МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «**ДОСЛІДЖЕННЯ МОЖЛИВОСТЕЙ РУШІЯ UNREAL ENGINE ДЛЯ СТВОРЕННЯ ВІРТУАЛЬНИХ ІГРОВИХ СЕРЕДОВИЩ**»

Виконала: студентка 2 курсу, групи 8.1223-з
спеціальності 122 комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми комп'ютерні науки
(назва освітньої програми)
М. О. Ковальчук
(ініціали та прізвище)
Керівник доцент кафедри комп'ютерних наук, доцент, д.е.н. Полуектова Н.Р.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент Проректор з наукової роботи Національного авіаційного університету, професор, д.т.н. Гнатюк С.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти магістр

Спеціальність 122 комп'ютерні науки
(шифр і назва)

Освітня програма комп'ютерні науки

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

_____ Шило Г.М.
(підпис)

" ____ " _____ 2025 р.

З А В Д А Н Н Я НА КВАЛІФІКАЦІЙНУ РОБОТУ

СТУДЕНТЦІ

Ковальчук Майї Олегівні
(прізвище, ім'я та по-батькові)

1. Тема роботи Дослідження можливостей рушія Unreal Engine для створення віртуальних ігрових середовищ

керівник роботи Полуектова Н.Р., доцент кафедри комп'ютерних наук, д.е.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » березня 2024 року № 384-с

2. Строк подання студентом роботи 19.02.2025

3. Вихідні дані до роботи *Постановка задачі:* Дослідити можливості рушія Unreal Engine для створення віртуальних ігрових середовищ.
Перелік літератури: Guillemette, R. Unreal Engine 5 Character Creation, Animation, and Cinematics. Mercury Learning and Information, 2023. Unreal Engine Documentation. URL: https://docs.unrealengine.com/ Офіційний веб-сайт Unreal Engine. URL: https://www.unrealengine.com/.
Вихідні програмні дані: Операційна система Windows 10 або вище. Програмне забезпечення Unreal Engine 5. Програмне забезпечення для 3D-моделювання (Blender, Autodesk Maya)..

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі: Розробити прототип віртуального ігрового середовища з використанням основних інструментів Unreal Engine. Провести тестування та оптимізацію розробленого середовища.
2. Основні теоретичні відомості. Концепція віртуального ігрового середовища. Огляд існуючих ігрових рушіїв. Основні особливості *UE*. Проблеми та виклики при створенні віртуальних ігрових середовищ Етапи моделювання віртуального ігрового середовища. Принципи створення 3d-сцени та персонажів. Симуляція фізичних процесів.
Розділ 1. Теоретичний аналіз проблеми дослідження
Розділ 2. Моделювання віртуального ігрового середовища
Розділ 3. Опис процесу розробки віртуального ігрового середовища

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) презентація

6. Консультанти розділів роботи

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

7. Дата видачі завдання 04.03.2024

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	22.04.24	
2.	Збір вихідних даних.	24.05.24	
3.	Обробка методичних та теоретичних джерел.	12.09.24	
4.	Розробка першого та другого розділу.	12.10.24	
5.	Розробка третього розділу.	12.12.24	
6.	Оформлення та нормоконтроль кваліфікаційної роботи магістра.	19.02.25	
7.	Захист кваліфікаційної роботи.	05.03.25	

Студент _____ М.О. Ковальчук
(підпис) (ініціали та прізвище)

Керівник роботи _____ Н. Р. Полуектова
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

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

РЕФЕРАТ

Кваліфікаційна робота магістра «Дослідження можливостей рушія Unreal Engine для створення віртуальних ігрових середовищ»: 62 с., 33 рис., 1 табл., 34 джерела, 4 додатки.

Ключові слова: Unreal Engine, віртуальне ігрове середовище, ігровий рушій, комп'ютерна графіка, 3D-моделювання.

Об’єкт дослідження – процес створення віртуальних ігрових середовищ з використанням сучасних рушіїв.

Мета роботи полягає в аналізі можливостей рушія Unreal Engine для створення віртуальних ігрових середовищ, а також у розробці прототипу такого середовища з використанням основних інструментів рушія.

Метод дослідження – теоретичний аналіз, емпіричне дослідження, моделювання та симуляція, тестування.

У роботі досліджено можливості Unreal Engine для створення віртуальних ігрових середовищ, включаючи етапи моделювання, принципи створення 3D-сцен та персонажів, методи симуляції фізичних процесів, питання інтерактивності, проєктування, реалізацію та тестування. Розроблено прототип, проведено тестування та оптимізацію, виявлено проблеми та виклики. Робота містить теоретичний аналіз, опис процесу та результати.

SUMMARY

Master's thesis “Research of the Unreal Engine capabilities for creating virtual game environments”: 62 p., 33 figures, 1 table, 34 sources, 4 appendices.

Keywords: Unreal Engine, virtual game environment, game engine, computer graphics, 3D modeling.

The object of research is the process of creating virtual game environments using modern engines.

The purpose of the study is to analyze the capabilities of the Unreal Engine for creating virtual game environments, as well as to develop a prototype of such an environment using the main tools of the engine.

Research methods: theoretical analysis, empirical research, modeling and simulation, testing.

The paper explores the capabilities of the Unreal Engine for creating virtual gaming environments, including modeling stages, principles of creating 3D scenes and characters, methods of simulating physical processes, interactivity, design, implementation, and testing. A prototype was developed, tested and optimized, and problems and challenges were identified. The work contains theoretical analysis, process description, and

ЗМІСТ

Завдання на кваліфікаційну роботу……..…..2

Реферат 4

Summary 5

Вступ..... 7

1 Теоретичний аналіз проблеми дослідження..... 9

1.1 Концепція віртуального ігрового середовища..... 9

1.2 Огляд існуючих ігрових рушіїв 12

1.3 Основні особливості Unreal Engine..... 15

1.4 Проблеми та виклики при створенні віртуальних ігрових середовищ 17

2 Моделювання віртуального ігрового середовища..... 20

2.1 Етапи моделювання віртуального ігрового середовища 20

2.2 Принципи створення 3D-сцени та персонажів 23

2.3 Симуляція фізичних процесів у віртуальному середовищі..... 28

2.4 Інтерактивність у віртуальному середовищі..... 31

3 Опис процесу розробки віртуального ігрового середовища 34

3.1 Проектування ігрового середовища на базі Unreal Engine 34

3.2 Реалізація ігрового рівня..... 36

3.3 Тестування та виправлення помилок..... 40

Висновки 45

Перелік посилань..... 48

Додатки..... 52

ВСТУП

Розвиток ігрової індустрії та постійне вдосконалення технологій створення віртуальних середовищ є одними з найбільш динамічних і перспективних напрямів сучасної комп'ютерної науки. Віртуальні середовища знаходять широке застосування не лише в розважальній сфері, але й в освітніх, медичних та інженерних проєктах, де вони стають інструментом для моделювання і симуляції реальних процесів. Рушій Unreal Engine (UE) є однією з провідних платформ для розробки віртуальних ігрових середовищ, завдяки своїй потужності, гнучкості та підтримці новітніх графічних і фізичних технологій. Дослідження можливостей цього рушія дозволить краще зрозуміти, як сучасні технології сприяють створенню реалістичних і високоефективних віртуальних середовищ, що є особливо важливим для розвитку інновацій в ігровій індустрії та суміжних галузях.

Мета дослідження полягає в аналізі можливостей рушія Unreal Engine для створення віртуальних ігрових середовищ, а також у розробці прототипу такого середовища з використанням основних інструментів рушія.

Для досягнення поставленої мети необхідно вирішити наступні **завдання**:

1. Проаналізувати існуючі ігрові рушії та їхні можливості для моделювання віртуальних середовищ.
2. Змоделювати бізнес-процеси RPG на базі рушія Unreal Engine;
3. Розробити віртуальне ігрове середовище на базі Unreal Engine, використовуючи основні принципи моделювання.
4. Провести тестування і оптимізацію розробленого середовища для забезпечення його стабільної роботи.
5. Виявити можливі проблеми та виклики, пов’язані з розробкою віртуальних ігрових середовищ у контексті продуктивності та якості графіки.

Об'єктом дослідження є процес створення віртуальних ігрових середовищ з використанням сучасних рушіїв.

Предметом дослідження є можливості та інструментарій Unreal Engine для моделювання та розробки віртуальних ігрових середовищ.

У роботі використано такі **методи дослідження**: *теоретичний аналіз наукової* і технічної літератури з теми ігрових рушіїв та технологій віртуальних середовищ; *емпіричне дослідження* шляхом розробки прототипу ігрового середовища на базі Unreal Engine; *моделювання та симуляція* для створення тривимірних ігрових сцен і фізичних процесів; моделювання бізнес процесів предметної області; *тестування* розробленого середовища для оцінки його продуктивності та якості.

Наукова новизна дослідження полягає у здійсненні комплексного порівняльного аналізу ключових характеристик сучасних ігрових рушіїв (Unreal Engine, Unity, Godot Engine, CryEngine, Amazon Lumberyard) з метою визначення оптимального інструментарію для створення реалістичних віртуальних середовищ. Уточнено визначення поняття «віртуальне ігрове середовище» як програмно-апаратного комплексу, що забезпечує занурення користувача в інтерактивний 3D-простір з виділенням його структурних компонентів.

Розроблено та оптимізовано прототип RPG на базі Unreal Engine, що демонструє можливості рушія для створення великомасштабних ігор з високою якістю графіки та продуктивністю.

Теоретичне значенняквалiфікаційної роботи полягає у розширенні теоретичних уявлень про процес створення віртуальних ігрових середовищ. Визначено критерії вибору ігрового рушія для розробки проектів різної складності.

Розроблений прототип може бути використаний як основа для створення повноцінної RPG. Отримані результати можуть бути застосовані для оптимізації існуючих ігрових проектів.

Магістерська робота складається з трьох розділів, вступу, висновків та списку використаних джерел.

1 ТЕОРЕТИЧНИЙ АНАЛІЗ ПРОБЛЕМИ ДОСЛІДЖЕННЯ

У першому розділі проведено теоретичний аналіз проблеми дослідження (визначено поняття "віртуальне ігрове середовище", структуру і компоненти віртуального ігрового середовища), розглянуто найпопулярніші ігрові рушії та особливості Unreal Engine.

1.1 Концепція віртуального ігрового середовища

Віртуальне ігрове середовище – це інтерактивний цифровий простір, створений за допомогою ігрових рушіїв, який забезпечує взаємодію гравця з 3D-світом. Воно характеризується інтерактивністю, реалістичністю, фізичною симуляцією та мережевим мультиплеєром [4915, 16].

Основними складовими віртуального ігрового середовища є графічний рушій, який відповідає за рендеринг, освітлення, тіні та анімацію персонажів, а також фізичний рушій, що реалізує реалістичне відображення законів фізики та симуляцію зіткнень. Штучний інтелект та логіка також відіграють важливу роль у створенні інтерактивного та реалістичного середовища [5].

Концепція віртуального ігрового середовища передбачає створення штучних інтерактивних світів за допомогою комп'ютерних технологій. Ці середовища дозволяють користувачам створювати аватари, взаємодіяти з віртуальними об'єктами та іншими гравцями в режимі реального часу. Віртуальні ігрові середовища можуть бути багатокористувацькими онлайн-іграми (ММО), спільними віртуальними середовищами для співпраці та соціальних взаємодій.

Віртуальні ігрові середовища характеризуються стійкістю, тобто вони продовжують існувати незалежно від присутності користувача, зберігаючи внесені зміни. Спільність дозволяє багатьом користувачам взаємодіяти в одному віртуальному світі одночасно. Взаємодія відбувається в режимі реального часу, що забезпечує високу ступінь інтерактивності та занурення в віртуальний світ.

Автоматизовані правила, наприклад, фізика світу, керують поведінкою об'єктів у віртуальному середовищі [1].

Прикладами таких середовищ можуть служити багатокористувацькі онлайн-ігри типу World of Warcraft та Final Fantasy XIV, а також спільні віртуальні середовища типу Second Life та VRChat. Технології віртуальної реальності (VR) та доповненої реальності (AR) значно розширюють можливості віртуальних ігрових середовищ, забезпечуючи більш повне занурення та інтерактивність.

З точки зору технологічної підтримки, віртуальні ігрові середовища сильно залежать від розвитку інтернет-технологій та мережевих інфраструктур, які забезпечують підключення та взаємодію між користувачами в режимі реального часу. Таким чином, концепція віртуальних ігрових середовищ інтегрує досягнення інформаційних технологій, соціальної психології та дизайну інтерактивних систем для створення нових форм спілкування та розваг.

Віртуальні ігрові середовища знаходять широке застосування в різних сферах, включаючи розваги (AAA-ігри та VR-контент), навчання (освітні симулятори та військові тренажери), архітектуру та дизайн (візуалізація проєктів у реальному часі), а також медицину (VR-реабілітація та тренування хірургів) [5, 7].

Перспективи розвитку віртуальних ігрових середовищ включають збільшення фотореалістичності за допомогою нових технологій рендерингу, використання нейромереж для генерації контенту, розширення VR/AR інтеграції у відеоігри та симулятори, а також поліпшення мережевих технологій для масових багатокористувацьких світів [17, 4].

Отже, під *віртуальним ігровим середовищем* будемо розуміти – *програмно-апаратний комплекс, що забезпечує користувачеві можливість занурення в інтерактивний тривимірний простір, де він може взаємодіяти з об'єктами та іншими користувачами за допомогою різних пристроїв введення-виведення.*

Віртуальне ігрове середовище є складною багатокомпонентною системою, що поєднує програмні та апаратні елементи для створення інтерактивного

користувацького досвіду. Основу цього середовища становить ігровий рушій, який виконує ключові функції, пов'язані з обробкою графіки, фізики, звуку, штучного інтелекту та мережевої взаємодії. До найпоширеніших рушіїв належать Unreal Engine, Unity та CryEngine, які забезпечують високий рівень деталізації та оптимізацію обчислювальних процесів.

Одним із важливих компонентів є *графічний підкомпонент*, який відповідає за візуалізацію тривимірного простору та об'єктів у грі. Він включає 3D-моделі, текстури, систему освітлення та ефекти, що формують реалістичне середовище. Розширені алгоритми рендерингу дозволяють досягти високої якості зображення та оптимізації продуктивності.

Фізичний підкомпонент імітує закони фізики, завдяки чому взаємодія об'єктів у грі виглядає реалістичною. Цей модуль відповідає за обробку зіткнень, гравітації, поведінки рідин, деформацій та інших фізичних ефектів. Сучасні фізичні рушії, такі як NVIDIA PhysX або Havok, дозволяють точно моделювати фізичні явища, що підвищує рівень занурення гравця в ігровий процес.

Звуковий підкомпонент забезпечує просторове звучання, відтворення звукових ефектів, музичного супроводу та голосової комунікації. Використання багатоканального аудіо та технологій просторового звучання дозволяє створювати атмосферу, що відповідає динаміці подій у грі. Завдяки цьому гравці можуть орієнтуватися у просторі за допомогою звуку, що є критично важливим у шутерах, симуляторах та VR-додатках.

Штучний інтелект (ШІ) використовується для моделювання поведінки неігрових персонажів (NPC) та інших інтерактивних об'єктів. Завдяки алгоритмам машинного навчання та поведінковим моделям NPC можуть адаптуватися до дій гравця, створюючи реалістичні сценарії взаємодії. Сучасні ігрові рушії дозволяють інтегрувати складні моделі ШІ, що забезпечує динамічний ігровий процес та підвищує рівень реалізму.

Мережевий підкомпонент є необхідним для багатокористувацьких ігор, оскільки забезпечує взаємодію між гравцями в режимі реального часу. Він відповідає за синхронізацію даних, управління мережевими затримками та

оптимізацію передавання інформації. Розробники використовують серверні технології та хмарні обчислення для забезпечення стабільного з'єднання та масштабованості ігрових світів.

Інтерфейс користувача (UI) дозволяє гравцеві взаємодіяти з віртуальним середовищем через меню, інформаційні панелі, індикатори здоров'я, міні-карти та інші елементи управління. Грамотне проєктування UI сприяє інтуїтивному використанню гри, спрощуючи навігацію та доступ до важливих функцій.

Пристрої введення-виведення виконують функцію комунікації користувача з грою. Серед них клавіатура, миша, геймпад, VR-шоломи та сенсорні екрани, що дають змогу зануритися в ігровий процес. Використання сучасних контролерів, таких як тактильний зворотний зв'язок або системи відстеження рухів, значно розширює можливості взаємодії у віртуальному середовищі.

Таким чином, структура віртуального ігрового середовища (Додаток Б) складається з багатьох взаємопов'язаних компонентів, що забезпечують реалістичність, інтерактивність та комфортність користувацького досвіду. Інтеграція передових технологій у кожен з цих аспектів сприяє створенню захоплюючих віртуальних світів, які відповідають сучасним вимогам ігрової індустрії.

1.2 Огляд існуючих ігрових рушіїв

Аналіз ігрових рушіїв є важливим етапом у розробці віртуальних ігрових середовищ, оскільки вибір правильного інструменту визначає можливості та ефективність кінцевого продукту. На сьогодні існує велика кількість ігрових рушіїв, кожен з яких має свої особливості, переваги та недоліки.

Одним з найпопулярніших є Unreal Engine, який відомий своєю потужністю та гнучкістю. Він надає широкий спектр інструментів для створення високоякісної графіки, реалістичної фізики та складних ігрових механік. Unreal

Engine підтримує різні платформи, включаючи ПК, консолі, мобільні пристрої та VR/AR-пристрої [19].

Unity – ще один популярний ігровий рушій, який відрізняється своєю простотою використання та кросплатформеністю. Він є чудовим вибором для розробки 2D та 3D ігор, а також інтерактивних додатків. Unity має велику спільноту розробників та широкий спектр навчальних матеріалів [10, 14].

Godot Engine – це безкоштовний і відкритий ігровий рушій, який стає все більш популярним завдяки своїй гнучкості та потужності. Він підтримує 2D та 3D графіку, має вбудований скриптовий рушій та надає широкий спектр інструментів для розробки ігор [9].

CryEngine – відомий своєю здатністю створювати високоякісну графіку та реалістичні віртуальні світи. Він часто використовується для розробки AAA-ігор з високими вимогами до графіки [2].

Amazon Lumberyard – це безкоштовний ігровий рушій, заснований на CryEngine. Він інтегрований з сервісами Amazon Web Services, що робить його привабливим для розробки онлайн-ігор [9].

У Додатку А наведено результати порівняльного аналізу п’яти поширених ігрових рушіїв: Unreal Engine, Unity, Godot Engine, CryEngine та Amazon Lumberyard. Аналіз здійснювався за низкою ключових параметрів, що визначають функціональні можливості, зручність використання та перспективи застосування кожного з рушіїв у розробці ігрових проєктів.

Зокрема, було розглянуто аспекти ліцензування та вартісної моделі, включаючи умови використання, наявність безкоштовних версій і розмір роялті. Оцінювалася якість документації, її повнота, актуальність та зручність навігації, а також рівень офіційної технічної підтримки. Аналіз торкався активності спільноти, доступності форумів, вікі-ресурсів та навчальних матеріалів, що відіграють ключову роль у вивченні й ефективному використанні рушіїв.

Досліджено поріг входження, складність освоєння інтерфейсу та програмування, а також наявність засобів для візуального програмування, що полегшують створення ігрових механік без необхідності глибокого знання мов

програмування. Окрему увагу приділено інструментам для анімації, які передбачають вбудовані засоби або можливість інтеграції з зовнішніми програмними пакетами.

Оцінено фізичний рушій, його реалістичність та можливості симуляції фізичних взаємодій. Вивчено інструменти для роботи зі звуком, включаючи вбудовані редактори та можливість інтеграції з професійними аудіоредакторами. Проаналізовано підтримку багатокористувацької гри, наявність вбудованих рішень для мережевої взаємодії, створення серверної та клієнтської частин.

Досліджено можливості створення інтерфейсу користувача (UI), зокрема наявність вбудованих бібліотек та підтримку сторонніх UI-фреймворків. Важливим аспектом оцінки стала підтримка інструментів для реалізації штучного інтелекту (AI), що включають створення поведінкових моделей NPC та підтримку технологій машинного навчання.

Також розглянуто екосистему плагінів та асетів: наявність маркетплейсів із готовими ресурсами, доступність та зручність інтеграції сторонніх модулів. Досліджено стабільність і надійність рушіїв, частоту випуску оновлень та оперативність виправлення помилок. Для кожного рушія наведено приклади успішних комерційних ігор або проєктів, що демонструють його можливості.

Оцінено здатність рушіїв до роботи з відкритими світами, включаючи інструменти для створення великих ігрових локацій та оптимізацію ландшафтів. Важливим параметром стала підтримка мобільних платформ, зокрема оптимізація продуктивності на мобільних пристроях та доступність засобів тестування.

Проаналізовано можливості для розробки VR/AR-додатків, підтримку конкретних пристроїв віртуальної та доповненої реальності, а також наявність спеціалізованих шаблонів. Досліджено інструменти для рендерингу в реальному часі, включаючи підтримку трасування променів та сучасних методів глобального освітлення.

Оцінено можливості рушіїв для роботи з даними: імпорт та експорт ресурсів, інтеграцію з базами даних та іншими зовнішніми системами. Важливим

параметром стала доступність навчальних матеріалів, курсів і сертифікаційних програм, що сприяють професійному зростанню розробників.

Аналіз торкався гнучкості та розширюваності рушіїв, що визначає їхню придатність до модифікації під специфічні вимоги проєктів. Окремо оцінено кросплатформеність редакторів, можливість їхньої роботи на різних операційних системах, а також інструменти для спільної розробки, що включають системи контролю версій та засоби командної комунікації.

Завершальний аспект дослідження – можливості для швидкого прототипування, що включають наявність готових шаблонів, інструментів для швидкого створення ігрових механік та підтримку процесу тестування концепцій.

Отримані результати дозволяють визначити оптимальний рушій залежно від потреб розробників, характеру проєкту та обмежень за ресурсами. Звісно, Unreal Engine та Unity є лідерами ринку, але Godot Engine та CryEngine також мають свої переваги.

1.3 Основні особливості Unreal Engine

Unreal Engine (UE) – це передовий ігровий рушій, розроблений компанією Epic Games, який широко застосовується не лише у сфері відеоігор, а й у розробці інтерактивних 3D-додатків, архітектурної візуалізації, кінематографії, віртуальної та доповненої реальності. Завдяки потужному графічному рушію та розширеним можливостям, Unreal Engine дозволяє створювати реалістичні візуальні ефекти, фізичні симуляції та масштабовані віртуальні середовища, що робить його універсальним інструментом для різних галузей.

Однією з ключових переваг Unreal Engine є його здатність забезпечувати фотореалістичну графіку завдяки сучасним технологіям рендерингу. Зокрема, підтримка трасування променів у реальному часі (Real-Time Ray Tracing) дає можливість отримати високоякісне освітлення, реалістичні тіні та рефлекси, що

значно покращує загальне візуальне сприйняття сцени. Інноваційна система глобального освітлення Lumen дозволяє автоматично підлаштовувати освітлення в залежності від змін у сцені, що особливо корисно для динамічних середовищ, таких як відкриті світи в іграх або інтерактивні симуляції.

Для оптимізації продуктивності та збільшення рівня деталізації UE використовує технологію Nanite, яка дозволяє імпортувати і працювати з геометрією, що складається з мільярдів полігонів без значного впливу на продуктивність. Це відкриває нові можливості для розробників, зокрема в створенні великих ігрових світів із надзвичайно детальними об’єктами, таких як кам’яні утворення, будівлі та природні ландшафти.

Суттєвою перевагою Unreal Engine є система візуального скриптування Blueprints, яка дає змогу створювати складну ігрову логіку без необхідності писати код. Це спрощує процес розробки та відкриває доступ до створення ігор для художників, дизайнерів та інших фахівців, які не мають досвіду програмування. Окрім цього, рушій підтримує програмування на мові C++, що дозволяє більш досвідченим розробникам створювати гнучкі та високопродуктивні рішення.

Unreal Engine також забезпечує широкі можливості для створення реалістичних персонажів завдяки інструменту MetaHumans. Ця технологія дозволяє швидко генерувати цифрових людей з високим рівнем деталізації, включаючи анімацію міміки, текстури шкіри та рухи тіла. Вона широко використовується не лише в іграх, але й у кінематографі, рекламній індустрії та наукових симуляціях.

Ще однією перевагою є кросплатформність рушія, що дає можливість розробляти проєкти для різних платформ, включаючи ПК, консолі (PlayStation, Xbox), мобільні пристрої (iOS, Android), а також пристрої віртуальної (VR) та доповненої реальності (AR). Така гнучкість робить Unreal Engine ідеальним вибором для компаній, які прагнуть розширити свою аудиторію та створювати продукти для різних категорій користувачів.

Активна спільнота та велика кількість навчальних матеріалів є ще одним значним плюсом Unreal Engine. Спільнота розробників постійно ділиться досвідом, створює безкоштовні ресурси та плагіни, що дозволяє швидко освоювати рушій та ефективно працювати з ним. Epic Games регулярно оновлює рушій, додаючи нові функціональні можливості та покращуючи продуктивність.

Таким чином, Unreal Engine є одним із найпотужніших інструментів для розробки інтерактивних проєктів, що поєднує передові технології рендерингу, оптимізації, штучного інтелекту та фізичної симуляції. Його гнучкість, потужність та кросплатформність роблять його ідеальним вибором як для інді-розробників, так і для великих компаній, що працюють у сфері відеоігор, архітектури, кінематографу та віртуальної реальності.

1.4 Проблеми та виклики при створенні віртуальних ігрових середовищ

Створення віртуальних ігрових середовищ (ВІС) є складним процесом, що вимагає комплексного підходу та врахування численних аспектів. Розробники стикаються з низкою викликів, які можна класифікувати на технічні, дизайнерські, педагогічні, соціальні, психологічні та економічні.

Технічні аспекти є ключовими для забезпечення функціональності та якості ВІС. Досягнення балансу між високою деталізацією та продуктивністю залишається однією з головних проблем, особливо у складних 3D-середовищах. Наприклад, використання технології трасування променів (ray tracing) у реальному часі значно покращує якість графіки, але вимагає значних обчислювальних ресурсів [11]. Вибір ігрового рушія, такого як Unreal Engine або Unity, потребує ретельного аналізу відповідності функціональних можливостей проєкту. Unreal Engine, наприклад, відомий своїми потужними інструментами для створення фотореалістичної графіки, тоді як Unity пропонує більшу гнучкість у розробці для різних платформ [5]. Інтеграція VR, AR, AI та фізичного

моделювання вимагає глибоких знань та досвіду, оскільки кожна з цих технологій має свої специфічні вимоги та обмеження [4, 12]. Ефективна взаємодія гравця з віртуальним світом, що включає передбачуваність фізики, зручність керування та реалістичність симуляцій, є критично важливою для забезпечення імерсивного досвіду [7].

Дизайнерські виклики пов'язані зі створенням імерсивного геймплею, що підтримує відчуття присутності, неперервність ігрового світу та його автентичність. Технології VR та штучного інтелекту використовуються для забезпечення глибокого занурення гравця та реалістичної поведінки NPC. Наприклад, використання алгоритмів машинного навчання дозволяє створювати адаптивних NPC, які реагують на дії гравця та змінюють свою поведінку залежно від контексту [17]. Розробка ефективного наративу та рівнів гри вимагає ретельного планування сюжетних ліній, що сприяють залученню користувачів та створюють виклик, який не є надто складним або занадто легким. Наприклад, використання технік сторітелінгу та геймдизайну дозволяє створювати ігри, які не тільки розважають, але й навчають [15].

Педагогічні аспекти стають особливо важливими при розробці навчальних ігор. Інтеграція навчального контенту в ігровий процес вимагає розробки ефективних методів оцінювання знань та залучення учнів до активного навчання через гейміфікацію. Наприклад, освітні ігри можуть містити вбудовані оцінки рівня знань гравців, алгоритми адаптивного навчання та динамічне коригування складності завдань [16].

Соціальні та психологічні аспекти також відіграють важливу роль. Дослідження свідчать, що довготривале перебування у віртуальних середовищах може впливати на психічне здоров'я гравців, а також на їхню соціальну взаємодію. Водночас такі середовища можуть сприяти розвитку навичок співпраці та критичного мислення [13].

Економічні аспекти розробки ігор включають високу вартість створення контенту, особливо у випадку використання VR-технологій. Також виникає потреба у залученні висококваліфікованих спеціалістів, що ускладнює процес

розробки та підвищує його вартість [6]. Важливим викликом є підтримка і оновлення ігрових середовищ після релізу, що вимагає постійних інвестицій у технічну підтримку, оновлення контенту та виправлення багів [1].

Ці проблеми та виклики вимагають комплексного підходу та співпраці фахівців з різних галузей, включаючи розробників, дизайнерів, педагогів, психологів та інших.

2 МОДЕЛЮВАННЯ ВІРТУАЛЬНОГО ІГРОВОГО СЕРЕДОВИЩА

Другий розділ присвячений моделюванню віртуального ігрового середовища на основі інструментарію UE; описано процес попереднього планування та створення концепту; створення геометрії, текстурування та освітлення; охарактеризовано проблеми анімації персонажів та об'єктів у віртуальному середовищі та реалізацію механік взаємодії гравця з об'єктами середовища. Програмування поведінки NPC (неігрових персонажів) та AI.

2.1 Етапи моделювання віртуального ігрового середовища

Моделювання віртуального ігрового середовища в Unreal Engine (UE) є комплексним процесом, що охоплює кілька ключових етапів: від концептуального проєктування до фінального тестування. Для кожного етапу доцільно використовувати UML-діаграми, які допоможуть структурувати процес розробки та взаємодію компонентів гри.

1) Попереднє планування та створення концепту, на цьому етапі формується основна ідея гри, визначаються її механіки, сюжет та стилістика. Ключовим документом на цьому етапі є Game Design Document (GDD), який містить детальний опис ігрового світу, персонажів та ігрових механік. UML-діаграма варіантів використання (рис. 2.1) допомагає візуалізувати взаємодію гравця (Player) з ігровою системою, що є важливим для розуміння основних функціональних вимог до проєкту.

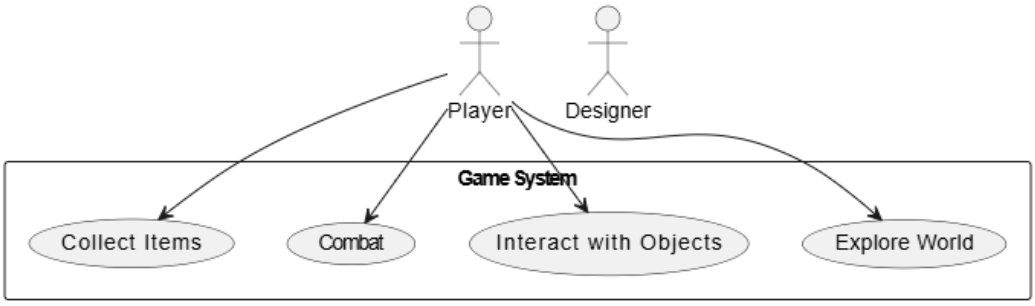


Рисунок 2.1 – UML-діаграма варіантів використання

2) Створення геометрії, текстурування та освітлення – охоплює моделювання 3D-об'єктів, текстурування та налаштування освітлення в UE. Класова структура гри, представлена на UML-діаграмі класів (рис. 2.2), відображає основні класи та їхні взаємодії, що є фундаментом для подальшої розробки.

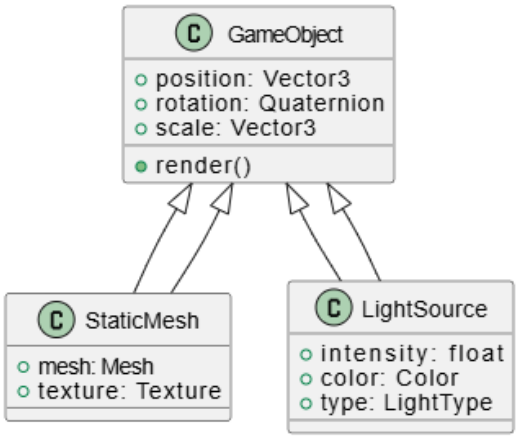


Рисунок 2.2 – UML-діаграма класів

3) Анімація персонажів та об'єктів, включає створення скелетної анімації та інтеграцію анімаційних Blueprint-систем. Діаграма станів для персонажа (рис. 2.3) допомагає візуалізувати переходи між різними станами персонажа, що є важливим для створення реалістичної поведінки.

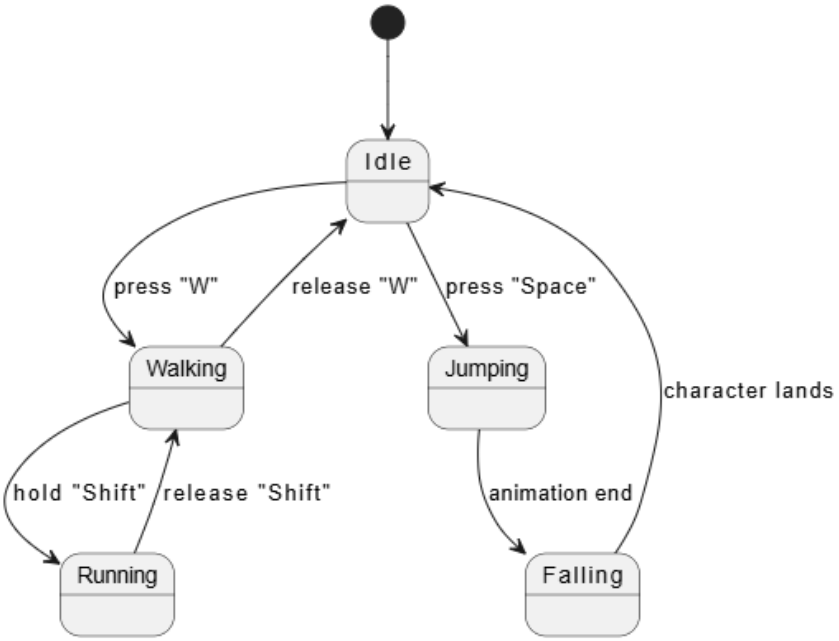


Рисунок 2.3 – UML-діаграма станів

4) Реалізація механік взаємодії гравця з об'єктами – передбачає використання Blueprint або C++ для створення логіки гри. Приклад реалізації взаємодії гравця з об'єктом у C++ демонструє, як програмісти можуть використовувати код для створення інтерактивних елементів у грі:

```
void AInteractableObject::OnInteract(APlayerCharacter* Player) {
    if (Player) {
        UE_LOG(LogTemp, Warning, TEXT("Player interacted with object"));
        // Виконати специфічну логіку (наприклад, відкрити двері)
    }
}
```

5) Програмування поведінки NPC та AI – реалізується через Behavior Tree та C++. UML-діаграма послідовності (рис. 2.4) ілюструє послідовність взаємодії NPC з гравцем, що є важливим для розуміння логіки поведінки NPC.

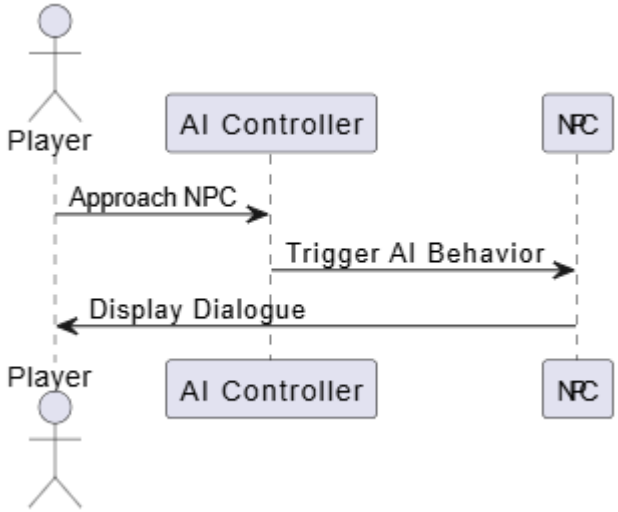


Рисунок 2.4 – UML-діаграма послідовності

6) Проєктування ігрового рівня. Передбачає розміщення об’єктів та їх налаштування. UML-діаграма компонентів (рис. 2.5) допомагає візуалізувати структуру ігрового рівня та його складові частини.

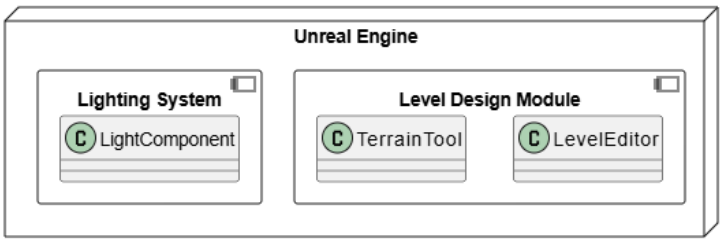


Рисунок 2.5 – UML-діаграма компонентів

7) Тестування та виправлення помилок. На фінальному етапі тестується стабільність та продуктивність гри. Приклад автоматизованого тесту на C++

демонструє, як програмісти можуть використовувати код для автоматизації процесу тестування:

```
IMPLEMENT_SIMPLE_AUTOMATION_TEST(FGameplayTest, "GameTests.BasicGameplay",
EAutomationTestFlags::EditorContext | EAutomationTestFlags::EngineFilter)

bool FGameplayTest::RunTest(const FString& Parameters) {
    // Перевірка ініціалізації гравця
    APlayerCharacter* Player = GetWorld()->SpawnActor<APlayerCharacter>();
    TestNotNull("Player exists", Player);
    return true;
}
```

Отже, моделювання віртуального ігрового середовища в UE – це складний процес, що включає концептуальне проєктування, 3D-моделювання, створення анімацій, програмування взаємодій та AI, а також тестування. Використання UML-діаграм допомагає структурувати проєкт і забезпечити ефективну взаємодію між усіма його складовими.

2.2 Принципи створення 3D-сцени та персонажів

Процес створення тривимірної сцени та персонажів у сучасних ігрових рушіях базується на фундаментальних принципах комп’ютерної графіки, художнього дизайну та алгоритмічних методів обробки даних. Він включає кілька ключових етапів: концептуальне моделювання, геометричне проєктування, текстурування, анімацію, освітлення та інтеграцію в ігрове середовище.

Перед створенням тривимірної сцени визначається її концепція та ключові елементи. На цьому етапі:

– Формується загальна ідея середовища та його функціональне призначення (наприклад, міський пейзаж, природний ландшафт, інтер’єр приміщення тощо).

– Виконуються ескізи основних об’єктів сцени, їхньої композиції та масштабності.

– Визначаються стилістичні особливості, такі як колірна гама, рівень деталізації та художня атмосфера.

– Для збереження цілісності дизайну використовується метод блокінгу сцени (blocking), який дозволяє створити низькополігональні заготівки всіх об'єктів перед їх деталізацією.

Створення 3D-об’єктів базується на полігональному моделюванні, де кожен об'єкт складається з множини полігонів (переважно трикутників або чотирикутників). Основні методи геометричного моделювання:

- Полігональне моделювання – створення об’єктів із множини вершин, ребер і граней (наприклад, для створення будівель, транспортних засобів, інтер’єрів).

- Скульптування (sculpting) – використання цифрових пензлів для додавання високополігональних деталей (застосовується переважно для персонажів та органічних форм).

- Процедурне моделювання – автоматизоване створення об'єктів на основі математичних алгоритмів (застосовується для генерації ландшафтів, рослинності, фрактальних структур).

Для персонажів використовується концепція топології сітки, що визначає рівномірність розподілу полігонів для коректної анімації. Важливо дотримуватись принципу low-poly для реального часу та high-poly для рендерингу та деталізації.

Текстурування є процесом нанесення двовимірних зображень на поверхню тривимірного об'єкта з метою додання деталізації. Основні етапи:

- UV-розгортка – перетворення тривимірної поверхні на двовимірну для коректного накладання текстури.

- PBR-матеріали (Physically-Based Rendering) – використання фізично коректних матеріалів для моделювання взаємодії світла з поверхнями (метал, дерево, скло тощо).

- Шейдери – програми для графічного процесора (GPU), які визначають, як об’єкт виглядатиме в сцені, включаючи тіні, відблиски та рефракцію.

Оптимізація текстур та матеріалів є важливою для продуктивності в реальному часі. Часто застосовується нормал-рефайлинг (normal mapping), що дозволяє імітувати дрібні деталі без додавання полігонів.

Анімація 3D-персонажів здійснюється за допомогою скелетної структури (rigging) та ключових кадрів (keyframe animation). Основні принципи:

- Скелетне моделювання (rigging) – створення кісткової структури для подальшої деформації полігональної сітки.

- Обмеження обертання (Inverse Kinematics, Forward Kinematics) – використання математичних алгоритмів для коректного руху кінцівок.

- Blend Shapes – використання морфологічних змін для анімації обличчя та міміки персонажа.

- Фізичні симуляції (Ragdoll Physics) – імітація динамічної поведінки тіла у випадку падіння чи зіткнення.

Анімація може бути створена вручну або за допомогою motion capture, що дозволяє отримати реалістичні рухи.

Освітлення відіграє ключову роль у створенні атмосфери сцени. Основні методи:

- Глобальне освітлення (Global Illumination, GI) – моделює розсіювання світла в середовищі для досягнення реалістичності.

- Динамічне та статичне освітлення – динамічні джерела світла змінюються в реальному часі, тоді як статичне освітлення запікається у вигляді текстур (lightmaps).

- Ефекти постобробки (Post-Processing Effects) – глибина різкості, розмиття руху, об'ємні тіні та світлові ореоли, що використовуються для покращення візуального стилю.

Налаштування освітлення має велике значення для оптимізації продуктивності та художнього сприйняття сцени.

Для забезпечення стабільної роботи в реальному часі необхідно враховувати принципи оптимізації:

- LOD-моделі (Level of Detail) – використання різних рівнів деталізації в залежності від відстані до об'єкта.

- Кешування текстур та матеріалів – зменшення обсягу відеопам’яті шляхом використання текстурних атласів.

- Occlusion Culling – відсіювання невидимих об'єктів для зменшення навантаження на GPU.

- GPU Instancing – повторне використання ідентичних об'єктів без дублювання їхніх даних у пам’яті.

Застосування цих методів дозволяє досягти високої продуктивності навіть у складних сценах.

Дотримання принципів оптимізації та фізичної достовірності є критично важливим для досягнення реалістичності та ефективності роботи сцени в реальному часі. Принципи створення 3D-сцени та персонажів базуються на фундаментальних засадах комп’ютерної графіки, дизайну та програмування. Розглянемо їх детальніше.

1. Принцип концептуальності. Кожна 3D-сцена та персонаж повинні відповідати певній концепції та стилістичному напряму. Усі елементи сцени повинні узгоджуватись між собою в межах художнього бачення. Важливі об’єкти повинні бути легко розпізнаваними та відповідати ігровим механікам. Варто пам’ятати, що баланс об’єктів, світла та кольорів створює правильну атмосферу.

Наприклад: у грі з постапокаліптичним сетингом використовуються тьмяні, вигорілі текстури, похмуре освітлення та хаотичне розміщення об'єктів, що підкреслює тематику виживання.

2. Принцип модульності та багаторазового використання. Сцена повинна складатися з багаторазово використовуваних модулів (наприклад, стіни, двері, меблі в приміщенні). Модульний підхід дозволяє комбінувати елементи для швидкого формування рівнів. Повторне використання моделей та текстур допомагає зберегти єдиний стиль.

Наприклад: в іграх типу MMORPG міста та села формуються за рахунок однакових наборів будівель та елементів оточення.

3. Принцип оптимізації та продуктивності. Створення 3D-сцени передбачає оптимізацію для забезпечення високої частоти кадрів (FPS). Зниження полігонального навантаження – використання low-poly моделей із нормал-мапами замість high-poly. LOD (Level of Detail) – використання спрощених моделей об'єктів на відстані для зниження навантаження на графічний процесор (GPU). Відсікання невидимих об'єктів (Occlusion Culling) – об’єкти поза камерою не рендеряться.

Наприклад: у відкритому світі гри GTA V використовуються різні рівні деталізації для будівель і машин залежно від відстані до гравця.

4. Принцип фізичної достовірності. Реалізація фізичних законів у 3D-сцені забезпечує реалістичність. Фізично коректні матеріали (PBR – Physically Based Rendering) – відтворюють поведінку світла на різних поверхнях (метал, скло, тканина). Система освітлення та тіней – враховує джерела світла, відбиття та розсіювання. Реалістична фізика – симуляція гравітації, твердих тіл, рідин, тканин тощо.

Наприклад: у Battlefield 2042 використовується фізично коректне освітлення та симуляція поведінки частинок пилу та вибухів.

5. Принцип інтерактивності. 3D-сцена має відповідати ігровим механікам та передбачати взаємодію гравця з об'єктами. Колізії та фізика взаємодії – об'єкти повинні мати правильні колізії, щоб гравець міг рухатися по них або взаємодіяти з ними. Анімації реакцій – двері повинні відкриватися, предмети – рухатися, NPC – реагувати на дії гравця. Динамічні зміни середовища – можливість руйнування, зміни погоди або часу доби.

Наприклад: у грі The Legend of Zelda: Breath of the Wild інтерактивність досягається за рахунок фізичних головоломок та можливості маніпуляцій з об’єктами.

6. Принцип масштабованості та адаптивності. 3D-сцена повинна бути гнучкою та підтримувати масштабування в залежності від ресурсів пристрою.

Гнучкі налаштування графіки – підтримка різних рівнів деталізації для слабких і потужних пристроїв. Оптимізація ресурсів – використання стрімінгу текстур та динамічного завантаження частин сцени. Підтримка різних платформ – мобільні пристрої, ПК, VR.

Наприклад: Unreal Engine дозволяє змінювати якість тіней, роздільну здатність текстур та кількість активних ефектів залежно від продуктивності пристрою.

2.3 Симуляція фізичних процесів у віртуальному середовищі

Симуляція фізичних процесів у віртуальному середовищі є ключовим аспектом створення реалістичних цифрових світів, особливо у комп'ютерних іграх, інженерних моделях, віртуальній реальності (VR) та навчальних симуляціях. Основним завданням симуляції є забезпечення реалістичної взаємодії об'єктів відповідно до фізичних законів, таких як гравітація, тертя, зіткнення, деформація та рідина.

Фізична симуляція в цифровому просторі спирається на кілька ключових аспектів. *Рух та кінематика* моделюють переміщення об'єктів у тривимірному просторі, використовуючи закони Ньютона для визначення траєкторій та швидкостей. *Сила та імпульс* описують вплив зовнішніх сил на об'єкти, включаючи гравітаційні, фрикційні та пружні взаємодії, що дозволяє відтворити реалістичні реакції на фізичні впливи. *Деформації та руйнування* враховують матеріальні властивості об'єктів, дозволяючи моделювати поведінку гнучких та ламких тіл під навантаженням. *Симуляція рідин та газів* використовує складні математичні моделі, такі як рівняння Нав'є-Стокса, для опису поведінки флюїдів, що є важливим для створення реалістичних ефектів води, диму та інших текучих середовищ.

Фізичні процеси в цифровому середовищі моделюються на основі класичної механіки та чисельних методів. Основні рівняння, що використовуються:

Рух об'єктів

Рівняння другого закону Ньютона є основою для розрахунку руху об'єктів під дією сил:

$$F = ma,$$

де F – сила, m – маса, a – прискорення.

Чисельна інтеграція, зокрема метод Ейлера, застосовується для дискретизації часу та оновлення положення та швидкості об'єктів:

$$v_{t+1} = v_t + a \cdot \Delta t,$$

$$x_{t+1} = x_t + v_t \cdot \Delta t,$$

де v – швидкість, x – положення, Δt – крок часу.

При *моделюванні колізій та взаємодії* об'єктів використовуються закони збереження імпульсу та коефіцієнт відновлення для визначення швидкостей об'єктів після зіткнення:

$$m_1 v_1 + m_2 v_2 = m_1 v_1' + m_2 v_2',$$

де v_1' та v_2' – швидкості після зіткнення.

Для обчислення відскоку враховується коефіцієнт відновлення e, що визначає ступінь пружності удару.

Існує кілька методів фізичної симуляції, кожен з яких має свої особливості та застосування. Rigid Body Dynamics (динаміка твердих тіл) моделює рух жорстких об'єктів без деформацій і широко використовується в ігрових рушіях, таких як NVIDIA PhysX, Bullet та Havok. Наприклад, в Unreal Engine, використовуючи C++, можна активувати фізичну симуляцію для об'єкта, задати його масу та додати імпульс для імітації поштовху:

```
UStaticMeshComponent* MeshComponent =
CreateDefaultSubobject<UStaticMeshComponent>(TEXT("Mesh"));
MeshComponent->SetSimulatePhysics(true);
MeshComponent->SetMassOverrideInKg(NAME_None, 50.0f);
MeshComponent->AddImpulse(FVector(1000, 0, 0));
```

Soft Body Dynamics (динаміка м'яких тіл) використовується для моделювання еластичних та гнучких об'єктів, застосовуючи метод скінченних елементів (FEM) для розрахунку деформацій. Fluid Simulation (симуляція рідин) в реальному часі використовує методи SPH (Smoothed Particle Hydrodynamics) та Lattice Boltzmann Method (LBM) для моделювання поведінки рідин. Наприклад, в Unreal Engine, система Niagara дозволяє створювати реалістичні ефекти води.

```
UNiagaraComponent* WaterSystem =
CreateDefaultSubobject<UNiagaraComponent>(TEXT("Water"));
WaterSystem->SetAsset(WaterNiagaraEffect);
WaterSystem->Activate();
```

Симуляція фізичних процесів у віртуальному середовищі дозволяє створювати реалістичні об’єкти та їх взаємодію. Використання сучасних фізичних рушіїв та алгоритмів забезпечує точність та ефективність симуляції в реальному часі. Це має широке застосування у розробці ігор, навчальних середовищ, інженерних аналізах та інших сферах, що потребують реалістичного відтворення фізичних явищ.

Припустимо, що у грі потрібно симулювати падіння каменя під дією сили тяжіння. Тоді план дій наступний:

1. Об'єкту надається фізичний компонент із заданою масою.

2. Визначається початкова швидкість нульова

3. Використовується рівняння:

$$h = h_0 - \frac{1}{2} g t^2,$$

де h_0 — початкова висота, $g=9.81$ *м/с²*.

4. Коли h=0, об'єкт або відскакує, або руйнується залежно від коефіцієнта відновлення.

Для моделювання автомобіля в Unreal Engine використовується `UWheeledVehicleMovementComponent`. Це система симуляції автомобільної фізики, яка включає – динаміку підвіски; взаємодію шин з поверхнею; аеродинамічний опір.

Отже, моделювання віртуального ігрового середовища – це багатогранний процес, що вимагає врахування багатьох аспектів.

2.4 Інтерактивність у віртуальному середовищі

Інтерактивність у віртуальному середовищі означає здатність користувача активно впливати на його елементи та отримувати відповідний відгук у реальному часі. Це ключова характеристика сучасних цифрових середовищ, що забезпечує взаємодію між користувачем і системою, а також між об'єктами середовища.

У контексті віртуального середовища інтерактивність реалізується через такі механізми: маніпуляція об'єктами (наприклад, переміщення, зміна форми, активація певних функцій); реагування середовища на дії користувача (зміна освітлення, фізичні ефекти, поведінка NPC); динамічні сценарії та геймплейні механіки (зміна сюжету залежно від рішень гравця).

Залежно від рівня залученості користувача та характеру взаємодії, інтерактивність можна поділити на такі типи:

- Рівень керування (Control-based interactivity) – передбачає прямий контроль користувачем об'єктів у середовищі. Наприклад, у грі The Legend of Zelda: Breath of the Wild гравець може маніпулювати фізичними об'єктами, використовуючи здібності персонажа.

- Сюжетна інтерактивність (Narrative interactivity) – забезпечує вплив користувача на розвиток подій. Наприклад, у Detroit: Become Human прийняті рішення змінюють сюжетні лінії, що веде до різних фіналів.

- Соціальна інтерактивність (Social interactivity) – включає взаємодію між користувачами в мультиплеєрному середовищі. Прикладом є MMORPG World of Warcraft, де гравці можуть утворювати гільдії, співпрацювати або змагатися.

- Реактивна інтерактивність (Reactive interactivity) – передбачає зміну середовища у відповідь на дії користувача без його безпосереднього контролю. Наприклад, в Red Dead Redemption 2 NPC реагують на зовнішні умови та дії гравця.

Умовно процес реалізації інтерактивності можна поділити на три основні компоненти:

1) Збір та обробка вводу користувача. Використовуються контролери, сенсорні пристрої, клавіатура, миша або VR-контролери. Наприклад, у Half-Life: Alyx система відстежує рухи рук гравця у віртуальному просторі.

2) Обчислення реакції системи. Відповідь системи моделюється за допомогою скриптів, алгоритмів штучного інтелекту або фізичних рушіїв. Наприклад, у Unreal Engine для програмування інтерактивних об’єктів можна використовувати Blueprints або C++.

3) Візуалізація результату. Оновлення стану середовища відповідно до змін, викликаних діями користувача. У VR-додатках, таких як Boneworks, фізичний рушій оновлює стан об’єктів у реальному часі відповідно до поведінки користувача.

Для демонстрації розглянемо базову реалізацію інтерактивного об'єкта в Unreal Engine за допомогою C++. Код взаємодії гравця з об’єктом:

```
void AInteractableObject::OnInteract(APlayerCharacter* Player) {
    if (Player) {
        UE_LOG(LogTemp, Warning, TEXT("Player interacted with object"));
        // Виконати специфічну логіку (наприклад, відкрити двері)
        OpenDoor();
    }
}
```

Blueprint-схема (псевдокод логіки Blueprints):

1. Перевірити, чи гравець наблизився до об'єкта
2. Активувати взаємодію при натисканні клавіші
3. Змінити стан об'єкта (наприклад, відкрити двері)

Отже, першим кроком є перевірка наближення гравця до об'єкта за допомогою сенсорів або тригерів, таких як компонент Box Collision. Наступним кроком є активація взаємодії при натисканні клавіші, що досягається використанням системи введення Unreal Engine. Останнім кроком є зміна стану об'єкта, наприклад, відкриття дверей, шляхом зміни його властивостей або відтворення анімації.

Реалізація інтерактивності у віртуальному середовищі є складним завданням, що супроводжується низкою викликів. Одним з ключових аспектів є оптимізація продуктивності, оскільки складні обчислення реакції середовища

можуть суттєво впливати на частоту кадрів (FPS) та загальну продуктивність гри. Для забезпечення природності взаємодії необхідно розробляти інтуїтивно зрозумілі інтерфейси, що дозволяють користувачам легко взаємодіяти з віртуальними об'єктами. Реалістична фізика є ще одним важливим аспектом, що вимагає використання потужних фізичних рушіїв, таких як NVIDIA PhysX, для забезпечення природної поведінки віртуальних об'єктів. У мультиплеєрних середовищах особливу увагу слід приділяти синхронізації мультиплеєрної взаємодії, щоб забезпечити коректну роботу інтерактивних механік в умовах мережевого геймплею.

Перспективи розвитку інтерактивності у віртуальних середовищах пов'язані з розвитком технологій. Очікується ширше використання штучного інтелекту для покращення поведінки NPC та адаптації сценаріїв до дій користувача. Розвиток VR та AR технологій дозволить створювати більш реалістичні та природні механіки взаємодії, що сприятиме глибшому зануренню користувачів у віртуальні світи. Глибша персоналізація ігрового досвіду стане можливою завдяки використанню поведінкових моделей користувача, що дозволить адаптувати віртуальне середовище до індивідуальних потреб та вподобань. Одним із перспективних напрямів є впровадження генеративних алгоритмів, таких як GPT-4, у NPC-діалоги, що дозволить створювати динамічні та непередбачувані сценарії взаємодії.

Отже, інтерактивність є фундаментальним аспектом віртуального середовища, що визначає ступінь занурення користувача та ефективність його взаємодії з цифровими об'єктами. Вона реалізується через механізми маніпуляції об'єктами, реакції системи та зміни середовища відповідно до дій користувача. Розвиток технологій, таких як VR, AI та фізичні рушії, дозволяє значно підвищити рівень інтерактивності та забезпечити більш реалістичну взаємодію у віртуальному просторі.

3 ОПИС ПРОЦЕСУ РОЗРОБКИ ВІРТУАЛЬНОГО ІГРОВОГО СЕРЕДОВИЩА

У третьому розділі описоно процес розробки віртуального ігрового середовища з детальним аналізом етапів проєктування, тестування та оптимізації.

3.1 Проектування ігрового середовища на базі Unreal Engine

Процес проєктування ігрового середовища можна умовно розділити на три основні етапи: проєктування концепції, реалізацію та тестування й оптимізацію. Кожен із цих етапів містить низку завдань, які забезпечують створення якісного, продуктивного та інтерактивного віртуального простору.

На першому етапі визначаються ключові параметри середовища, що формують його унікальність, функціональність і художню виразність. Концептуальне моделювання починається зі створення концепт-документа, який містить: опис стилістики середовища (реалістичний, стилізований, футуристичний тощо); визначення механік взаємодії гравця зі світом; специфікацію ключових об'єктів (NPC, будівлі, ландшафт).

Створення референсів та прототипування. на цьому етапі використовуються: 2D-ескізи та концепт-арти, що задають загальний стиль. Прототипи рівнів у Unreal Engine, які дозволяють оцінити масштабність середовища.

Реалізація та програмування середовища – цей етап охоплює створення 3D-моделей, текстурування, налаштування фізики, освітлення та інтеграцію звукових ефектів.

Unreal Engine дозволяє імпортувати 3D-моделі з Maya, Blender або 3ds Max. Використовуються моделі з високою деталізацією для великих планів. Для

фонових елементів застосовуються Low-poly моделі для оптимізації продуктивності.

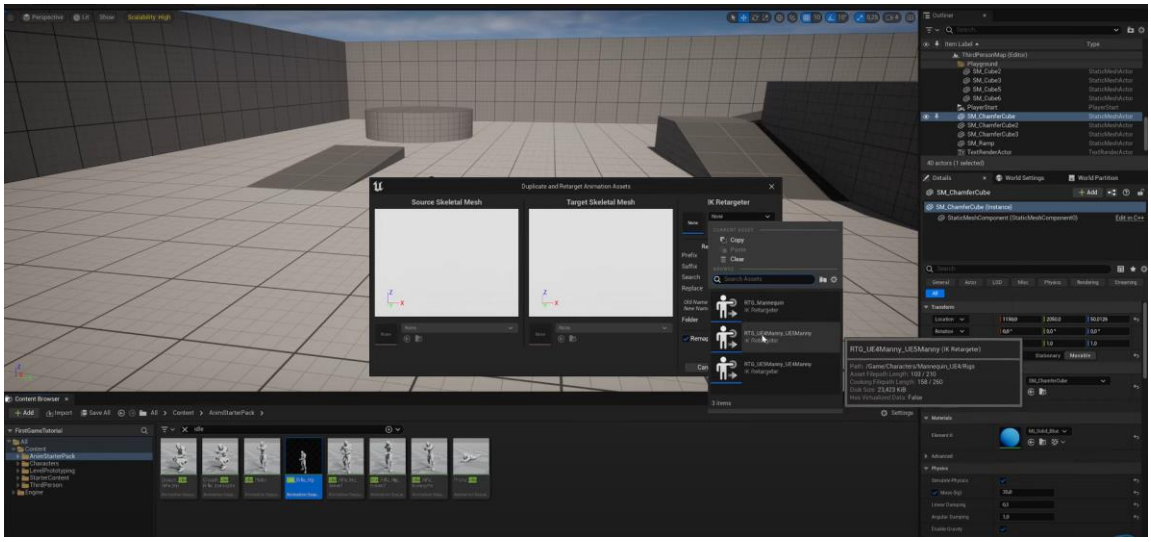


Рисунок 3.1 – Налаштування ретаргетингу анімацій у редакторі Skeletal Mesh

Для створення реалістичних текстур застосовуються PBR (Physically Based Rendering) матеріали. Unreal Engine має потужний Material Editor, що дозволяє створювати процедурні матеріали. Наприклад: для імітації води в UE використовується система шейдерів, що реагує на освітлення та рухи гравця.

Використання статичного та динамічного освітлення допомагає у підвищенні продуктивності. Ray Tracing для реалістичної взаємодії світла з об'єктами. Наприклад: у грі Cyberpunk 2077 використовується Ray Tracing для створення кінематографічного освітлення.

Як згадували в попередніх розділах, Unreal Engine має вбудований Physics Engine для моделювання гравітації та колізій. Наприклад, реалізація руйнування об'єктів у Battlefield V виконується через систему Destruction Physics.

Приклад реалізації фізичної взаємодії (C++ у Unreal Engine):

```
void APhysicsObject::OnHit(AActor* OtherActor) {
    if (OtherActor->ActorHasTag("Player")) {
        UGameplayStatics::ApplyDamage(OtherActor, 10.0f, nullptr, this,
UDamageType::StaticClass());
    }
}
```

У цьому прикладі об'єкт завдає шкоди гравцю при зіткненні.

Після створення середовища важливо провести тестування та оптимізувати продуктивність. Перевіряємо колізії об'єктів та коректність взаємодії гравця. Тестуємо інтерактивність середовища (взаємодія з NPC, об'єктами).

Використання Level Streaming для підвантаження частин карти. Оптимізація текстур через Mipmapping. Використання Nanite (технологія в Unreal Engine 5) для динамічного рендерингу високополігональних моделей.

3.2 Реалізація ігрового рівня

Розробка ігрового рівня в "Iter Astris" є комплексним процесом, що поєднує наукову точність астрономічних моделей з інтерактивними ігровими механіками, забезпечуючи високий рівень занурення гравця у дослідницьку діяльність. Основна мета – створити реалістичне, масштабне та динамічне середовище, що дозволить гравцеві вивчати космічні явища через взаємодію з ігровим світом.

Рівень у "Iter Astris" складається з кількох ключових зон, кожна з яких представляє конкретний аспект астрономічного дослідження. До них належать:

Орбітальна станція "Astra Nova" – центральний хаб, де гравець отримує завдання, аналізує зібрані дані та взаємодіє з іншими членами Астронаукового Альянсу.

Гравітаційний сектор – зона, присвячена дослідженню гравітаційних хвиль, що використовує симуляцію просторово-часових деформацій.

Чорна діра "XG-7" – місце експедиційного аналізу фізичних явищ, пов’язаних із горизонтом подій та акреційним диском.

Система екзопланет "Aurora-13" – рівень, на якому гравець застосовує методи транзитної спектроскопії для відкриття нових планет.

Проєктування цих зон базується на принципах наукової достовірності, доповненої геймплейними механіками, що стимулюють активне дослідження.

Ця концепція знаходить відображення у структурній організації ігрового світу, що представлена у відповідній діаграмі класів (рис. 3.2). Представлена діаграма класів описує структуру віртуального ігрового середовища космічної тематики. Діаграма складається з чотирьох класів: AGameLevel, APlanet, ABlackHole та ASpaceStation.

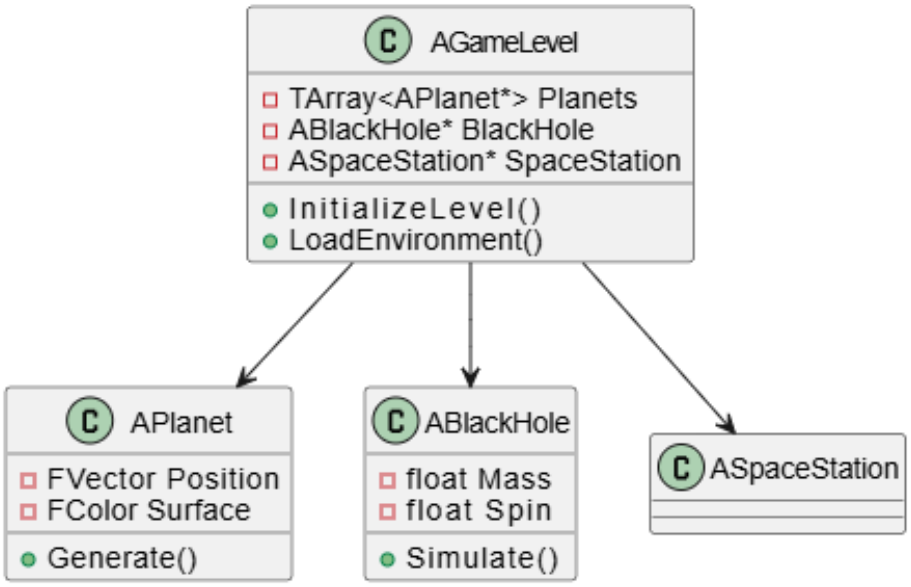


Рисунок 3.2 – UML-діаграма класів основних об’єктів рівня

Клас AGameLevel є центральним елементом діаграми, що представляє рівень гри. Він містить три атрибути: Planets, BlackHole та SpaceStation, які є колекцією планет, чорною дірою та космічною станцією відповідно. Клас також має два методи: InitializeLevel та LoadEnvironment, які, використовуються для ініціалізації рівня та завантаження оточення.

Клас APlanet представляє планету та містить два атрибути: Position та Surface, які описують положення планети та її поверхню. Клас також має метод Generate, який, використовується для генерації планети.

Клас ABlackHole представляє чорну діру та містить два атрибути: Mass та Spin, які описують масу та обертання чорної діри. Клас також має метод Simulate, який, використовується для симуляції поведінки чорної діри. Клас ASpaceStation представляє космічну станцію.

Клас AGameLevel містить об'єкти класів APlanet, ABlackHole та ASpaceStation, що відображає композиційну структуру рівня гри.

Реалізація ігрового рівня у Unreal Engine починається зі створення базової геометрії за допомогою інструментів Landscape та Static Meshes. Наприклад, для моделювання чорної діри використовується матеріал, що симулює гравітаційне викривлення світла, а для орбітальної станції застосовуються модулі із параметричними деталями:

```
// Створення матеріалу гравітаційного викривлення
UMaterialInstanceDynamic* BlackHoleMaterial =
UMaterialInstanceDynamic::Create(BaseMaterial, this);
BlackHoleMaterial->SetScalarParameterValue(TEXT("GravityWarp"), 1.5f);
```

У процесі роботи використовується Procedural Mesh Component, що дозволяє генерувати планети із заданими параметрами:

```
// Генерація планети з динамічним ландшафтом
APlanet* NewPlanet = GetWorld()->SpawnActor<APlanet>();
NewPlanet->GenerateSurfaceFeatures();
```

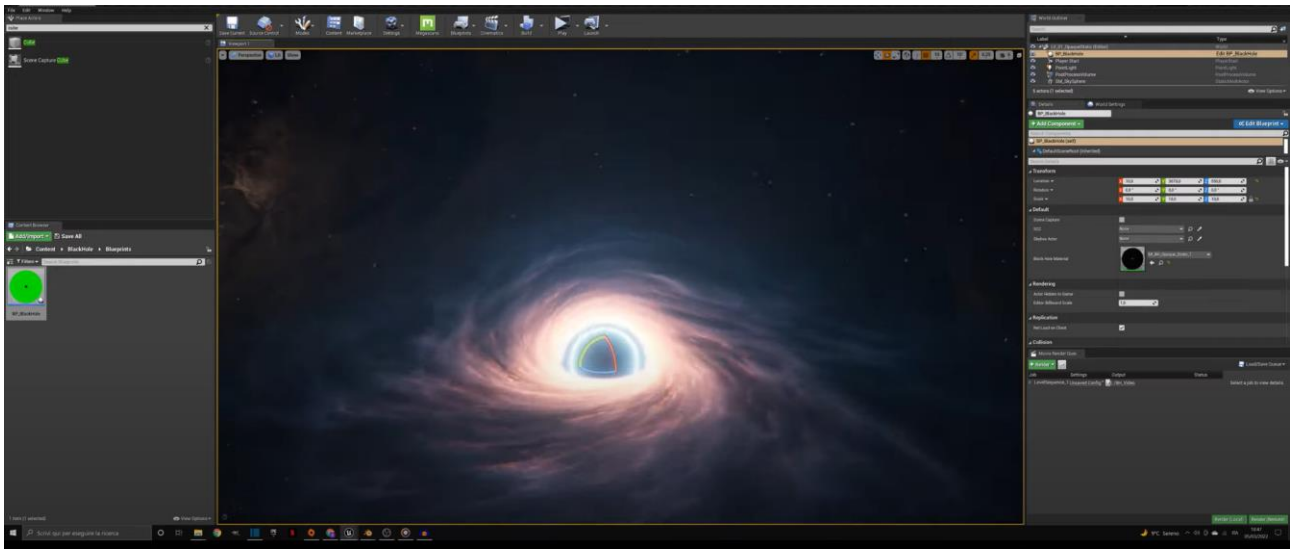


Рисунок 3.3 – Процес моделювання чорної діри

Для створення атмосфери та реалістичності використовується технологія PBR (Physically-Based Rendering), яка враховує фізичні властивості матеріалів. Орбітальна станція оснащена джерелами динамічного освітлення для симуляції денного та нічного циклів. Чорна діра має спеціальний шейдер для рендерингу акреційного диска із ефектом гравітаційного лінзування.

Використання Blueprints та C++ дозволяє створити динамічні об’єкти. Наприклад, зміна орбіти екзопланети відбувається за рахунок обчислення гравітаційної взаємодії:

```
void AExoplanet::UpdateOrbit(float DeltaTime) {
    FVector GravityForce = ComputeGravity();
    Velocity += GravityForce * DeltaTime;
    SetActorLocation(GetActorLocation() + Velocity *
DeltaTime);
}
```



Рисунок 3.4 – Процес зміни орбіти екзопланети

Одним із ключових аспектів рівня є інтерактивні механізми, що дозволяють гравцеві взаємодіяти з досліджуваними об’єктами. Наприклад, для сканування простору гравець використовує спектральний аналізатор, який імітує методи спектроскопії:

```
void ASpectralAnalyzer::AnalyzeLightSpectrum(FColor LightData) {
    float Wavelength = ComputeWavelength(LightData);
    UE_LOG(LogTemp, Warning, TEXT("Detected wavelength: %f nm"), Wavelength);
}
```

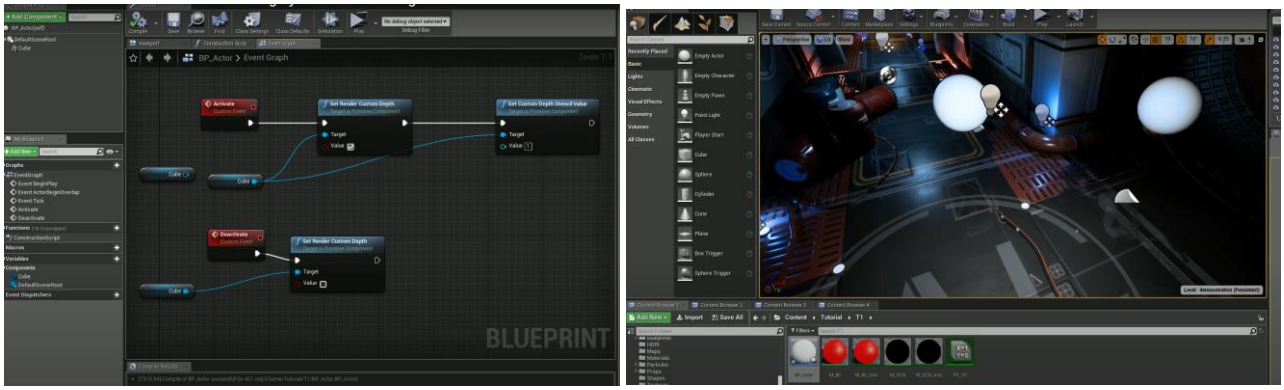


Рисунок 3.5 – Процес налаштування сканування простору

Гравітаційні хвилі в грі представлені як спотворення простору-часу, які візуалізуються через динамічні матеріали:

```
void AGravityWaveDetector::DetectWave() {
    float Distortion = ComputeDistortion();
    MaterialInstance->SetScalarParameterValue(TEXT("WaveIntensity"),
Distortion);
}
```

Використання Unreal Engine, зокрема Blueprints та C++, дозволяє створити реалістичну симуляцію космічних явищ, які допомагають гравцеві отримати інтерактивний освітній досвід.

3.3 Тестування та виправлення помилок

Тестування та виправлення помилок є завершальним і критично важливим етапом розробки віртуального ігрового середовища, який безпосередньо впливає на якість продукту, стабільність його роботи та загальне враження користувача. Враховуючи складність сучасних ігрових рушіїв, таких як Unreal Engine (UE), процес тестування охоплює широкий спектр перевірок: від коректної роботи механік і фізичних взаємодій до оптимізації продуктивності та усунення логічних помилок у коді.

Юніт-тестування передбачало перевірку окремих функціональних блоків коду, таких як механіки руху персонажа, взаємодія об'єктів або алгоритми штучного інтелекту. В Unreal Engine це реалізували за допомогою Automated Tests у C++. Приклад коду для юніт-тесту у UE5:

```
1. #include "CoreMinimal.h"
2. #include "Misc/AutomationTest.h"
3. IMPLEMENT_SIMPLE_AUTOMATION_TEST(FJumpTest, "Game.Character.Jump",
4. EAutomationTestFlags::EditorContext | EAutomationTestFlags::EngineFilter)
5. bool FJumpTest::RunTest(const FString& Parameters)
6. {
7. ACharacter* TestCharacter = NewObject<ACharacter>();
8. TestCharacter->Jump();
9. TestTrue("Character should be jumping", TestCharacter-
   >GetCharacterMovement()->IsFalling());
10.  return true;}
```

Цей тест допоміг перевірити, чи змінюється стан персонажа на "falling" після виклику функції Jump(), що підтверджує коректність реалізації механіки стрибка.

Наступним було функціональне тестування – яке оцінює, чи відповідає поведінка гри очікуваному дизайну. Воно включало перевірку механік руху персонажа (біг, стрибки, взаємодія з перешкодами); взаємодію з об'єктами (відкривання дверей, активація тригерів); логіку поведінки NPC та AI.

У UE можна створити тестовий Blueprint, який перевірятиме коректність роботи тригерів. Наприклад, об’єкт змінює колір при взаємодії:

Подія `OnBeginOverlap` → Перевірка, чи персонаж торкнувся об'єкта.

Зміна матеріалу → Візуальний індикатор взаємодії.

Логування результату → Відображення повідомлення у `Output Log`

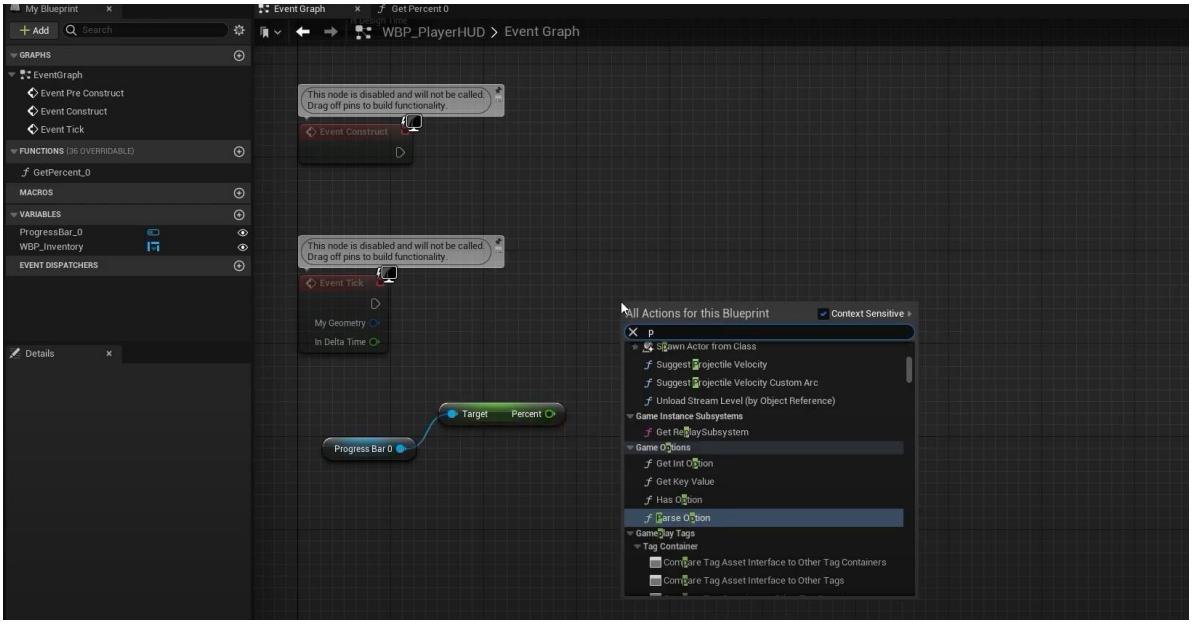


Рисунок 3.6 – Blueprint-схема взаємодії з об'єктом у редакторі UE

Регресійне тестування необхідне після внесення змін до коду для перевірки, чи нові зміни не призвели до виникнення нових помилок. Наприклад, після оптимізації фізичних ефектів слід переконатися, що вони не вплинули на поведінку NPC або рух гравця.

Якщо після зміни швидкості руху персонажа в коді з'явилася проблема з анімаціями, слід перевірити, чи правильно синхронізуються стан анімації та параметри руху.

```
void AMyCharacter::MoveForward(float Value)
{
    if (Controller && Value != 0.0f)
    {
        FVector Direction = FRotationMatrix(Controller->GetControlRotation()).GetUnitAxis(EAxis::X);
        AddMovementInput(Direction, Value);
        UE_LOG(LogTemp, Warning, TEXT("Moving forward with value: %f"), Value);
    }
}
```

Для аналізу значень параметра Value, що дозволить виявити аномалії у русі було використано логування.

Наступним видом, використаного тестування було – стрес-тестування, воно оцінює стабільність гри в екстремальних умовах, наприклад: взаємодія великої кількості NPC у сцені; високий рівень деталізації середовища; робота з максимальною кількістю одночасних клієнтів у мультиплеєрі. Запуск сцени зі 100 NPC, які одночасно виконують анімації та взаємодіють із середовищем, дозволяє оцінити, чи гра зберігає продуктивність без значних просідань FPS.

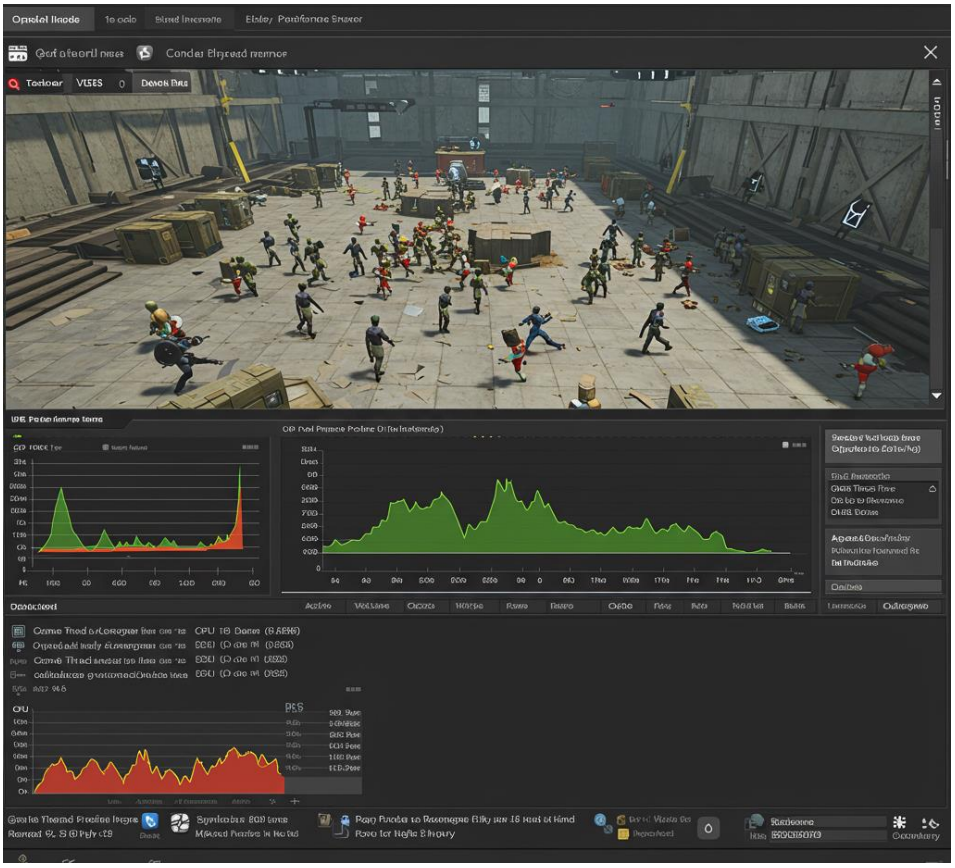


Рисунок 3.7 – Тестування продуктивності Unreal Engine Performance Profiler у вікні Insights

Після завершення етапу тестування ключовим завданням стає виправлення виявлених помилок та оптимізація продуктивності гри для забезпечення її стабільної та ефективної роботи.

Усунення багів (Debugging) передбачало систематичний аналіз та виправлення помилок у коді. Для цього використовували такі методи:

Аналіз логів, він полягав у використанні макросу UE_LOG() для виявлення аномалій та помилок, що фіксуються в журналі подій. Це дозволило відстежувати послідовність виконання коду та виявляти місця, де виникають непередбачувані ситуації.

Використання Breakpoints передбачало покрокове виконання коду за допомогою точок зупинки, що дозволяло детально аналізувати стан програми на кожному кроці та виявляти логічні помилки.

Інструменти профілювання, зокрема Unreal Insights, використовувалися для аналізу продуктивності гри, виявлення "вузьких місць" та оптимізації використання ресурсів.

Зменшення кількості полігонів у 3D-моделях дозволило знизити навантаження на графічний процесор (GPU) та підвищити продуктивність рендерингу.

LOD (Level of Detail) передбачало використання спрощених моделей об'єктів на віддалених відстанях, що дозволило знизити навантаження на GPU без суттєвої втрати якості зображення.

Оптимізація освітлення полягала у використанні static lighting замість dynamic lighting, що дозволило зменшити кількість обчислень, необхідних для рендерингу освітлення.

Зменшення кількості викликів рендерингу (Draw Calls) дозволило знизити навантаження на CPU та GPU, що сприяло підвищенню продуктивності.

Застосування цих методів дозволило забезпечити стабільну та продуктивну роботу гри, що є важливим для забезпечення комфортного ігрового досвіду.

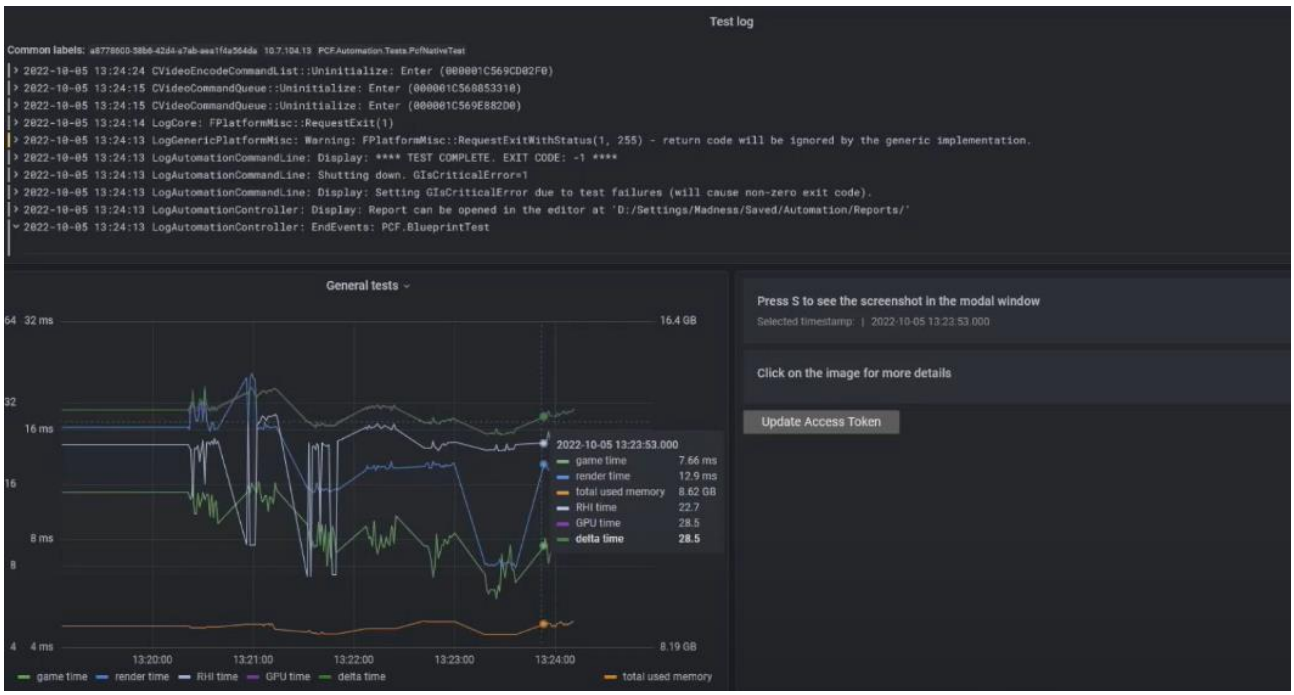


Рисунок 3.8 – Інтерфейс інструменту для моніторингу та аналізу продуктивності в Unreal Engine

Таким чином, тестування та виправлення помилок у віртуальному ігровому середовищі є комплексним процесом, що включає перевірку логіки механік, стабільності рушія, оптимізації продуктивності та аналіз взаємодії елементів гри. Використання автоматизованих тестів, регресійного аналізу та стрес-тестування дозволяє запобігти появі критичних помилок та забезпечити стабільну роботу продукту.

ВИСНОВКИ

1. У дослідженні було проведено глибокий аналіз найпопулярніших ігрових рушіїв: Unreal Engine, Unity, Godot Engine, CryEngine та Amazon Lumberyard. Для кожного рушія було розглянуто: графічні можливості (підтримка трасування променів, глобального освітлення, об’ємного рендерингу); фізичні рушії (використання NVIDIA PhysX, Chaos Physics в Unreal Engine, Havok у Unity); оптимізаційні інструменти (підтримка рівнів деталізації (LOD), асинхронне завантаження ресурсів, кешування текстур); кросплатформеність (можливість розгортання на ПК, мобільних пристроях, VR/AR, хмарних платформах); гнучкість і розширюваність (наявність маркетплейсів для розширення функціоналу (Unreal Marketplace, Unity Asset Store)); програмування та скриптові мови (порівняння C++ (Unreal Engine) та C# (Unity), можливості Blueprint у UE).

У результаті було визначено, що Unreal Engine забезпечує найкращі можливості для створення реалістичних віртуальних середовищ завдяки передовим графічним технологіям та розвиненій системі візуального програмування.

2. Уточнено, визначення базового поняття «*віртуальне ігрове середовище*» як програмно-апаратний комплекс, що забезпечує користувачеві можливість занурення в інтерактивний тривимірний простір, де він може взаємодіяти з об'єктами та іншими користувачами за допомогою різних пристроїв введення-виведення. Виділено його структурні компоненти.

Змодельовано усі необхідні бізнес-процеси RPG на базі рушія Unreal Engine, відповідно до стандартів ігрової індустрії, а саме: систему персонажів (Character System) – реалізовано механізми кастомізації, параметри стану (здоров’я, витривалість, рівень персонажа); штучний інтелект (AI) – використано Behavior Tree для NPC, налаштовано логіку поведінки ворогів та союзників; систему діалогів – реалізовано діалогову систему на основі Blueprint з підтримкою нелінійного розвитку подій; розроблено механізм обміну ресурсів, інвентар, систему крафтингу; систему квестів – використано патерни стану (State

Machine) для динамічного управління завданнями; бойову механіку – реалізовано систему комбінацій ударів, блокування, відступів з анімаційними трансформаціями через Animation Blueprint.

3. Процес створення ігрового середовища охоплював такі етапи: прототипування (створено тестовий рівень із базовими механіками (пересування, взаємодія з об’єктами)); створення 3D-моделей (використано Blender для моделювання низькополігональних і високополігональних об’єктів); текстурування (застосовано PBR-матеріали для реалістичного освітлення); освітлення (використано систему Lumen для динамічного освітлення сцени); анімація (налаштовано систему Rigging та застосовано Motion Capture для реалістичних рухів персонажів); інтерактивність (реалізовано скрипти взаємодії (наприклад, відчинення дверей, збір предметів, перемикачі механізмів)).

4. Для забезпечення стабільної роботи було проведено оптимізаційні заходи – профілювання продуктивності (використано Unreal Insights для аналізу FPS та завантаження процесора/GPU); оптимізація графіки (використано LOD-моделі, імплементацію Nanite для деталізації); зменшення витрат ресурсів (застосовано методи Occlusion Culling та Mipmapping для текстур); юніт-тести та автоматизовані перевірки (реалізовано тести AI, перевірку взаємодії гравця з об’єктами). У результаті середовище працює стабільно з частотою 60 FPS на цільових платформах.

5. Виявлено проблеми продуктивності та якості графіки: високодеталізовані моделі значно знижують FPS; складні симуляції призводять до перевантаження процесора; трасування променів суттєво впливає на продуктивність. Для балансу якості графіки та продуктивності довелось використовувати компромісні рішення, такі як Hybrid Rendering.

Тестування мультиплеєру виявило проблеми із затримками (ping > 80 мс у певних умовах). Для вирішення цих проблем було застосовано ряд оптимізаційних стратегій, зокрема, впровадження Level Streaming для покращення завантаження рівнів, а також AI LOD для оптимізації роботи NPC.

Проведене дослідження підтверджує, що всі поставлені завдання були виконані якісно та в повному обсязі. Розроблене віртуальне середовище базується на передових технологіях та демонструє високу якість графіки, продуктивність і гнучкість у налаштуванні.

Отримані результати можуть бути використані для подальших досліджень у сфері геймдеву, особливо щодо оптимізації великих відкритих світів, мультиплеєрних механік та інтеграції нейромереж у розробку ігор.

ПЕРЕЛІК ПОСИЛАНЬ

1. Accommodating for Learning Disabilities at Events. 2020. URL: https://medium.com/@teoohevents/accommodating-for-learning-disabilities-at-events-f627c156ff3d

2. Cry Engine. URL: https://www.cryengine.com/

3. Designing and Developing Game-Like Learning Experience in Virtual Worlds: Challenges and Design Decisions of Novice Instructional Designers // International Journal of Virtual and Augmented Reality (IJVAR). 2023. № 7. С. 1-19.

4. F. Grudzewski, M. Awdziej, G. Mazurek, K. Piotrowska. Virtual Reality in Marketing Communication – the Impact on the Message, Technology and Offer Perception – Empirical Study Economics and Business Review, 4 (2018), pp. 36-50, 10.18559/ebr.2018.3.4

5. Guillemette, R. Unreal Engine 5 Character Creation, Animation, and Cinematics. Mercury Learning and Information, 2023

6. Hepp A., Krotz F. Mediatized Worlds – Understanding Everyday Mediatization // Mediatized Worlds. Culture and Society in a Media Age. Basingstoke : Palgrave Macmillan, 2014. P. 1–14.

7. Murru, G., Fratarcsngeli, M., Empler, T., Augmented Visualization on Handheld Devices for Cultural Heritage, Practical Augmented Visualization on Handheld Devices for Cultural heritage, in: V. Skala (Eds.), WSCG 2013 - Communication Papers Proceedings, University of West Bohemia, Plzen, Czech Republic, 2013, pp. 97-103.

8. Problems of the Intelligent Virtual Learning Environment Development // Proceedings of the 2018 International Conference on Intelligent Systems (IS). 2018. С. 27-31.

9. Sobota B., Pietriková E. The Role of Game Engines in Game Development and Teaching. Computer Science for Game Development and Game Development for Computer Science. IntechOpen, Aug. 18, 2023. URL: http://dx.doi.org/10.5772/intechopen.1002257

10. Unity engine. Unity. URL: https://unity.com

11. Unreal Engine Documentation. URL: https://docs.unrealengine.com/

12. Virtual Reality in Marketing Communication – the Impact on the Message, Technology and Offer Perception – Empirical Study Economics and Business Review. 2023. № 23 (3). С. 109-128.

13. Волинець В. О. Віртуальна реальність у соціокультурному просторі сучасності // Культура України. Серія: Культурологія. 2016. Вип. 52. С. 120–128.

14. Габрусєв В. Ю., Вельгач А. В., Кулянда О. О. Дослідження функціональних особливостей рушія UNITY 3D на прикладі реалізації 3D міні-гри. Науковий часопис Національного педагогічного університету імені М. П. Драгоманова. Серія 2 : Комп'ютерно-орієнтовані системи навчання : зб. наук. праць. 2022. Вип. 22 (29). С. 152-159. DOI: 10.31392/NPU-nc.series 2.2020.22(29).21.

15. Дзьобань О. Сучасний віртуальний простір: конгеніальність віртуальності й міфу // Стратегічні пріоритети. 2017. № 3 (44). С. 168.

16. Любивий Я. Віртуальне суспільство доби постмодерну. Соціокультурні та теоретичні засади філософії постмодерну. Київ : Інститут філософії імені Григорія Сковороди НАН України, 2017. С. 192–193

17. Офіційний веб-сайт Unreal Engine. URL: https://www.unrealengine.com/

18. Пінчук О.П., Лупаренко Л. А. *Дидактичний потенціал використання цифрового контенту з доповненою реальністю* Сучасні інформаційні технології та інноваційні методики навчання в підготовці фахівців: методологія, теорія, досвід, проблеми (63). 2022. стор. 39-57. ISSN 978-966-2337-01-3

19. Степорук Н. Unity проти Unreal Engine – який рушій обрати для гри та чому. Зважуємо всі за та проти з розробниками. URL: https://gamedev.dou.ua/articles/unity-orunreal-engine/

20. Чепелюк, А. В., Грушко В. В., Федоренко А. М., Роль віртуальної реальності в модернізації фізичної культури: можливості й виклики для навчання

та тренувань. *Педагогічна Академія: наукові записки*, no. 10 (Вересень 13, 2024). URL: https://pedagogical-academy.com/index.php/journal/article/view/334.

21. Яценко О., Нечипорук В. Огляд переваг та недоліків ігрових рушіїв // Перспективи інноваційного розвитку науки і технологій. 2023. С. 44-53.

22. Дячук С. Ф., Коноваленко І. В., Шкодзінський О. К. Віртуальне освітнє середовище ТНТУ на базі LMS ATutor // Міжнародний наук.-практичний семінар «Теорія і практика дистанційного навчання іноземних громадян: вітчизняний та міжнародний досвід». – ХНУРЕ, 12 листопада 2014. – С.11–15.

23. Wang H. Application of Virtual Reality Technology in Physics Experiment Teaching // Proceedings of the MECAE 2018.- Atlantis Press, 2018.- pp. 266-271. doi: 10.2991/mecae18.2018.55

24. Ковальчук, М. О., and Н. Є. Колесник. "Графічний дизайн та комп'ютерна графіка: монографія." *Житомир, ТОВ* 505. 2020. 440 с.

25. Ковальчук, М. О. Особливості створення навчального мультимедійного курсу. *Молодий вчений*, 2023, 9 (1): 156-159.

26. Маєвський, О. М.; Ковальчук, М. О. Сучасні технології підготовки мультимедійних творчих продуктів. У Prospective And Priority Directions Of Scientific Research In Technical And Agricultural Sciences (с. 191-214). Boston: International Science Group. *Prospective And Priority Directions Of Scientific Research In Technical And Agricultural Sciences*, 2023, 191-214.

27. 3D моделювання: навчальний посібник / М. Ковальчук, В. Українець, В. Терещук. Житомир : Пол. нац. ун-ет., 2024. 64 с.

28. Карабін О. Й., Бабій Н. Б. Розробка 3D ігор як засіб формування алгоритмічного мислення. «Сучасні інформаційні технології та інноваційні методики навчання: досвід, тенденції, перспективи»: матеріали VI міжнарод. наук.-практ. інтернет-конф., м. Тернопіль, 12.11.2020 р.–13.11.2020 р. Тернопіль, 2020. № 6. С. 187–189

29. Anagh R.S. «Unreal Engine: Powering the Unreal Realities of Digital Creativity». URL: https://www.linkedin.com/pulse/article-unreal-engine-anagh-r-s-zrlpc

30. .Learn how to use new GPU lightmaping in Unreal Engine [Електронний ресурс]. – 2020. – URL: https://evermotion.org/tutorials/show/12189/learn-how-to-use-new-gpulightmaping-in-unreal-engine.

31. GPU Lightmass Global Illumination [Електронний ресурс]. URL: https://docs.unrealengine.com/4.26/enUS/RenderingAndGraphics/GPULightmass/.

32. Plugins [Електронний ресурс]. URL: https://docs.unrealengine.com/5.0/en-US/plugins-in-unreal-engine/

33. Prickett T. M. Diving deep into the NVIDIA Ampere GPU architecture [Електронний ресурс] / Timothy Morgan Prickett. – 2020. URL: https://www.nextplatform.com/2020/05/28/diving-deep-intothe-nvidia-ampere-gpu-architecture/.

ДОДАТКИ

Додаток А. Характеристика ігрових рушіїв

Характеристики	Unreal Engine	Unity	Godot Engine	CryEngine	Amazon Lumberyard
Ліцензування та вартість	Повний безкоштовний доступ до всіх можливостей, при доході проекту понад $ 1 млн. 5% від доходу	Безкоштовна (функції частково обмежені), при доході проекту понад $ 100 тис. вартість ліцензії залежить від прибутку	Повний безкоштовний доступ до всіх можливостей, при доході проекту понад $ 5 тис. 5% від прибутку	Повний безкоштовний доступ до всіх можливостей, при доході проекту понад $ 5 тис. 5% від прибутку	Безкоштовний
Якість документації та підтримка	Повнота, але складна навігація, офіційна підтримка, велика спільнота, багато навчальних матеріалів	Повна, актуальна, зручна навігація, офіційна підтримка, дуже велика спільнота, безліч навчальних матеріалів	Повна, актуальна, зручна навігація, офіційна підтримка, велика спільнота, багато навчальних матеріалів	Неповна, застаріла, складна навігація, обмежена офіційна підтримка, невелика спільнота	Досить повна, але потребує оновлення, офіційна підтримка, велика спільнота, багато навчальних матеріалів
Спільнота та підтримка	Активна, велика спільнота, форуми, вікі, навчальні матеріали, офіційна підтримка	Активна, дуже велика спільнота, форуми, вікі, навчальні матеріали, офіційна підтримка	Активна, велика спільнота, форуми, вікі, навчальні матеріали, офіційна підтримка	Активна, але невелика спільнота, обмежені форуми, вікі, навчальні матеріали, обмежена офіційна підтримка	Активна, велика спільнота, форуми, вікі, навчальні матеріали, офіційна підтримка
Поріг входження та складність	Складний інтерфейс, не вимагає знання мов програмування, але бажано знати C++, складне освоєння	Простий інтерфейс, не вимагає знання мов програмування, але бажано знати C#, легке освоєння	Простий інтерфейс, не вимагає знання мов програмування, але бажано знати C#, легке освоєння	Складний інтерфейс, вимагає знання C++, складне освоєння	Простий інтерфейс, не вимагає знання мов програмування, але бажано знати C++, легке освоєння
Можливості візуального програмування	Потужна система Blueprint	Візуальний редактор шейдерів	Візуальний редактор шейдерів	Система Flow Graph	Візуальний редактор Lua
Інструменти для анімації	Потужні вбудовані інструменти, інтеграція з Maya та Blender	Вбудовані інструменти, підтримка Mixamo	Вбудовані інструменти, підтримка Spine	Вбудовані інструменти, підтримка FBX	Вбудовані інструменти, підтримка FBX
Фізичний рушій	PhysX, Chaos Physics	PhysX	Вбудований 3D-фізичний рушій,	Вбудований 3D-фізичний рушій	PhysX

Характеристики	Unreal Engine	Unity	Godot Engine	CryEngine	Amazon Lumberyard
			підтримка Bullet		
Інструменти для роботи зі звуком	Потужна система звуку, підтримка Wwise	Вбудовані інструменти, підтримка FMOD	Вбудовані інструменти, підтримка FMOD	Вбудовані інструменти, підтримка FMOD	Вбудовані інструменти, підтримка Wwise
Можливості для мережевої гри	Потужна підтримка, клієнт-серверна архітектура	UNet, Mirror, Photon	Вбудована підтримка, ENet	Вбудована підтримка, CryNetwork	GridMate, GameLift
Інструменти для створення інтерфейсу користувача (UI)	UMG UI Designer	Unity UI, NGUI, TextMesh Pro	Вбудовані інструменти, підтримка Control nodes	Вбудовані інструменти, CryUI	Вбудовані інструменти, Lumberyard UI
Можливості для штучного інтелекту (AI)	Потужна система AI, Behavior Trees	NavMesh, ML-Agents	Вбудовані інструменти, NavMesh	Вбудовані інструменти, AI System	Вбудовані інструменти, Script Canvas
Екосистема плагінів та асетів	Unreal Engine Marketplace, велика кількість плагінів	Unity Asset Store, величезна кількість асетів	Asset Library, велика кількість асетів	CryEngine Marketplace, обмежена кількість плагінів	AWS Marketplace, велика кількість плагінів
Стабільність та надійність	Висока стабільність, регулярні оновлення	Висока стабільність, регулярні оновлення	Висока стабільність, регулярні оновлення	Середня стабільність, нерегулярні оновлення	Висока стабільність, регулярні оновлення
Приклади успішних проєктів	Fortnite, Gears of War, Batman: Arkham City	Temple Run, Among Us, Genshin Impact	Ori and the Blind Forest, Hollow Knight, Brotato	Crysis, Far Cry, Hunt: Showdown	Star Citizen, Crucible, New World
Можливості для роботи з відкритим світом	Потужні інструменти, підтримка великих локацій	Terrain Engine, World Streamer	Вбудовані інструменти, підтримка великих локацій	Вбудовані інструменти, SVOTI	Terrain Editor, підтримка великих локацій
Підтримка мобільних платформ	Висока оптимізація, інструменти для тестування	Висока оптимізація, інструменти для тестування	Висока оптимізація, інструменти для тестування	Середня оптимізація, обмежені інструменти	Висока оптимізація, інструменти для тестування
Інструменти для VR/AR	Потужна підтримка, шаблони проєктів	Потужна підтримка, AR Foundation, XR Interaction Toolkit	Потужна підтримка, OpenXR	Потужна підтримка, CryVR	Потужна підтримка, VR Gems
Можливості для рендерингу в реальному часі	Трасування променів, Lumen, Nanite	Universal Render Pipeline, High Definition Render Pipeline	Vulkan, OpenGL ES 3.0	SVOGI, Temporal Anti-Aliasing	Physically Based Rendering, Global Illumination
Інструменти для роботи з даними	Імпорт/експорт даних,	Імпорт/експорт даних, Unity Analytics	Імпорт/експорт даних, JSON, XML	Імпорт/експорт даних, XML	Інтеграція з AWS,

Характеристики	Unreal Engine	Unity	Godot Engine	CryEngine	Amazon Lumberyard
	інтеграція з базами даних				Lumberyard Data
Можливості для навчання та освіти	Unreal Engine Online Learning, сертифікація	Unity Learn, Unity Certification	Godot Documentation, Godot Tutorials	CryEngine Documentation, CryDev.net	Lumberyard Documentation, Lumberyard Tutorials
Гнучкість та розширюваність	Висока гнучкість, відкритий код (частково)	Висока гнучкість, C# API	Висока гнучкість, відкритий код	Середня гнучкість, обмежений API	Висока гнучкість, відкритий код (частково)
Кросплатформеність редактора	Windows, macOS, Linux	Windows, macOS, Linux	Windows, macOS, Linux, BSD, Haiku	Windows	Windows
Інструменти для спільної роботи	Perforce, Git, UnrealGameSync	Unity Collaborate, Plastic SCM, Git	Git, Subversion	Perforce, Git	Perforce, Git
Можливості для прототипування	Швидке прототипування за допомогою Blueprint	Швидке прототипування за допомогою Unity Editor	Швидке прототипування за допомогою Godot Editor	Прототипування за допомогою Flow Graph	Прототипування за допомогою Script Canvas

Додаток Б. Структура віртуального ігрового середовища

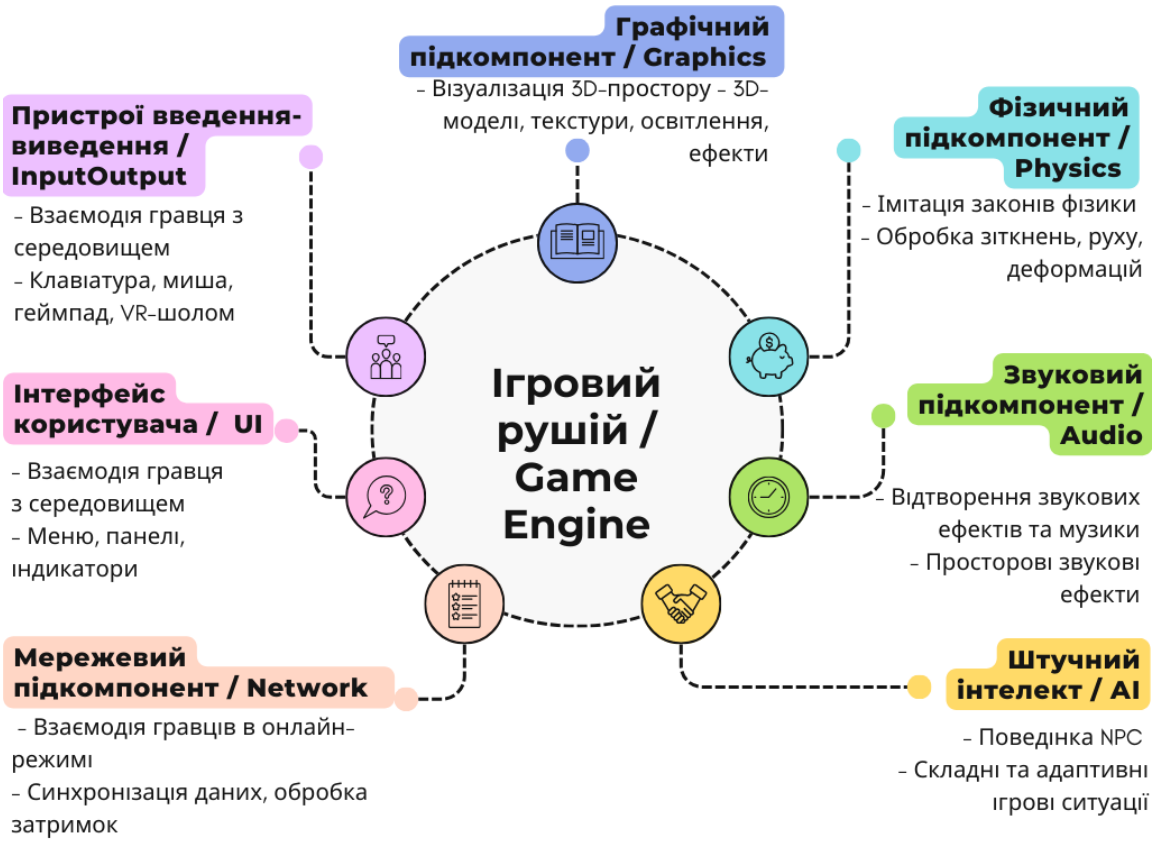


Рисунок 1 – Схема структури віртуального ігрового середовища

Додаток В. Розробка ігрового рівня

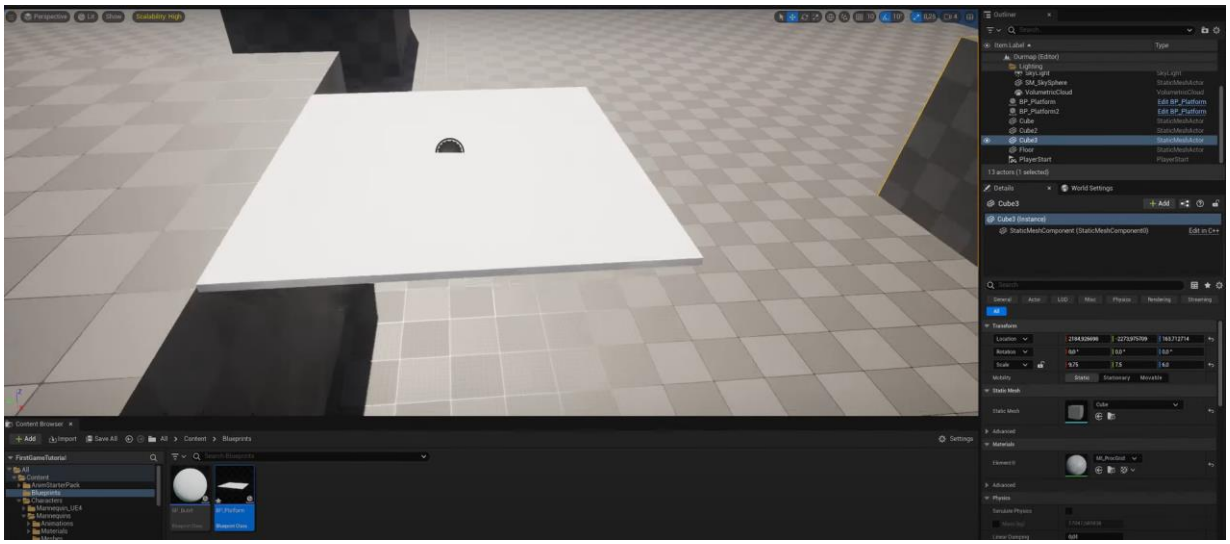


Рисунок 1 – Візуалізація об'єкта в Unreal Engine

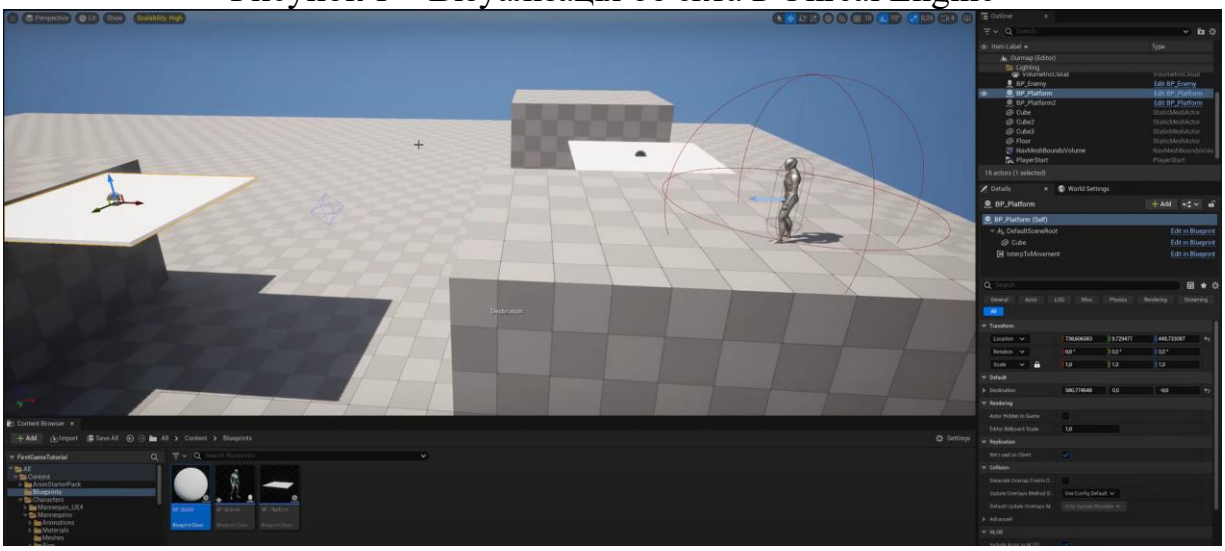


Рисунок 2 – Візуалізація рівня в Unreal Engine з персонажем та елементами оточення.

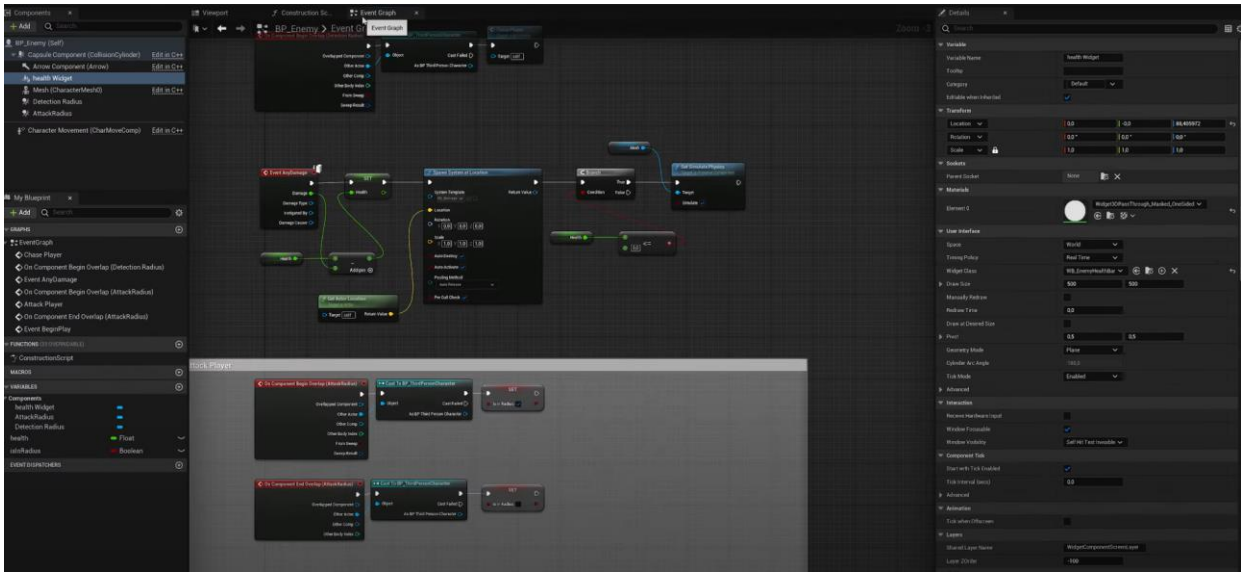


Рисунок 3 – Blueprint-схема для обробки подій та зміни параметрів об'єкта в Unreal Engine.

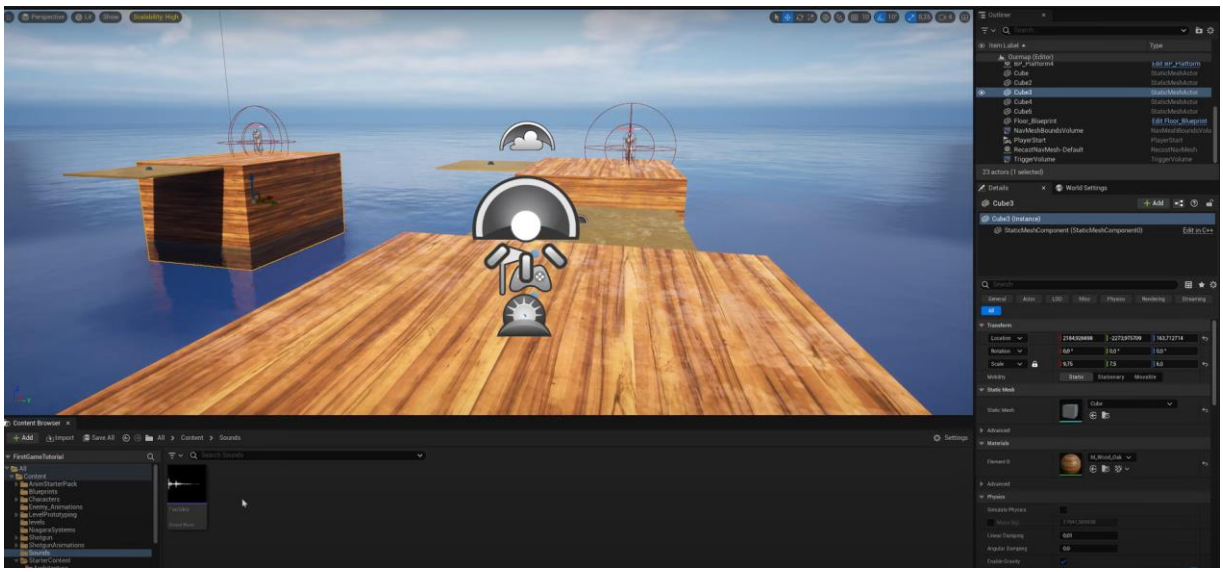


Рисунок 4 – Візуалізація ігрового рівня в Unreal Engine з розміщеними об'єктами

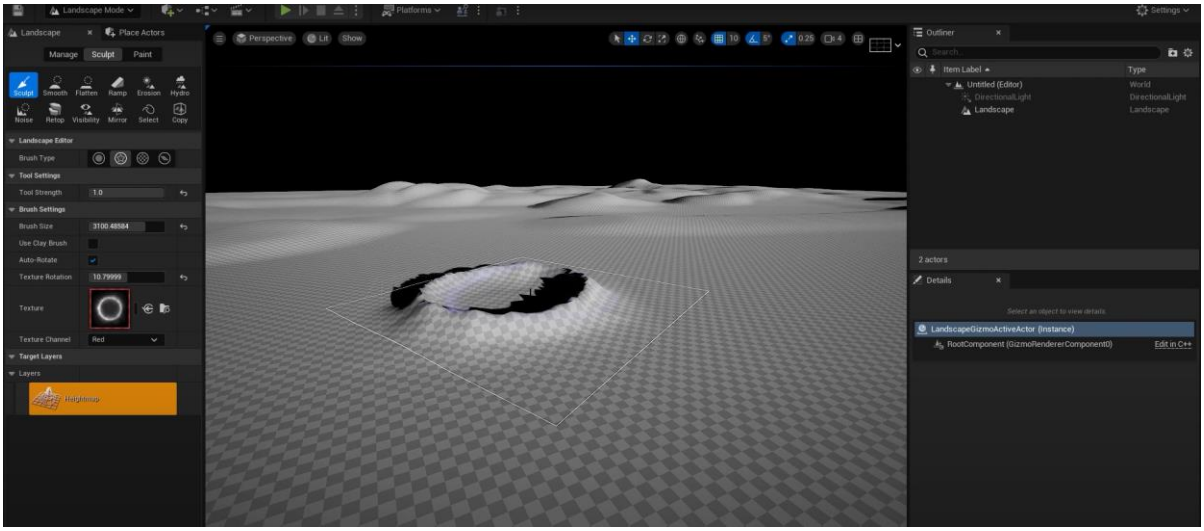


Рисунок 5 – Налаштування параметрів пензля для створення кратера на ландшафті в Unreal Engine

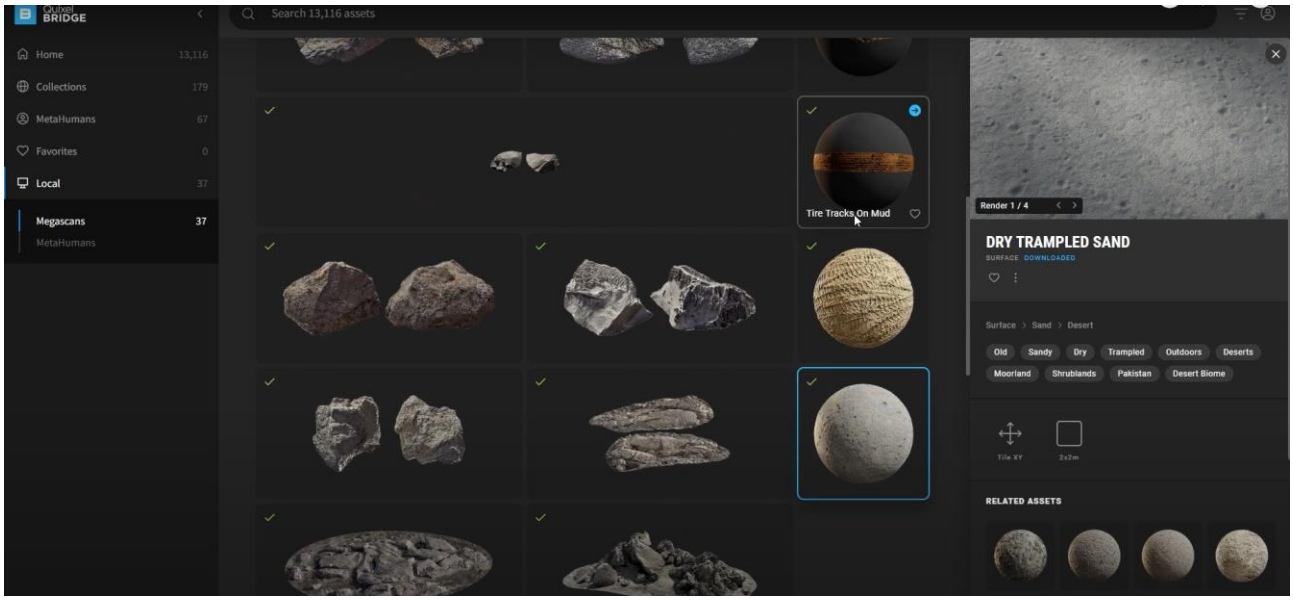


Рисунок 6 – Вибір ассетів для віртуального середовища в програмі Bridge

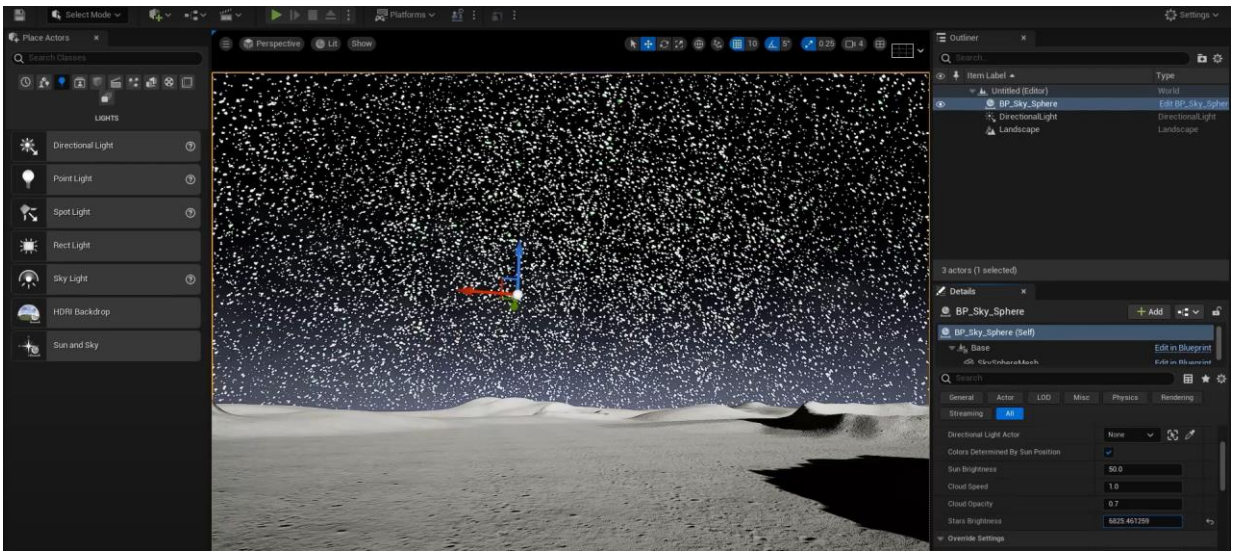


Рисунок 7 – Налаштування параметрів сфери неба та джерела освітлення для створення ефекту зоряного неба

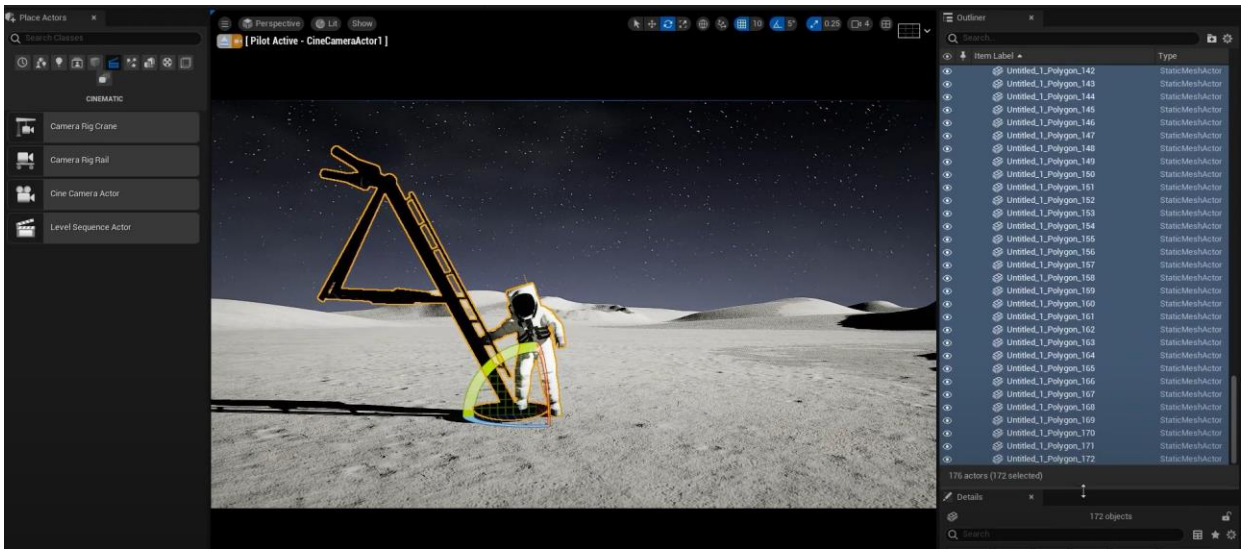


Рисунок 8 – Налаштування параметрів Cine Camera Actor

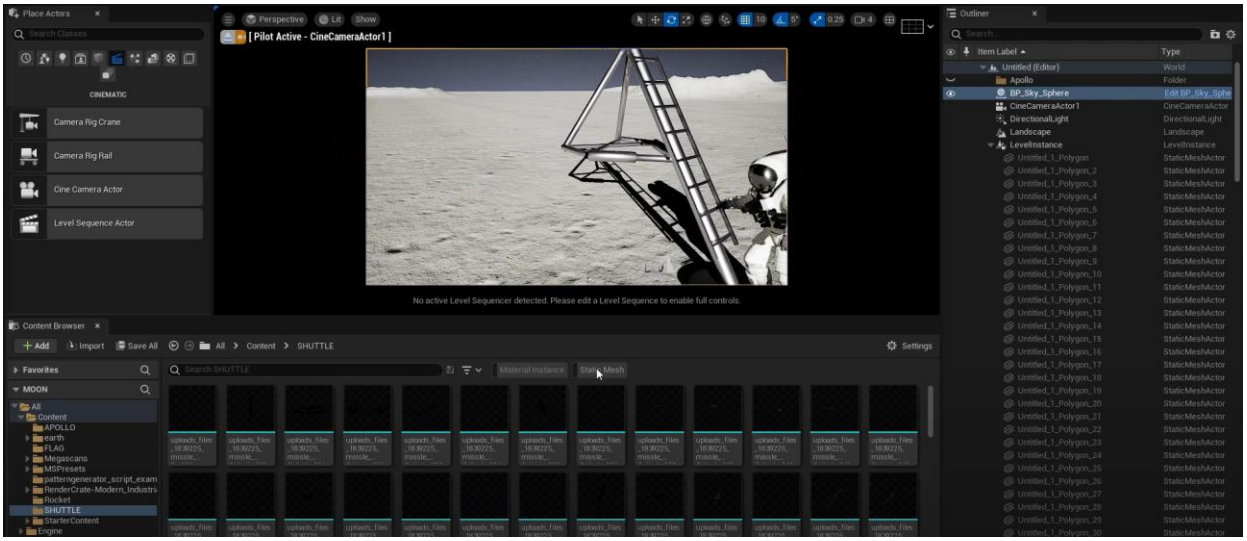


Рисунок 9 – Налаштування параметрів камери та розміщення об'єктів у сцені для створення кінематографічного кадру

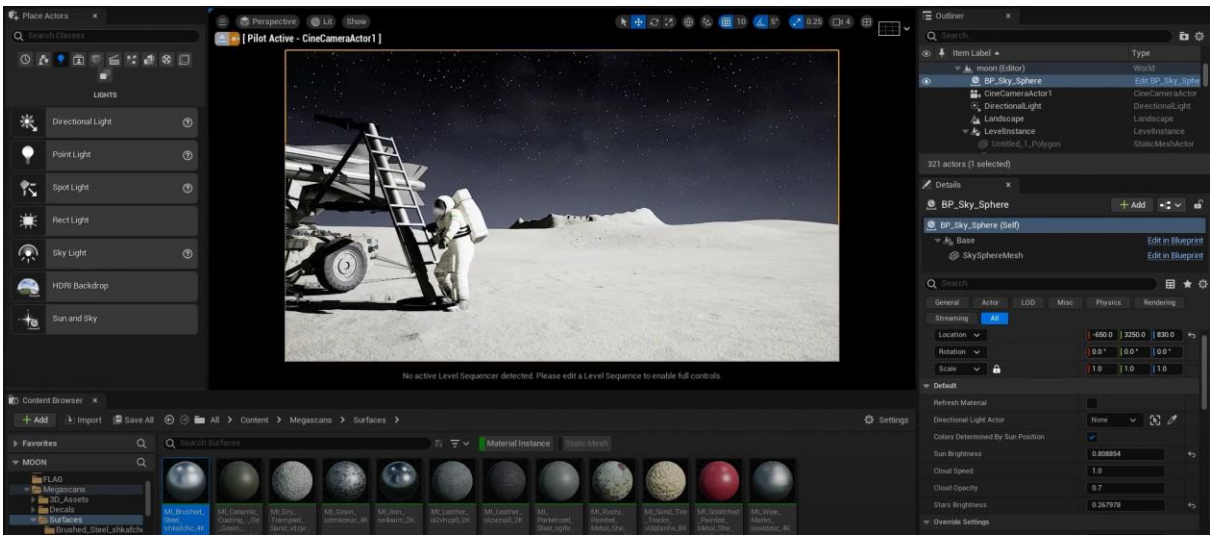


Рисунок 10 – Композиція кінематографічної сцени в Unreal Engine з використанням Cine Camera Actor та Level Sequence

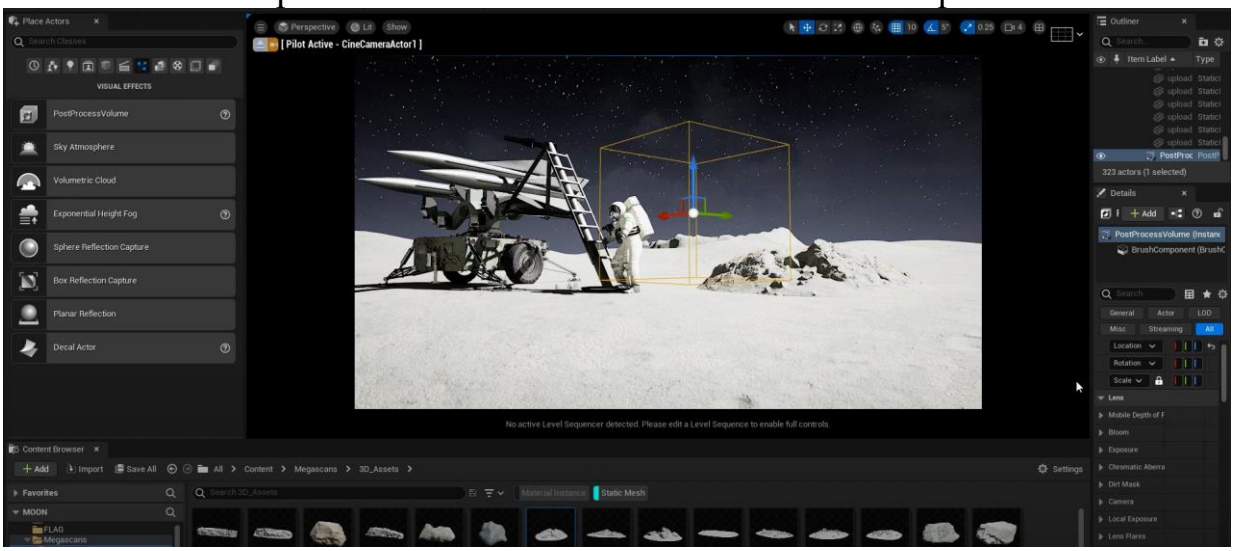


Рисунок 11 – Використання Post Process Volume для покращення візуального стилю кінематографічного кадру

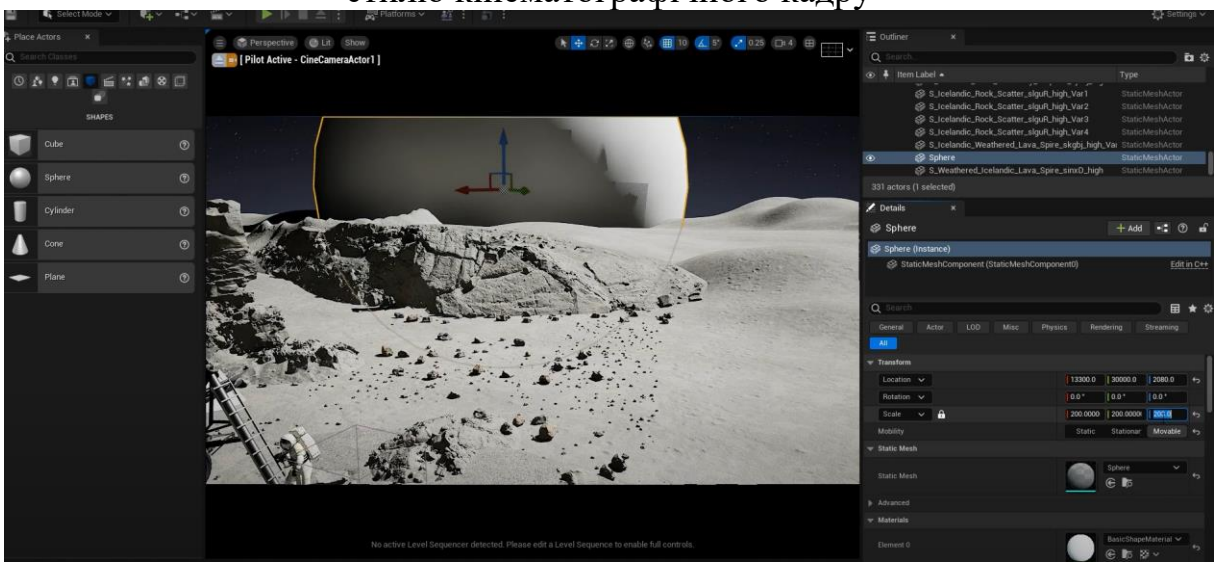


Рисунок 12 – Використання панелі Details для налаштування масштабу та позиції об'єкта "Sphere"

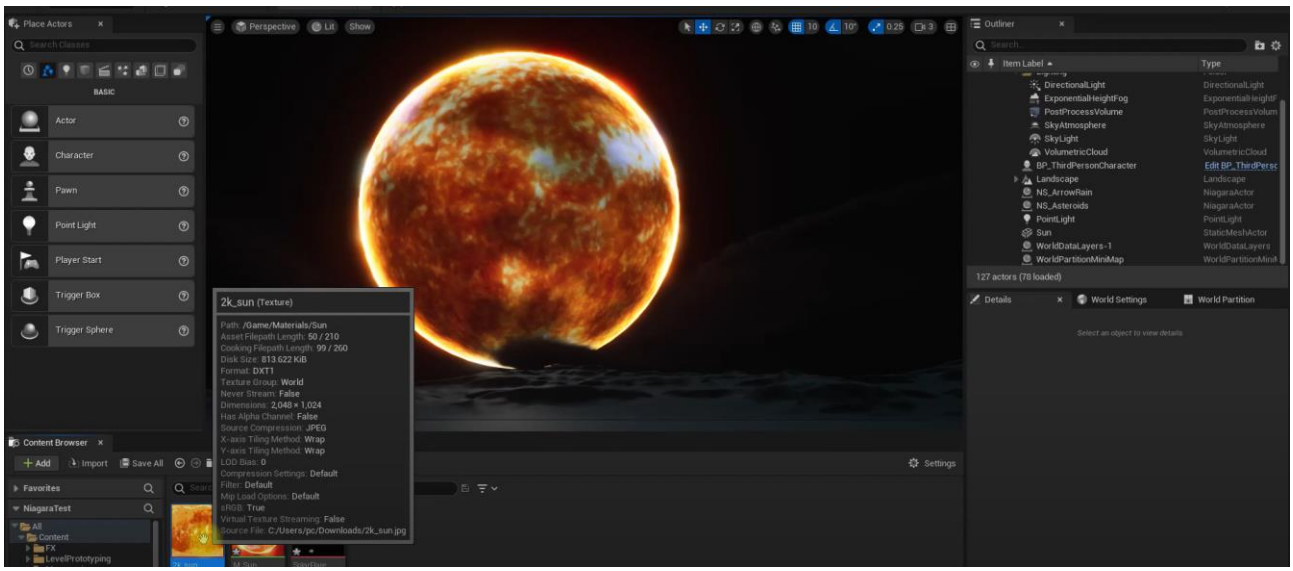
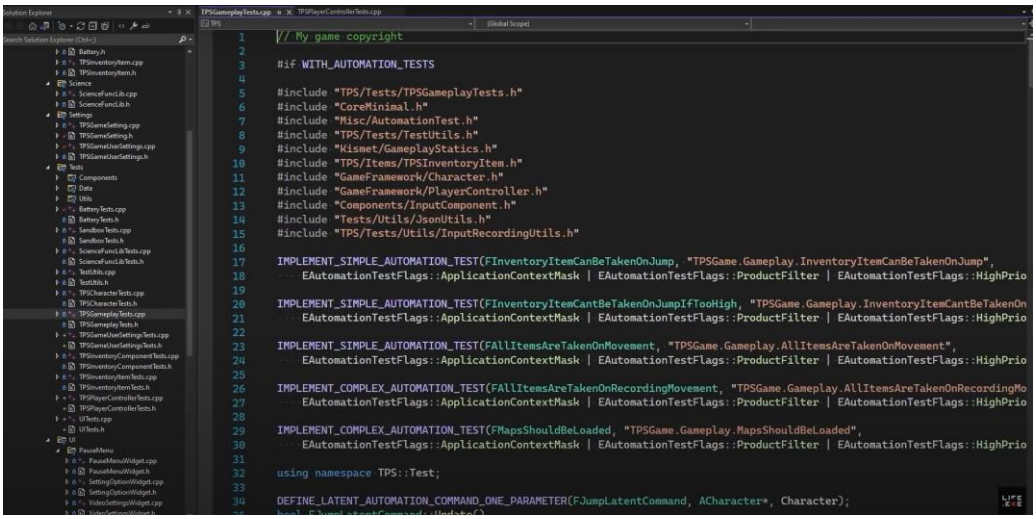


Рисунок 13 – Робочий простір Unreal Engine з налаштуванням параметрів матеріалу для створення ефекту палаючої планети

Додаток Д. UI тестування віртуального ігрового середовища



```cpp
// My game copyright

#if WITH_AUTOMATION_TESTS

#include "TPS/Tests/TPSGameplayTests.h"
#include "CoreMinimal.h"
#include "Misc/AutomationTest.h"
#include "TPS/Tests/TestUtils.h"
#include "Kismet/GameplayStatics.h"
#include "TPS/Items/TPSInventoryItem.h"
#include "GameFramework/Character.h"
#include "GameFramework/PlayerController.h"
#include "Components/InputComponent.h"
#include "Tests/Utils/JsonUtils.h"
#include "TPS/Tests/Utils/InputRecordingUtils.h"

IMPLEMENT_SIMPLE_AUTOMATION_TEST(FInventoryItemCanBeTakenOnJump, "TPSGame.Gameplay.InventoryItemCanBeTakenOnJump",
    EAutomationTestFlags::ApplicationContextMask | EAutomationTestFlags::ProductFilter | EAutomationTestFlags::HighPrio

IMPLEMENT_SIMPLE_AUTOMATION_TEST(FInventoryItemCantBeTakenOnJumpIfTooHigh, "TPSGame.Gameplay.InventoryItemCantBeTakenOn
    EAutomationTestFlags::ApplicationContextMask | EAutomationTestFlags::ProductFilter | EAutomationTestFlags::HighPrio

IMPLEMENT_SIMPLE_AUTOMATION_TEST(FAllItemsAreTakenOnMovement, "TPSGame.Gameplay.AllItemsAreTakenOnMovement",
    EAutomationTestFlags::ApplicationContextMask | EAutomationTestFlags::ProductFilter | EAutomationTestFlags::HighPrio

IMPLEMENT_COMPLEX_AUTOMATION_TEST(FAllItemsAreTakenOnRecordingMovement, "TPSGame.Gameplay.AllItemsAreTakenOnRecordingMo
    EAutomationTestFlags::ApplicationContextMask | EAutomationTestFlags::ProductFilter | EAutomationTestFlags::HighPrio

IMPLEMENT_COMPLEX_AUTOMATION_TEST(FMapsShouldBeLoaded, "TPSGame.Gameplay.MapsShouldBeLoaded",
    EAutomationTestFlags::ApplicationContextMask | EAutomationTestFlags::ProductFilter | EAutomationTestFlags::HighPrio

using namespace TPS::Test;

DEFINE_LATENT_AUTOMATION_COMMAND_ONE_PARAMETER(FJumpLatentCommand, ACharacter*, Character);
```

Рисунок 1 – Приклад реалізації автоматизованих тестів для перевірки коректності роботи механік інвентаря та переміщення об'єктів в Unreal Engine

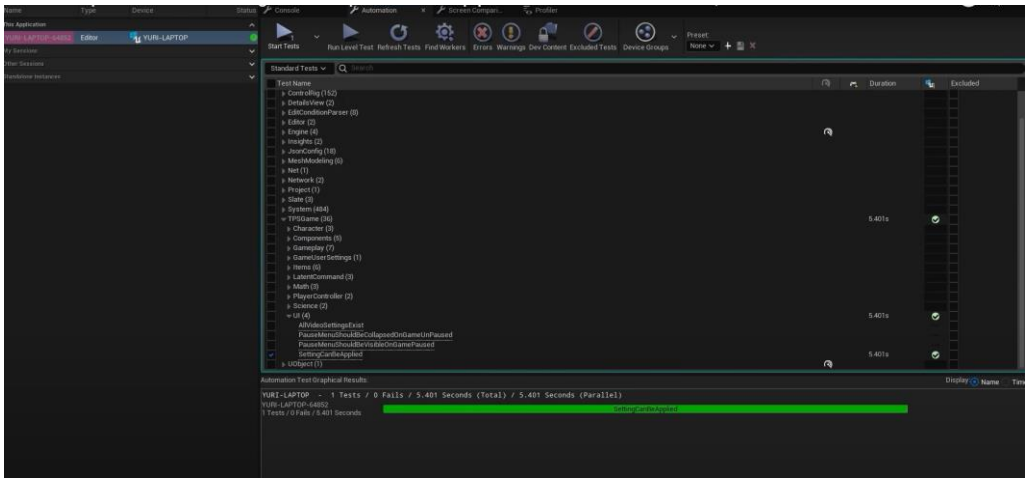
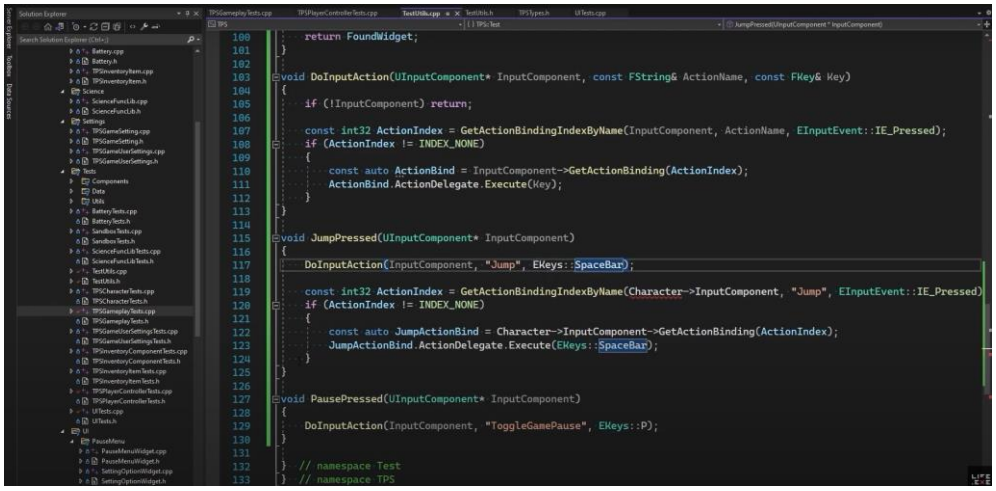


Рисунок 2 – Вікно Automation Tool з результатами виконання автоматизованих тестів



```cpp
    return FoundWidget;
}

void DoInputAction(UInputComponent* InputComponent, const FString& ActionName, const FKey& Key)
{
    if (!InputComponent) return;

    const int32 ActionIndex = GetActionBindingIndexByName(InputComponent, ActionName, EInputEvent::IE_Pressed);
    if (ActionIndex != INDEX_NONE)
    {
        const auto ActionBind = InputComponent->GetActionBinding(ActionIndex);
        ActionBind.ActionDelegate.Execute(Key);
    }
}

void JumpPressed(UInputComponent* InputComponent)
{
    DoInputAction(InputComponent, "Jump", EKeys::SpaceBar);

    const int32 ActionIndex = GetActionBindingIndexByName(Character->InputComponent, "Jump", EInputEvent::IE_Pressed)
    if (ActionIndex != INDEX_NONE)
    {
        const auto JumpActionBind = Character->InputComponent->GetActionBinding(ActionIndex);
        JumpActionBind.ActionDelegate.Execute(EKeys::SpaceBar);
    }
}

void PausePressed(UInputComponent* InputComponent)
{
    DoInputAction(InputComponent, "ToggleGamePause", EKeys::P);
}

} // namespace Test
} // namespace TPS
```

Рисунок 3 – Приклад реалізації функцій обробки натискання клавіш "Spacebar" та "P" для управління персонажем в Unreal Engine



Рисунок 4 – Вид з кабіни пілота космічного корабля з відображенням часу проходження рівня та індикаторами стану

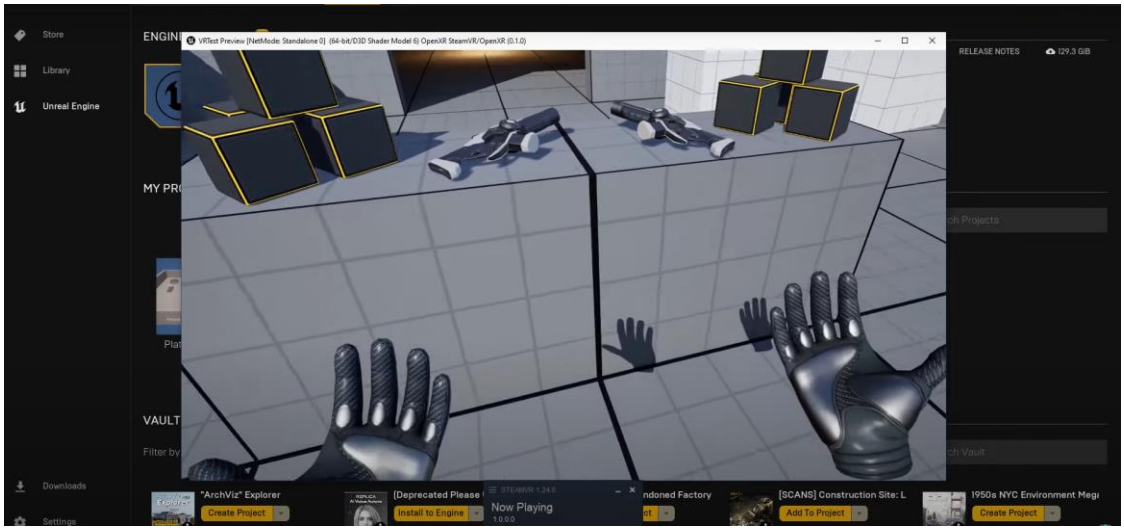


Рисунок 5 – Інтерфейс Unreal Engine Launcher з візуалізацією віртуального середовища

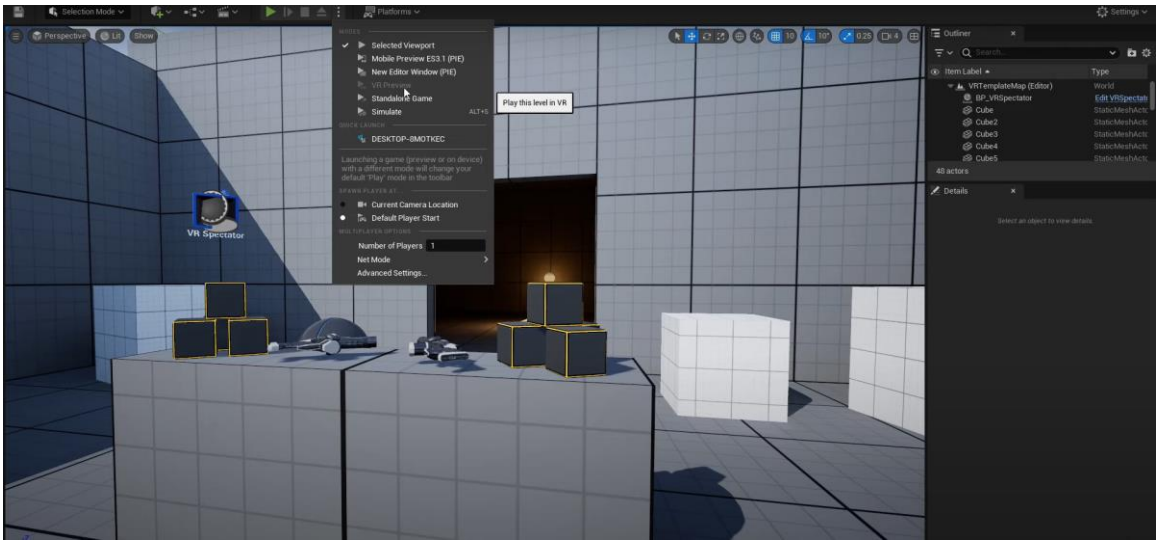


Рисунок 6 – Підготовка тестового рівня до запуску симуляції в Unreal Engine з налаштуванням параметрів режиму гри