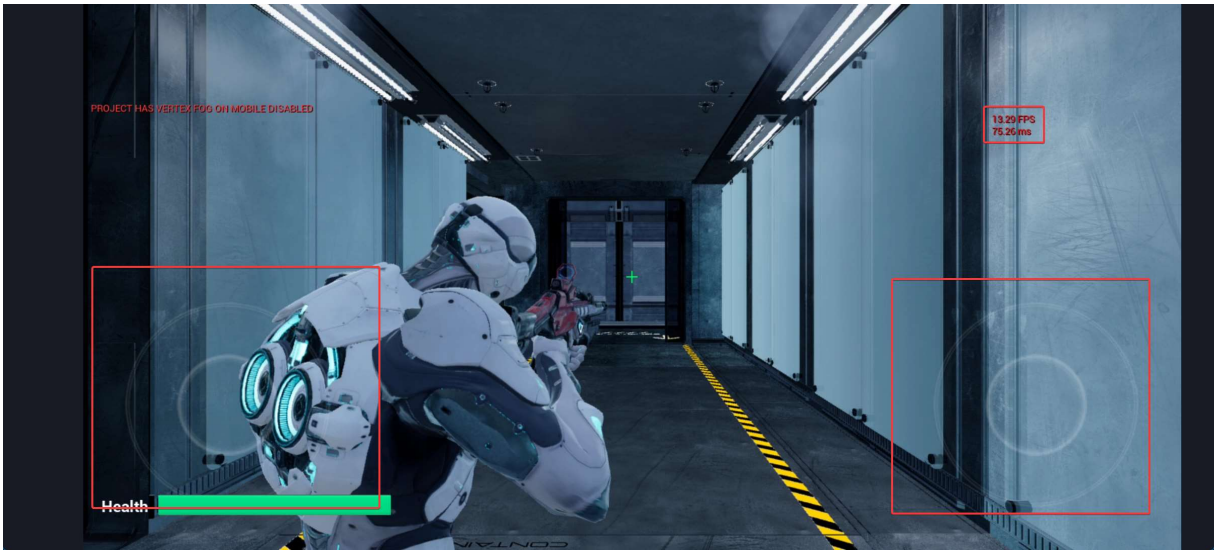

Рисунок 3.10 – Демонстрація низької частоти кадрів, та відсутності кнопки «стріляти» чи «стрибати»

Аналізуючи отримані проблеми під час першої спроби запуску ігрового застосунку, важливо звернути увагу на оптимізацію частоти кадрів для забезпечення плавності гри та вчасного виявлення та вирішення проблем з інтерфейсом, що можуть обмежувати функціональність головного персонажа. Ці кроки допоможуть покращити враження користувачів від гри та забезпечити її більш гармонійну та зручну геймплейну динаміку.

РОЗДІЛ 4. ОПТИМІЗАЦІЯ ІГРОВОГО ЗАСТОСУНКУ ПІД МОБІЛЬНІ ПЛАТФОРМИ

4.1 Профайлінг гри та визначення ботлнеків через використання профайлера

У гейм-розробці та оптимізації ігор термін «ботлнек» (англ. Bottleneck) використовується для позначення обмеження або слабкого місця, яке гальмує швидкодію системи. Виявлення та вирішення ботлнеків є важливою задачею при оптимізації ігрового застосунку для досягнення кращої продуктивності та плавності геймплею.

Профайлінг є процесом аналізу ігрового застосунку з метою виявлення проблемних ділянок коду або ресурсів, які спричиняють незадовільну продуктивність [20, 21]. Для здійснення профайлінгу використовуються спеціальні інструменти, такі як профайлери, які дозволяють вимірювати та аналізувати різні аспекти гри, такі як час виконання функцій, використання пам'яті, частота кадрів тощо.

У середовищі Unreal Engine доступні різні інструменти профайлінгу, такі як «Unreal Insights» та «Stat System», які дозволяють відстежувати та аналізувати різні параметри гри для виявлення ботлнеків. Крім того, у Android Studio також є інструменти профайлінгу, такі як «Android Profiler», які дозволяють відстежувати роботу додатка на мобільному пристрої.

Для збору статистики в грі Unreal Engine можна використовувати команду «stat StartFile», яка записує статистику у файл. Ця статистика зберігається у форматі «.ue4stats» та містить інформацію про час виконання різних функцій, використання пам'яті та інші показники продуктивності.

Для аналізу цих даних можна використовувати інструмент «Session Frontend», який надає графічний інтерфейс для розбору та аналізу статистики зі збережених файлів. В Session Frontend доступні різні вікна, такі як «Timing

Graph», «Memory Graph», «CPU Graph» та інші, які дозволяють детально вивчити показники продуктивності гри та знайти можливі ботлнеки.

Однак кращою альтернативою є Unreal Insights, це додаток, що надається рушієм, однак запускається окремо від нього. На відміну від Session Frontend, даний інструмент збирає статистику в режимі реального часу, а не зберігає її в файл на пристрої, а також надає більш розгорнуту інформацію про стан ігрового застосунку. В подальшому буде використовуватись лише цей інструмент.

Для його запуску необхідно зайти в папку (шлях до рушія на ПК) Engine\Binaries\Win64, та натиснути на файл UnrealInsights.exe. Далі, використовуючи інструмент ADB необхідно ввести три команди, як показано на рис. 4.1, для налаштування пристрою та запуску ігрового застосунку в режимі збору аналітичних даних.

```
C:\Users\stef4>adb.exe reverse tcp:1980 tcp:1980

C:\Users\stef4>adb shell setprop debug.ue4.commandline -tracehost=127.0.01

C:\Users\stef4>adb shell am start -n com.stef.SimpleShooter/com.epicgames.ue4.GameActivity
Starting: Intent { cmp=com.stef.SimpleShooter/com.epicgames.ue4.GameActivity }
```

Рисунок 4.1 – Команди, необхідні для профайлінгу з допомогою Unreal Insights

Після цього на ПК, в самому профайлері необхідно обрати теперішню сесію, яка в колонці «Status» матиме значення «LIVE», що означає що це активна сесія, а запис даних з попередніх запусків.

У даному ігровому застосунку було помічено, що процесор постійно очікує на графічний чіп. Як можемо бачити з рис. 4.2, на Render потоці більшу частину часу процесор виконує завдання «WaitUntilTasksComplete», що власне і позначає очікування на GPU. Це може свідчити про те, що графічна частина є основною проблемою, яка обмежує швидкодію гри. Аналізуючи статистику та використовуючи інструменти профайлінгу, розробники можуть зосередитись на оптимізації графічних ресурсів, алгоритмів рендерингу та інших аспектів, щоб покращити продуктивність гри та забезпечити плавний геймплей для користувачів.

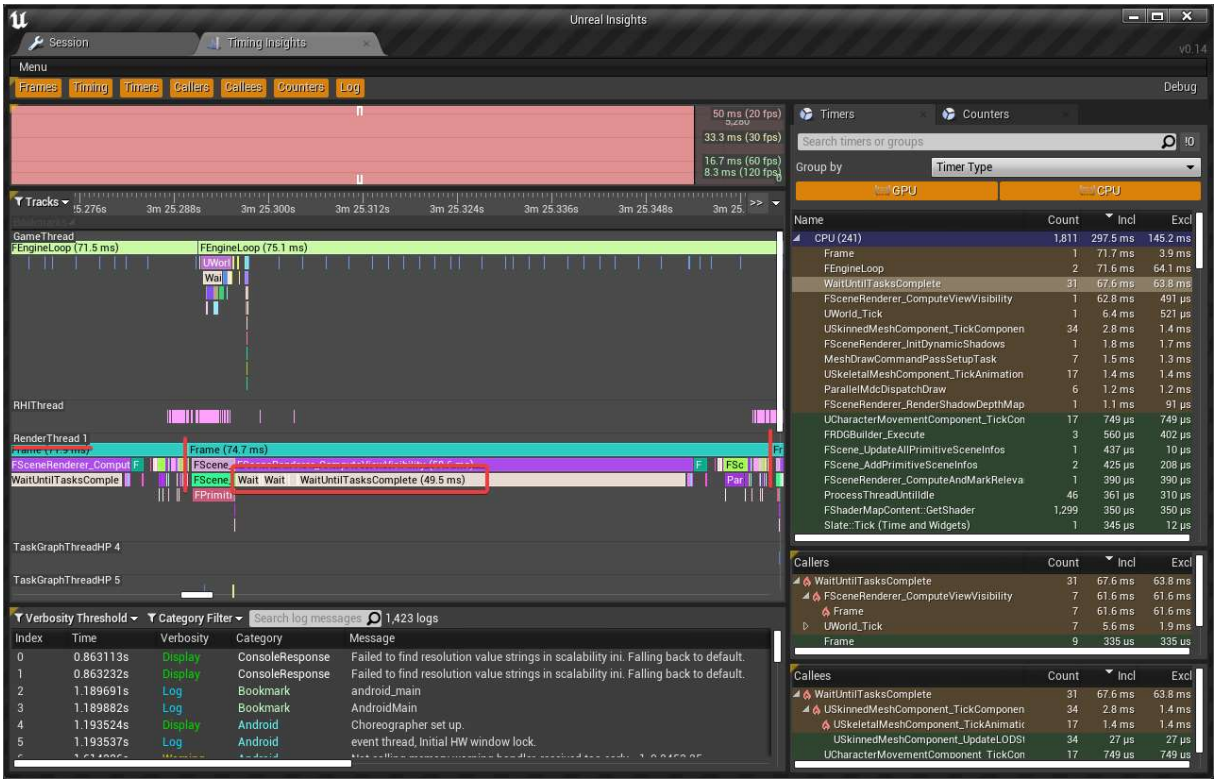

Рисунок 4.2 – Демонстрація проблеми з очікуванням CPU на закінчення дій GPU

4.2 Оптимізація графіки та програмного коду

У процесі оптимізації графіки та програмного коду, для поліпшення продуктивності гри були вжиті наступні кроки:

1. Зміна цільової платформи з ПК на мобільну:

У меню редагування проекту Unreal Engine, необхідно змінити налаштування «Target Hardware» на мобільну платформу. Це повідомляє рушій, що основною платформою запуску ігрового застосунку буде мобільний пристрій. На основі цього в Unreal Engine присутній ряд оптимізації шейдерів, освітлення та інших ресурсоємних елементів ігрового застосунку. Також зміна платформи передбачає вбудовану перевірку на сумісність коду та графічних об’єктів з мобільним пристроєм.

2. Device Profile Configuration:

Device Profile (конфігурація пристрою) є файлом конфігурації, який містить параметри, які набирають чинності в залежності від пристрою, на якому

запущена гра. Цей файл розташовується за шляхом «шлях до рушія/Config/DefaultDeviceProfiles.ini». Він дозволяє налаштувати параметри, такі як роздільна здатність екрану, рівень деталей та інші, для різних пристроїв на основі їх продуктивності.

При запуску гри рушій визначає деякі важливі дані про пристрій, такі як: виробника та модель, модель GPU та CPU, об’єм оперативної пам’яті та ін., як можна побачити на рис. 4.3.

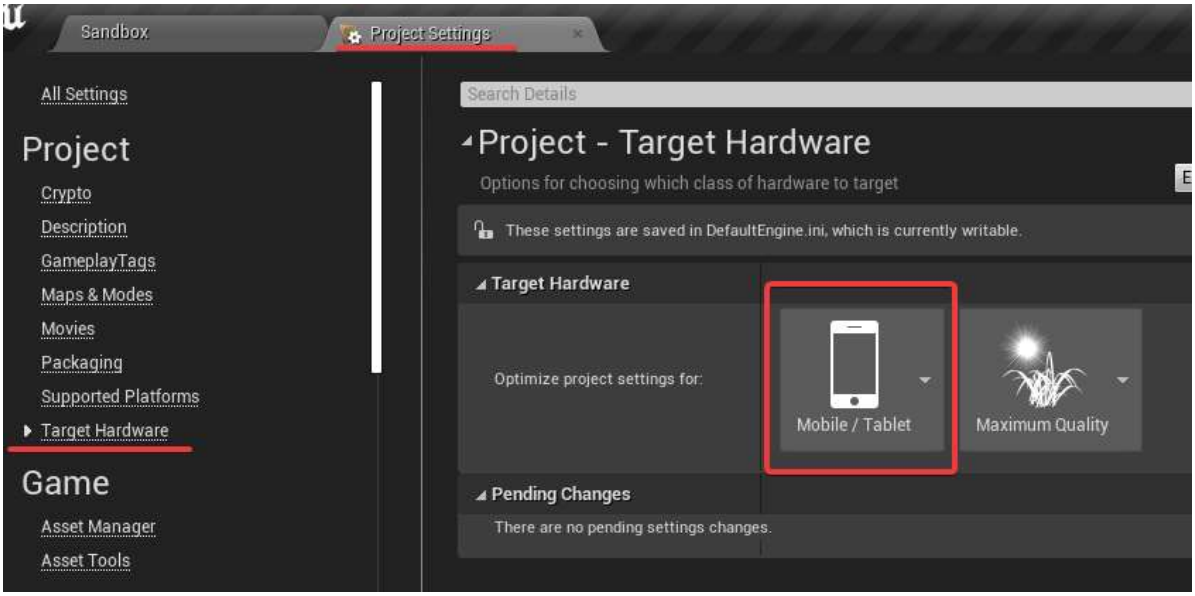

Рисунок 4.3 – Результат зміни Target Hardware на мобільну платформу

На основі цих даних розробник має можливість обирати конкретні налаштування швидкодії гри в залежності від певної групи пристроїв, найбільша увага звертається саме на графічний процесор. В рушії з самого початку наявні певні конфігурації за замовчуванням, які показують чудовий результат, однак часом з’являється потреба змінити значення певних змінних, або ж розробник має можливість впровадити свої параметри та може обирати їх значення в цьому ж файлі.

У межах даної роботи, для досягнення оптимальної продуктивності на мобільних пристроях, було змінено максимальну роздільну здатність гри шляхом модифікації змінної r.MobileContentScaleFactor для глобальних

категорій пристроїв з назвами: «Android_High», «Android_Mid», «Android_Low» як це показано на рис. 4.4.

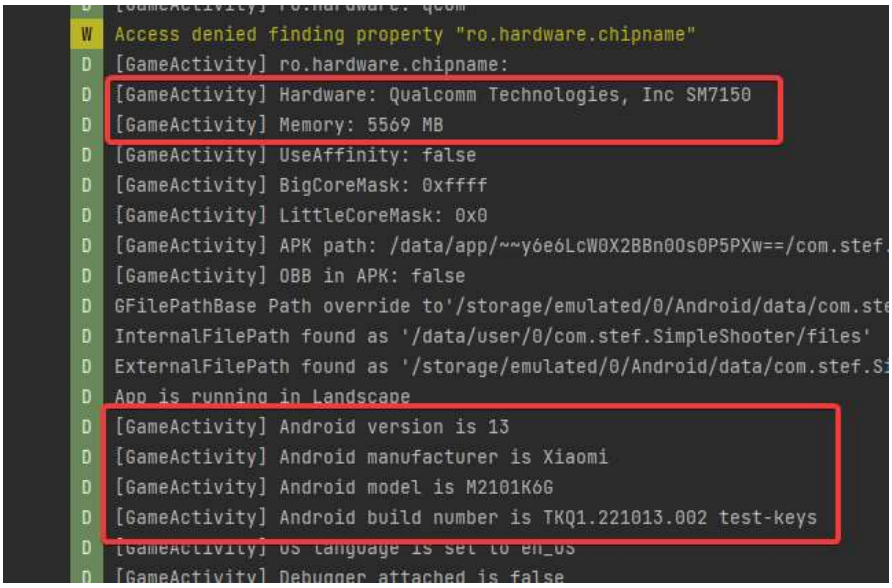

Рисунок 4.4 – Логи з даними про пристрій на якому запущений ігровий додаток

Ці групи пристроїв наявні за замовчуванням і вони символізують загальну швидкодію пристрою, усі подальші категорії на основі графічного ядра і тд. часто посилаються на них, запозичуючи налаштування за принципом наслідування, що означає, що зміна параметру максимальної роздільної здатності в цих групах, вплине на усі наявні підгрупи пристроїв.

Зниження максимальної роздільної здатності дозволяє прискорити швидкодію ігрового застосунку через те, що графічному ядру необхідно обробляти меншу кількість пікселів, що значно зменшує загальний час «Рендерингу» кадра.

3. Зменшення кількості полігонів з використанням LODs:

Полігон – це геометричний елемент, який визначає форму 3D-моделі. Кількість полігонів впливає на продуктивність гри, оскільки більша кількість полігонів вимагає більше ресурсів для обробки. Якщо детально оглянути ігрову сцену та подивитись показники графічних об’єктів, то можна помітити, що в середньому на один об’єкт припадає близько кількох тисяч полігонів, що є надто великою кількістю для мобільних пристроїв [21-23].

Для хорошої швидкодії рекомендується використовувати, так звані «Low-Poly» моделі. Low-Poly (англ. низько полігональні) об’єкти – це об’єкти які з самого початку створені з низькою кількістю полігонів, з метою заощадження ресурсів графічного та центрального обчислювального ядра що в свою чергу покращує продуктивність ігрового застосунку [23].

```
[Android_Low DeviceProfile]
DeviceType=Android
BaseProfileName=Android
+CVars=r.MobileContentScaleFactor=0.4
; Scalability groups, see AndroidScalability.ini
+CVars=sg.ViewDistanceQuality=0
+CVars=sg.AntiAliasingQuality=0
+CVars=sg.ShadowQuality=0
+CVars=sg.PostProcessQuality=0
+CVars=sg.TextureQuality=0
+CVars=sg.EffectsQuality=0
+CVars=sg.FoliageQuality=0

[Android_Mid DeviceProfile]
DeviceType=Android
BaseProfileName=Android
+CVars=r.MobileContentScaleFactor=0.6
+CVars=sg.ViewDistanceQuality=1
+CVars=sg.AntiAliasingQuality=1
+CVars=sg.ShadowQuality=1
+CVars=sg.PostProcessQuality=1
+CVars=sg.TextureQuality=1
+CVars=sg.EffectsQuality=1
+CVars=sg.FoliageQuality=1

[Android_High DeviceProfile]
DeviceType=Android
BaseProfileName=Android
+CVars=sg.ViewDistanceQuality=2
+CVars=sg.AntiAliasingQuality=2
+CVars=sg.ShadowQuality=2
+CVars=sg.PostProcessQuality=2
+CVars=sg.TextureQuality=2
+CVars=sg.EffectsQuality=2
+CVars=sg.FoliageQuality=2
+CVars=r.MobileContentScaleFactor=0.75
```

Рисунок 4.5 – Зміни в файл конфігурації рушія для зменшення роздільної здатності

Для того, щоб надати об’єкту певних деталей, зберігаючи низьку кількість полігонів, використовуються специфічні види текстур, що імітують наявність цих деталей, не на рівні геометрії об’єкта а на рівні його відображення на екран, що є певним компромісом між якістю зображення та швидкістю його обробки. Оскільки в нашому додатку усі моделі є високополігональними, то застосовується ще одна методика оптимізації LOD.

LODs (рівні деталізації) є технікою, яка дозволяє автоматично відображати менш деталізовані версії моделей для віддалених об'єктів. Тобто

моделі, які знаходяться далеко від гравця і графічні особливості яких стає складно розрізнити, автоматично заміняються попередньо заготовленими менш деталізованими моделями з нижчою кількістю полігонів, що дозволяє зменшити завантаженість GPU, практично не шкодячи візуальній якості ігрового застосунку [18]. В Unreal Engine це можна налаштувати через меню «Static Mesh Settings» моделі.

У даному проекті було використано автоматичну генерацію LODs, щоб створити менш деталізовані версії моделей без участі 3D-Художника, а також зробити це в короткий термін. Рушій самостійно за допомогою вбудованих алгоритмів оптимізує «Mesh» моделей, намагаючись максимально зберігати оригінальну форму об’єкта, однак прибираючи мілкі деталі, яких не видно на відстані.

Однак це вирішує лише половину нашої проблеми, віддалені моделі тепер відображаються з меншою кількістю полігонів, однак ближні об’єкти досі залишаються складними для обробки. Для вирішенні цього була використана можливість визначення змінної MinLOD для кожної з платформ. Це дозволяє встановлювати мінімальний рівень деталізації для моделей в залежності від платформи. Оскільки створити 3D-модель для кожного значення відстані від гравця до об’єкта – неможливо, зазвичай створюється лише невелика їх кількість, близько 4, де 1-й рівень деталізації – це найбільш деталізована модель, а 4-й – найменш деталізована. При віддалені від цієї моделі на певний поріг відстані, рівень деталізації автоматично збільшується.

Налаштування MinLOD дозволяє визначити мінімальний рівень деталізації і цим обмежити використання моделей з високою кількістю полігонів та відображати лише легкі для обробки графічним ядром моделі.

У межах даного завдання мінімальним був обраний 3-тій рівень, як показано на рис. 4.6. Це зберігає баланс між втратою якості моделі та швидкістю його виводу на екран.

Також варто зауважити, що налаштування MinLOD можна встановлювати для кожної платформи окремо, що дозволяє підтримувати один проект для різних пристроїв і підкреслює кросплатформні можливості рушія Unreal Engine.

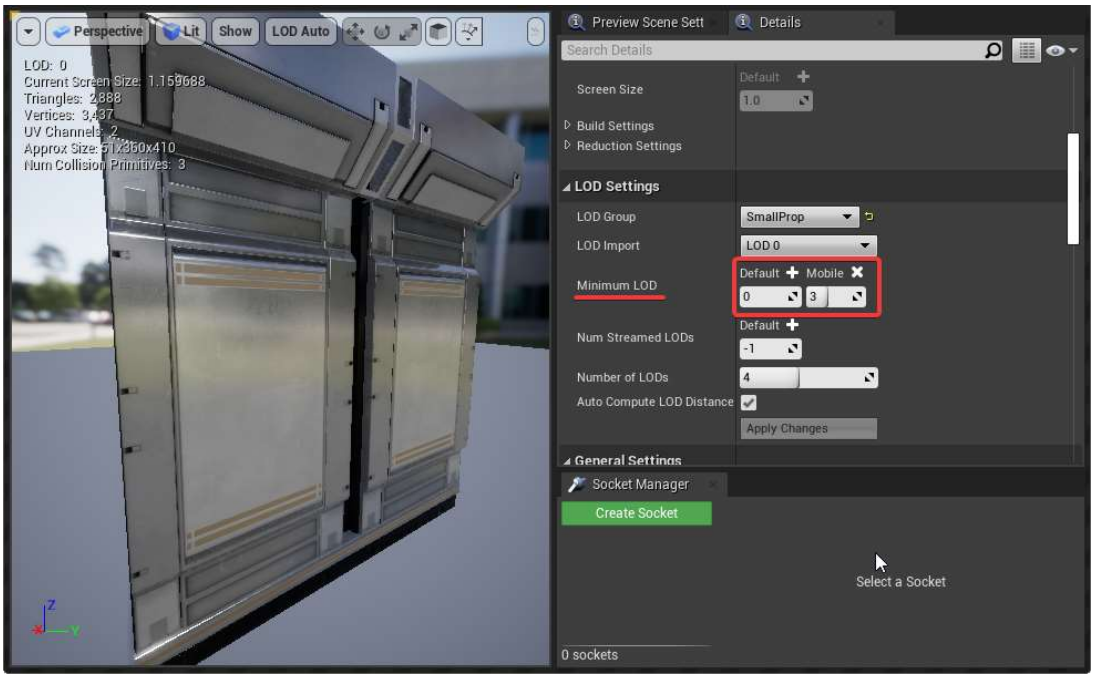

Рисунок 4.6 – Демонстрація зміни налаштування MinLOD

Важливо відзначити, що після впровадження змін в LOD, були помічені деякі графічні артефакти, які виникають як наслідок оптимізації. Один з таких артефактів можна побачити на рис. 4.7, де видно що сусідні об’єкти підлоги не на одному рівні.

Рисунок 4.7 – Приклад графічного артефакту

Щоб виправити ці артефакти, необхідно співпрацювати з командою 3D-художників, оскільки ці проблеми не можуть бути вирішені на рівні розробника гри і не входять в тему даної роботи.

4. Оптимізація завантаження CPU:

Хоча в даному проекті проблем із завантаженням CPU не виникло, все ж можна надати деякі рекомендації щодо оптимізації цього аспекту.

Основними джерелами навантаження на центральний процесор є наступні: фізична симуляція, програвання звуків та виконання ігрового коду.

1. Для оптимізації роботи фізичних симуляції, рекомендується використовувати простіші моделі колізій та обмежувати обчислення фізичних ефектів тільки на об'єктах, які їх потребують.

2. Для оптимізації роботи звуку, можна використовувати асинхронне завантаження звукових ресурсів та попереднє кешування, щоб зменшити затрати CPU на завантаження та відтворення звуків.

3. Для прискорення коду в ігровій індустрій існує ряд компромісів між точністю та швидкістю обчислень. Для прикладу, хотілося б продемонструвати рішення розробників з компанії ID Software, які створили всесвітньовідому багатокористувацьку гру в жанрі «Shooter» Quake III Arena. Даний ігровий застосунок був революцією на ринку, оскільки запропонував революційну якість 3D графіки, однак для цього необхідно було оптимізувати велику кількість обчислень які застосовувались для необхідних симуляцій та відображення картинки. Розробники гри змогли винайти більш оптимальний метод обрахування оберненого квадратного кореня а саме $\frac{1}{\sqrt{x}}$. Саме це значення використовувалось для нормалізації вектора, що застосовується в розрахунку освітлення та його відбиття від поверхонь.

Тогочасні комп’ютери мале певні труднощі з обчисленням квадратного кореня, оскільки це вимагало великої кількості операцій ділення, тож вони використали наступний алгоритм:

– інтерпретували число з рухомою комою у вигляді цілого числа, що дає грубе наближення квадратного логарифму;

– використали тотожність $\log_2\left(\frac{1}{\sqrt{x}}\right) = -\frac{1}{2}\log_2 x$

– виконали один крок методу Ньютона для кращої апроксимації цієї функції.

Дани метод дає похибку близьку 0,175 %, що практично ніяк не впливає на якість вихідного зображення, однак значно зменшує кількість обчислень, сильно заощаджуючи ресурс CPU. На сьогодні дана оптимізація вже не релевантна, оскільки сучасні процеси підтримують особливу інструкцію «rsqrtss», яка виконується на апаратному рівні, має менше похибку та витрачає менше часу.

Також існують більш очевидні оптимізації. Для прикладу можна використовувати квадрат відстані, замість звичайної відстані у випадках, коли потрібно порівняти їх між собою. Це дозволяє уникнути виконання складних операцій квадратного кореня та прискорює порівняння відстаней.

Ц цілому, оптимізація графіки та програмного коду є важливим кроком для покращення продуктивності гри. Правильне налаштування параметрів, використання LODs, оптимізація роботи CPU та компроміси між точністю та швидкістю обчислень допомагають забезпечити кращу продуктивність та покращити якість ігрового досвіду.

4.3 Проблеми користувацького інтерфейсу та методи їх вирішення

При портуванні гри з ПК на мобільні пристрої виникає необхідність змінювати користувацький інтерфейс (UI), оскільки користувачі не мають клавіатури або мишки, а вся взаємодія з ігровим застосунком відбувається лише за допомогою сенсорного дисплею.

У межах даного проекту користувач міг виконати стрибок за допомогою клавіші «пробіл», а також стріляти натисканням на ліву кнопку миші, однак зараз такої можливості немає як можна побачити на рис. 4.8.

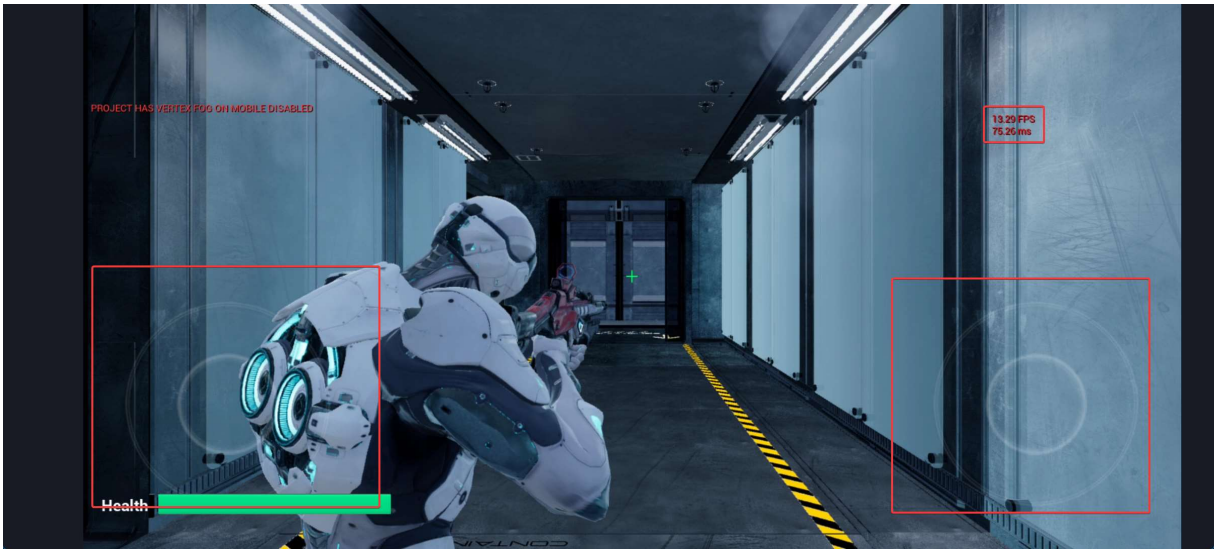

Рисунок 4.8 – Демонстрація нефункіонального користувацького інтерфейсу

Ці дії є основними для будь якої гри жанру Shooter, без них стає неможливим виконати ціль гри, а в контексті цієї роботи – це нейтралізувати усіх загарбників космічної бази.

Для виправлення цієї проблеми необхідно відредагувати або створити новий, або ж відредагувати наявний Widget Blueprint, який відповідає за відображення HUD. В UnrealEngine існує такий інструмент як Widget який використовується для позначення візуальних елементів користувацького інтерфейсу (UI), які використовуються для взаємодії з грою. Віджети є частиною системи UMG (Unreal Motion Graphics), яка надає можливості створення і керування UI у Unreal Engine [1, 9, 12].

Widget є наборами візуальних елементів, таких як кнопки, текстові поля, зображення, прогрес-бари тощо, які можна розміщувати на екрані гри. Вони дозволяють створювати інтерактивні елементи, з якими користувач може взаємодіяти, наприклад, натискати кнопки, вводити текст, переміщати повзунки тощо [13].

У свою чергу Blueprint – це система візуального програмування, що дозволяє розробникам створювати функціональність гри шляхом з'єднання вузлів ігрової логіки у візуальному середовищі. Blueprint є графічними

скриптами, які дозволяють створювати і змінювати функціональність гри без написання коду.

Blueprint використовують вузли і з'єднання між ними для визначення поведінки об'єктів, персонажів, AI, ефектів, фізики, інтерфейсів користувача та іншої логіки гри. Кожен вузол виконує певну функцію, таку як обчислення, взаємодія з іншими об'єктами, зміна значень змінних та інше. З'єднання між вузлами визначає потік і порядок виконання логіки.

Таким чином нам необхідно додати на екран необхідні кнопки для стрибку «Jump» та «Fire». Після чого необхідно перейти в меню Events (англ. Події) та натиснути на плюсик біля події «On Clicked», як продемонстровано на рис. 4.9.

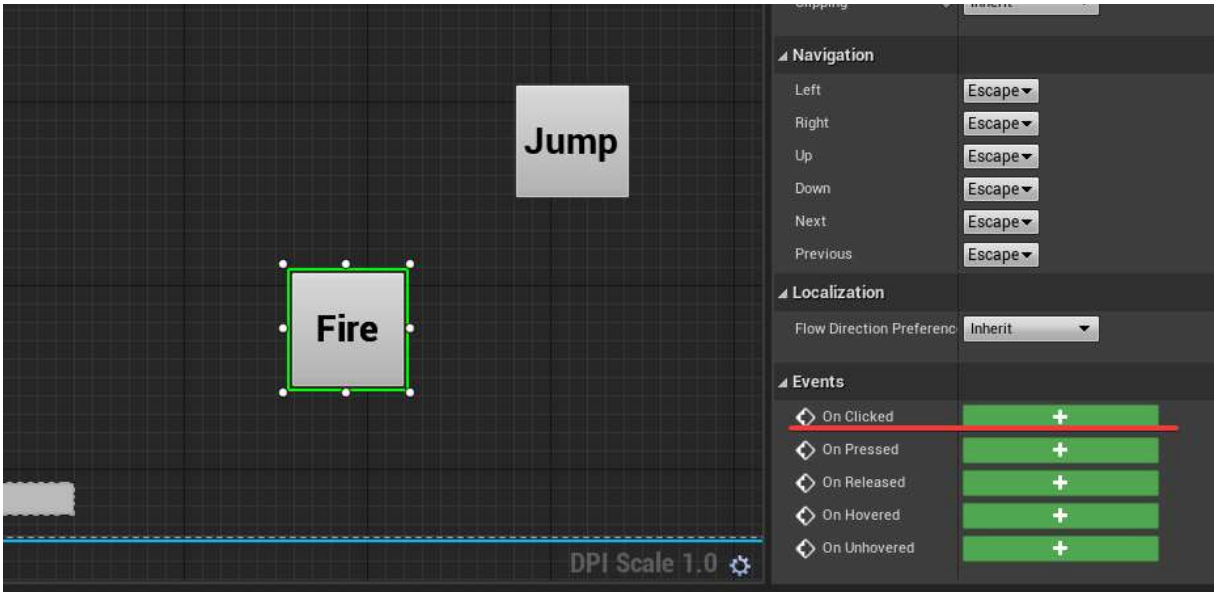

Рисунок 4.9 – Приклад оформлення кнопок користувацького інтерфейсу, та додавання реакції на подію кліку

У результаті відкриється меню редагування блюпрінту даного віджета і з’явиться вузол події кліку на відповідну кнопку. Необхідно додати вузли Jump та Fire, які були завчасно реалізовані, та під’єднати їх до відповідних кнопок як це показано на рис. 4.10. Після внесення цих змін до Widget Blueprint, необхідно виконати його компіляцію та зберегти зміни.

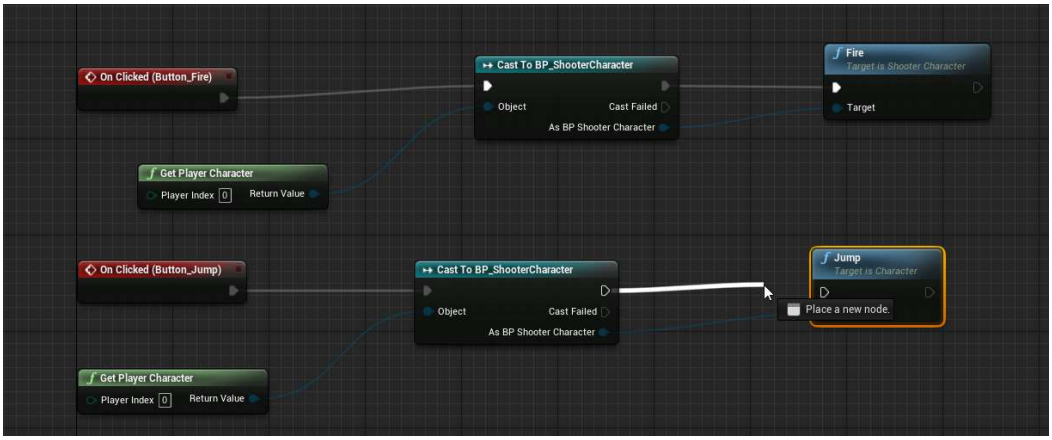

Рисунок 4.10 – Імплементація реакції на подію кліку

Тепер гра буде мати відповідні кнопки для стрільби та стрибка, що забезпечить повну функціональність гри на мобільних пристроях, як показано на рис. 4.11.

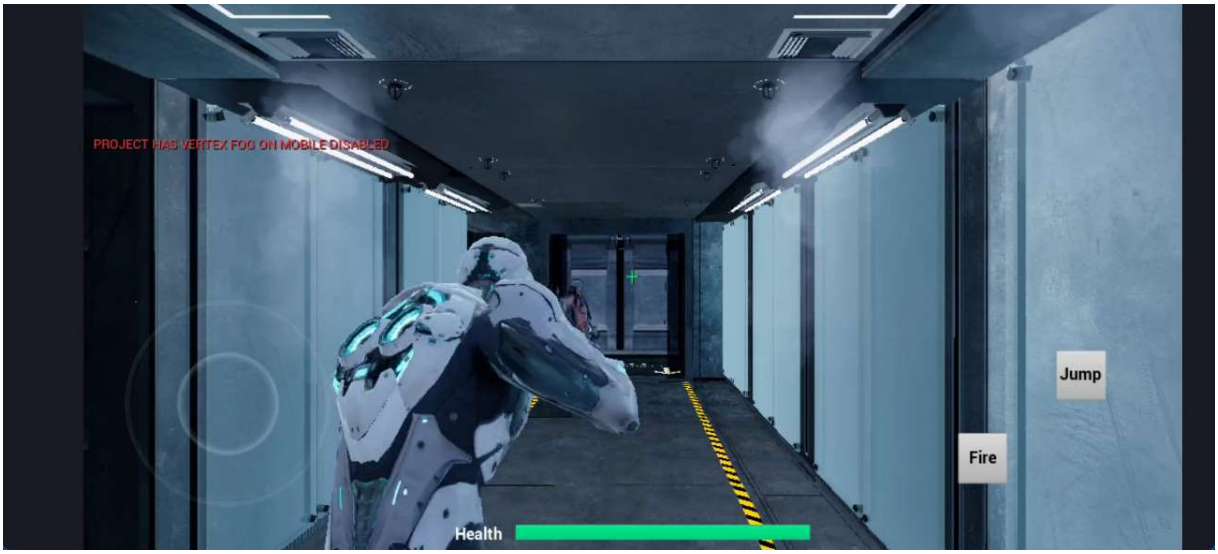

Рисунок 4.11 – Демонстрація виправленого користувацького інтерфейсу

Описані вище кроки є важливими для вирішення проблеми відсутності можливості стріляти та стрибати в портованій грі на мобільну платформу. Вони дозволять забезпечити коректну взаємодію гравця з грою через сенсорний дисплей, що є ключовим аспектом портування ігрових застосунків на мобільні пристрої.

Важливо зауважити, що дизайн користувацького інтерфейсу вимагає великої кількості знань та досвіду і зазвичай виконується окремою людиною, кваліфікованою в UI/UX. У межах даного дослідження було розглянуто лише технічну сторону даної проблеми, коли як розробка конкретної реалізації може слугувати темою окремої роботи.

РОЗДІЛ 5. ЗАПУСК ПОРТОВАНОГО ЗАСТОСУНКУ

5.1 Перевірка частоти кадрів та функціональності ігрових механік

Під час перевірки частоти кадрів та функціональності ігрових механік, було виявлено покращення у частоті кадрів з 15 до 30 кадрів за секунду, а в деяких місцях навіть до 60 кадрів за секунду. Ще більших значень було досягнуто в застосунку скомпільованому без інструментів розробника, які значно впливають на швидкодію. Максимальна частота кадрів змінилась не суттєво, однак зросла стабільність та плавність ігрового застосунку. Мінімальне значення частоти кадрів не перетинало позначку 30 FPS, Це позитивно впливає на плавність гри та загальне відчуття швидкодії геймплею.

Однак, можна помітити велику різницю у частоті кадрів залежно від місцезнаходження гравця. Деякі місця на ігровій сцені вимагають додаткової роботи з боку команди 3D-художників для досягнення максимальної оптимізації. Це може включати спрощення графічних об'єктів, оптимізацію текстур чи освітлення, спрощення анімацій персонажів, а також налаштування параметрів, які впливають на продуктивність.

Наслідком покращення частоти кадрів може бути помітне зниження якості деяких зображень. Проте, варто зазначити, що таке зниження є нормальним для мобільних платформ, де ресурси обмежені. Подальші спрощення графічних об'єктів та додаткові налаштування можуть допомогти покращити якість зображення при збереженні стабільної частоти кадрів.

У рамках даної роботи були розглянуті лише базові проблеми, з якими може стикнутись розробник на перших етапах портування гри на мобільні платформи.

Оптимізація графіки та функціональності є складним процесом, який може вимагати подальшої роботи та співпраці між розробниками, художниками та тестувальниками для досягнення найкращих результатів.

5.2 Порівняльний аналіз продуктивності ігрового застосунку на мобільних пристроях до та після оптимізації

Для порівняння швидкодії додатку, ми будемо звертати увагу на наступні характеристики:

– час, витрачений графічним ядром на відмальовування кадру;
– час, витрачений центральним процесором на обробку ігрової логіки;
– час простою центрального процесора в очікуванні на графічне ядро;
– суб'єктивна оцінка плавності та приємності ігрового досвіду.

Як можемо бачити на рис. 5.1 у полі «Frame», до внесеної оптимізації графічне ядро на один кадр витрачало 65 мс, тоді як після змін витрачається 43 мс, що вказує на те, що нам вдалось знизити навантаження на GPU.

Рисунок 5.1 – Різниця в часі затраченому на один кадр

До внесення змін, як показано на рис. 5.1 у полі «Game», центральний процесор витрачав на обчислення 13 мс, а після змін цей час уже становить 12 мс. Важливо зауважити, що в цей час не враховується очікування на графічне ядро. Це було очікувано, оскільки жодних оптимізацій CPU в межах даного проекту не проводилось, оскільки найбільш навантаженим був саме GPU.

Щоб побачити саме час простою центрального процесора, потрібно використати Unreal Insights, про який згадувалось вище. Отже, до змін процесор в режимі очікування знаходився 40 мс, після змін лише 20мс, що якраз збігається з даними про те, що графічне ядро почало працювати на 20 мс менше за один кадр (рисунок 5.2).

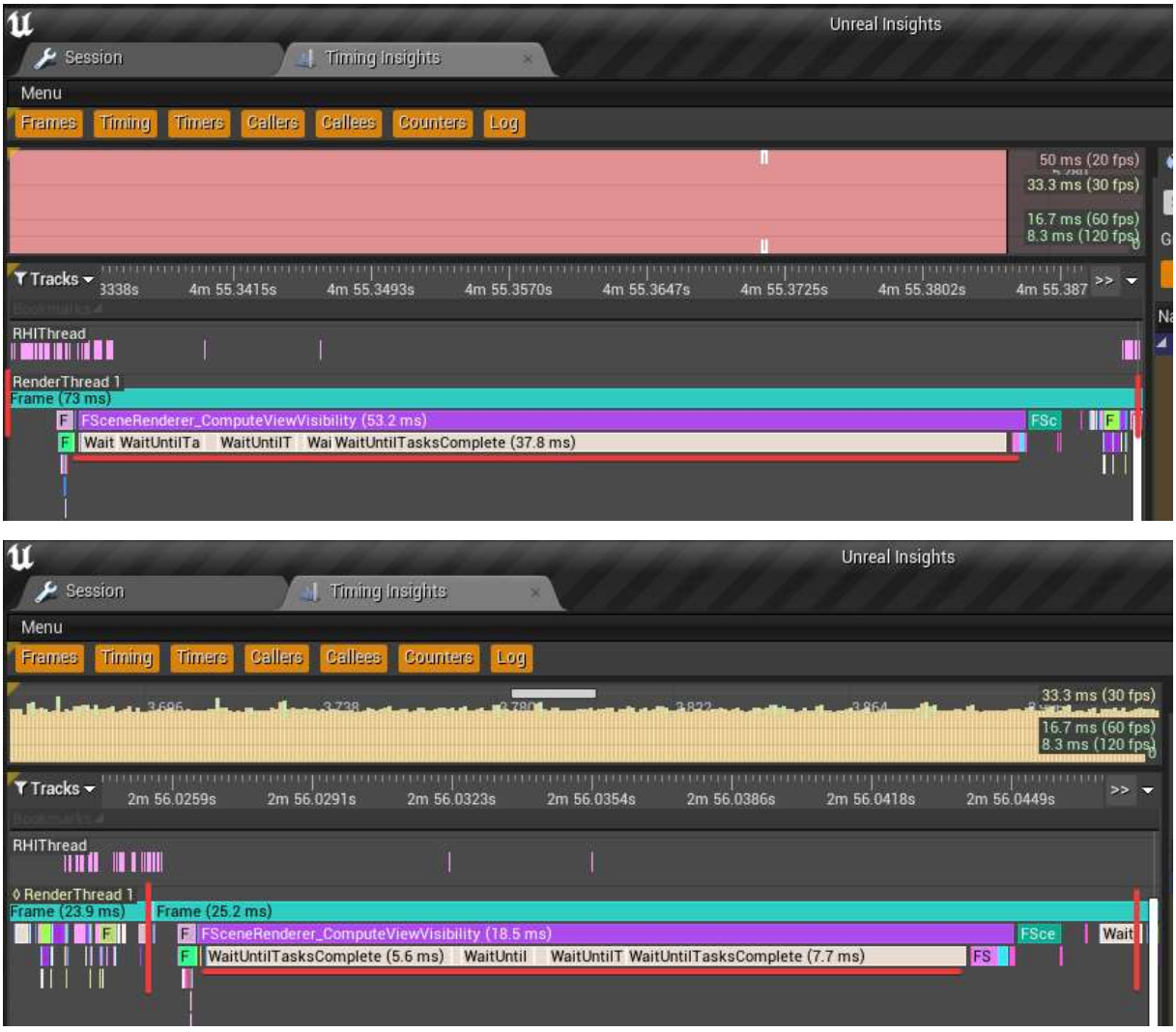

Рисунок 5.2 – Демонстрація часу затраченого CPU в очікування на GPU до та після виконаних оптимізацій

Можемо помітити, що в нашому випадку все вдалось бездоганно. Час витрачений виключно центральним процесором знизився лише за рахунок меншого очікування на GPU, однак не враховуючи цього параметру, він був

доволі прийнятним в межах нашої роботи, та не вимагав додаткових оптимізацій.

Певні методи зниження цього показника були описані в роботі раніше. Щодо суб’єктивної оцінки плавності ігрового досвіду варто зазначити, що на мою думку грати стало набагато зручніше та приємніше, немає сильних зависань, що заважали виконанню цілі гри, а також з’явилась можливість змагатися з противниками, а не лише тікати від них, що безсумнівно збільшує зацікавленість.

ВИСНОВКИ

У роботі проведено дослідження рушіїв Unity, Unreal Engine та CryEngine. Основна увага приділена рушію Unreal Engine, який широко застосовується в індустрії геймдеву, оскільки надає велику кількість інструментів для розробки ігор, а також широкий спектр підтримуваних платформ, включаючи мобільні пристрої.

У роботі досліджено процес портування ігрового застосунку на мобільні платформи з використанням Unreal Engine. З цією метою проаналізовано процес налаштування пристроїв з ОС Android для розробки ігрових застосунків у середовищі Unreal Engine.

Також у роботі визначено налаштування середовища Unreal Engine для розробки під ОС Android, яке включає в себе встановлення Android Studio, завантаження Android SDK Command-line Tools та налаштування шляхів до SDK, NDK та JDK. Це дозволяє Unreal Engine коректно взаємодіяти з Android-платформою та забезпечує можливість розробки ігрових застосунків для мобільних пристроїв.

Проведено запуск ігрового застосунку та здійснено аналіз наявних проблем, зокрема, значна увага частоті кадрів та проблемам з інтерфейсом, що можуть обмежувати функціональність головного персонажа

Здійснено профайлінг гри та визначено ботлнеки з метою виявлення проблемних ділянок коду або ресурсів, які спричиняють незадовільну продуктивність з допомогою інструментів «Unreal Insights», та «Session Frontend».

Проведено оптимізацію графіки та програмного коду, для поліпшення продуктивності гри. Встановлено правильне налаштування параметрів, використано LODs, та налаштування його мінімального значення.

Проведено порівняльний аналіз продуктивності ігрового застосунку на мобільних пристроях до та після оптимізації. Час витрачений виключно центральним процесором знизився лише за рахунок меншого очікування на

GPU, однак не враховуючи цього параметру, він був доволі прийнятним в межах нашої роботи, та не вимагав додаткових оптимізацій.

Під час виконання роботи вдалось ознайомитись з багатьма різними інструментами та методиками, дізнатись більше інформації про рушій «Unreal Engine», та про його можливості портування і оптимізації.

Було сформовано ряд практичних рекомендацій для розробників, які в майбутньому можуть стикнутись з даним завданням. Запропонована вище інструкція хоч і підкріплена прикладами з «Unreal Engine», але може бути використана і в будь якому іншому рушії, оскільки перераховані методики є агностичними від фреймворку

Враховуючи вище зазначене, можна сказати, що поставлена мета була досягнута та всі завдання дослідження виконано. Запропоновані рекомендації щодо ефективного портування ігрових застосунків на мобільні платформи з використанням Unreal Engine можуть використовуватись гейм-девелоперами для вирішення подібного кола проблем.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Роберт К. Мартін. Чиста архітектура. Пер. з англ. І. Бондар-Терещенко. Х. : Вид-во Фабула, 2019. 273 с. ISBN: 978-617-09-5286-8.
2. Alireza Tavakkoli, Game Development and Simulation with Unreal Technology, Second Edition, A K PETERS, 2018. ISBN: 1138092193, 9781138092198.
3. John M. Blain. The Complete Guide to Blender Graphics: Computer Modeling & Animation, 6th Edition. Publisher: CRC Press, 2020. 550 p. ISBN-10: 0367553619.
4. Unreal Engine: Сайт продукту. [Електронний ресурс]. URL: https://www.unrealengine.com/en-US/.
5. Joanna Lee. Learning Unreal Engine Game Development. Publisher: Packt Publishing, 2016. 274 p. ISBN-10: 1784398152, 9781784398156.
6. Isbister, Katherine. Celia Hodent. Game Usability: Advancing the Player Experience. 2nd Edition. CRC Press, 2022. 438 p. ISBN-10: 036761992X.
7. Unity Technologies. Unity Documentation: Official Documentation for Unity Engine. [Online]. URL: https://docs.unity3d.com/.
8. Unity Technologies. Unity Learn: Tutorials and Guides for Unity Engine. [Online]. Available: URL://learn.unity.com/.
9. Shannon, Tom. Unreal Engine 4 for Design Visualization: Developing Stunning Interactive Visualizations, Animations, and Renderings (Game Design). Pearson Education (US). 384 p. ISBN: 9780134680705.
10. Epic Games. Unreal Engine Documentation: Official Documentation for Unreal Engine. [Online]. URL: https://docs.unrealengine.com/
11. Norman, Donald A. The Design of Everyday Things. Basic Books, 2013. 386 p.
12. Satheesh PV, Unreal Engine 4 Game Development Essentials: Master the basics of Unreal Engine 4 to build stunning video games, Packt Publishing, 2016. ISBN: 978-1-78439-196-6.
13. Justin Plowman, 3D Game Design with Unreal Engine 4 and Blender, Packt Publishing, 2016. ISBN: 1785881469,9781785881466.

14. Katax Emperore, Devin Sherry. Unreal Engine Physics Essentials. Publisher: Packt Publishing, 2015. 327 p.
15. Jesse Schell. The Art of Game Design: A Book of Lenses. CRC Press, 3rd Edition. 652 p. 2019. ISBN-10: 1138632058.
16. Android Developers. Android Studio Documentation: Official Documentation for Android Studio. [Online]. URL: https://developer.android.com/studio/documentation.
17. John P. Doran. Unreal Engine Game Development Cookbook. Publisher: Packt Publishing, 2015. 326 p. ISBN-10: 1784398160.
18. Andrew Sanders, An Introduction to Unreal Engine 4 (Focal Press Game Design Workshops), CRC Press, 2016. ISBN 13: 978-1-4987-6509-1.
19. Marcos Romero; Brenden Sewell, Blueprints Visual Scripting for Unreal Engine, Packt Publishing Limited, 2019. ISBN: 9781789347067.
20. Aram Cookson, Ryan Dowling Soka, Clinton Crumpler, Unreal Engine 4 Game Development in 24 Hours, Sams Publishing, 2016. ISBN: 9780134389103.
21. Adams, Ernest. Fundamentals of Game Design. New Riders, Publisher: New Riders; 3-rd edition, 2013. 576 p. ISBN-10: 0321929675.
22. Frank D. Luna. Introduction to 3D Game Programming with Directx 11. Publisher: Wiley. 600 p. ISBN: 9781936420223.
23. Schell Jesse, Zak Parrish and Joel Van Eenwyk. Mastering Unreal Technology: The Art of Level Design. Publisher: Sams: First Edition, 2005. 984 p. ISBN-10: 9780672326929.
24. Hartson, Rex, and Pardha Pyla. The UX Book: Process and Guidelines for Ensuring a Quality User Experience. Morgan Kaufmann, 2015. 976 p. ISBN-10: 0123852412.
25. McCaffrey, Mitch, Unreal Engine VR Cookbook: Developing Virtual Reality with UE4, Pearson Education, 2017. ISBN-13: 978-0-13-464917-7.