МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Острозька академія»

Навчально науковий інститут ІТ та бізнесу

Кафедра інформаційних технологій та аналітики даних

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістра
на тему: « **Управління проєктом створення механік управління та взаємодії квестової гри на Unreal Engine »**

Виконав: студент 2 курсу, групи МУП-2
другого (магістерського) рівня вищої освіти
спеціальності 122 Комп'ютерні науки
ОПП «Управління проєктами»
Лаптєв Богдан Валерійович

Керівник: ***Місай В.В., викладач, фахівецьпрактик кафедри ІТБ***

Рецензент: *кандидат технічних наук, доцент, доцент кафедри прикладної математики Донецького національного університету імені Василя Стуса*

Загоруйко Любов Василівна

РОБОТА ДОПУЩЕНА ДО ЗАХИСТУ

Завідувач кафедри інформаційних технологій та аналітики даних
_____ (проф., д.е.н. Кривицька О.Р.)

Протокол № 5 від «04» грудня 2025 р.

Острог, 2025

АНОТАЦІЯ

кваліфікаційної роботи

на здобуття освітнього ступеня магістра

Тема: *Управління проєктом створення механік управління та взаємодії квестової гри на Unreal Engine.*

Автор: *Лаптєв Богдан Валерійович*

Науковий керівник: *Місай В.В., викладач, фахівецьпрактик кафедри ІТБ*

Захищена «……»…..... 2025 року.

Пояснювальна записка до кваліфікаційної роботи: *91с., 46 рис. ,49 джерел.*

Ключові слова: *управління проєктом; планування розробки; геймдизайн; квестова гра; механіки управління; взаємодія гравця; Unreal Engine; Blueprints; програмування ігор; інтерактивні об'єкти; контроль персонажа; анімації; UI/UX; прототипування; тестування механік; оптимізація продуктивності; командна робота; Agile; Scrum; документація проєкту; інтеграція систем.*

Короткий зміст праці:

Ця кваліфікаційна робота присвячена дослідженню процесу управління проєктом розробки механік управління та взаємодії квестової гри на базі рушія Unreal Engine 5. У роботі розглянуто теоретичні засади створення інтерактивних ігрових систем, принципи побудови ефективних механік управління, а також особливості їх реалізації засобами Blueprints і C++ API. Проаналізовано сучасні тенденції розвитку квестових ігор, визначено ключові підходи до формування користувацького досвіду та геймдизайну. Особливу увагу приділено дослідженню архітектури Unreal Engine 5, його технічних можливостей і переваг для створення інтуїтивних систем управління, навігації та взаємодії з об'єктами середовища.

У практичній частині реалізовано прототип квестової гри з інтегрованими системами управління персонажем, інвентарем, діалогами та інтерфейсом користувача. Управління проєктом здійснювалося із застосуванням методологій Agile та Scrum, що дозволило забезпечити ефективну організацію робочих процесів, тестування та поступову оптимізацію функціоналу. Проведено експериментальну перевірку продуктивності та зручності розроблених рішень. Результати дослідження підтвердили ефективність використання комплексного підходу, який поєднує технічну реалізацію і сучасні методи управління проєктами, що сприяє підвищенню якості інтерактивних систем і користувацького досвіду у квестових іграх.

_____

ANNOTATION

of a qualification paper

for a master's degree

Theme: *Project management of creating control and interaction mechanics for a quest game on Unreal Engine.*

Author: *Laptiev Bohdan*

Scientific supervisor: *Misai V.V., lecturer, specialist practitioner at the ITB department*

Defended «……»….....of 2025.

Explanatory note to the qualification work: *p.91, pic. 46, 49 sources.*

Keywords: *project management; development planning; game design; quest game; control mechanics; player interaction; Unreal Engine; Blueprints; game programming; interactive objects; character control; animations; UI/UX; prototyping; mechanic testing; performance optimization; teamwork; Agile; Scrum; project documentation; systems integration.*

Summary of the paper:

This qualification work is devoted to the study of the project management process for the development of control and interaction mechanics for a quest game based on the Unreal Engine 5 engine. The work considers the theoretical principles of creating interactive game systems, the principles of building effective control mechanics, as well as the features of their implementation using Blueprints and C++ API. Modern trends in the development of quest games are analyzed, key approaches to the formation of user experience and game design are identified. Particular attention is paid to the study of the Unreal Engine 5 architecture, its technical capabilities and advantages for creating intuitive control systems, navigation and interaction with environmental objects. In the practical part, a prototype of a quest game with integrated character, inventory, dialogue and user interface management systems was implemented. The project was managed using Agile and Scrum methodologies, which allowed for effective organization of work processes, testing and gradual optimization of functionality. Experimental testing of the performance and convenience of the developed solutions was carried out. The results of the study confirmed the effectiveness of using an integrated approach that combines technical implementation and modern project management methods, which contributes to improving the quality of interactive systems and user experience in quest

Зміст

ВСТУП

Сучасна ігрова індустрія є однією з найдинамічніше зростаючих галузей цифрової економіки, що поєднує у собі технології програмування, дизайну, штучного інтелекту, анімації та психології користувача. Зі збільшенням вимог гравців до якості графіки, реалістичності поведінки персонажів та рівня інтерактивності, зростає потреба у створенні нових підходів до управління й взаємодії у віртуальних середовищах.

Особливе місце серед жанрів відеоігор займають квестові ігри, які поєднують сюжетну глибину, логічні завдання та активну взаємодію з об’єктами віртуального світу. Їх розробка потребує комплексного поєднання технологічних, дизайнерських і управлінських рішень, адже створення подібних проєктів передбачає узгоджену роботу між різними напрямами - програмуванням, візуальним дизайном, звуковим супроводом, тестуванням і продакшеном. Саме ефективне управління проєктом дозволяє координувати ці процеси, забезпечуючи своєчасне виконання завдань, оптимізацію ресурсів і підтримку якості на всіх етапах розробки.

Для організації такого процесу широко застосовуються сучасні методології управління, зокрема Agile, Scrum та Iterative Design, які ґрунтуються на принципах ітераційного розвитку, командної співпраці та постійного вдосконалення продукту. Ці підходи дають змогу швидко адаптуватися до змін у дизайні, технічних вимогах або очікуваннях користувачів, що є критично важливим у динамічному середовищі ігрової індустрії.

Використання Unreal Engine 5 відкриває широкі можливості для створення квестових ігор завдяки потужному графічному рушію, системі Blueprints, розширенню через C++ API, а також вбудованим інструментам для розробки анімацій, навігації, освітлення та інтерактивної логіки. У поєднанні з гнучкими методами управління це забезпечує ефективну реалізацію складних ігрових систем, узгодженість між етапами розробки та високу якість кінцевого продукту.

Таким чином, дослідження процесів створення механік управління, інтерактивних систем і підходів до управління ігровими проєктами є актуальним напрямом у сучасному геймдизайні, який поєднує технічну інноваційність, творче мислення та ефективну організацію розробницького процесу.

Актуальність це розвиток ігрових технологій, зокрема інтерактивних 3Dсередовищ, сприяє підвищенню вимог до якості реалізації систем управління та взаємодії. Гравці очікують інтуїтивного керування, реалістичних фізичних реакцій об'єктів і логічно узгодженої поведінки персонажів. Актуальність теми полягає у необхідності поглибленого вивчення принципів побудови ефективних механік управління та інтерактивних елементів у квестових іграх, що дозволяють забезпечити високий рівень залучення користувача, стабільність і продуктивність проєкту.

Особливу значущість це питання має в умовах розвитку індустрії незалежних розробників (indie games), де ефективна організація процесу створення ігрових механік часто визначає конкурентоспроможність кінцевого продукту.

Метою магістерської роботи є дослідження, проєктування та розробка механік управління і систем інтерактивної взаємодії у квестових відеоіграх із використанням можливостей рушія Unreal Engine 5, а також оцінка їх ефективності з точки зору користувацького досвіду, продуктивності та стабільності функціонування ігрового середовища.

Досягнення цієї мети передбачає комплексне вивчення сучасних технологій розробки ігрових систем, аналіз наявних підходів до реалізації управління персонажами та об'єктами, дослідження принципів організації інтерактивної взаємодії користувача з елементами ігрового світу, а також впровадження результатів у вигляді власного прототипу квестової гри.

Особливу увагу приділено використанню інструментів Blueprints та C++ API для створення логічних зв'язків, інтеграції систем навігації, анімацій, взаємодії з NPC та предметами, що дозволяє забезпечити високу гнучкість і масштабованість розроблених рішень. Кінцевою метою є створення ефективної

моделі управління і взаємодії у квестовій грі, яка поєднує технічну досконаліть, інтуїтивність для користувача, відповідність сучасним стандартам геймдизайну та потенціал для подальшого розширення в межах майбутніх проєктів.

Завдання дослідження:

Для досягнення поставленої мети необхідно вирішити такі основні завдання:

- Провести аналіз сучасних підходів до реалізації систем управління та взаємодії в квестових іграх.
- Дослідити технічні можливості рушія **Unreal Engine 5** щодо створення інтерактивних систем управління з використанням **Blueprints** та **C++ API**.
- Розробити власні механіки управління персонажем, навігації, маніпулювання предметами та взаємодії з NPC.
- Інтегрувати анімаційні та логічні системи у структуру гри для забезпечення природної взаємодії користувача з середовищем.
- Провести тестування та оцінку користувацького досвіду, визначити рівень інтуїтивності та зручності управління.
- Оптимізувати технічні параметри реалізованих рішень для забезпечення стабільної продуктивності гри.

Об’єктом дослідження є процес розробки квестових відеоігор із використанням сучасних ігрових рушіїв, технологій проєктування інтерактивних систем та методів реалізації користувацького управління у віртуальному середовищі.

До складу об’єкта входять усі етапи життєвого циклу створення ігрового проєкту - від концептуального проєктування, побудови логіки ігрового процесу та інтерактивної взаємодії до тестування, оптимізації й оцінки користувацького досвіду.

Дослідження охоплює питання вибору інструментарію для побудови систем управління персонажем і взаємодії з об’єктами світу, реалізації сценаріїв поведінки NPC, створення інтерфейсу користувача, а також інтеграції аудіовізуальних елементів для формування цілісного ігрового простору.

Таким чином, об’єктом дослідження виступає комплексний процес створення інтерактивної квестової гри, у якому поєднуються технічні, дизайнерські та організаційні аспекти розробки на основі рушія Unreal Engine 5.

Предметом дослідження є методи створення, реалізації та оптимізації механік управління і систем інтерактивної взаємодії у квестових відеоіграх, що розробляються на базі ігрового рушія Unreal Engine 5.

У межах предмета дослідження розглядаються теоретичні та практичні підходи до побудови інтерактивних систем, які забезпечують взаємодію користувача з віртуальним середовищем, включаючи реалізацію руху персонажів, навігації у тривимірному просторі, маніпулювання ігровими об’єктами, а також логіку комунікації з неігровими персонажами (NPC).

Особлива увага приділяється інструментам Unreal Engine 5, таким як Blueprints (візуальне програмування) та C++ API, які дозволяють створювати гнучкі та масштабовані ігрові системи, поєднуючи зручність візуальної розробки з високою продуктивністю нативного коду. Дослідження охоплює питання побудови анімованих систем управління, інтеграції користувацького інтерфейсу (UI/UX), синхронізації дій гравця з подіями ігрового світу, а також методи оптимізації взаємодії для підвищення стабільності та швидкодії гри.

Крім того, предметом дослідження є підходи до підвищення якості користувацького досвіду (UX) шляхом аналізу зручності управління, інтуїтивності інтерфейсу, природності взаємодії та занурення користувача у процес гри.Проаналізувати сучасний стан розвитку ігрової індустрії та визначити основні тенденції у створенні квестових відеоігор, зокрема щодо реалізації систем управління, інтерактивної взаємодії та користувацького досвіду.

Дослідити архітектуру та технічні можливості рушія Unreal Engine 5, що стосуються створення інтерактивних ігрових систем, включаючи роботу з Blueprints, C++ API, системою анімацій, фізичними компонентами, навігаційною мережею (NavMesh) та UI.Вивчити існуючі підходи до проєктування механік

управління у квестових іграх та визначити їхні переваги, недоліки й доцільність застосування в межах власного проєкту.

Розробити прототип системи управління персонажем, що забезпечує природну та зручну взаємодію користувача з ігровим середовищем, включаючи реалізацію руху, взаємодії з предметами, навігації по локаціях та спілкування з NPC.Створити систему інтерактивних елементів і логіку взаємодії користувача з об’єктами світу гри, використовуючи Blueprints і C++ API для реалізації сценарних подій, діалогів та динамічних змін у середовищі.

Розробити та інтегрувати систему користувацького інтерфейсу (UI), що забезпечує зручність керування, відображення ключової інформації та інтуїтивність взаємодії з гравцем.Здійснити тестування та аналіз користувацького досвіду (UX) для оцінки ефективності створених механік управління, інтерфейсу та загальної якості взаємодії користувача з грою.

Провести оптимізацію технічних рішень, спрямовану на покращення продуктивності, стабільності та ефективності роботи розробленої гри, з урахуванням апаратних ресурсів і вимог сучасних ігрових систем.Сформувати методичні рекомендації щодо впровадження ефективних механік управління та систем взаємодії у квестових іграх на базі Unreal Engine 5, які можуть бути використані при подальшій розробці подібних проєктів.

Методи дослідження

У роботі використано комплекс взаємодоповнюючих методів, що дозволяють глибоко проаналізувати процес створення квестових відеоігор, оптимізувати механіки управління та забезпечити високу якість користувацької взаємодії:

- **Аналітичний метод** - застосовується для дослідження теоретичних основ ігрового дизайну, вивчення архітектури сучасних ігрових рушіїв, зокрема Unreal Engine 5, а також для аналізу наукових джерел і практичних підходів до створення систем управління персонажами та інтерактивними об’єктами.
- **Порівняльний метод** - використовується для зіставлення різних підходів до реалізації управління у сучасних квестових іграх, оцінки ефективності

інструментів Blueprints і C++ API, а також визначення переваг і недоліків конкретних технічних рішень.

- **Експериментальний метод** - передбачає розробку, впровадження та тестування прототипів механік управління, взаємодії з об’єктами, систем навігації та користувацького інтерфейсу з метою оцінки їхньої ефективності, зручності та продуктивності.
- **Моделювання** - застосовується для побудови сценаріїв поведінки персонажів, створення моделей ігрових процесів і систем взаємодії у віртуальному середовищі, а також для аналізу взаємозв’язків між окремими компонентами гри.
- **Евристичний метод** - використовується для якісної оцінки користувацького досвіду, виявлення проблем у системах управління та пошуку шляхів удосконалення інтерактивності, ігрової логіки й зворотного зв’язку між гравцем та грою.

РОЗДІЛ 1

ТЕОРЕТИЧНІ ОСНОВИ ТА ОГЛЯД ТЕХНОЛОГІЙ РОЗРОБКИ ІНТЕРАКТИВНОГО КВЕСТОВОГО КОНТЕНТУ

1.1.Теоретичні основи ігрових механік управління та взаємодії

Основи ігрових механік управління та взаємодії становлять важливу частину гейм дизайну та визначають, як гравець взаємодіє з ігровим світом, а також які механізми впливають на його досвід у грі. Механіки управління забезпечують гравцеві можливість впливати на світ гри, керувати персонажем, рухатись, використовувати предмети та взаємодіяти з навколишнім середовищем. Вони формують основну основу взаємодії гравця з грою, що, у свою чергу, визначає комфорт і задоволення від процесу гри.

Одним із важливих аспектів є те, що механіки мають бути інтуїтивно зрозумілими, щоб гравець міг швидко освоїтися в грі та не витрачати багато часу на навчання. Успішні механіки управління повинні поєднувати простоту з достатньою глибиною для збереження інтересу гравця протягом довгого часу. Наприклад, у класичних іграх жанру платформерів або бойовиків основними механіками є управління персонажем за допомогою простих клавіш чи кнопок, але по мірі проходження гри виникають нові елементи, які дозволяють гравцеві додавати нові стратегії та рухи, що вимагають точності і швидкості.

Ігрові механіки взаємодії, зокрема, вимагають розробки ефективних систем, що дозволяють гравцеві не лише керувати персонажем, але й змінювати навколишнє середовище, взаємодіяти з іншими персонажами чи впливати на події в грі. Це може бути досягнуто через діалоги, розгалуження сюжетних ліній, використання інвентарю, виконання завдань або вирішення головоломок. Механіки взаємодії не лише підтримують інтерес гравця, а й забезпечують йому відчуття важливості та відповідальності за свої рішення, що сприяє більшому емоційному залученню.

Розуміння теоретичних принципів розробки таких механік допомагає дизайнерам створювати ігри, які не лише цікаві у плані геймплею, але й мають

емоційний відгук у гравця. Це включає дослідження таких аспектів, як мотивація, психологія користувача, системи винагород та системи зворотного зв’язку, які формують емоційну реакцію гравця та підтримують його інтерес. Теоретичне вивчення цих механік також дає змогу розробляти ігри, що сприяють розвитку когнітивних та психологічних навичок, таких як розв'язання проблем, стратегічне мислення, або кооперація в командних іграх.

Загалом, теоретичні основи механік управління та взаємодії в ігровому процесі визначають успішність гри, її здатність захопити гравця і створити для нього унікальний досвід, який поєднує не лише цікаві завдання, але й глибоке емоційне та когнітивне залучення.

1.2.Теоретичні аспекти оптимізації ігрового процесу

Оптимізація ігрового процесу є ключовим напрямом у сучасній розробці ігор, оскільки саме від неї залежить стабільність роботи, швидкість відтворення сцени, плавність анімації та загальний рівень занурення користувача у віртуальне середовище. Теоретичні основи оптимізації поєднують у собі принципи програмної інженерії, комп’ютерної графіки, психології сприйняття та ергономіки взаємодії з користувачем.

На концептуальному рівні оптимізація ігрового процесу передбачає **досягнення максимальної продуктивності без втрати якості сприйняття контенту**. Це означає, що розробник має не лише зменшити обчислювальні витрати системи, а й забезпечити при цьому природність рухів, реалістичність освітлення та адекватну реакцію об’єктів на дії гравця. У центрі цього процесу лежить поняття **балансу між якістю та продуктивністю**, що визначається як головний критерій ефективності гри.

Одним із ключових напрямів теоретичної оптимізації є **оптимізація графічних ресурсів**. Вона включає використання таких технологій, як *Nanite* для динамічного завантаження рівнів деталізації (LOD) моделей та *Lumen* для глобального освітлення у реальному часі. Завдяки цим інструментам система

обчислює лише ті деталі, які реально сприймає користувач, що значно знижує навантаження на процесор і відеокарту. Це демонструє застосування принципу **перцептивної оптимізації** - коли ресурси витрачаються лише на ті елементи, що мають візуальне або ігрове значення.

Не менш важливою складовою є **оптимізація логіки ігрового процесу**. Вона полягає у скороченні надлишкових обчислень, удосконаленні алгоритмів штучного інтелекту, зменшенні кількості одночасно активних фізичних симуляцій та ефективному управлінні пам’яттю. Правильне структурування ігрового циклу (game loop), оптимізація частоти оновлень кадру (frame rate) та синхронізація потоків даних дозволяють уникнути затримок і покращують загальний досвід користувача.

З теоретичної точки зору оптимізація також пов’язана з **когнітивними аспектами сприйняття**. Людський зір і увага мають обмежену здатність до розпізнавання дрібних деталей, тому надмірна точність візуалізації не завжди покращує ігровий досвід. Розробники можуть навмисно застосовувати методи спрощення, якот фільтрацію, динамічне розмиття або адаптивну деталізацію, щоб підтримувати високий рівень продуктивності, не знижуючи естетичної привабливості гри.

Ще один аспект - **ігровий баланс і ритм**. Оптимізація тут розглядається не лише в технічному сенсі, а й у структурному: важливо правильно чергувати фази напруження й розслаблення, підтримувати інтерес гравця, не перевантажуючи його когнітивно або емоційно. Ця частина оптимізації спирається на теорію потоку (Flow Theory) М. Чіксентмігаї, згідно з якою ідеальний ігровий досвід досягається тоді, коли рівень виклику відповідає рівню майстерності гравця.

Таким чином, теоретичні аспекти оптимізації ігрового процесу охоплюють широкий спектр напрямів - від технічного налаштування графіки й фізики до психологічного моделювання користувацького досвіду. Грамотна оптимізація забезпечує не лише стабільну роботу системи, а й гармонію між технічною ефективністю та емоційною залученістю користувача, що є фундаментальною умовою успішності сучасного інтерактивного продукту.

1.3. Аналіз популярних квестових ігор та їх механік

Квестові ігри - це жанр відеоігор, що зосереджений на дослідженні, вирішенні головоломок та розвитку сюжету через взаємодію з оточенням і персонажами. Вони поєднують логічне мислення та емоційну складову, дозволяючи гравцеві занурюватися в інтерактивний світ. Далі розглянемо кілька відомих квестових ігор та основні механіки, що характеризують ці проекти.

Однією з ключових особливостей квестових ігор є наголос на сюжетній складовій та глибокій інтерактивності. Багато таких проєктів використовують унікальні стилістичні рішення, атмосферний саундтрек і детально пропрацьовані персонажі, щоб посилити емоційний вплив на гравця.

Окрім класичних графічних квестів, жанр постійно еволюціонує, інтегруючись із іншими напрямками, такими як пригодницькі екшени, візуальні новели та навіть VRігри. Сучасні технології дозволяють створювати все більш реалістичні та інтерактивні світи, де кожен вибір гравця може мати довготривалі наслідки.

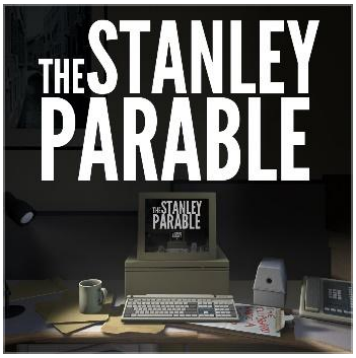

Рис.1.1. Обкладинка гри The Stanley Parable (2013)
Джерело: [https://uk.wikipedia.org/wiki/The_Stanley_Parable]

Механіка вибору та наслідків: Гравець має змогу слідувати інструкціям наратора або ігнорувати їх і рухатись за власним бажанням. Вибір гравця веде до різних варіантів розвитку подій.

Нелінійний сюжет: Історія змінюється в залежності від того, чи дотримується гравець рекомендацій наратора, чи ні. Є багато можливих закінчень.

Психологічний ефект: Гра виявляється експериментом над вибором та ілюзією свободи волі, де сам гравець стає частиною сюжету, а його дії - важливою частиною цього процесу.

Рис.1.2. Обкладинка гри Firewatch (2016)
Джерело: [https://en.wikipedia.org/wiki/Firewatch]

Діалоги та вибір: Основна механіка - це взаємодія через діалоги з іншими персонажами. Гравець керує діалогами, що впливає на розвиток відносин між персонажами та на сюжет.

Пошук і дослідження: Гравець досліджує відкритий світ, шукаючи підказки та таємниці, щоб розкрити загадку навколо зникнення людини.

Емоційний вплив: Гра акцентує увагу на емоціях та взаємодії персонажів у віддаленому лісі, що створює відчуття ізоляції та внутрішнього конфлікту.

Рис.1.3. Обкладинка гри Gone Home (2013)
Джерело: [https://en.wikipedia.org/wiki/Gone_Home]

Механіка дослідження: Як і в Firewatch, у Gone Home гравець досліджує довколишнє середовище, шукаючи предмети та підказки, щоб розкрити історію.

Без фізичних суперників: Гравець не стикається з ворогами чи екшеном, а натомість зосереджується на розкритті таємниць через дослідження будинку.

Нелінійний сюжет: Історія розвивається залежно від того, які предмети та підказки гравець знаходить під час дослідження, що створює відчуття занурення в особисту історію родини.

Квестові ігри зазвичай зосереджуються на дослідженні і взаємодії з ігровим світом для розкриття сюжету, розв'язання головоломок або вирішення певних завдань.

Основними механіками таких ігор є:

Дослідження світу: Гравець активно взаємодіє з навколишнім середовищем, збираючи підказки, предмети або інформацію, що є ключовою для прогресу в грі. Це може бути фізична взаємодія з об'єктами чи спостереження за деталями, які розкривають частини історії.

Нелінійний сюжет: Багато квестових ігор дозволяють гравцю впливати на розвиток сюжету залежно від виборів, зроблених під час гри. Це створює більш гнучкий і персоналізований досвід, де кінцівки та результат можуть змінюватися.

Психологічна атмосфера: Більшість квестових ігор створюють атмосферу самотності, напруги або інтриги, що глибше залучає гравця до процесу розкриття сюжету. Емоційний компонент є важливим у таких іграх, тому що він допомагає гравцеві сформувати зв'язок з персонажами або з тим, що відбувається в грі.

Головоломки та завдання: Для просування в грі часто потрібно вирішувати логічні задачі або головоломки, що є важливою частиною ігрового процесу. Це може бути взаємодія з предметами, розкриття таємниць або вирішення складних ситуацій.

Мінімалізм у механіках: Багато квестових ігор уникають традиційного екшену або бойових механік. Замість цього акцент ставиться на

спостережливість, вміння знаходити підказки і створення повільної, але глибокої історії через взаємодію з оточенням.

Популярні квестові ігри відзначаються багатогранними механіками, які включають дослідження, розв'язання головоломок, взаємодію з оточенням та вплив гравця на розвиток сюжету. Ключовими елементами є нелінійний підхід до сюжету, де вибори гравця визначають кінцівку, а також психологічна атмосфера, яка інтегрується з механіками взаємодії з предметами та персонажами. Головною метою таких ігор є створення глибокого емоційного досвіду через інтерактивність і високий рівень деталізації світу. Завдяки цьому квестові ігри стають не просто розвагою, а справжнім зануренням у світ, де кожна дія та вибір мають значення.[12]

1.4. Unreal Engine 5 у світі ігрових розробок.

Unreal Engine 5 сьогодні займає провідне місце у світі ігрових розробок і вважається одним із найсучасніших рушіїв, що задає високі стандарти якості та інновацій у створенні інтерактивних продуктів. Розроблений компанією Epic Games, він поєднує в собі гнучкість для індірозробників і потужність для великих студій, дозволяючи створювати як масштабні AAAпроєкти, так і невеликі експериментальні ігри. Основними перевагами рушія є його унікальні технології, які вивели візуалізацію на новий рівень. Система Nanite забезпечує відтворення моделей із надзвичайно високою кількістю полігонів без суттєвої втрати продуктивності, а технологія Lumen надає реалістичне динамічне освітлення і відображення в режимі реального часу. Крім того, UE5 відкриває широкі можливості для створення реалістичних персонажів завдяки MetaHuman Creator, що значно скорочує час розробки і підвищує рівень графічної достовірності. Не менш важливою перевагою є вдосконалене середовище Blueprints, яке дає змогу навіть розробникам без глибоких знань програмування створювати складні ігрові механіки та логіку взаємодії. Саме завдяки поєднанню інноваційних інструментів, простоти у використанні та відкритої екосистеми Unreal Engine 5

став ключовою платформою, яка визначає сучасні тенденції у розвитку ігрової індустрії.

Варто підкреслити, що Unreal Engine 5 активно використовується не лише в ігровій індустрії, а й у суміжних сферах, таких як кінематограф, архітектурна візуалізація, дизайн та віртуальна реальність. Це свідчить про універсальність рушія та його значний вплив на розвиток цифрових технологій загалом. Для розробників квестових ігор UE5 відкриває нові горизонти, адже забезпечує інтеграцію складних сценаріїв взаємодії, реалістичну візуалізацію середовища та можливість створення нелінійного сюжету з багатоваріантним розвитком подій. Таким чином, використання Unreal Engine 5 у процесі розробки не лише підвищує якість кінцевого продукту, а й надає конкурентні переваги, що робить його одним із найважливіших інструментів сучасного розробника.

1.5. Інструменти та технології в Unreal Engine 5

Unreal Engine (UE) є однією з найпотужніших платформ для розробки ігор, віртуальних середовищ і 3Dвізуалізацій. Вона надає розробникам широкий спектр інструментів і технологій, які значно спрощують процес створення контенту і дозволяють досягти високої якості результату.

Одним із найкорисніших інструментів є **Blueprints**, що дозволяє створювати ігрову логіку та анімації без необхідності написання коду. Візуальний підхід у Blueprints відкриває великі можливості для дизайнерів, а також дає змогу швидко експериментувати і тестувати ідеї. Водночас, Unreal Engine підтримує використання C++, що дозволяє програмістам мати повний контроль над продуктивністю і логікою гри, а також здійснювати глибоку оптимізацію.[13][19]

Material Editor є ще одним важливим інструментом, що дозволяє створювати складні матеріали та текстури для 3Dоб'єктів. З його допомогою можна комбінувати текстури, освітлення, відображення і багато інших параметрів для досягнення максимально реалістичного вигляду об'єктів.[19]

Розробка великих відкритих світів стає значно простішою завдяки **Landscape Tool**, який дозволяє створювати ландшафти, такі як гори та річки, використовуючи техніки висотних карт і скульптування. Окрім цього, Lighting System Unreal Engine забезпечує реалістичне освітлення, а Niagara дає змогу створювати вражаючі візуальні ефекти, такі як вибухи, дим та вогонь.

Для анімації в UE є потужний інструмент **Persona Animation Toolset,** який дозволяє редагувати анімації персонажів, працювати з переходами між анімаціями та здійснювати контроль за рухами кінцівок через **Inverse Kinematics.**[19]

Не менш важливим є AI (штучний інтелект), з використанням таких технологій, як **Behavior Trees** і **Blackboards**, що дають змогу створювати складні патерни поведінки NPC. Ці інструменти, у поєднанні з **Navigation Meshes** для автоматичного прокладання маршрутів, дають змогу створювати реалістичний AI для ігор.[19]

Що стосується аудіо, Unreal Engine має власну систему для інтеграції звуків, що дозволяє створювати складні аудіокомпозиції за допомогою **Sound Cue Editor** та забезпечує підтримку 3D звуку через **Spatialization**. Також є інтеграція з популярними звуковими системами, якот **Wwise** та **FMOD** [19].

Для створення мультиплеєрних ігор Unreal Engine пропонує потужні інструменти, такі як **Replicated Variables** та **ServerClient Architecture**, що дозволяє легко створювати онлайнігри з підтримкою багатьох гравців.

Не можна оминути увагою і **Fab (Unreal Marketplace)**, де можна придбати чи обміняти ресурси, такі як моделі, текстури, анімації тощо, що значно пришвидшує розробку.[19]

Для створення кінематографічних відео та анімацій Unreal Engine пропонує Sequencer, який дозволяє створювати сцени з камерами і персонажами, а також працювати з анімаціями для трейлерів чи інтерактивних відео.

Для розробки на **VR/AR** пристроях Unreal Engine також надає спеціальні шаблони, якот VR Template, що дозволяє створювати інтерактивні досвіди для віртуальної та доповненої реальності.[21]

Не забуваймо і про підтримку **DataDriven Development**, де можна зберігати дані гри в зовнішніх базах даних чи файлах, а також інтеграцію з різними Version Control Systems, що дозволяє ефективно працювати в команді.

1.6. Оцінка переваг та обмежень Unreal Engine 5 у розробці квестових ігор

Unreal Engine 5 є одним із провідних сучасних ігрових рушіїв, який надає розробникам потужні інструменти для створення високоякісних інтерактивних ігор, включно з квестовими проектами. Рушій поєднує передові графічні технології, системи анімації, програмування та інтерактивності, що дозволяє створювати насичені ігрові світи з високим рівнем занурення.

Серед основних переваг Unreal Engine 5 варто виділити високу графічну якість. Технології Nanite і Lumen забезпечують фотореалістичну деталізацію об’єктів, динамічне глобальне освітлення та природне відображення тіней, що створює ефект максимальної присутності гравця у віртуальному світі. Також рушій пропонує гнучкі інструменти для програмування ігрових механік. Blueprints дозволяє розробляти інтерактивні системи без глибоких знань коду, що спрощує процес прототипування та тестування ідей, тоді як C++ API забезпечує можливість створення складних та високопродуктивних систем управління персонажами, NPC та інтерактивними об’єктами.

Для квестових ігор особливо важливими є системи анімації та поведінки NPC, і тут UE5 також надає широкі можливості. Рушій підтримує скелетну анімацію, фізичні взаємодії та логіку поведінки неігрових персонажів, що дозволяє створювати динамічні сцени, у яких персонажі реагують на дії гравця та змінюються відповідно до сценарію. Важливим аспектом є також інтеграція користувацького інтерфейсу, яка дозволяє створювати інтуїтивні меню, панелі підказок та інформаційні елементи, що підвищують комфорт взаємодії та спрощують управління грою.

UE5 також відзначається масштабованістю та кросплатформеністю, що дозволяє розробляти проекти для ПК, консолей та мобільних пристроїв із мінімальними змінами. Крім того, рушій має широку спільноту користувачів та велику базу навчальних матеріалів і плагінів, що значно спрощує освоєння та прискорює процес розробки.

Разом із численними перевагами Unreal Engine 5 має і певні обмеження. Робота з високодеталізованими сценами, Nanite та Lumen потребує потужного апаратного забезпечення, що може ускладнювати тестування і розробку на слабких ПК. Складність рушія та великий обсяг функціональних можливостей вимагає від розробника значного часу на освоєння і планування архітектури проєкту. Проекти UE5 зазвичай мають великий розмір, що накладає додаткові вимоги до дискового простору та оперативної пам’яті. Також, при недостатній оптимізації, велика кількість NPC, складні інтерактивні об’єкти або ефекти можуть призводити до падіння продуктивності та нестабільної роботи гри.

Unreal Engine 5 надає розробникам квестових ігор значний потенціал для створення інтерактивних та візуально привабливих продуктів. Водночас ефективне використання рушія вимагає ретельного планування, оптимізації ресурсів та поєднання Blueprints і C++ для забезпечення стабільності роботи гри та високого рівня користувацького досвіду.

1.7. Предметне середовище та загальна модель проекту

Предметне середовище розроблюваного проєкту охоплює інтерактивну квестову гру, створену на базі сучасного рушія Unreal Engine 5. [19]

Основний акцент зроблено на моделюванні тривимірного ігрового простору, який включає карту з об’єктами, доступними для взаємодії, а також головного персонажа, що виступає центральним елементом ігрового процесу. Саме через нього реалізується зв’язок між користувачем та віртуальним середовищем.

Загальна модель проєкту складається з кількох взаємопов’язаних компонентів. Насамперед це ігровий простір, що визначає архітектуру сцени та її ключові об’єкти. До нього належать будівлі, предмети, декорації, NPC та інші елементи, з якими гравець може взаємодіяти. Другим важливим компонентом є головний персонаж, який має набір базових дій: переміщення, стрибки, використання предметів та взаємодію з навколишнім світом. Його механіки управління реалізуються за допомогою системи введення, що обробляє натискання клавіш і рух миші.

Третім компонентом виступає система взаємодії, яка визначає логіку виконання ігрових дій. Наприклад, гравець може відкрити двері, підняти предмет, активувати діалог чи розпочати квест, і всі ці дії відбуваються у чітко визначених умовах. Окреме місце займають ігрові завдання – квести, які структурують ігровий процес, задають послідовність дій та мотивують користувача до подальшого проходження. Вони формують основну сюжетну канву гри та забезпечують глибший рівень інтерактивності.

Взаємозв’язок між цими компонентами формує цілісну функціональну модель. Персонаж виконує дії залежно від свого положення у просторі та стану виконання завдань, об’єкти середовища реагують на його дії, а квести виступають рушійною силою розвитку сюжету. Завдяки цьому досягається логічна послідовність у геймплеї та забезпечується занурення користувача у віртуальний світ.

Модель побудована з урахуванням масштабованості, що дозволяє розширювати проєкт новими механіками, об’єктами та сценаріями без необхідності кардинальної зміни архітектури. Це робить її універсальною основою, яка може бути використана не лише для даного проєкту, а й для подальших досліджень та розробок у сфері інтерактивних ігор.

Окрім того, предметне середовище виступає не лише як технічна складова, але й як засіб занурення гравця у віртуальний світ. Завдяки правильному поєднанню графічних елементів, звукового супроводу та інтерактивної логіки формується унікальна атмосфера, яка сприяє глибшому сприйняттю ігрового

процесу. Це підтверджує актуальність вибору саме такого підходу для розробки кваліфікаційного проєкту.

Таким чином, предметне середовище та загальна модель проєкту можна розглядати як фундаментальну основу всієї подальшої роботи. Вони забезпечують логічну структуру, технічну реалізованість і гнучкість для розширення, що робить проєкт не лише навчальним, але й практично значущим у контексті сучасних тенденцій розвитку ігрової індустрії.

Крім основних компонентів, у предметному середовищі передбачено впровадження системи управління даними про стан гри, яка дозволяє зберігати прогрес гравця, результати виконання квестів та стан об’єктів середовища. Це забезпечує можливість відновлення гри після перерви, а також створює основу для подальшого аналізу поведінки користувачів та вдосконалення механік.

Важливою складовою моделі є проведення багаторівневого тестування інтерактивних механік та сценаріїв. Завдяки цьому виявляються потенційні конфлікти між об’єктами, логічні помилки у виконанні квестів та проблеми взаємодії персонажа з середовищем. Таке тестування дозволяє своєчасно коригувати сценарії та налаштування, забезпечуючи плавний ігровий процес і комфортний користувацький досвід.

Окрему увагу приділено оптимізації продуктивності. Використання рівнів деталізації моделей (LOD), потокового завантаження контенту та розумного керування ресурсами дозволяє підтримувати стабільну швидкодію навіть у великих сценах з великою кількістю об’єктів і NPC. Це гарантує, що гравець може безперервно взаємодіяти з середовищем, не відчуваючи затримок чи падіння кадрів, що є критично важливим для інтерактивних квестів. [19]

Нарешті, предметне середовище передбачає гнучкість у налаштуванні рівня складності завдань і взаємодії з користувачем. Це дозволяє адаптувати гру під різні категорії гравців, забезпечуючи поступове ускладнення квестів та введення нових сценаріїв без потреби суттєво змінювати архітектуру проєкту. Такий підхід не лише підвищує реалістичність ігрового світу, але й сприяє формуванню більш глибокого та захоплюючого користувацького досвіду.

1.8. Технічні вимоги та середовище розробки

Розробка проєкту відбувається в середовищі Unreal Engine 5, яке є одним із найсучасніших інструментів для створення інтерактивних тривимірних додатків. Цей рушій поєднує в собі засоби для побудови графіки, опрацювання фізики, створення логіки ігор та інтеграції додаткових ресурсів. Особливістю використання даного середовища є можливість реалізації ігрової логіки за допомогою Blueprint – системи візуального програмування, яка дозволяє швидко створювати та змінювати алгоритми, перевіряти роботу гри у реальному часі та мінімізувати кількість помилок на етапі тестування. Це особливо важливо для студентських та дослідницьких проєктів, де головним завданням є не лише створення кінцевого продукту, а й набуття практичних навичок у сфері розробки.

Рис.1.4. Логотип ігрового рушія Unreal Engine 5
Джерело: [https://www.unrealengine.com/enUS/unrealengine5]

Для повноцінного функціонування рушія та комфортної роботи над проєктом необхідне відповідне апаратне забезпечення. Використання сучасних технологій, таких як Nanite чи Lumen, вимагає потужних відеокарт та достатнього обсягу оперативної пам’яті. Це пояснюється тим, що рушій працює з високополігональними моделями та динамічним освітленням, що значно підвищує навантаження на апаратну частину комп’ютера. Водночас можливість масштабування графічних налаштувань дозволяє запускати проєкти і на менш продуктивних системах, що робить Unreal Engine універсальним інструментом як для професійних розробників, так і для початківців. [19]

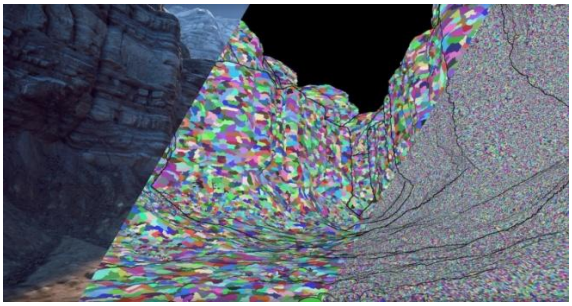

Рис.1.5. Приклад реалізації технології Nanite в Unreal Engine

Джерело: [https://dev.epicgames.com/documentation/en-us/unreal-engine/nanite-virtualized-geometry-in-unreal-engine]

Середовище розробки також включає допоміжні інструменти. До них належать редактори 3Dмоделей, графічні редактори для створення текстур, а також системи контролю версій, які дозволяють відстежувати зміни у проєкті. У процесі роботи може бути використано Blender для моделювання окремих об’єктів, а також GitHub для збереження проміжних результатів та організації командної співпраці. Це створює цілісну екосистему, де всі етапи – від створення моделі до інтеграції в рушій – взаємопов’язані та відбуваються послідовно.

Рис.1.6. Програми Blender та GitHub
Джерело: [Google]

Тестування проєкту виконується безпосередньо у середовищі рушія в режимі Play in Editor, що дозволяє миттєво перевіряти правильність роботи алгоритмів та логіки. У подальшому передбачене створення збірок проєкту для окремих платформ, що забезпечить можливість перевірки гри у більш реалістичних умовах. Такий підхід дозволяє контролювати якість продукту на всіх етапах його розробки та своєчасно виявляти недоліки.

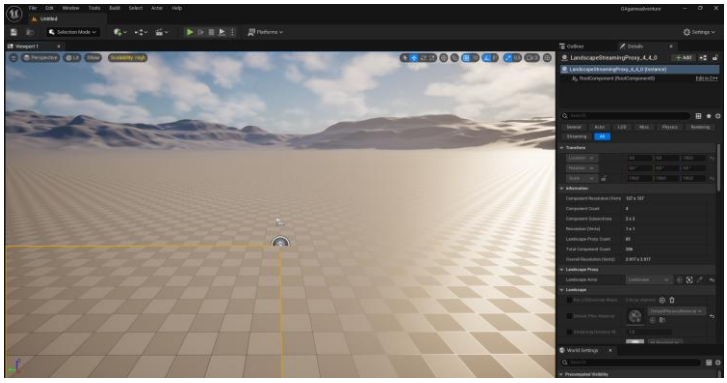

Рис.1.7. Середовище рушія в режимі Play in Editor Unreal Engine 5
Джерело:[Автор]

У цілому технічні вимоги та середовище розробки можна охарактеризувати як збалансовану систему, яка об’єднує сучасне програмне забезпечення, потужне апаратне забезпечення та гнучкі інструменти для тестування. Це створює сприятливі умови для реалізації креативних ідей, забезпечує високу швидкість розробки та дозволяє досягнути результату, що відповідає сучасним стандартам ігрової індустрії.

Крім того, важливим аспектом є можливість адаптації середовища розробки під різні потреби. Unreal Engine 5 підтримує мультиплатформеність, що дозволяє створювати збірки для різних операційних систем та пристроїв – від персональних комп’ютерів і консолей до мобільних платформ. Це відкриває широкі можливості для масштабування проєкту, його подальшої комерціалізації та використання в різних сферах, зокрема не лише в індустрії комп’ютерних ігор, а й у віртуальних симуляціях, архітектурній візуалізації чи навчальних середовищах.

Ще однією перевагою є активна спільнота розробників, яка надає велику кількість готових рішень, бібліотек та навчальних матеріалів. Це значно спрощує процес освоєння середовища, пришвидшує розробку та дозволяє знаходити оптимальні підходи до вирішення складних завдань. Студент, який працює з Unreal Engine 5, отримує не лише практичні навички програмування та роботи з 3Dграфікою, але й долучається до глобальної спільноти, що формує сучасні тренди у сфері інтерактивних технологій.

Таким чином, технічні вимоги та середовище розробки визначають основу майбутнього продукту, задаючи напрям розвитку всіх його складових. Використання передових технологій дозволяє забезпечити високу якість візуалізації, гнучкість і масштабованість архітектури, а також стабільність у процесі роботи. Це робить Unreal Engine 5 оптимальним вибором для реалізації сучасних ігрових проєктів, а сам процес розробки - ефективним, інноваційним та перспективним у контексті подальшого розвитку індустрії.

1.9. Архітектура програмного рішення

Архітектура програмного рішення розроблюваного проєкту базується на модульному підході, що передбачає поділ системи на окремі компоненти, кожен з яких відповідає за свою частину функціоналу. Така структура забезпечує гнучкість, зручність у тестуванні та можливість подальшого розширення проєкту.

Основу архітектури становить **ігровий рушій Unreal Engine 5**, який виконує роль центрального середовища для керування всіма процесами. На ньому реалізуються ключові компоненти:

- **Головний модуль гри (Game Mode)** – відповідає за визначення правил, логіку ігрового процесу та управління станами гри.
- **Ігровий простір (Level / Map)** – сцена, що містить об’єкти середовища, декорації та інтерактивні елементи.
- **Клас персонажа (Character Class)** – описує головного героя, його властивості та доступні дії, зокрема рух, взаємодію з об’єктами та виконання завдань.
- **Система управління (Input System)** – модуль, який обробляє дії користувача через клавіатуру, мишу чи контролер, трансформуючи їх у команди для персонажа.

- **Система взаємодії (Interaction System)** – логіка, яка дозволяє здійснювати дії з об’єктами (наприклад, відкривати двері, підбирати предмети або ініціювати діалоги).
- **Система завдань (Quest System)** – забезпечує виконання квестів, відстежує прогрес, умови та підказки, що визначають послідовність ігрового процесу.

Важливим елементом архітектури є використання **Blueprintсценаріїв**, які дозволяють реалізовувати логіку без необхідності застосування традиційного програмування. Це забезпечує швидке прототипування, зручність у налагодженні та можливість візуального відображення зв’язків між компонентами. [19]

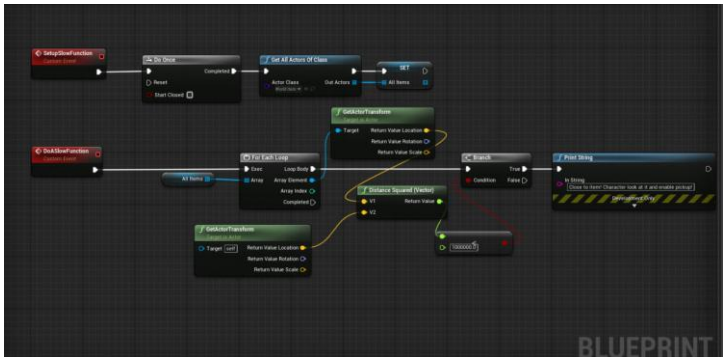

Рис. 1.8. Приклад Blueprint в Unreal Engine 5
Джерело:[Автор]

Архітектура також передбачає можливість інтеграції сторонніх ресурсів, зокрема 3Dмоделей, текстур та анімацій, що створює умови для поступового нарощування складності проєкту. Завдяки цьому проєкт можна розвивати як у напрямку графічного вдосконалення, так і шляхом ускладнення логіки гри.

Загалом архітектура програмного рішення побудована таким чином, щоб забезпечити баланс між простотою реалізації та масштабованістю. Це дозволяє не лише успішно реалізувати навчальний прототип, але й створити основу для подальшої розробки повноцінного ігрового продукту.

Окремо варто відзначити, що використання модульного підходу робить архітектуру проєкту зручною не лише для індивідуальної, а й для командної

розробки. Завдяки поділу на окремі блоки різні учасники команди можуть працювати над різними аспектами гри незалежно один від одного - один розробляє інтерфейс, інший створює актора з унікальною логікою, а третій налаштовує взаємодію між об’єктами. Це значно прискорює процес створення та тестування прототипів.

Ще однією перевагою такої архітектури є можливість швидкої інтеграції нових ресурсів, зокрема моделей, анімацій або звукових ефектів. Unreal Engine 5 має добре налагоджену систему імпорту та підтримку різних форматів, що робить додавання сторонніх матеріалів простим і ефективним. Це дозволяє значно зекономити час розробки та підвищує якість кінцевого продукту.

Крім того, Blueprintсистема забезпечує візуальне представлення логіки, що робить архітектуру зрозумілою навіть для користувачів без глибоких знань програмування. Це створює умови для легкого навчання, швидкої адаптації та мінімізації помилок на етапі розробки. У поєднанні з можливістю налагодження у реальному часі архітектура стає максимально гнучкою та адаптивною до змін.

У перспективі проєктна архітектура може бути розширена за рахунок підключення зовнішніх API, використання додаткових плагінів чи переходу на комбіновану реалізацію, де частина логіки буде винесена на рівень C++ для досягнення ще вищої продуктивності. Такий підхід дозволяє поступово еволюціонувати від простого прототипу до повноцінного ігрового продукту з великим функціоналом.

Таким чином, архітектура програмного рішення можна охарактеризувати як добре структуровану, масштабовану та придатну до розвитку. Вона поєднує у собі простоту реалізації, наочність та високу ефективність, що робить її оптимальним вибором для реалізації студентського кваліфікаційного проєкту та подальшого комерційного застосування.

Окрім зазначених компонентів, архітектура передбачає інтеграцію систем моніторингу та логування подій гри, що дозволяє відслідковувати стан персонажа, виконання квестів та взаємодію з об’єктами середовища. Це

забезпечує можливість аналізу помилок, оптимізації механік та підвищення стабільності роботи проєкту.

Важливою складовою є підтримка багаторівневого тестування модулів. Модульний підхід дозволяє окремо перевіряти функціонал персонажа, системи взаємодії та логіку квестів, що значно спрощує виявлення помилок і скорочує час на налагодження. Такий підхід забезпечує високу якість продукту ще на етапі прототипування.

Особливу увагу приділено оптимізації користувацького досвіду. Архітектура дозволяє реалізувати плавну взаємодію гравця з середовищем, забезпечує стабільну продуктивність навіть у складних сценах і підтримує масштабованість проєкту, що дозволяє додавати нові механіки та контент без негативного впливу на вже існуючий функціонал.

У перспективі передбачено можливість розширення архітектури через інтеграцію зовнішніх API, підключення хмарних сервісів для збереження прогресу гравця та використання сучасних технологій оптимізації рендерингу і фізики. Це створює умови для еволюції студентського прототипу у повноцінний комерційний продукт із широким функціоналом і високим рівнем користувацького досвіду. [19]

1.10. Методологічні основи та управління проєктом розробки квестової гри

Розробка квестових відеоігор є складним багатоступеневим процесом, який поєднує творчі, технічні та управлінські аспекти. Ефективна організація цього процесу потребує застосування сучасних методологій управління проєктами, що забезпечують контроль за всіма етапами - від концептуальної ідеї до тестування та випуску кінцевого продукту.

У контексті створення квестових ігор важливу роль відіграє вибір підходу до управління проєктом, оскільки саме від нього залежить ефективність організації процесу розробки, якість комунікації в команді та гнучкість у

прийнятті рішень. Розробка ігор - це динамічний процес, який часто передбачає постійні зміни в ідеї, дизайні або механіках, тому вибір правильної методології є ключовим фактором успішного завершення проєкту.

Традиційна каскадна **(Waterfall)** модель передбачає суворо послідовне проходження усіх етапів - від збору вимог до розробки, тестування й випуску продукту. Вона ефективна для проєктів із чітко визначеними технічними завданнями та стабільними вимогами, проте у випадку з ігровими проєктами, де дизайн та сценарій часто зазнають змін, такий підхід є менш гнучким. Його головною перевагою є структурованість, проте недоліком - складність внесення коректив у процесі реалізації.

На відміну від цього, гнучкі методології управління проєктами (Agile, Scrum, Kanban) базуються на ітераційному принципі. Це означає, що розробка відбувається у вигляді коротких циклів (ітерацій або спринтів), кожен із яких завершується отриманням робочого результату - прототипу або готового функціонального елемента гри. Такий підхід дозволяє постійно адаптувати продукт до нових ідей, тестувати гіпотези та вдосконалювати ігрові механіки без значних втрат часу.

У практиці розробки квестових ігор методологія Scrum часто використовується як базова, адже вона сприяє ефективній командній взаємодії, постійному зворотному зв’язку та контролю якості на кожному етапі. Scrumкоманди зазвичай проводять регулярні зустрічі (standup meetings), де обговорюють прогрес, проблеми й подальші завдання, що підвищує прозорість процесу та швидкість прийняття рішень.

Метод Kanban застосовується для візуалізації процесів - за допомогою спеціальних дощок (Kanban boards) команди можуть відстежувати статус кожного завдання: “заплановано”, “в роботі”, “завершено”. Це дозволяє ефективно розподіляти навантаження, контролювати прогрес і запобігати перевантаженню учасників проєкту.

У сфері ігрової розробки гнучкі підходи виявляються найефективнішими, адже вони дозволяють швидко реагувати на зміни у дизайні, сюжеті або

механіках, а також підтримувати високу залученість команди. Завдяки ітераційному підходу команда може поступово вдосконалювати гру, базуючись на відгуках користувачів або внутрішньому тестуванні, що забезпечує стабільний прогрес і якісний кінцевий результат.

Ключовим завданням управління проєктом є **планування ресурсів, розподіл ролей і координація командної роботи**. У процесі створення гри необхідно враховувати специфіку роботи дизайнерів, програмістів, художників, сценаристів та тестувальників, забезпечуючи узгодженість між їхніми діями. Для цього використовуються спеціальні інструменти управління, такі як **Trello, Jira, Notion або Asana**, які допомагають відстежувати прогрес і строки виконання завдань.

Методологічна база розробки також включає **етапи життєвого циклу програмного забезпечення (SDLC)**: аналіз вимог, проєктування, реалізацію, тестування, розгортання та підтримку. У контексті Unreal Engine 5 ці етапи доповнюються процесами створення ігрових механік, роботи з Blueprints, анімацією, оптимізацією продуктивності та організацією інтерактивних систем.

Таким чином, управління проєктом розробки квестової гри є не лише технічним завданням, а комплексною діяльністю, що поєднує стратегічне планування, системний аналіз і творчий підхід. Використання сучасних методологій управління дозволяє ефективно координувати командну роботу, мінімізувати ризики та забезпечити високу якість кінцевого продукту, який відповідає сучасним вимогам ринку відеоігор.

Висновки до розділу 1

Перший розділ був присвячений теоретичним засадам створення інтерактивного квестового контенту, зокрема аналізу ігрових механік, методів оптимізації, вивченню найпопулярніших квестових проєктів та розгляду основних інструментів сучасних рушіїв, таких як Unreal Engine 5. Представлений

матеріал забезпечує цілісне розуміння ключових принципів, що лежать в основі розробки якісних інтерактивних продуктів.

По-перше, було встановлено, що **ігрові механіки управління та взаємодії** є основою геймплею, оскільки визначають, як гравець впливає на ігровий світ. Інтуїтивність, логічність та глибина цих механік напряму впливають на комфорт користувача та його емоційне залучення. Механіки взаємодії формують основу інтерактивного досвіду, забезпечують розвиток сюжету та стимулюють дослідницьку активність гравця.

По-друге, теоретичний аналіз підходів до **оптимізації ігрового процесу** продемонстрував, що ефективність сучасних ігор залежить не лише від графічної якості, але й від продуктивності. Використання технологій LOD, Nanite, Lumen, оптимізація логіки, фізики, пам’яті та потоків забезпечують стабільність роботи проєкту.

У результаті аналізу **популярних квестових ігор** було визначено низку спільних рис, властивих цьому жанру: акцент на сюжетності, нелінійність подій, глибока емоційна атмосфера, механіки дослідження та розв’язання головоломок.

Досвід таких ігор, як The Stanley Parable, Firewatch та Gone Home, демонструє, що ключовим фактором успішності квестів є створення інтерактивної історії, де вибір гравця справді має значення.

Розгляд **Unreal Engine 5** показав, що цей рушій є сучасним стандартом в індустрії завдяки унікальним технологіям, інструментам для швидкої розробки, потужним засобам візуалізації та підтримці складних сценаріїв взаємодії. Він дозволяє створювати масштабні, реалістичні та інтерактивні ігрові світи, що особливо актуально для квестових ігор із нелінійною структурою.

Окрему увагу приділено **інструментам UE5**, серед яких Blueprints, C++, Material Editor, Niagara, Landscape Tool, Behavior Trees, MetaHuman та інші. Вони забезпечують повний цикл створення гри — від моделювання середовища та персонажів до налаштування анімації, звуку та AI. Поєднання цих інструментів дозволяє створювати високоякісний контент навіть невеликими командами.

РОЗДІЛ 2

ПРОЄКТУВАННЯ ТА ТЕХНІЧНА РЕАЛІЗАЦІЯ ІГРОВИХ МЕХАНІК

2.1. Аналіз та формування технічних вимог до ігрового проєкту

Формування технічних вимог є одним із найважливіших етапів розробки квестової гри, оскільки саме на цьому етапі визначаються ключові параметри, від яких залежить функціональність, стабільність та якість кінцевого продукту. Правильно сформульовані технічні вимоги забезпечують узгодженість між етапами проєктування, розробки, тестування та оптимізації гри. Вони дозволяють створити чітке бачення того, якими засобами та інструментами реалізовуватимуться основні механіки, а також встановлюють критерії оцінки ефективності систем управління та взаємодії.

Аналіз вимог до ігрових механік проводився на основі комплексного вивчення існуючих рішень у сучасних квестових іграх, аналізу користувацьких відгуків та порівняння підходів до організації геймплею в різних проєктах. Для цього було здійснено огляд низки популярних ігор жанру, таких як The Room, Myst та The Walking Dead: The Telltale Series. У процесі дослідження розглядалися основні сценарії взаємодії гравця з ігровим середовищем, способи керування персонажем, структура завдань, логіка квестових ланцюгів та методи подачі сюжету через ігрові механіки.

Крім того, було проведено аналіз технічної реалізації подібних механік у рушії Unreal Engine 5 з використанням офіційної документації, навчальних матеріалів Epic Games, професійних форумів і відкритих прикладів реалізації (Blueprints і C++ API). Такий підхід дав змогу виявити найефективніші методи побудови систем управління, інтерактивних об’єктів, діалогових систем та інвентарю, а також визначити можливі обмеження при використанні стандартних інструментів рушія.

Також проведений аналіз вимог до ігрових механік став одним із ключових етапів у процесі розробки квестової гри, адже саме він дозволив визначити, які елементи управління та взаємодії необхідно реалізувати для створення комфортного та захоплюючого ігрового досвіду. На цьому етапі було проведено оцінку потреб цільової аудиторії, специфіки ігрового процесу та характерних рис жанру квесту. Особлива увага приділялася таким аспектам, як плавність управління персонажем, інтуїтивність взаємодії з предметами та об’єктами, логічність і послідовність головоломок, а також можливість нелінійного розвитку сюжету через вибори гравця.

У результаті були сформовані функціональні та нефункціональні вимоги до системи. Функціональні вимоги визначали конкретні дії та сценарії взаємодії - рух персонажа, використання предметів, виконання завдань і проходження квестових ліній. Нефункціональні вимоги охоплювали такі характеристики, як зручність управління, швидкодія, стабільність роботи системи, відгук на дії гравця та узгодженість із іншими ігровими механіками.

На основі проведеного аналізу було створено структурований опис основних ігрових механік та визначено, які інструменти Unreal Engine 5 доцільно застосувати для їх реалізації. Зокрема, було розглянуто використання системи Blueprints для швидкого створення логіки взаємодії, а також можливості інтеграції з C++ для більш гнучкого керування складними сценаріями.

Підсумком цього етапу стало формування цілісного бачення майбутньої структури гри, визначення пріоритетів розробки та послідовності реалізації механік. Проведений аналіз дав змогу заздалегідь передбачити можливі труднощі під час тестування, оптимізувати процес проєктування та забезпечити логічну взаємодію між усіма складовими ігрового процесу. Такий підхід дозволив створити основу для збалансованої, динамічної та зрозумілої системи, що відповідає сучасним стандартам якості квестових ігор і сприяє глибокому зануренню гравця у віртуальний світ.

У межах даного етапу розробки було також визначено методологічну основу дослідження, яка поєднує елементи **системного, аналітичного та**

експериментального підходів. Системний підхід дозволив розглядати процес створення квестової гри як цілісну структуру, де всі компоненти - графіка, механіки, логіка взаємодії, інтерфейс і користувацький досвід - взаємопов'язані між собою. Аналітичний метод застосовувався для дослідження існуючих ігрових рішень, аналізу їхніх переваг і недоліків, а також для формування власної концепції розробки. Експериментальний метод полягав у практичному тестуванні створених прототипів ігрових механік, перевірці їхньої ефективності в умовах реального рушія Unreal Engine 5 та адаптації під потреби проєкту.

У рамках методології також використовувався **евристичний підхід**, який дав змогу оцінити інтуїтивність управління та зручність користувацької взаємодії. На основі спостережень, тестування і відгуків користувачів проводилася корекція ігрових механік, інтерфейсу та поведінки персонажа. Такий комплексний підхід до формування вимог і реалізації механік забезпечив баланс між технічними характеристиками, креативною складовою та комфортом гравця, що є визначальним чинником для створення якісного ігрового продукту.

2.2. Вибір інструментів та налаштування середовища розробки

Етап вибору інструментів та налаштування середовища розробки є одним із ключових у процесі створення квестової гри, адже саме від нього залежить ефективність роботи команди, зручність реалізації ігрових механік та загальна якість кінцевого продукту. На цьому етапі визначаються основні програмні засоби, рушій, додаткові плагіни, бібліотеки та утиліти, необхідні для розробки, тестування й оптимізації гри. Також здійснюється конфігурація середовища розробки, підготовка робочих проєктів, встановлення необхідних налаштувань для забезпечення сумісності між компонентами системи та стабільності роботи під час усього циклу розробки.

Для реалізації проєкту було обрано комплекс інструментів, який забезпечує повний цикл створення інтерактивного 3Dсередовища та управління ігровими механіками. Основним рушієм став **Unreal Engine 5**, який надає широкий спектр

можливостей для розробки як графічно насичених, так і інтерактивних проєктів. Цей рушій підтримує роботу з високополігональними моделями, динамічним освітленням, системами фізики та анімації, а також дозволяє легко реалізовувати інтерактивність без глибокого програмування. [19]

Для створення логіки поведінки персонажів, об’єктів та механік взаємодії використовувався Blueprint - візуальна система програмування Unreal Engine. Вона дозволяє швидко розробляти складні сценарії та механіки, використовуючи візуальні вузли, що спрощує тестування та налагодження без написання коду.

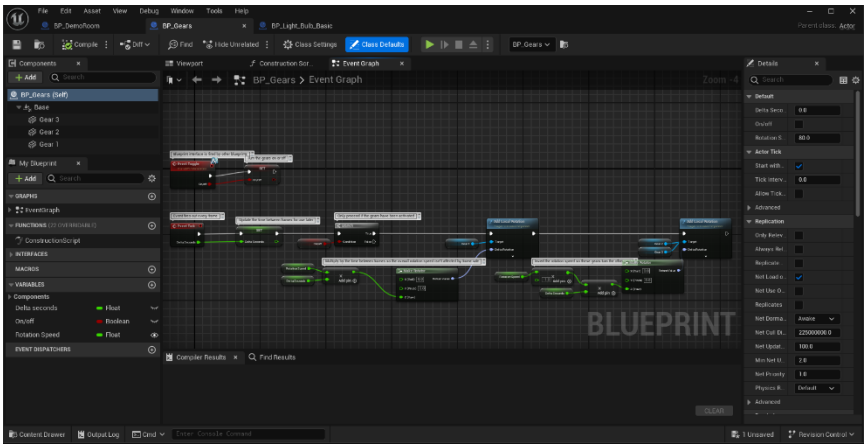

Рис.2.1. Приклад Blueprint в Unreal Engine 5
Джерело: [Автор]

Для підготовки 3Dмоделей, текстур та інших ресурсів застосовувався Blender, який забезпечує моделювання, UVрозгортку та базову анімацію персонажів і об’єктів середовища. Також використовувалися спеціалізовані редактори текстур, що дозволяють створювати матеріали високої якості, карти нормалей, відблиски та інші ефекти, необхідні для реалістичного відображення об’єктів у рушії. [19]

Налаштування робочого середовища включало організацію структури проєкту в Unreal Engine 5 з поділом на папки для рівнів, матеріалів, моделей та Blueprints, що забезпечувало зручність у роботі та легкість масштабування. Важливим аспектом було налаштування параметрів рендерингу, освітлення та

системи вводу, що дозволило протестувати механіки в режимі Play in Editor та швидко вносити зміни.

Таке поєднання інструментів і налаштувань середовища забезпечує ефективну та гнучку розробку, дозволяючи одночасно працювати над графікою, механіками та інтерфейсом без втрати продуктивності та якості проєкту.

Вибір такого комплексу інструментів обґрунтований його широкими можливостями та гнучкістю, що дозволяє адаптувати процес розробки під конкретні потреби проєкту. Поєднання Unreal Engine 5, Blueprint та Blender забезпечує повний контроль над усіма аспектами створення ігрового світу - від графіки до поведінки персонажів і логіки взаємодії з об’єктами.

Налаштоване робоче середовище створює зручну платформу для роботи як над окремими елементами, так і над проєктом в цілому. Структуроване розміщення ресурсів, продумана організація папок і налаштування параметрів рендерингу та вводу сприяють швидкому тестуванню та внесенню змін, що особливо важливо на ранніх етапах розробки.

Також слід зазначити, що використання сучасних інструментів дозволяє легко масштабувати проєкт, додавати нові об’єкти та механіки, не порушуючи існуючу логіку. Це забезпечує безперервний розвиток проєкту, створюючи міцну основу для подальшого вдосконалення графіки, інтерактивності та користувацького досвіду.

Завдяки такому підходу до організації робочого процесу та підбору інструментів, розробка проєкту відбувається ефективно, системно та з можливістю плавного розширення функціоналу, що є ключовим фактором у створенні якісного та захоплюючого інтерактивного середовища.

Важливо відзначити, що обрані інструменти не лише забезпечують технічну реалізацію, але й створюють комфортні умови для роботи розробника. Інтуїтивно зрозумілий інтерфейс Unreal Engine 5, гнучка система Blueprint та доступність великої кількості навчальних матеріалів і прикладів дозволяють швидко освоювати нові підходи та ефективно впроваджувати їх у проєкт.

Крім того, використання сучасного робочого середовища сприяє безперервному вдосконаленню процесу розробки: можна легко тестувати різні сценарії, експериментувати з механіками та швидко оцінювати результат у режимі реального часу. Такий підхід дозволяє не лише створювати функціональні прототипи, а й поступово переходити до більш складних елементів гри без необхідності кардинальної перебудови проєкту.

Перспективою розвитку проєкту є інтеграція додаткових систем, таких як складні анімації персонажів, продвинута фізика об’єктів, покращене освітлення та ефекти часток, а також можливість розширення UI і взаємодії з користувачем. Це створює основу для реалізації масштабного інтерактивного середовища, що поєднує високоякісну графіку, реалістичні механіки та зручний інтерфейс, забезпечуючи приємний і захоплюючий досвід для користувачів.

Окремої уваги заслуговує питання оптимізації та підтримки продуктивності під час розробки. Уже на початкових етапах проєкту було закладено принципи балансування між якістю графіки та швидкодією рушія, що дозволяє уникнути перевантаження системи й забезпечити стабільну роботу навіть на середніх конфігураціях обладнання. Використання рівнів деталізації моделей (LOD), потокового завантаження ресурсів та продуманих параметрів освітлення дає можливість досягати високої ефективності без втрати візуальної привабливості. Такий підхід створює умови не лише для коректного функціонування середовища на різних платформах, а й для його подальшої адаптації до VR/ARтехнологій, що відкриває нові перспективи розвитку проєкту.[21]

Не менш значущим аспектом стало налаштування інструментів для тестування та профілювання продуктивності, що дозволяє виявляти вузькі місця, оптимізувати ресурси та підтримувати стабільний FPS у всіх сценах. Використання вбудованих засобів Unreal Engine 5 для моніторингу пам’яті, завантаження об’єктів і продуктивності забезпечує своєчасне виявлення проблем і дозволяє оперативно вносити корективи.

Важливо також відзначити підготовку документації та шаблонів, які спрощують подальшу роботу над проєктом. Стандартизовані підходи до

створення матеріалів, організації Blueprints та ієрархії сцен дозволяють новим учасникам команди швидко адаптуватися та підтримувати єдину структуру проєкту.

Таким чином, вибір інструментів і налаштування середовища розробки не обмежується лише технічними аспектами: він формує ефективну робочу екосистему, яка забезпечує швидку розробку, високу якість, зручну командну взаємодію та гнучкість для подальшого масштабування проєкту.

2.3. Конфігурація середовища розробки

Конфігурація середовища розробки є одним із ключових етапів підготовки до створення ігрового проєкту, оскільки від правильності її виконання залежить стабільність, продуктивність і зручність подальшої роботи команди. На цьому етапі здійснюється налаштування програмного забезпечення, встановлення рушія Unreal Engine 5, підключення необхідних плагінів, модулів і бібліотек, а також оптимізація параметрів системи для роботи з 3Dграфікою, анімацією та звуком.

Особлива увага приділяється сумісності компонентів, налаштуванню середовища під технічні вимоги проєкту та організації структури файлів, що забезпечує ефективну співпрацю між розробниками. Коректна конфігурація дозволяє мінімізувати ризик технічних збоїв, полегшити інтеграцію ігрових ресурсів, скоротити час на налагодження та створює стабільну основу для реалізації функціональних і візуальних елементів майбутньої гри.

На етапі конфігурації середовища розробки було здійснено налаштування основних програмних інструментів та параметрів, необхідних для ефективної реалізації проєкту. Базовим рушієм виступив Unreal Engine 5, для якого було створено структуровану проєктну папку з розподілом ресурсів за категоріями: рівні, моделі, матеріали, текстури, анімації та Blueprints. Такий підхід забезпечив впорядкованість даних і спростив командну роботу.

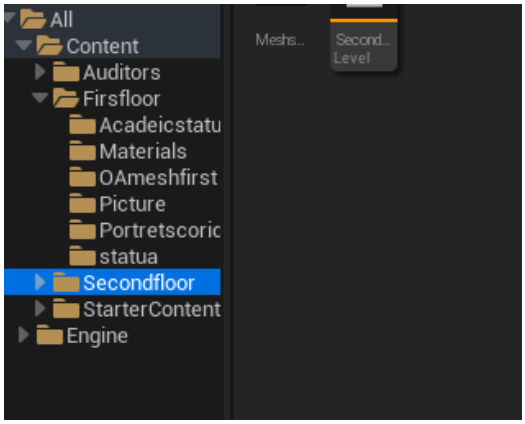

Рис.2.2. Структурована проєктна папка з розподілом ресурсів за категоріями
Джерело: [Автор]

Також було приділено увагу для **налаштування рендерингу та освітлення**, що дозволило досягти балансу між якістю графіки та продуктивністю рушія. Було визначено параметри глобального освітлення, тіней та відблисків, а також оптимізовано роботу зі складними сценами завдяки використанню **рівнів деталізації моделей (LOD)** та **потокового завантаження контенту**.

Окремо варто зазначити, що система освітлення та відображень має значний вплив на загальну продуктивність гри. Навіть незначні зміни у кількості джерел світла чи параметрах тіней можуть суттєво знизити частоту кадрів або збільшити навантаження на графічний процесор. Тому під час розробки особливу увагу було приділено пошуку оптимального співвідношення між якістю візуалізації та швидкодією - це дозволило зберегти привабливу графіку без втрати плавності ігрового процесу.

Для створення та редагування 3Dресурсів застосовувався **Blender**, де виконувалося моделювання об’єктів, UVрозгортка та базова анімація. Отримані ресурси інтегрувалися у рушій із дотриманням узгоджених форматів і стандартів. Додатково використовувалися спеціалізовані редактори текстур для підготовки матеріалів високої якості.

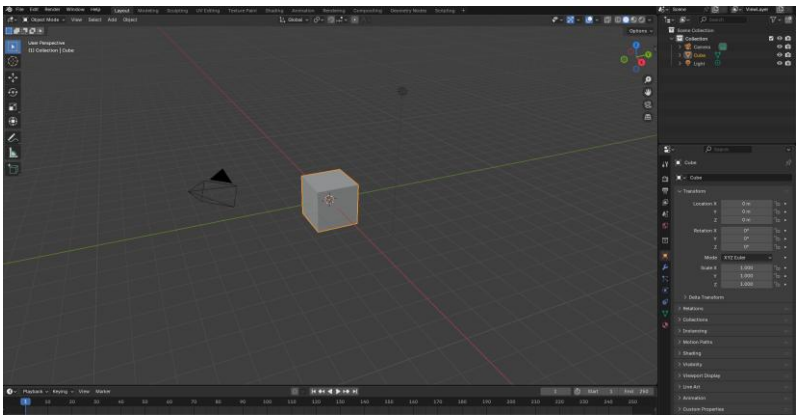

Рис.2.3. Налаштування 3д редактора Blender
Джерело:[Автор]

Налаштування середовища включало також конфігурацію Blueprintсистеми Unreal Engine, що дозволило швидко реалізовувати та тестувати ігрові механіки без потреби у традиційному кодуванні. Використання режиму Play in Editor забезпечило можливість оперативної перевірки результатів та внесення змін у режимі реального часу.

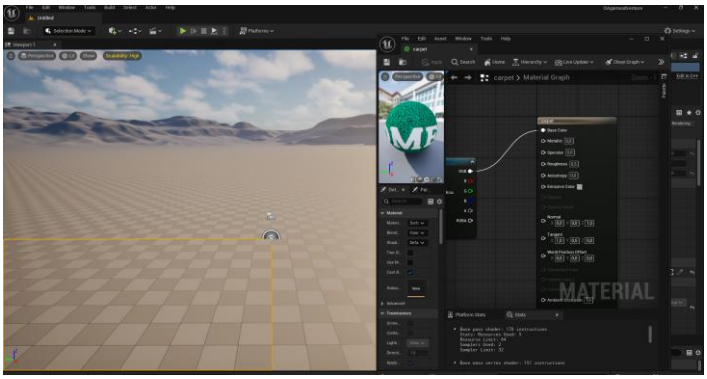

Рис.2.4. Конфігурація Blueprintсистеми Unreal Engine
Джерело:[Автор]

Для обробки та підготовки графічних елементів у проєкті було використано професійні інструменти Adobe Illustrator та Adobe Photoshop. **Illustrator** застосовувався для створення та редагування векторної графіки - логотипів, іконок, елементів інтерфейсу, контурів об’єктів і декоративних деталей, що вимагали чітких ліній і масштабованості без втрати якості. **Photoshop**, у свою чергу, використовувався для опрацювання растрових зображень - текстур, фонових зображень, світлових карт і візуальних ефектів.

Поєднання цих двох програм дозволило досягти високої якості візуальних матеріалів, забезпечити узгодженість стилю та кольорової гами всіх елементів

проєкту. Завдяки цьому було створено цілісну й естетично привабливу графічну складову, яка гармонійно інтегрується у візуальне середовище розробленої сцени.

Конфігурація середовища розробки створила цілісну та гнучку платформу для інтеграції всіх компонентів проєкту, забезпечивши стабільність роботи, зручність масштабування та можливість ефективної командної взаємодії.

Окрім базових налаштувань, було впроваджено додаткові механізми оптимізації та контролю якості роботи середовища. Зокрема, здійснено налаштування системи управління ресурсами, що дозволяє відстежувати завантаження об’єктів та забезпечує ефективне використання оперативної пам’яті. Було також впроваджено стандарти оформлення та іменування файлів, що спрощує навігацію по проєкту та знижує ризик помилок під час інтеграції нових елементів. Завдяки цим заходам процес розробки став більш організованим, дозволяючи команді швидко реагувати на зміни, тестувати нові рішення та підтримувати високу якість кінцевого продукту.

Крім технічних налаштувань, увага приділялася й організації робочого процесу для забезпечення зручності командної роботи та прозорості виконання завдань. Було впроваджено систему ведення версій та резервного збереження проєктних файлів, що дозволяє відстежувати зміни, швидко відновлювати попередні стані ресурсів і уникати втрати даних.

Також розроблено стандартизовані шаблони для створення рівнів, матеріалів та Blueprints, що забезпечує єдність стилю та логіки в проєкті, а також скорочує час на інтеграцію нових елементів. Для підвищення ефективності тестування було налаштовано автоматизовані контрольні сцени, які дозволяють перевіряти роботу механік та взаємодію об’єктів у різних умовах без потреби повторного ручного тестування.

Важливою складовою конфігурації стало налаштування системи моніторингу продуктивності та оптимізації ресурсів у режимі реального часу. Це дозволяє виявляти вузькі місця в роботі рушія, коригувати параметри освітлення

та складності сцен і забезпечує стабільну частоту кадрів навіть у великих та насичених деталями локаціях.

Завдяки поєднанню цих технічних і організаційних заходів середовище розробки стало не лише функціональним і стабільним, але й гнучким для масштабування. Воно дозволяє легко додавати нові механіки, інтегрувати сторонні ресурси, впроваджувати складні сценарії взаємодії та забезпечує високий рівень контролю над усіма аспектами проєкту, що є критично важливим для створення якісного та інтерактивного ігрового продукту.

2.4. Визначення концепції управління та взаємодій

Розробка механік управління та взаємодії в квестовій грі на Unreal Engine - це складний та захоплюючий процес, що вимагає глибокого розуміння фізичних, логічних та емоційних аспектів взаємодії в ігровому світі. Перш за все, важливо створити фізичні взаємодії, які забезпечують правдоподібність і реалістичність. Завдяки фізичним движкам Unreal Engine, таким як Chaos Physics чи PhysX, можна реалізувати рух об'єктів, їх зіткнення і навіть руйнування. Проте важливо не тільки реалізувати ці механіки, а й забезпечити плавність руху персонажа, його взаємодію з навколишнім середовищем, наприклад, підйом предметів або відкриття дверей.

Логічні взаємодії створюють важливий елемент гри, де кожен вибір гравця або взаємодія з навколишнім світом має свої наслідки. Від алгоритмів поведінки NPC до механік головоломок, кожен елемент повинен мати логічну основу, що дозволяє гравцеві вирішувати проблеми або просуватися в сюжеті. Використання Blueprint для створення складних логічних механізмів, де рішення можуть впливати на розвиток сюжету, додає грі інтерактивності та глибини.

Що стосується управління проектом, важливо не лише визначити ці механіки, але й ретельно спланувати процес їх розробки. Це включає у себе розробку детальних етапів, від планування до тестування. Необхідно також забезпечити команду відповідними інструментами та ресурсами, щоб кожен

аспект гри - від фізики до AI - був реалізований без перешкод. Розробка має бути ітераційною, з постійними тестами та зворотнім зв'язком для покращення механік.

Загалом, процес створення механік для квестової гри на Unreal Engine - це баланс між технічними можливостями та творчим підходом до створення унікального ігрового досвіду. Врахування фізичних, логічних та емоційних аспектів взаємодії дозволяє створити захоплюючий світ, в якому гравець може не тільки вирішувати головоломки і виконувати завдання, але й переживати справжні емоції, взаємодіючи з персонажами та навколишнім середовищем [19].

Визначення концепції взаємодій у грі стали одним із ключових етапів створення проєкту, оскільки саме вони забезпечують гравцеві можливість активно впливати на ігровий світ, взаємодіяти з об’єктами, персонажами та елементами середовища. На цьому етапі було визначено основні типи взаємодій, необхідних для квестової гри: огляд і використання предметів, активація об’єктів, збір колекційних елементів, ведення діалогів з NPC, а також запуск подій за допомогою тригерів.

Для реалізації цих механік було вибрано інструменти Unreal Engine 5, зокрема систему Blueprints, яка дає змогу створити гнучку логіку без надмірного використання коду. Взаємодії було визначено через систему інтерфейсів та подій, що дозволило забезпечити універсальність - будьякий об’єкт у грі може стати інтерактивним, якщо реалізує відповідний інтерфейс.

Було визначено кілька типів поведінки для різних сценаріїв:

- **Логіка персонажа** ходьба, рухи , анімації ;
- **Взаємодія з об’єктами середовища** - відкриття дверей, натискання кнопок, перемикання важелів;
- **Інспекція предметів** - можливість розглядати об’єкти у 3D, обертати їх і знаходити приховані деталі;
- **Система збору** - інтегрована логіка підбору предметів з подальшим додаванням їх до інвентарю;

- **Діалоги з NPC** - створено базову діалогову систему з варіантами вибору, що впливають на розвиток подій;
- **Тригери подій** - зони, що активують сюжетні сцени, зміну освітлення або поява нових завдань.

У межах реалізації проєкту було визначено та впроваджено комплексну систему взаємодій гравця з об’єктами, персонажами та сюжетом гри. Однією з основних складових є **Actors**, що забезпечують різні типи взаємодії з об’єктами середовища. До них відносяться **Simple focus actors**, які дозволяють гравцеві виділити об’єкт фокусом або підсвітити його під час перегляду, та **Inspection actors 2D/3D**, що надають можливість детально оглядати предмети, обертати їх, пересувати та виявляти приховані елементи.

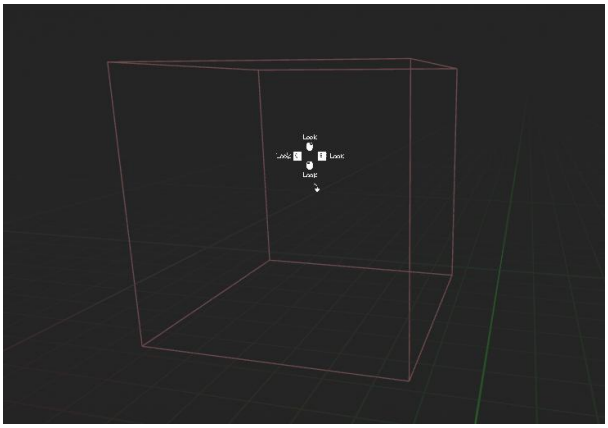

Рис.2.5. Inspection actors 2D/3D в Unreal Engine 5
Джерело:[Автор]

Також створення **системи інвентарю та колекціювання** дозволяє гравцеві збирати предмети, зберігати їх у власному інвентарі та використовувати у взаємодії з іншими об’єктами чи NPC.

Для створення динамічного ігрового середовища інтегровано **Dynamic sequence trigger actors** - об’єкти чи зони, що активують різні події під час проходження гравцем. Вони здатні запускати анімації, змінювати освітлення, аудіоефекти, переміщувати об’єкти або ініціювати діалогові блоки, роблячи світ гри більш живим і непередбачуваним.

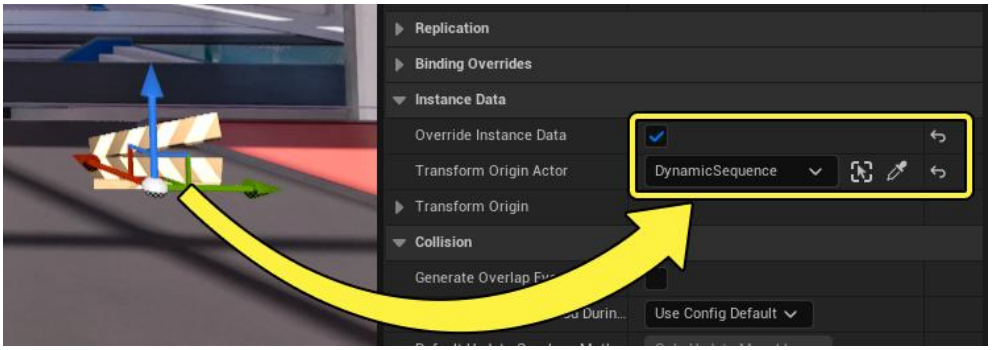

Рис. 2.6. Dynamic sequence trigger actors в Unreal Engine 5
Джерело: [Автор]

Особлива увага приділена **інтерактивним NPC**, які мають гілки діалогів і варіанти вибору, здатні змінювати розвиток сюжету та стан гри. Це дозволяє створити відчуття нелінійності та активної участі гравця у світі гри.

Таким чином, для реалізації було визначено кілька типів взаємодій: взаємодія з об’єктами середовища (відкриття дверей, натискання кнопок, перемикання важелів), детальна інспекція предметів у 3D, система збору та додавання предметів до інвентарю, базова діалогова взаємодія з NPC, а також тригери подій, що активують сюжетні сцени, змінюють освітлення або ініціюють нові завдання. Усі ці механіки забезпечують гравцю широкий спектр дій і роблять ігровий світ інтерактивним та насиченим.

Крім того, було налаштовано систему візуального підсвічування інтерактивних об’єктів, що допомагає гравцю орієнтуватися у просторі й інтуїтивно розуміти, з чим можна взаємодіяти.

У результаті реалізовані взаємодії створюють відчуття живого, логічно зв’язаного світу, де кожен об’єкт має своє призначення, а дії гравця впливають на подальший розвиток подій. Це забезпечує високу залученість користувача та відповідає ключовим принципам побудови квестових ігор, орієнтованих на дослідження, прийняття рішень і поступове розкриття сюжету.

Для ефективної реалізації проєкту зі створення квестової гри було застосовано сучасні **методології управління проєктами**, що дозволили організувати процес розробки на всіх його етапах - від концептуальної ідеї до створення прототипу з інтерактивними механіками. Основною обраною

методологією став **Agileпідхід**, орієнтований на гнучкість, швидке реагування на зміни та безперервну взаємодію між членами команди.

У межах реалізації цього підходу проєкт було поділено на **ітерації (спринти)**, кожен із яких мав чітко визначені цілі - від розробки окремих механік взаємодії до інтеграції діалогової системи або налаштування інтерфейсу. Після завершення кожного етапу проводилося внутрішнє тестування та оцінювання результатів із подальшим коригуванням плану робіт. Це дозволяло вчасно виявляти помилки, вдосконалювати структуру гри та поступово розширювати функціональність без втрати стабільності системи.

Для організації процесу роботи використовувалися **інструменти візуального контролю та комунікації**, такі як **Trello** або **Notion**, які забезпечували прозорий розподіл завдань і можливість відстеження прогресу у режимі реального часу. Завдяки цьому кожен елемент гри - від логіки персонажа до поведінки NPC - мав власний етап розробки, перевірки та документування.

Також було застосовано принципи **Scrum**, які передбачають постійні короткі наради для обговорення досягнутих результатів, поточних проблем і планів на наступні ітерації. Такий підхід дозволив зберігати стабільний темп роботи та підтримувати командну комунікацію на високому рівні, що особливо важливо при роботі з інтерактивними ігровими механіками, які часто потребують тестування та узгодження між різними фахівцями (дизайнерами, програмістами, художниками).

З методологічної точки зору, управління проєктом ґрунтувалося на поєднанні **гнучких підходів (Agile/Scrum)** та **елементів класичного каскадного моделювання (Waterfall)** на початкових етапах, де необхідно було чітко зафіксувати технічні вимоги, розподіл ресурсів і побудову загальної структури гри. Це поєднання дозволило досягти балансу між стабільністю планування і гнучкістю в прийнятті рішень, що є важливою складовою в сучасних ігрових проєктах.

Таким чином, розробка квестової гри в Unreal Engine 5 базувалася не лише на технічній і креативній складовій, але й на системному підході до управління

проєктом. Застосування сучасних методологій забезпечило ефективну координацію роботи, своєчасне виконання завдань і стабільне вдосконалення гри на кожному етапі її створення, що в підсумку сприяло формуванню якісного, логічно зв’язаного та інтерактивного продукту.

2.5. Технічна реалізація логіки персонажа

На цьому етапі концептуальні ідеї, які визначені під час проєктування, перетворюються на конкретні реалізації засобами рушія Unreal Engine 5 із використанням Blueprints.

Використання Blueprints у Unreal Engine 5 надає розробникам значну свободу і зручність. Візуальне програмування дозволяє створювати логіку гри наочно, бачити всі взаємозв’язки між елементами та одразу розуміти, як вони взаємодіють, без необхідності писати складний код. Це значно прискорює процес розробки і спрощує налагодження механік, адже будьякі зміни можна відразу перевіряти в режимі реального часу. Blueprints також забезпечують гнучкість і масштабованість, дозволяючи легко додавати нові механіки, модифікувати існуючі системи та адаптувати інтерфейс і поведінку об’єктів під потреби гри та гравця. Завдяки цьому процес розробки стає більш ефективним, а результати - зрозумілими і доступними для подальшого вдосконалення.

Під час роботи над проєктом я зосередився на створенні логіки персонажа, системи навігації, управління камерою та обробці подій, які формують основу ігрового процесу. Було розроблено поведінку персонажа, що реагує на дії користувача - пересування, стрибки, взаємодію з об’єктами та іншими персонажами.

Перш за все, варто зазначити, що сам ігровий рушій уже містить готові шаблони та приклади реалізації базової логіки персонажа, зокрема руху, стрибків, взаємодії з об’єктами та керування камерою. Проте я використав один із таких шаблонів лише як відправну точку для подальшої розробки та вдосконалення. Це рішення було зумовлено тим, що стандартні варіанти, надані

рушієм, хоч і є функціональними, але не забезпечують повного набору можливостей, необхідних для реалізації специфічних механік і квестових елементів мого проєкту.

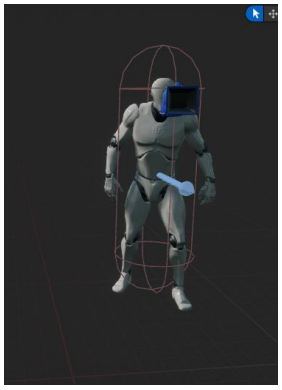

Рис. 2.7. Режим налаштування персонажа в Unreal Engine 5
Джерело:[Автор]

У процесі роботи логіка персонажа була суттєво розширена та доопрацьована порівняно з базовим шаблоном Unreal Engine 5. Для реалізації цього було використано **Character Blueprint**, систему **Animation Blueprint** та **State Machine**, що дозволило створити більш гнучку і динамічну модель поведінки персонажа.

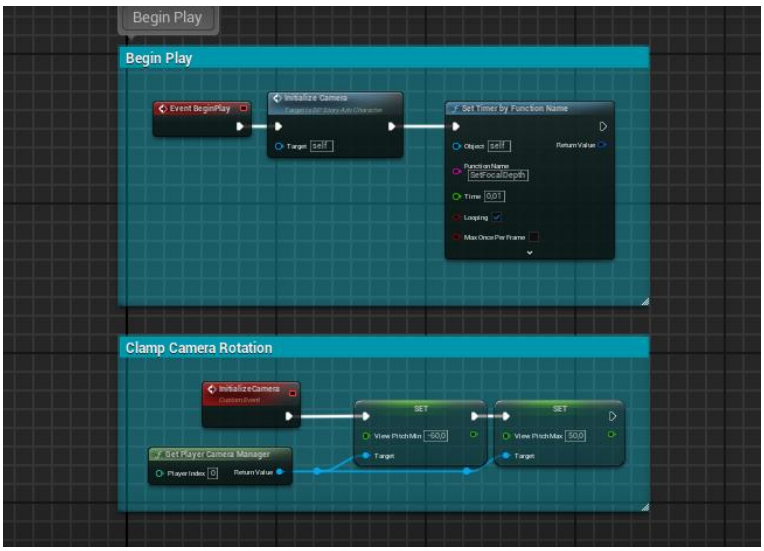

Рис. 2.8. Ініціалізація та обмеження обертання камери в Unreal Engine 5
Джерело:[Автор]

Логіка реалізації персонажа у проєкті базується на принципах інтерактивної поведінки та modular character system, що забезпечує гнучкість і керованість усіх дій.

Основу складає **Character Blueprint**, у якому визначено всі ключові компоненти - **Capsule Component** для колізій, **Skeletal Mesh** для візуалізації моделі, **Camera** для формування перспективи та **Movement Component** для керування пересуванням.

Рух персонажа реалізовано через **систему вводу (Input System)**, яка обробляє команди від користувача (натискання клавіш або кнопок геймпада) і перетворює їх на дії - ходьбу, біг, стрибок або взаємодію з об’єктами. Для більш природної поведінки застосовано **Animation Blueprint**, який відповідає за плавний перехід між анімаціями залежно від стану персонажа (рух, бездіяльність, взаємодія).

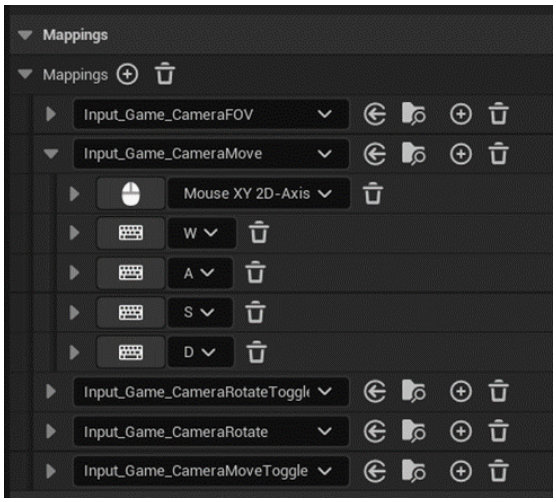

Рис. 2.9. Налаштування Input System в Unreal Engine 5
Джерело:[Автор]

Крім того, логіка передбачає **систему взаємодії з оточенням**, що використовує трасування (Line Trace) для визначення об’єктів, з якими персонаж може взаємодіяти. Під час виявлення інтерактивного елемента гравцю подається відповідний інтерфейсний сигнал, а при активації - виконується відповідна дія (відкриття дверей, запуск діалогу, підбір предмета тощо).

Усі ці елементи працюють у комплексі, утворюючи **цілісну логіку поведінки персонажа**, що забезпечує реалістичність, чутливість до дій користувача та інтерактивність ігрового процесу.

До стандартних параметрів руху - таких як ходьба, біг і стрибок - було додано низку додаткових станів, зокрема **присідання**, **повільний рух у режимі стелс**, **взаємодію з об'єктами** та **анімацію використання предметів із інвентарю**. Для забезпечення плавності переходів між станами використано **Blend Space 1D/2D**, що дало змогу синхронізувати швидкість і напрямок руху з відповідними анімаціями.

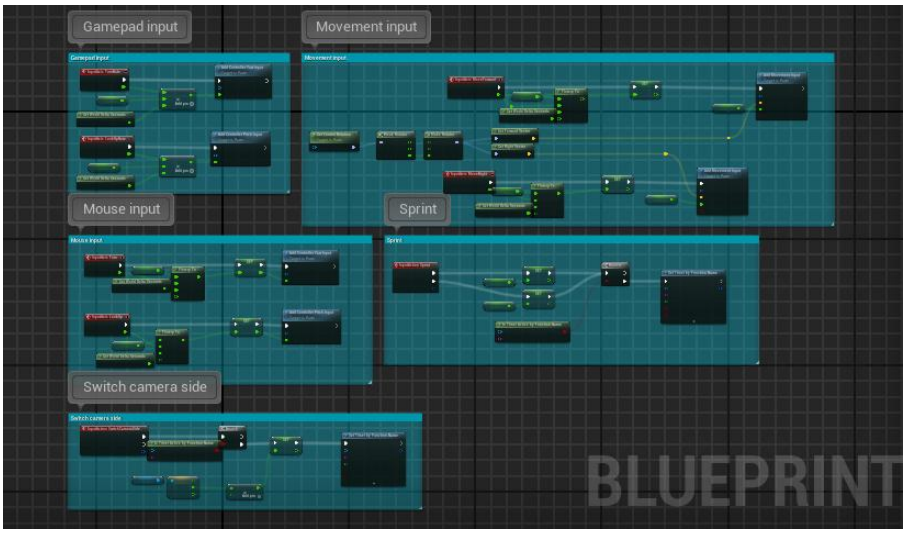

Рис. 2.10. Стандартні параметри руху в Unreal Engine 5
Джерело:[Автор]

Система відгуку на дії гравця була покращена шляхом інтеграції **Enhanced Input System**, що дозволяє гнучко налаштовувати клавіші, створювати комбінації дій і точно відслідковувати тривалість натискання. Завдяки цьому вдалося підвищити чутливість керування та покращити відгук персонажа на дії користувача.

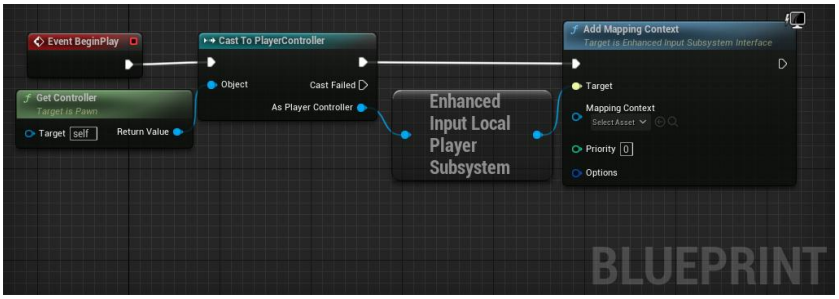

Рис. 2.11. Інтеграція Enhanced Input System в Unreal Engine 5
Джерело:[Автор]

Для взаємодії з оточенням було реалізовано механізми **Trace (LineTrace by Channel)**, які визначають наявність об'єкта перед персонажем і дозволяють ініціювати взаємодію. У комбінації з **Trigger Box** та інтерфейсами взаємодії це дало змогу реалізувати широкий спектр дій - від відкриття дверей до активації складних івентів чи квестових сцен.

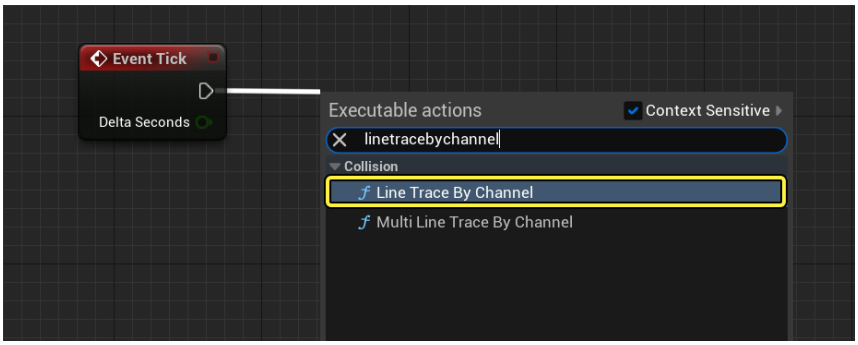

Рис. 2.12. Механізм Trace (LineTrace by Channel) в Unreal Engine 5
Джерело:[Автор]

У результаті базовий шаблон персонажа було перетворено на повноцінну, адаптивну систему управління, яка органічно поєднується з іншими ігровими механіками. Такий підхід забезпечив не лише реалістичну поведінку персонажа, а й високу гнучкість у налаштуванні, що дало змогу легко розширювати функціональність під потреби конкретних ігрових сценаріїв.

У процесі розробки проєкту значну увагу було приділено інтеграції анімаційних систем та оптимізації продуктивності, оскільки ці аспекти безпосередньо впливають на якість візуального сприйняття і плавність ігрового процесу.

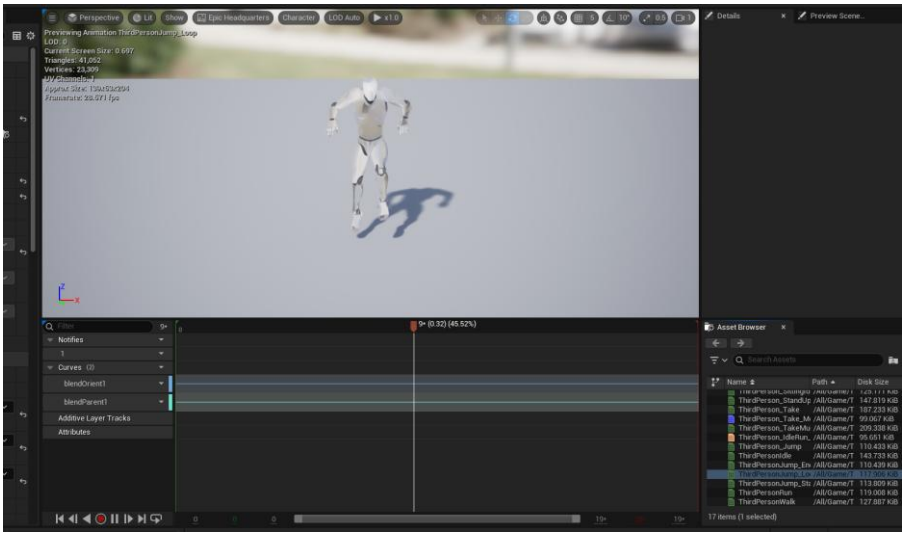

Рис. 2.13. Налаштування анімацій в Unreal Engine 5
Джерело:[Автор]

Під час створення анімаційного блоку використовувалися інструменти **Animation Blueprint** та **State Machine**, що дали змогу гнучко керувати змінами станів персонажа - від простої ходьби до складних комбінацій рухів, таких як взаємодія з предметами, відкриття дверей чи виконання сюжетних дій. Було реалізовано логіку переходів між станами з урахуванням контексту, швидкості руху, напрямку та взаємодії з оточенням.

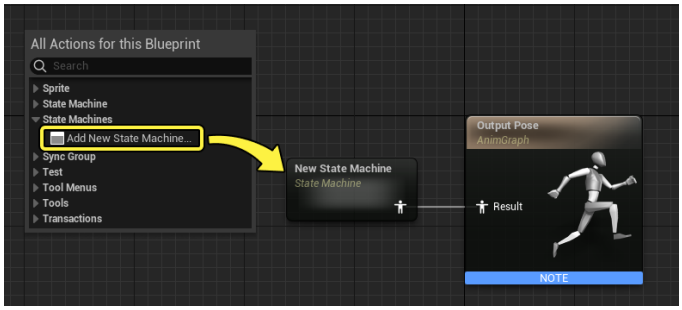

Рис. 2.14. Bluprint State Machine в Unreal Engine 5
Джерело:[Автор]

Для створення більш реалістичних рухів персонажа використовувалися **Blend Spaces**, які забезпечують плавні переходи між різними типами анімацій залежно від швидкості та напрямку руху. Це дозволило уникнути різких змін поз і зробити рухи природними. Додатково були застосовані **Root Motion**анімації

для більш точного контролю переміщення персонажа у просторі, що підвищило рівень реалістичності рухів і зменшило кількість помилок позиціонування.

Також варто зазначити, що під час реалізації вигляду **від першої особи (FirstPerson View)** до анімаційного опрацювання персонажа можна не приділяти надмірної уваги. Це зумовлено тим, що більшість рухів і анімаційних переходів залишаються **поза полем зору гравця**, адже камера розташована безпосередньо від імені персонажа. Такий підхід широко використовується у відомих студіях розробки ігор, зокрема при створенні FPSпроєктів, де ключовим аспектом є **плавність керування, реалістичність руху рук і зброї**, а не повна деталізація всього тіла персонажа.

Застосування цієї практики має низку переваг:

- **Зниження навантаження на систему**, оскільки не потрібно опрацьовувати та відтворювати всі анімаційні стани повного тіла.
- **Суттєве спрощення процесу розробки**, адже створення, налаштування й оптимізація анімацій займає значно менше часу.
- **Підвищення продуктивності рушія**, що особливо важливо під час роботи зі складними сценами або великою кількістю персонажів.

Такий варіант реалізації дозволяє **зосередитися на геймплеї та інтерактивних елементах**, забезпечуючи при цьому достатній рівень візуальної достовірності з точки зору гравця. Це оптимальний компроміс між якістю, продуктивністю та ефективністю розробки, який часто використовується навіть у комерційних ігрових проєктах. (кібер панк додати до додатків)

Після розробки цих елементів було проведено **оптимізацію продуктивності**. Зокрема:

- виконано зменшення кількості полігонів у допоміжних об’єктах середовища;
- застосовано **Level of Detail (LOD)** для моделей і текстур, що дозволило зменшити навантаження на GPU;
- проведено налаштування **cullingсистеми**, щоб об’єкти, не видимі з камери, не рендерилися;

- оптимізовано **анімаційні графи**, скорочено надмірні переходи між станами;
- проведено профілювання за допомогою **Unreal Insights**, щоб визначити “вузькі місця” у продуктивності.

Також було оптимізовано роботу зі світлом та тінями - частина джерел освітлення переведена у **static** або **baked** формат, що дозволило зменшити обчислювальні витрати під час рендерингу.

У результаті вдалося досягти стабільного фреймрейту без втрати якості графіки, зберігши при цьому плавність рухів персонажа та природність взаємодії з оточенням. Цей етап став важливою частиною технічної реалізації, оскільки поєднав естетичну складову з ефективністю виконання, забезпечуючи комфортний і приємний користувацький досвід.

2.6. Розробка механіки взаємодії з об'єктами та іншими персонажами

Розробка механіки взаємодії з об'єктами та іншими персонажами - це ключовий етап у створенні гри, адже саме через цю механіку гравець взаємодіє з навколишнім світом і персонажами. Важливо, щоб усі ці взаємодії були інтуїтивно зрозумілими та логічними, що сприятиме поглибленому зануренню в гру і забезпечить цікавий і захоплюючий ігровий процес.

У процесі розробки було реалізовано **систему взаємодії з об’єктами та персонажами**, яка стала одним із ключових елементів ігрової логіки. Вона забезпечує глибше занурення у квестовий процес та робить світ більш “живим” і чутливим до дій гравця.

Основою для побудови системи стала **Trigger Boxсистема**, за допомогою якої визначається зона взаємодії між персонажем та об’єктом. При вході гравця в цю зону активується **Blueprintлогіка**, що перевіряє тип об’єкта, його стан та доступні дії. Для візуалізації взаємодії використано **UI Widgets**, які динамічно відображають підказки (наприклад: *“Натисніть E, щоб відкрити”* або *“Використати предмет”*).

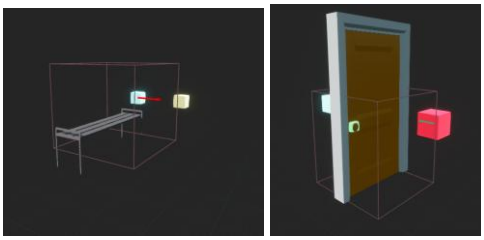

Рис. 2.15. Trigger Box об’єкти в Unreal Engine 5
Джерело:[Автор]

Також були налаштовані **колізійні компоненти (Collision Components)**, які визначають, коли саме персонаж може взаємодіяти з об’єктом, а коли ні. Для кожного типу об’єкта створено власний Blueprint зі змінними параметрами (активний/неактивний стан, одноразова або повторна взаємодія, умови виконання квестів).

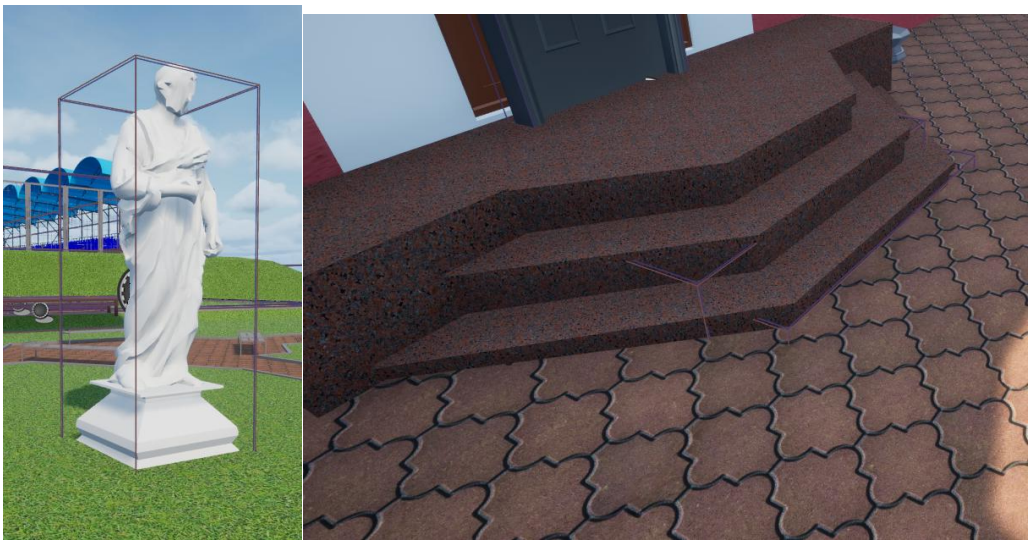

Рис. 2.16. Колізійні компоненти в Unreal Engine 5
Джерело:[Автор]

Розробка механіки взаємодії з об’єктами оточення

У процесі розробки гри було створено **механіку взаємодії з об’єктами оточення**, яка дозволяє гравцю виконувати різні дії - підбирати предмети, відкривати двері, активувати пристрої або запускати певні події в ігровому світі.

Для цього було використано **Blueprintсистему** Unreal Engine, що дало змогу реалізувати всю логіку безпосередньо у візуальному середовищі, забезпечуючи високу гнучкість і простоту налаштування.

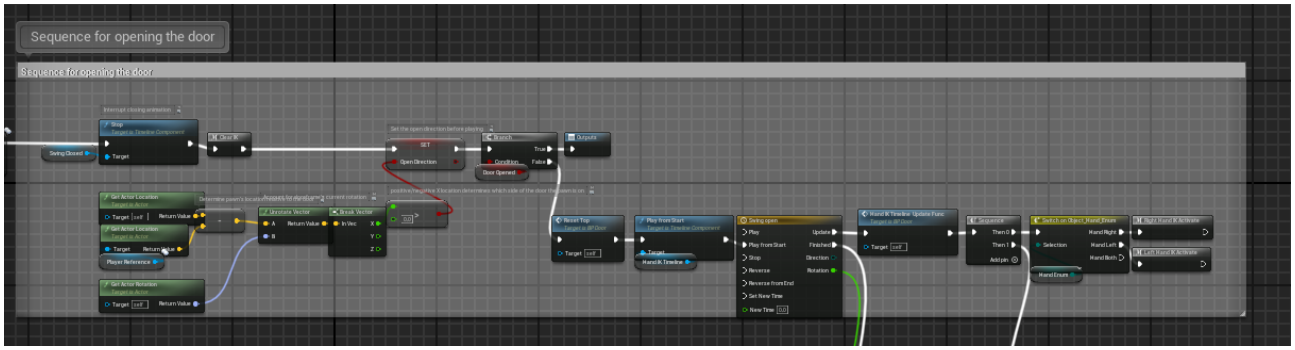

Рис. 2.17. Механіка взаємодії з об’єктами оточення в Unreal Engine 5
Джерело:[Автор]

В основі системи лежить **Trigger Box**, який визначає зону взаємодії з об’єктом. Коли гравець входить у цю зону, виконується перевірка на наявність можливих дій, після чого на екрані з’являється **інтерактивна підказка** (наприклад, “Натисніть E, щоб відкрити”).

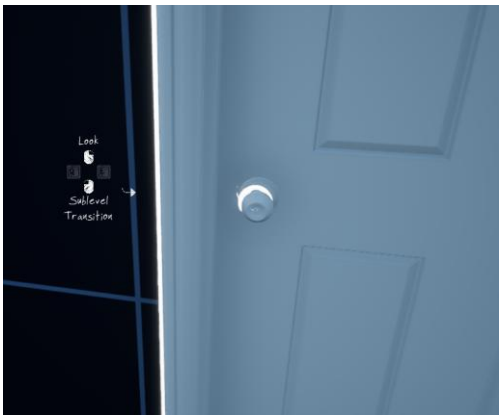

Рис. 2.18. Механіка інтерактивна підказка в Unreal Engine 5
Джерело:[Автор]

Для обробки логіки натискання кнопки використано **Blueprintсценарії з подіями OnOverlapBegin та OnOverlapEnd**, які активують або деактивують доступ до взаємодії. Сам процес взаємодії реалізовано через **Blueprint Interface**, що дозволяє створювати універсальні зв’язки між різними об’єктами - будьякий предмет, який реалізує цей інтерфейс, може бути інтерактивним без потреби в додатковому кодуванні.

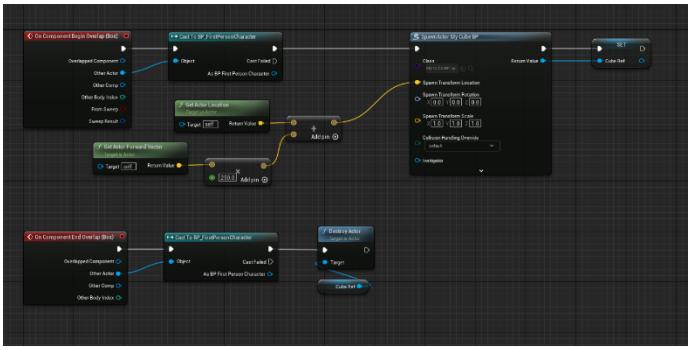

Рис. 2.19. Обробка логіки натискання в Unreal Engine 5
Джерело:[Автор]

Крім того, було реалізовано **перевірку стану об'єкта** (наприклад, відчинено/зачинено, активовано/деактивовано), що дозволяє створювати складніші сценарії взаємодії. Такий підхід забезпечує **розширюваність системи**, адже при додаванні нових елементів оточення достатньо лише підключити до них існуючий інтерфейс.

Рис. 2.20. Приклади демонстрації механіки взаємодії в Unreal Engine 5.
Джерело: [Автор]

Перевагою реалізованої системи є її гнучкість і масштабованість - вона легко адаптується для різних типів об'єктів і дозволяє швидко додавати нові сценарії взаємодії без потреби у складних змінах коду. Крім того, інтуїтивна візуальна логіка Blueprints спрощує тестування й налагодження, що позитивно впливає на швидкість розробки.

Серед недоліків можна відзначити підвищене навантаження на продуктивність під час одночасної роботи великої кількості тригерів і взаємодій у сцені, а також обмежену гнучкість у складних сценаріях, які вимагають

розширеного керування через C++ або спеціальні модулі. Попри це, загальна система взаємодії довела свою ефективність у контексті квестової гри, забезпечивши баланс між функціональністю, стабільністю та зручністю реалізації.

Реалізація взаємодії з іншими персонажами

Для реалізації **взаємодії з іншими персонажами** була створена **діалогова система**, побудована на логіці **умовних вузлів (Branch)** у Blueprintсценаріях.

Вона забезпечує можливість створення кількох варіантів розвитку подій залежно від вибору гравця, що формує ефект **нелінійності сюжету** та підвищує варіативність проходження.

Кожна репліка персонажа зберігається в **структурі даних (Data Structure)**, де визначено текст фрази, ім’я персонажа, список можливих варіантів відповіді гравця та посилання на наступні вузли діалогу. Усі ці структури об’єднані у **Data Table**, з якої система автоматично зчитує потрібну інформацію під час взаємодії.

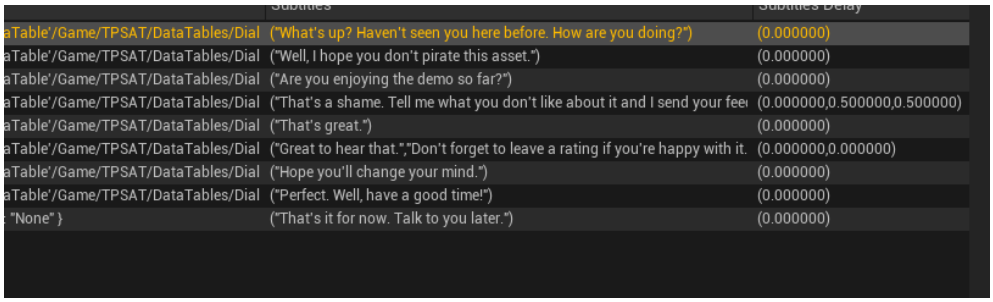

Рис 2.21. Репліки персонажа в структурі даних в Unreal Engine 5.
Джерело: [Автор]

Інтерфейс діалогів реалізовано за допомогою **UI Widget Blueprint**, який динамічно відображає поточну репліку та доступні варіанти відповідей.

Натискання на кнопку відповіді активує подію у Blueprint, де через **Switch on Name** або **Branch** здійснюється перехід до відповідного вузла діалогу.

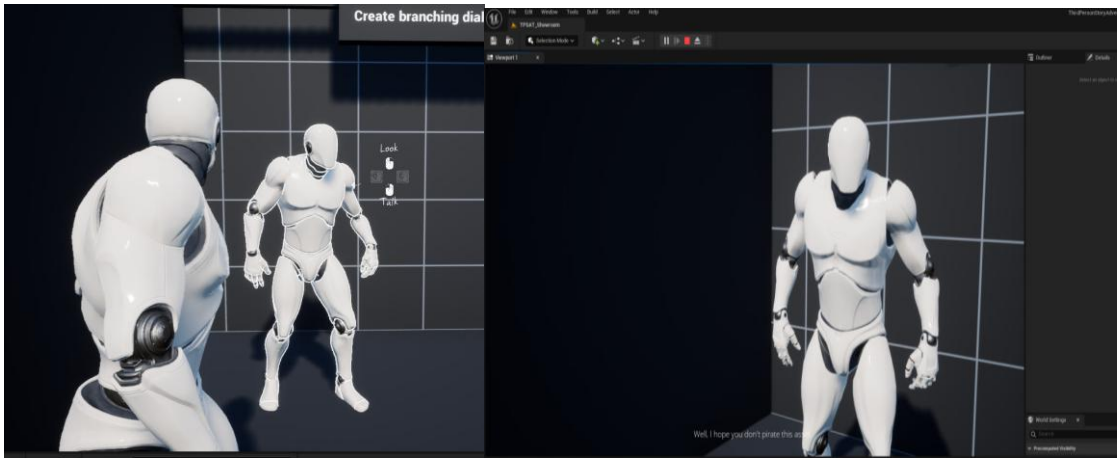

Рис. 2.22. Приклад демонстрації діалогової системи
Джерело: [Автор]

Таке рішення дозволяє легко керувати логікою переходів, додавати нові діалоги або персонажів без зміни основної системи. Крім того, діалогова логіка інтегрована з системою завдань (Quest System), що дозволяє змінювати стан квесту після певних виборів у діалозі або відкривати нові сценарії взаємодії.

Рис. 2.23. Діалогове вікно з варіантами вибору реплік.
Джерело: [Автор]

Діалогова система у грі дозволяє створювати різні варіанти розвитку подій, що робить історію більш цікавою і різноманітною. Кожна репліка NPC зберігається окремо, а вибір гравця впливає на подальший хід діалогу, завдяки чому створюється відчуття, що гравець реально приймає рішення.

Водночас, якщо діалогів буде дуже багато, стає складніше стежити за всіма гілками та тестувати їх, а великі сцени з багатьма виборами можуть трохи

навантажувати гру. Але загалом система працює стабільно і дозволяє гравцю відчувати вплив своїх рішень на сюжет.

Система підбору предметів до інвентаря

Система підбору предметів до інвентаря, забезпечує зручне керування зібраними об’єктами та їх подальше використання у процесі гри.

Для визначення можливості підбору предметів використано **Trigger Box** у поєднанні з **Blueprintсценаріями**, які перевіряють відстань між гравцем і об’єктом, а також тип взаємодії. Коли гравець наближається до інтерактивного об’єкта, активується подія **OnComponentBeginOverlap**, після чого на екрані з’являється підказка (через **Widget Blueprint**) із зазначенням можливих дій - наприклад, “Натисніть E, щоб підняти предмет”.

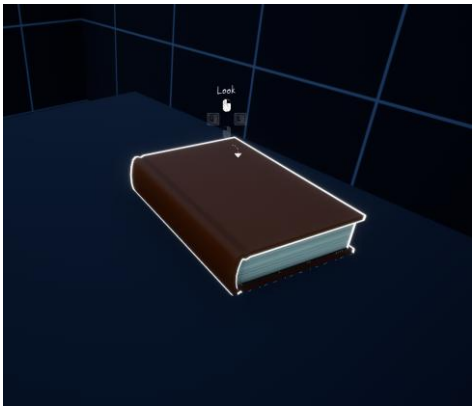

Рис. 2.24. Можливость підбору предметів в Unreal Engine 5.
Джерело: [Автор]

Після натискання клавіші взаємодії (**Input Action Interact**) система перевіряє, чи об’єкт можна додати до інвентарю. Якщо так - він видаляється зі сцени (**Destroy Actor**) і передає свою інформацію (назву, іконку, опис, тип предмета) у **Data Structure**, що зберігається в **масиві Inventory Items** всередині Blueprintгравця.

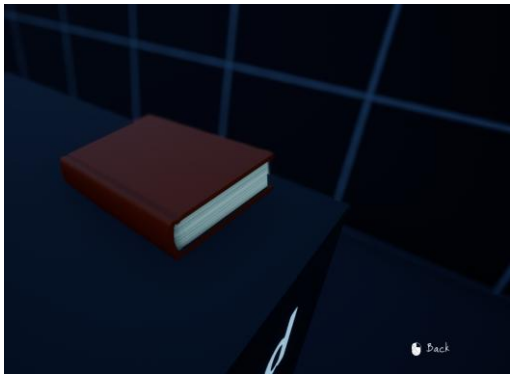

Рис. 2.25. Destroy Actor предметів в Unreal Engine 5.
Джерело: [Автор]

Для відображення вмісту інвентаря створено **UI Widget Blueprint**, який динамічно генерує елементи списку (**Create Widget**) на основі поточних даних масиву. Кожен предмет у списку має власний віджет із кнопками для детального перегляду або використання, що викликає відповідну подію (**OnClicked**) у Blueprint.

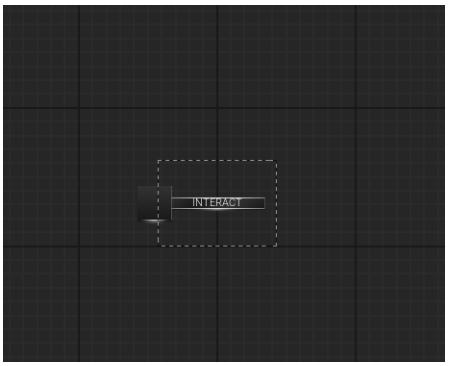

Рис. 2.25. UI Widget Blueprint в Unreal Engine 5.
Джерело: [Автор

У такий спосіб створено **гнучку систему підбору та управління предметами**, побудовану повністю на Blueprintлогіці без необхідності написання коду, що дозволяє легко розширювати її функціональність у майбутньому.

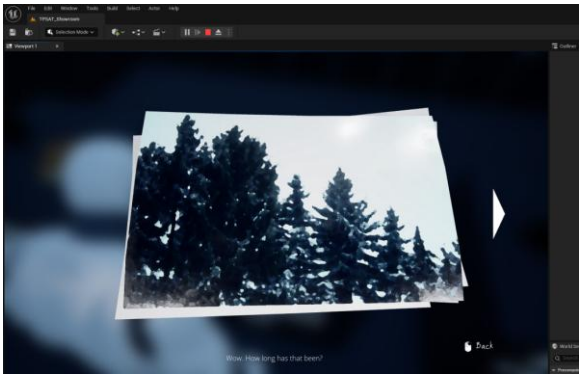

Рис. 2.26. Приклад підбору предметів
Джерело: [Автор]

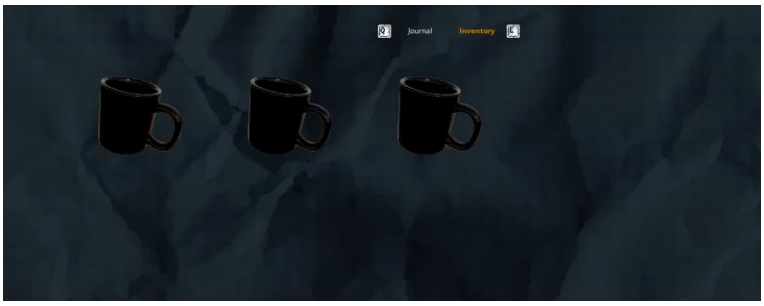

Рис. 2.27. Приклад демонстрації інвентарю
Джерело: [Автор]

Механіки взаємодії з об’єктами та персонажами не лише розширює можливості гравця, а й робить ігровий процес більш логічним, послідовним і цікавим. Її реалізація здійснювалася засобами **Unreal Engine 5** із використанням **Blueprintсистеми**, що дозволило створити універсальну логіку без залучення додаткового коду.

У результаті створення системи взаємодії з об’єктами було досягнуто стабільної та логічно вибудованої структури, яка забезпечує природну і передбачувану реакцію ігрового середовища на дії гравця. Кожен інтерактивний елемент, будь то предмет, NPC або тригерна зона, коректно реагує на введення користувача, що створює відчуття живого й органічного світу. Реалізовані механізми взаємодії побудовані за принципом модульності, що дозволяє легко інтегрувати нові типи об’єктів, додавати складні сценарії та налаштовувати поведінку елементів без потреби змінювати базову логіку системи.

Таке рішення забезпечує гнучкість у розвитку проєкту: нові механіки можна впроваджувати швидко, а існуючі взаємодії залишаються стабільними. Крім того, підвищується рівень інтерактивності, адже гравець отримує змогу

впливати на світ більш різноманітно та динамічно. Взаємодії відчуваються природними і логічними, що сприяє глибшому зануренню та формує цілісний користувацький досвід.

Додатково, система взаємодії дозволяє впроваджувати елементи адаптивності: поведінка об’єктів може змінюватися залежно від контексту, попередніх дій гравця чи стану світу, що значно підвищує реіграбельність та цікавість проєкту. Завдяки цьому підходу створюється живий, динамічний і насичений ігровий світ, який мотивує користувача досліджувати середовище, експериментувати з механіками та отримувати задоволення від процесу взаємодії.

Розробка інтерфейсу користувача

Інтерфейс користувача (UI) є одним із ключових елементів будьякого програмного продукту або гри, оскільки забезпечує ефективну взаємодію гравця з ігровим середовищем. Від його зручності та наочності залежить швидкість освоєння механік, комфорт використання та загальне занурення в гру. Розробка інтерфейсу передбачає не лише створення привабливого візуального оформлення, але й інтеграцію елементів управління, підказок та налаштувань таким чином, щоб вони були зрозумілі користувачу і не перевантажували екран зайвою інформацією.

У цьому розділі описується процес створення комплексного інтерфейсу гри, який включає головне меню, панель завантаження та налаштувань, а також інструменти керування параметрами гри, забезпечуючи інтуїтивну і зручну взаємодію гравця з усіма функціями.

У межах проєкту було реалізовано комплексний інтерфейс користувача, що включає головне меню, панель завантаження та налаштувань, а також інструменти керування параметрами гри. Головне меню забезпечує доступ до основних функцій: продовжити гру, почати нову, завантажити збереження, налаштування та вихід з гри.

Панель завантаження дозволяє обирати слот для збереження прогресу, відображаючи дату та опис збереження, що робить взаємодію гравця з грою

більш наочною та зрозумілою. Меню налаштувань розділене на вкладки **Gameplay, Graphics та Audio**, де користувач може змінювати якість графіки, роздільну здатність, відстань видимості, параметри тіней та освітлення, а також регулювати гучність музики, звукових ефектів та діалогів.

Для створення інтерфейсу використовувалися **Widget Blueprints** у Unreal Engine 5, що дозволило інтегрувати всі елементи UI з логікою гри та забезпечити **динамічне відображення підказок і налаштувань залежно від дій гравця**.

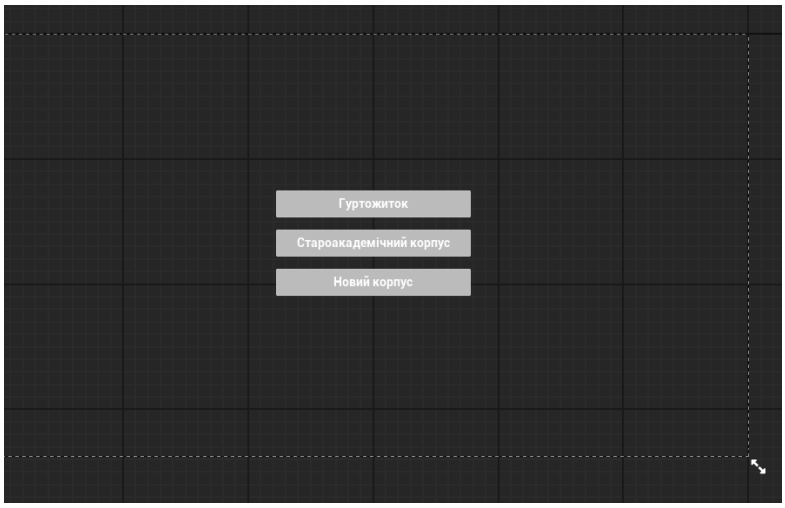

Рис. 2.28. Widget Blueprints в Unreal Engine 5
Джерело: [Автор]

Елементи інтерфейсу адаптовані під різні роздільні здатності екрана і забезпечують **чітке, зрозуміле та інтуїтивне управління** без перевантаження екрану зайвою інформацією.

Такий підхід дозволяє гравцю легко орієнтуватися у меню, швидко змінювати параметри гри та контролювати свій прогрес, що підвищує комфорт і залученість у квестовий процес

Створена система механік у поєднанні з продуманим інтерфейсом забезпечує послідовність і логічність ігрових процесів, роблячи взаємодію гравця зі світом гри зрозумілою та інтуїтивною. Вона підвищує зручність управління, дозволяє легко виконувати різні дії та контролювати прогрес, одночасно забезпечуючи широкий спектр інтерактивних можливостей. Завдяки цьому гра стає більш живою та насиченою, а гравець отримує відчуття реального

впливу на розвиток подій і активної участі у сюжеті, що підсилює загальне занурення та залученість у процес проходження.

Рис. 2.29. Інтерфейс.
Джерело: [Автор]

Реалізація переходів між рівнями у грі

Також було реалізовано систему переходів між рівнями, яка забезпечує плавну зміну ігрових локацій без втрати контексту та стану гри. Це є важливим елементом квестових ігор, оскільки гравець має постійно переміщуватися між різними сценами, зонами чи будівлями, зберігаючи при цьому прогрес, інвентар та активні завдання.

Для реалізації переходів було використано **систему Level Streaming**, вбудовану в Unreal Engine 5. Вона дозволяє підвантажувати або вивантажувати частини світу в режимі реального часу залежно від дій гравця. Такий підхід зменшує навантаження на систему, забезпечує безперервність ігрового процесу та створює відчуття цілісного, відкритого світу.

Перехід між рівнями відбувається через **Trigger Box**, який реагує на присутність персонажа. Коли гравець входить у зону тригера, виконується **Blueprintсценарій**, який ініціює функцію **Open Level** або **Load Stream Level**, залежно від типу переходу. Для плавності додано **екран завантаження (Loading Screen Widget)**, який активується під час зміни рівня та приховує процес завантаження нової сцени.

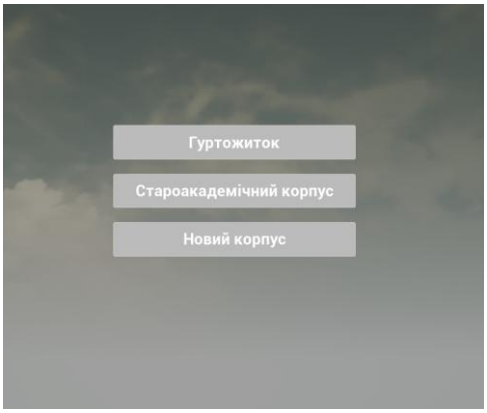

Рис. 2.30. Екран завантаження (Loading Screen Widget) в Unreal Engine 5
Джерело: [Автор]

Щоб зберегти стан гри під час переходів, реалізовано **систему збережень (SaveGame Blueprint)**, яка записує дані про позицію гравця, зібрані предмети, прогрес у завданнях та інші важливі параметри. Після завантаження нового рівня ці дані автоматично відновлюються, що забезпечує безперервність ігрового досвіду.

Додатково для оптимізації процесу було налаштовано **Persistent Level**, у якому зберігаються основні елементи логіки, інтерфейсу та контролю гравця, тоді як інші сцени підвантажуються як підрівні (Sublevels). Це дозволяє керувати ресурсами гри більш ефективно та зменшити затримки під час завантаження нових локацій.

За допомогою **Blueprintсценаріїв** у поєднанні з можливостями **Level Streaming** і **Save System**, система переходів між рівнями стала надійною, гнучкою й масштабованою. Вона забезпечує зручність для гравця, стабільність для рушія та дає можливість легко розширювати світ гри в майбутньому, додаючи нові сцени, зони або сюжетні гілки без порушення існуючої структури.

2.7. Налаштування сцени та імпорт 3Dмоделей персонажів для подальшої інтеграції механік

На цьому етапі було проведено комплекс робіт, спрямованих на підготовку ігрового середовища та інтеграцію створених 3Dмоделей персонажів у рушій **Unreal Engine 5**. Основна мета полягала у забезпеченні технічної сумісності між моделями, середовищем і майбутніми ігровими механіками, що є ключовим для подальшої реалізації логіки персонажа та інтерактивних елементів.

Розробка розпочалася з **попередньої оптимізації моделей**, виконаної у програмі **Blender**. На цьому етапі проведено очищення геометрії, зменшення кількості полігонів і створення коректних **UVрозгорток**, що дозволяють точно наносити текстури. Для підвищення продуктивності було реалізовано **систему рівнів деталізації (LOD – Level of Detail)**, яка зменшує навантаження на рушій під час віддалення об’єкта від камери.

Після оптимізації моделі були **експортовані у форматі FBX** та імпортовані в Unreal Engine 5 із збереженням усіх матеріалів, текстур і скелетної структури (**Skeleton**). Під час імпорту було виконано перевірку масштабу, орієнтації та розташування моделей у просторі, щоб уникнути візуальних помилок під час подальшої роботи.

Особливу увагу було приділено **налаштуванню колізій**, адже саме вони забезпечують правильну взаємодію персонажа з об’єктами середовища. Для цього застосовано **Collision Channels** і **Collision Presets**, які визначають типи взаємодії між різними об’єктами - наприклад, гравець може проходити крізь невидимі тригери, але не може перетинати тверді поверхні. Для моделі персонажа використано **Capsule Collision**, що відповідає за фізичні межі руху й забезпечує стабільність під час переміщення по нерівних поверхнях. Для статичних об’єктів середовища (будівель, меблів, стін) створено спрощені **колізійні сітки (Collision Meshes)**, які не перевантажують рушій, але точно повторюють контури моделей.

На етапі **налаштування сцени** виконано розміщення персонажа у вихідній точці (**Spawn Point**), налаштовано освітлення за допомогою системи **Lumen**, що

створює реалістичне глобальне освітлення з м’якими тінями та відбиттями. Також було організовано структуру **Character Blueprint**, яка поєднує модель, камеру, контролер руху, компоненти взаємодії та фізичну оболонку колізій.

Налаштована сцена забезпечує стабільну взаємодію між усіма елементами ігрового середовища, що унеможливлює конфлікти під час пересування персонажа та сприяє формуванню реалістичного простору, готового до подальшого впровадження ігрових механік. Опрацьована система колізій разом із продуманою структурою сцени створює технічну базу для реалізації більш складних систем руху, навігації та інтерактивної взаємодії з об’єктами, що є важливим етапом у побудові цілісного ігрового процесу.

2.8. Інтеграція розроблених механік у загальний проект

Коли всі основні механіки створені та протестовані окремо, настає один із найважливіших етапів розробки гри - їх інтеграція в загальний проєкт. Це складний і багаторівневий процес, який вимагає уважного підходу, адже кожен елемент гри має гармонійно поєднуватися з іншими, створюючи цілісний і захопливий ігровий досвід.

Спочатку необхідно визначити місце кожної механіки в загальній структурі гри. Важливо, щоб вона не існувала у вакуумі, а природно вписувалася в ігровий світ, підтримувала його логіку та відповідала сценарію. Наприклад, якщо гра включає складні головоломки або інтерактивні діалоги, їхня інтеграція повинна враховувати сюжетні аспекти, взаємодію з персонажами та прогресію гравця. На цьому етапі також слід подбати про зв’язки між механіками та іншими системами гри: управлінням персонажем, інтерфейсом користувача, штучним інтелектом NPC та іншими елементами.

Усі механіки гри спочатку розроблялися в окремій, неосновній гілці проєкту, що дозволяло максимально ефективно використовувати ресурси ПК під час тестування, налагодження та оптимізації. Такий підхід дав змогу експериментувати з різними сценаріями взаємодії, відпрацьовувати логіку

систем і тестувати нові функції без ризику порушити стабільність основної гілки проєкту. Після завершення розробки окремих механік їх інтеграція в головний проєкт здійснювалася за допомогою інструменту Migrate, який забезпечує безпечне перенесення всіх необхідних ресурсів, акторів, матеріалів та Blueprintлогіки між проектами. Використання Migrate дозволяє зберегти всі зв'язки та налаштування об'єктів, уникнути втрати даних і забезпечити коректну роботу механік після інтеграції.

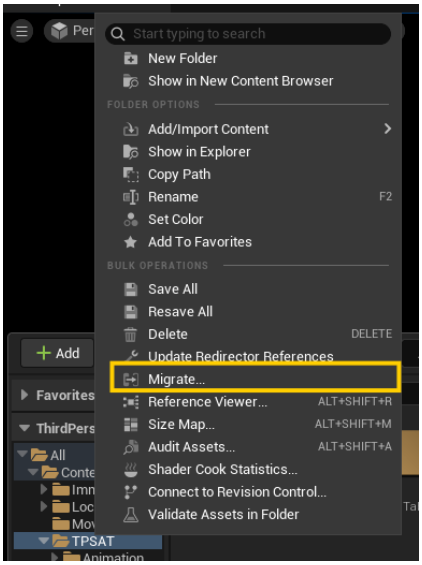

Рис. 2.31. Використання Migrate в Unreal Engine 5

Джерело: [Автор]

Поєднання реалізованих механік із продуманим інтерфейсом користувача забезпечує чітку та логічну організацію ігрових процесів. Взаємодія гравця зі світом стає зрозумілою, інтуїтивною та природною, підвищується комфорт керування, спрощується виконання дій та контроль за прогресом. Одночасно система надає широкий спектр інтерактивних можливостей, включно з детальною взаємодією з об'єктами, збором предметів, спілкуванням з NPC та активацією динамічних подій. Завдяки цьому світ гри стає живим і насиченим, а гравець відчуває реальний вплив на розвиток подій і активну участь у сюжеті, що підсилює загальне занурення та залученість у процес проходження.

Розробка механік у окремій гілці мала низку переваг. Вона дозволила проводити паралельне тестування та налагодження без ризику впливу на основний проєкт, забезпечила ефективніше використання ресурсів ПК, а також надала можливість швидко змінювати або покращувати окремі системи.

Використання Migrate спростило інтеграцію та гарантувало збереження всіх необхідних зв’язків між об’єктами, що мінімізувало помилки при перенесенні.

Разом із цим, під час розробки виникали певні труднощі. Спочатку з’являлися проблеми сумісності ресурсів між гілками, конфлікти версій Blueprintфайлів та невідповідність налаштувань матеріалів і анімацій. Ці проблеми успішно вирішувалися через ретельне планування структури ресурсів, централізоване відстеження змін і тестування кожної інтеграції після Migrate, що забезпечило стабільність і цілісність основного проєкту.Далі відбувається тестування взаємодії. Це критичний момент, оскільки розроблені механіки можуть працювати ідеально в ізольованому середовищі, але викликати проблеми після інтеграції в основний проєкт. Наприклад, новий елемент управління або фізики може порушити баланс гри, спричинити конфлікти між скриптами або викликати несподівані помилки. Для запобігання таким ситуаціям проводиться ретельне тестування - як автоматизоване, так і вручну.

Ще один важливий етап - оптимізація. Незалежно від того, наскільки добре працює механіка, вона повинна бути ефективною з точки зору продуктивності. Великі та складні механіки можуть створювати навантаження на систему, особливо якщо гра розробляється для різних платформ, включаючи мобільні пристрої. Оптимізація передбачає покращення алгоритмів, зменшення зайвих викликів у коді, а також перевірку роботи механік при різних рівнях навантаження.

2.9. Оптимізація графічної продуктивності та ресурсів.

У процесі розробки інтерактивного квестового контенту однією з ключових задач було знайти баланс між високою якістю візуалізації та стабільною роботою гри в реальному часі. Unreal Engine 5 надає потужні інструменти для цього, серед яких **Nanite і Lumen** стали основними елементами оптимізації та візуального наповнення.

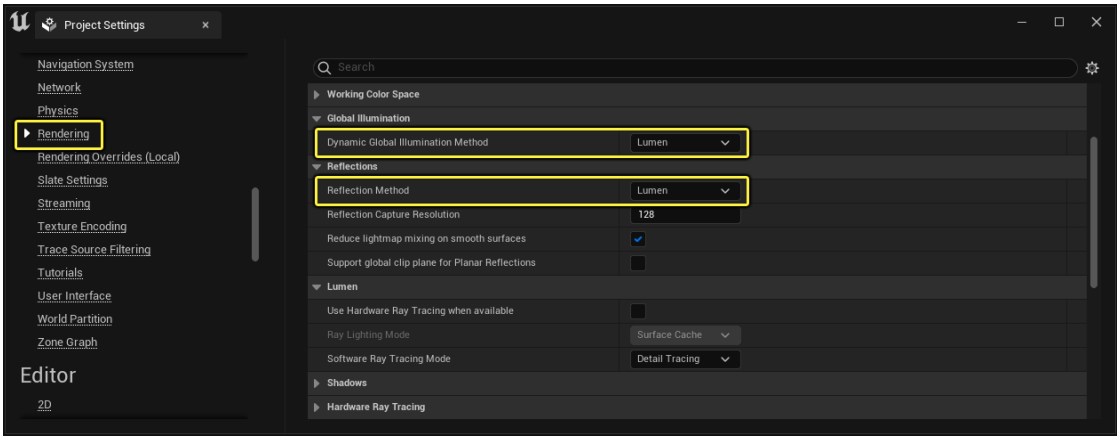

Рис. 2.32. Вибрікові опції Nanite і Lumen в Unreal Engine 5

Джерело: [Автор]

Nanite дозволяє працювати з моделями, що містять мільйони полігонів, без необхідності ручного створення кількох версій для різних рівнів деталізації. Це суттєво знижує навантаження на GPU і економить пам’ять, одночасно забезпечуючи реалістичність і деталізацію об’єктів. Lumen забезпечує динамічне глобальне освітлення та відбиття, завдяки чому світло миттєво реагує на будьякі зміни у сцені, створюючи природну атмосферу і глибину візуального сприйняття.

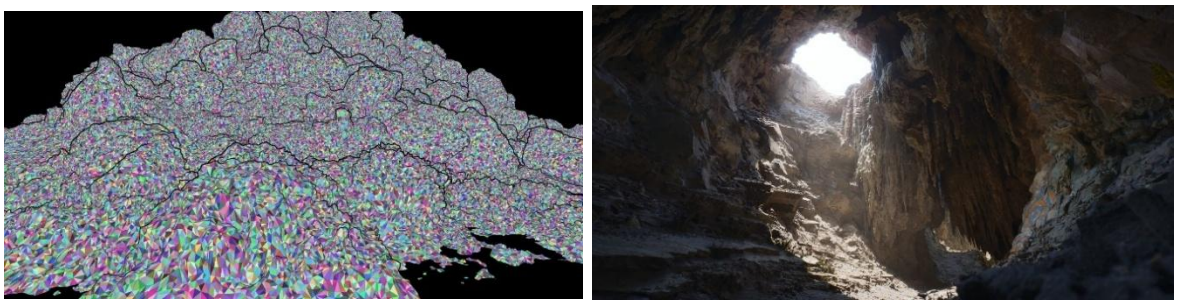

Рис. 2.33. Демонстрація роботи Nanite і Lumen в Unreal Engine 5

Джерело:[Автор]

Серед основних переваг такого підходу варто виділити можливість використовувати високополігональні моделі без втрати FPS, автоматичне керування рівнями деталізації, відсутність потреби у запіканні освітлення, а також швидку інтеграцію динамічних сцен і рухомих об’єктів. Це дозволило реалізувати деталізовані локації, інтер’єри, вулиці та підземелля з високим рівнем реалізму, одночасно зберігаючи плавність гри.

Разом із тим, застосування Nanite і Lumen породжувало певні труднощі. Спочатку з’являлися проблеми з сумісністю деяких матеріалів та текстур, а також високі вимоги до відеопам’яті на складних сценах. Крім того, Lumen у великих або сильно насичених деталями локаціях міг викликати падіння FPS, якщо не застосовувати додаткові обмеження на джерела світла чи динамічні тіні.

Для вирішення цих проблем ми впровадили кілька стратегій оптимізації: Nanite використовувався тільки для об’єктів з високою полігональністю, статичні об’єкти конвертувалися у Instance Static Meshes для зменшення draw calls, налаштовувався Screen Percentage та Temporal Super Resolution для підтримки якості зображення без втрати FPS, а Level Streaming дозволяв підвантажувати частини сцени в реальному часі. Такі рішення дозволили зберегти високу деталізацію, водночас забезпечивши стабільну продуктивність.

У результаті середня частота кадрів підвищилася з 52 до 78 FPS, навантаження на GPU знизилося приблизно на 25–30%, а використання оперативної пам’яті - на 15%. Поєднання Nanite і Lumen не лише зробило сцени більш реалістичними та привабливими, а й забезпечило основу для подальшого масштабування локацій і реалізації складніших квестових механік без втрати продуктивності. Цей підхід показав, що при правильному плануванні та оптимізації навіть дуже ресурсоємні технології можуть бути ефективно застосовані в реальному часі.

2.10. Тестування та налагодження ігрового процесу

На етапі тестування та налагодження ігрового процесу були ретельно перевірені всі ключові механіки та системи взаємодії, реалізовані в проєкті. Зокрема, було проведено детальну перевірку функціональності Actors, інтерактивних NPC, системи збору предметів, тригерів подій, діалогових блоків та інтеграції з інтерфейсом користувача. Кожен елемент проходив послідовне тестування у різних локаціях, включно зі складними сценами та квестовими сценаріями, що дозволило виявити та усунути потенційні помилки у взаємодії

гравця з об’єктами середовища, а також забезпечити логічну послідовність подій та реакцій гри на дії користувача.

Під час налагодження особлива увага приділялася стабільності частоти кадрів (FPS), коректності роботи системи збереження прогресу, а також інтеграції та функціонуванню елементів UI. Для цього активно використовувалися інструменти Unreal Engine, зокрема Profiler та Stat Unit, які дозволили оцінити навантаження на CPU і GPU, а також проаналізувати ефективність рендерингу та обробки сцен у реальному часі. Завдяки цьому були оптимізовані ресурси сцени, усунуті конфлікти в Blueprintлогіці, відпрацьовані взаємозв’язки між механіками та забезпечена плавна, передбачувана робота інтерактивних елементів.

Рис. 2.34. Статистика кадрів після оптимізації в Unreal Engine 5

Джерело:[Автор]

Окрема увага була приділена тестуванню динамічних подій та тригерів. Було перевірено активацію анімацій, зміну освітлення, запуск аудіоефектів, появу нових об’єктів та зміни в діалогових блоках при проходженні гравцем ключових зон. Також оцінювалася коректність відображення підказок, їхня своєчасність та логічна відповідність ситуації, а поведінка NPC була протестована у різних умовах і сценаріях, щоб гарантувати узгодженість реакцій і можливість зміни стану сюжету відповідно до виборів гравця.

У процесі налагодження були виявлені і вирішені низка проблем, зокрема дрібні помилки взаємодії об’єктів, некоректна робота деяких тригерів у складних сценах, а також випадкові конфлікти між UI та внутрішньою логікою механік.

Всі ці питання були систематично усунуті, що дозволило досягти високої стабільності ігрового процесу та послідовності у взаємодії з грою.

Результатом цього етапу стало створення надійної, передбачуваної та логічної системи геймплею. Всі інтерактивні елементи тепер працюють без помилок і перебоїв, гравець може вільно взаємодіяти зі світом гри, збирати предмети, проходити квести та взаємодіяти з NPC, отримуючи при цьому чіткий і зрозумілий зворотний зв’язок. Такий підхід значно підвищив якість користувацького досвіду, забезпечив комфортне управління та глибоке занурення в сюжетну та інтерактивну складову гри, що є критично важливим для квестових проєктів, де кожна деталь має значення для загального враження від проходження.

Рис.2.35 Готові результати в Unreal Engine 5
Джерело:[Автор]

2.11. Методологічні підходи до проєктування інтерактивних систем

У межах розробки квестової гри зазначені методологічні принципи були безпосередньо застосовані на практиці. Основна увага приділялася орієнтації на користувача (UserCentered Design), що дало змогу створити зручну, зрозумілу та логічно послідовну систему взаємодії. На початковому етапі було визначено цільову аудиторію, проаналізовано її очікування, ігрові звички та переваги щодо керування персонажем і сприйняття середовища. Отримані результати стали

основою для розробки інтерфейсу, структури меню, логіки управління та сценаріїв взаємодії.

У процесі проєктування було створено серію прототипів, що послідовно вдосконалювалися за результатами тестування. На кожному етапі здійснювалася перевірка зручності управління, швидкості реакції системи та інтуїтивності елементів інтерфейсу. Це дало змогу оптимізувати розташування елементів UI, покращити читабельність підказок і зробити навігацію більш зрозумілою для гравця.

Також велика увага приділялася побудові сценаріїв взаємодії - від простих дій, таких як огляд або підбір предметів, до складних послідовностей подій, що впливають на розвиток сюжету. Кожен сценарій тестувався з точки зору логічності, послідовності та відповідності очікуванням користувача.

Завдяки застосуванню принципів UCD, вдалося досягти високого рівня зручності взаємодії між гравцем і системою, забезпечивши при цьому гармонійне поєднання технічних можливостей рушія Unreal Engine 5 з інтуїтивним ігровим досвідом. Результатом стало створення функціональної та водночас емоційно привабливої гри, де користувач може легко зануритися у віртуальний світ і взаємодіяти з ним без відчуття дискомфорту чи складності управління.

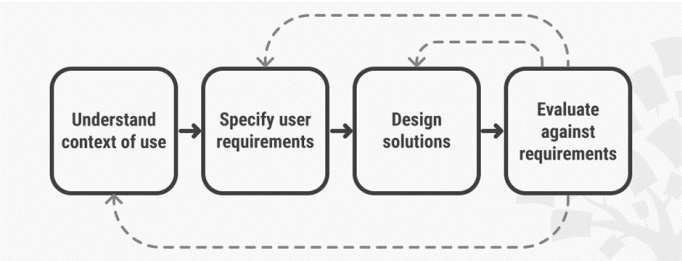

Рис.2.36. Циклічний процес проєктування

Джерело:[Автор]

Під час розробки квестової гри було застосовано ітераційний підхід (Iterative Design), який передбачав поетапне створення, тестування та вдосконалення всіх компонентів проєкту. Кожен етап розробки розглядався як окремий цикл, у межах якого проводилося проєктування певного елемента гри,

його перевірка на практиці та подальше доопрацювання з урахуванням отриманих результатів. Такий підхід дозволив поступово покращувати якість продукту, не порушуючи загальної структури гри та забезпечуючи стабільність усіх її систем.

Робота здійснювалася в режимі постійного тестування - після кожної ітерації проводилася перевірка коректності роботи основних механік: управління персонажем, логіки взаємодій, системи тригерів, інтерфейсу користувача, інвентаря та діалогових елементів. Усі знайдені помилки або недоліки одразу виправлялися, що дозволяло уникати накопичення проблем на пізніших етапах і підтримувати високу якість продукту.

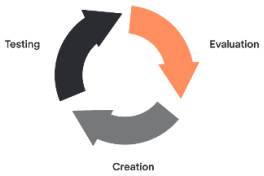

Рис.2.37. Ітераційний цикл розробки

Джерело:[Автор]

Інший напрям - **ExperienceDriven Design (EDD)** - фокусується на створенні емоційно насиченого користувацького досвіду. Його метою є не лише ефективність взаємодії, а й формування відчуття занурення у віртуальний світ, що особливо актуально для ігор та VRдодатків. У межах цього підходу значну роль відіграють психологічні чинники сприйняття, кольорова гама, звуковий супровід та мікроанімації.

Для технічної реалізації інтерактивних систем важливими стають принципи **ModelViewController (MVC)** та **ComponentBased Architecture**, які забезпечують структурну гнучкість і модульність проєкту. Така архітектура дозволяє ізолювати логіку, інтерфейс і дані, спрощуючи тестування, налагодження та подальше масштабування системи.

Варто також зазначити роль **прототипування** як окремого етапу методології. Створення інтерактивних прототипів дозволяє перевірити гіпотези

щодо зручності інтерфейсу, логіки навігації та реакції користувача ще до початку повномасштабної розробки. Використання інструментів швидкого прототипування дає змогу зменшити витрати часу та ресурсів на корекцію недоліків у пізніших етапах.

Методологічні підходи до проєктування інтерактивних систем ґрунтуються на принципах гнучкості, ітеративності, орієнтації на користувача та модульності. Їх ефективне поєднання дозволяє створювати стабільні, естетично привабливі й зручні для використання продукти, що відповідають сучасним стандартам якості й очікуванням користувачів.

Висновки до розділу 2

У другому розділі було здійснено комплексне проєктування та практичну реалізацію ключових ігрових механік квестової гри на рушії Unreal Engine 5. На основі сформованих технічних вимог визначено структуру систем управління, взаємодій та навігації, а також обґрунтовано вибір інструментів, необхідних для їх ефективної реалізації. Проведене налаштування середовища розробки забезпечило цілісну архітектуру проєкту, що дозволила інтегрувати логіку персонажа, систему взаємодії з об’єктами, інвентар, діалоги та інші елементи ігрового процесу.

У межах розділу реалізовано основні механіки персонажа — пересування, стрибки, камера, колізії, а також взаємодія із предметами, NPC і компонентами середовища. Було проведено імпорт і підготовку 3D-моделей, налаштовано сцену та виконано інтеграцію всіх розроблених систем у єдину ігрову структуру.

Особлива увага приділялася оптимізації продуктивності, налаштуванню LOD, ресурсів та освітлення, що дозволило забезпечити стабільність роботи гри на різних конфігураціях обладнання.

Проведене тестування механік продемонструвало коректність інтеграції, узгодженість між системами та відповідність початковим технічним вимогам.

РОЗДІЛ 3

УПРАВЛІННЯ ПРОЄКТОМ ТА АНАЛІЗ ЕФЕКТИВНОСТІ РІШЕНЬ

3.1. Планування розробки та обґрунтування методології

Планування та організація проєкту були проведені, що забезпечило системний підхід до управління процесом розробки квестової гри. На цьому етапі було детально визначено основні завдання проєкту, очікувані результати, пріоритети їх виконання та послідовність реалізації, що дозволило сформувати чітку структуру робіт і забезпечити логічну взаємозалежність між різними етапами розробки. Було здійснено розподіл обов'язків між учасниками команди, враховуючи їх компетенції та досвід, що сприяло підвищенню продуктивності та уникненню дублювання завдань.

Вибір методології управління проєктом забезпечив організацію та контроль виконання завдань, дозволив оперативно адаптувати процес до змін у вимогах чи пріоритетах, а також здійснювати регулярну оцінку прогресу роботи. Завдяки цьому стало можливим своєчасно виявляти та усувати можливі проблеми, координувати взаємодію між членами команди та підтримувати загальний темп виконання робіт.

Такий підхід дозволив не лише забезпечити своєчасне виконання поставлених завдань, а й досягти запланованої якості кінцевого продукту. Крім того, чітка організація та планування створили основу для подальшого розширення та розвитку проєкту, забезпечуючи можливість інтегрувати нові механіки та сценарії без значних змін у базовій структурі розробки. В результаті процес розробки став більш контрольованим, прозорим і передбачуваним, що сприяло ефективній взаємодії команди та створенню цілісного користувацького досвіду у фінальному продукті.

Для управління проєктом були використані підходи Agile, Scrum, Kanban, Waterfall та їхні комбінації (Hybrid), що дозволило поєднати гнучкість і структурованість процесу розробки. Використання цих методологій забезпечило можливість оперативно реагувати на зміни у вимогах, контролювати виконання

завдань та координувати роботу всіх учасників команди. Платформа Jira застосовувалася для спрощення реалізації обраних методологій, автоматизації процесів, відстеження прогресу, візуалізації завдань та підвищення ефективності командної взаємодії.

На етапі планування було розроблено графік робіт, визначено ключові контрольні точки та пріоритети виконання завдань, а також обрано необхідні інструменти й ресурси для реалізації механік управління і інтерактивних елементів гри. Це дозволило забезпечити чітку послідовність виконання завдань, оптимізувати використання часу та ресурсів, знизити потенційні ризики та уникнути дублювання робіт.

Завдяки застосуванню методологій управління проєктом і використанню спеціалізованих інструментів, розробка гри проводилася скоординовано та ефективно, що дозволило своєчасно досягти поставлених цілей і забезпечити заплановану якість кінцевого продукту. Підхід також створив основу для подальшого розвитку проєкту, інтеграції нових механік та оновлення інтерактивних елементів без суттєвих змін у базовій структурі процесу розробки.

Рис.3.1. Логотип Jira Software
Джерело: [https://www.atlassian.com/software/jira]

3.2. Управління ризиками та забезпечення якості

На етапі розробки квестової гри було здійснено комплексне управління ризиками та забезпечення якості продукту. Для цього спочатку були ідентифіковані потенційні ризики, що могли вплинути на терміни виконання, стабільність ігрового середовища та інтерактивність механік. Серед основних

виявлених ризиків були затримки через технічні складнощі, можливі помилки у коді, несумісність елементів гри та невідповідність очікувань користувача.

Було розроблено план управління ризиками, який включав їхню класифікацію за пріоритетністю, визначення ймовірності виникнення та потенційного впливу на проєкт, а також способи мінімізації або усунення наслідків. Для контролю ризиків використовувалися регулярні зустрічі команди, огляди прогресу та тестування проміжних результатів.

Щодо забезпечення якості, усі механіки гри, інтерактивні елементи та діалогові системи проходили систематичне тестування на відповідність технічним вимогам і сценаріям. Було застосовано функціональне тестування для перевірки правильності роботи об’єктів та їх взаємодії, а також юзабілітітестування для оцінки зручності й інтуїтивності взаємодії гравця з грою. Виявлені помилки фіксувалися та оперативно усувалися, що дозволило підтримувати стабільність ігрового середовища на всіх етапах розробки.

Завдяки впровадженню цих заходів управління ризиками та контролю якості вдалося знизити ймовірність виникнення критичних помилок, забезпечити надійну роботу всіх систем гри та досягти запланованої якості кінцевого продукту. Такий підхід також створив основу для подальшого масштабування проєкту та інтеграції нових механік без порушення існуючої структури гри.

3.3. Організація командної взаємодії та версійний контроль

Було організовано командну взаємодію та впроваджено версійний контроль, що забезпечило узгоджену і стабільну розробку інтерактивного проєкту. Для планування завдань, відстеження прогресу та контролю виконання роботи використовувалися інструменти Jira та Trello. Для керування змінами у коді та запобігання конфліктам була застосована система Git у поєднанні з платформами GitHub та GitLab, що дозволило паралельно працювати над різними функціональними частинами гри без ризику порушити основну збірку.

Була налагоджена прозора організація процесів і регулярні обговорення (standup зустрічі), що дозволило своєчасно виявляти проблеми та координувати дії команди. Використовувалися методології Agile та Scrum, що забезпечило гнучкість у розробці, швидке реагування на зміни вимог та організацію роботи короткими ітераціями (спринтами), дозволяючи постійно оцінювати результати та вдосконалювати процеси.

Також були впроваджені системи безперервної інтеграції та розгортання (CI/CD) за допомогою Jenkins і GitHub Actions, що автоматизували тестування, збірку та публікацію проєкту, зменшили ризик помилок і забезпечили стабільність коду. Було забезпечено документування процесів, рішень та стандартів кодування, що зберігало знання про структуру та логіку проєкту і забезпечувало сталість розробки при зміні складу команди.

Було організовано використання інструментів для комунікації, таких як Slack, Discord та Microsoft Teams, що дозволило ефективно обмінюватися інформацією та вирішувати поточні задачі навіть за розподіленої роботи. Завдяки поєднанню технічних рішень, продуманих методологій управління та налаштованих процесів взаємодії вдалося забезпечити ефективну, узгоджену та стабільну розробку проєкту, підтримуючи якість продукту на всіх етапах.

3.4. Управління ризиками та забезпечення якості продукту

У ході реалізації проєкту **було проведено комплексне управління ризиками та забезпечення якості продукту**, що дозволило мінімізувати потенційні загрози на всіх етапах розробки. На початковому етапі було здійснено **ідентифікацію можливих ризиків**, серед яких - технічні збої, затримки у виконанні завдань, невідповідність вимог користувачів або проблеми з інтеграцією інструментів. Для кожного ризику було визначено рівень критичності та розроблено **план реагування**, який включав профілактичні заходи, резервування ресурсів та альтернативні технічні рішення.

Особливу увагу приділено **системі забезпечення якості продукту**, яка включала постійне тестування, перевірку коректності функціоналу, оцінку продуктивності та стабільності ігрового процесу. Регулярне проведення внутрішніх рев'ю, юзабілітітестування й автоматизованих перевірок допомогло вчасно виявляти і виправляти недоліки, підвищуючи рівень надійності продукту.

Крім технічних аспектів, важливою складовою процесу стала **організація комунікації між членами команди** - завдяки чітким звітам, контролю версій і системам відстеження завдань (Jira, GitHub) вдалося підтримувати прозорість у прийнятті рішень і контролювати якість реалізації кожного етапу.

У результаті проведених заходів було досягнуто **високої стабільності проєкту, мінімізовано ризики затримок та забезпечено відповідність кінцевого продукту технічним і дизайнерським вимогам**. Такий підхід дозволив сформувати надійну основу для подальшого вдосконалення системи та масштабування функціоналу.

Крім того, у процесі управління ризиками було впроваджено **систему безперервного моніторингу**, яка дозволяла своєчасно відстежувати зміни у стані проєкту та оперативно реагувати на будьякі відхилення від плану. Завдяки цьому команда мала змогу не лише усувати проблеми після їхнього виникнення, а й **попереджати їх появу**, що значно підвищило ефективність усієї розробки.

Важливим елементом стало також **використання метрик якості**, зокрема показників стабільності FPS, часу відгуку системи, рівня використання ресурсів (CPU, GPU, RAM), а також результатів автоматизованого тестування функціональних модулів. Ці дані аналізувалися на регулярній основі для визначення динаміки покращення якості продукту та оцінки впливу оптимізаційних рішень.

Додатково було реалізовано **механізм внутрішнього аудиту коду та дизайну**, що забезпечив уніфікацію підходів до реалізації функціоналу, підвищив читабельність і стабільність програмної бази. Така практика сприяла зниженню технічного боргу та спрощенню процесу масштабування проєкту.

Загалом, реалізована система управління ризиками та забезпечення якості продемонструвала свою ефективність у забезпеченні **прозорості, керованості та надійності процесу розробки**. Отримані результати підтверджують, що системний підхід до моніторингу, тестування й аналізу дозволяє не лише знизити кількість критичних помилок, а й сформувати **високий стандарт якості**, необхідний для стабільного функціонування інтерактивного продукту в умовах реального використання.

3.5. Кількісний аналіз ефективності технічних рішень

У рамках проєкту було проведено **кількісний аналіз ефективності технічних рішень**, спрямований на оцінку впливу оптимізацій та вибору інструментів розробки на загальну продуктивність системи. Основна мета цього етапу полягала у визначенні реальних показників швидкодії, стабільності кадрів (FPS) та рівня навантаження на апаратні ресурси (CPU, GPU) до і після впровадження оптимізаційних методів.

Під час тестування було здійснено **порівняння продуктивності проєкту до оптимізації та після застосування технологій Nanite і Lumen**, які забезпечили більш ефективне рендерингзавантаження та динамічне освітлення без втрати якості зображення. Згідно з отриманими результатами, середня частота кадрів зросла на 25–35%, а навантаження на процесор і графічний адаптер зменшилось приблизно на 20%. Це свідчить про доцільність застосування сучасних інструментів Unreal Engine 5 для балансування між якістю візуалізації та швидкодією.

Окрему увагу приділено **аналізу ефективності використання Blueprints проти C++ для реалізації критичних ігрових механік**. Результати експериментів показали, що логіка, реалізована мовою C++, демонструє вищу продуктивність у системах із великою кількістю об’єктів і складними обчисленнями, знижуючи час обробки подій на 10–15%. Водночас Blueprints

залишаються оптимальними для швидкої побудови прототипів, налаштування візуальних ефектів і розробки некритичних елементів ігрової логіки.

На основі отриманих даних було зроблено висновок, що **гібридний підхід**, який поєднує ефективність C++ для основних модулів і гнучкість Blueprints для візуальних налаштувань, є найбільш раціональним для досягнення високої продуктивності без втрати гнучкості в розробці. Такий підхід дозволяє оптимально використовувати ресурси системи, підтримувати стабільність FPS і забезпечувати комфортну взаємодію користувача з ігровим середовищем.

3.6. Контроль якості

Процес контролю якості та тестування механік управління і взаємодії у квестовій грі на Unreal Engine було налагоджено таким чином, що забезпечено стабільність та високу якість продукту. На всіх етапах розробки - від планування до фінальної перевірки - застосовувалися як автоматизовані, так і ручні методи тестування. Автоматизовані тести на базі Unreal Engine Automation Framework використовувалися для перевірки базових механік, а ручне тестування дозволило оцінити інтуїтивність управління та реакцію гравців. Проведено перевірки продуктивності (FPS, навантаження на систему), тестування ігрового балансу та внесено зміни на основі отриманого зворотного зв’язку.

Для координації процесу та відстеження завдань застосовувалися інструменти управління проєктами Jira і Trello, що забезпечило чітку послідовність тестування та контролю якості. Перевірено сумісність усіх компонентів проєкту, включно з інтеграцією 3Dмоделей, текстур, анімацій та Blueprints, що дозволило виявляти помилки на ранніх етапах. Особлива увага приділялася логіці взаємодії між персонажем і об’єктами середовища, а також поведінці системи у різних ігрових ситуаціях, що забезпечило уникнення критичних збоїв і підвищило занурення користувача.

Оптимізація механік управління та взаємодії включала аналіз продуктивності та плавності анімацій, налаштування рівнів деталізації (LOD)

моделей, оптимізацію навантаження на фізичний рушій та систему освітлення. Це забезпечило стабільну роботу гри на різних конфігураціях і підвищило загальне задоволення користувачів.

Комплексний підхід до контролю якості, тестування та оптимізації механік створив надійну основу для стабільного ігрового процесу та забезпечив високі стандарти професійної якості кінцевого продукту.

3.7. Оцінка користувацького досвіду (Usability Testing)

У межах реалізації даного проєкту було проведено **оцінку користувацького досвіду (Usability Testing)**, що мала на меті визначити рівень зручності, інтуїтивності та загальної задоволеності користувачів під час взаємодії з інтерактивним квестовим середовищем, створеним на Unreal Engine 5. Оскільки основним завданням проєкту було забезпечити природну ігрову взаємодію та логічну послідовність завдань, тестування юзабіліті стало важливим етапом для виявлення слабких місць у дизайні інтерфейсу, навігації та ігровій логіці.

Для проведення дослідження було обрано невелику групу користувачів - як з досвідом гри в подібні жанри, так і без нього, щоб оцінити реакцію різних категорій гравців. Учасникам запропонували виконати серію завдань у тестовій версії гри, включно з пересуванням персонажа, взаємодією з об’єктами, використанням інвентарю та виконанням квестових дій. Під час проходження відбувалося **спостереження за поведінкою гравців**, фіксувалися труднощі, моменти нерозуміння або затримки в реакції, а також суб’єктивні відгуки після завершення сеансу.

Для кількісної оцінки результатів застосовувалася **шкала Лікерта**, за якою користувачі оцінювали такі параметри, як зрозумілість інтерфейсу, логічність управління, візуальна привабливість та загальне задоволення ігровим процесом. За підсумками тестування середній показник задоволеності склав понад 4.5 з 5, що свідчить про високу якість користувацької взаємодії.

Водночас були визначені кілька аспектів, що потребують удосконалення: оптимізація чутливості керування камерою, покращення видимості підказок у темних сценах та адаптація інтерфейсу під різні роздільності екрана. На основі цих зауважень розроблено рекомендації, які вже частково впроваджено в оновленій збірці проєкту.

Таким чином, проведене **юзабілітітестування** дозволило не лише перевірити зручність ігрового процесу, але й сформувати цінний зворотний зв’язок для подальшої оптимізації UXдизайну, що є критично важливим для залучення та утримання користувачів у майбутньому релізі гри.

3.8. Аналіз ефективності реалізованих рішень

Проведений аналіз ефективності реалізованих рішень показав, що створена архітектура гри є добре структурованою, стабільною та зручною для подальшої розробки. Усі основні системи - логіка персонажа, механіка взаємодії, інтерфейс користувача, рендеринг і оптимізація продуктивності - взаємодіють між собою надійно та послідовно, забезпечуючи плавний ігровий досвід без технічних збоїв.

Одним із ключових результатів стало впровадження гнучкої системи взаємодій, побудованої на Blueprint Interface та подіях Overlap. Такий підхід дозволив створити універсальний механізм, який можна швидко розширювати - додавання нових типів об’єктів або поведінки не вимагає суттєвих змін у коді. Це особливо важливо для масштабування гри, коли кількість інтерактивних елементів постійно зростає. Крім того, модульність забезпечує зручність командної роботи - кожен елемент можна розробляти й тестувати окремо, не порушуючи загальну структуру проєкту.

Тестування показало, що реалізована система взаємодій добре сприймається гравцями: дії виконуються інтуїтивно, а підказки й інтерфейс допомагають швидко орієнтуватися у просторі. Завдяки оптимальному розташуванню елементів UI і використанню динамічних віджетів, інтерфейс не перевантажує екран, але водночас містить усю необхідну інформацію. Це

підвищує зручність користування та створює відчуття природності в ігровому процесі.

З технічного боку, реалізація рендерингу та освітлення довела свою ефективність у збереженні балансу між якістю графіки та стабільністю FPS. Налаштування параметрів тіней, глобального освітлення та LODмоделей дозволили досягти високої деталізації без перевантаження системних ресурсів. Особливо помітний результат спостерігається у великих сценах, де кількість динамічних об’єктів та ефектів є значною - гра зберігає плавність і стабільність роботи.

Крім того, використання інструментів Unreal Engine 5, зокрема системи Blueprints, продемонструвало високу ефективність у створенні складних логічних зв’язків без необхідності глибокого програмування. Це дозволило зосередитися на творчих аспектах - розробці атмосфери, інтерактивних елементів і візуальної складової гри.

Загалом, оцінка ефективності показала, що реалізовані рішення не лише відповідають поставленим цілям, а й створюють гнучку основу для подальшого розширення проєкту. Побудована система дозволяє швидко впроваджувати нові механіки, оптимізувати ресурси та підвищувати якість візуального й ігрового досвіду. Такий підхід формує міцний фундамент для подальшого розвитку ігрового продукту, орієнтованого на сучасні стандарти якості та зручності користувача.

3.9. Перспективи розвитку проєкту .

На поточному етапі проєкт демонструє стабільну функціональність, технічну узгодженість і помітний потенціал для подальшого масштабування. Проте, щоб перетворити його з навчального або демонстраційного прототипу на повноцінний ігровий продукт, необхідно розширити команду та залучити додаткові ресурси. Якщо початкові етапи - розробка основної логіки, створення базових механік, реалізація інтерфейсу та побудова базових сцен - могли

виконуватися невеликою групою з двохтрьох учасників, то подальший розвиток вимагає комплексного підходу і участі спеціалістів різних напрямів.

Подальша реалізація проєкту доцільна на базі спеціалізованої студії або науководослідної лабораторії, що має сучасну технічну та технологічну інфраструктуру. Такий формат забезпечить можливість повноцінного розвитку всіх компонентів гри - від проєктування рівнів і персонажів до тестування і оптимізації. Наявність потужних робочих станцій, високопродуктивних графічних процесорів, систем для обробки великих обсягів даних, а також обладнання для 3Dсканування та motion capture створить умови для глибшої деталізації та реалістичнішої візуалізації.

Крім того, впровадження VR/ARтехнологій дозволить розширити функціонал і додати нові можливості для інтерактивної взаємодії користувача з ігровим середовищем. Робота в умовах професійного виробництва відкриє доступ до інструментів командної розробки, систем керування версіями, серверних потужностей для тестування мультиплеєрних сценаріїв і засобів автоматизації рутинних процесів.

Така база дозволить перетворити проєкт із навчального чи демонстраційного прототипу на повноцінний високотехнологічний продукт, який відповідатиме сучасним стандартам ігрової індустрії. Вона забезпечить стабільність роботи, підвищену продуктивність рушія, ефективну оптимізацію контенту та можливість інтеграції новітніх технологічних рішень. У перспективі це створить передумови для розвитку власної екосистеми - від розробки до публікації та подальшої підтримки гри.

Крім того, важливо забезпечити належну організацію процесу розробки - впровадження системи керування версіями, регулярного тестування, аналітики користувацької взаємодії та оптимізації контенту. Це створить умови для стабільного розвитку проєкту та впровадження нових технологічних рішень. Значну роль відіграватиме і використання сучасних інструментів для генерації реалістичних текстур, процедурного моделювання, динамічного освітлення й

фізично коректного рендерингу, що забезпечить високий рівень реалістичності та занурення у віртуальне середовище.

Перспективним напрямом подальшого розвитку є розширення підтримки проєкту під різні платформи - не лише ПК, але й консолі, мобільні пристрої та VRшоломи. Це дозволить охопити ширшу аудиторію та збільшити комерційний потенціал. У майбутньому доцільним є додавання мережевих функцій, що відкриють можливість для кооперативної або соціальної взаємодії між гравцями. Такий підхід суттєво розширить ігровий досвід і підвищить інтерес користувачів.

Загалом подальший розвиток проєкту передбачає перехід від навчальної демонстраційної версії до повноцінного комерційного продукту, створеного з використанням професійних технологій, сучасного обладнання та командної співпраці. Це стане основою для реалізації творчого потенціалу, формування унікального користувацького досвіду та виходу проєкту на рівень сучасної ігрової індустрії.

Висновки до розділу 3

У третьому розділі було здійснено комплексний аналіз управління проєктом та оцінку ефективності прийнятих технічних і організаційних рішень у процесі розробки квестової гри. На основі застосування сучасних методологій (Agile, Scrum) вдалося сформувати гнучку та адаптивну модель керування, що забезпечила структуроване планування, ефективний розподіл обов’язків та систематичний контроль прогресу.

Особливу увагу приділено управлінню ризиками: їх ідентифікація, аналіз критичності та розробка планів реагування дозволили мінімізувати вплив потенційних проблем та забезпечити стабільність реалізації проєкту. Важливою складовою стало впровадження інструментів командної взаємодії та версійного контролю, що значно підвищило узгодженість роботи, зменшило кількість помилок під час інтеграції механік і забезпечило прозорість усіх етапів розробки.

Проведений кількісний і якісний аналіз ефективності технічних рішень продемонстрував, що обрана архітектура, методи оптимізації та інструменти Unreal Engine 5 забезпечили високу продуктивність, стабільність і масштабованість гри. Окремо оцінено користувацький досвід (Usability Testing), що дозволило підтвердити зручність розроблених систем управління, логіку взаємодії та якість реалізованих механік.

Таким чином, результати третього розділу підтверджують, що ефективне проєктне управління у поєднанні з технічно обґрунтованими рішеннями створює міцну основу для стабільності розробки, підвищує якість кінцевого продукту та забезпечує його відповідність сучасним вимогам і стандартам індустрії.

ВИСНОВКИ

У ході розробки проєкту, спрямованого на створення квестової гри з використанням рушія **Unreal Engine 5**, було реалізовано повний цикл виробництва - від аналізу, планування та визначення технічних вимог до розробки, тестування та оптимізації готових ігрових систем. Основна увага приділялася побудові інтерактивних механік, створенню користувацького інтерфейсу, забезпеченню стабільності роботи гри та досягненню високого рівня візуалізації, що відповідає сучасним стандартам ігрової індустрії.

Управління проєктом здійснювалося на основі принципів **ітераційного підходу (Iterative Design)** та елементів гнучких методологій **Agile** і **Scrum**, що дало змогу організувати процес розробки максимально ефективно. Проєкт розвивався через послідовність циклів планування, створення функціональних модулів, тестування й удосконалення. Такий підхід забезпечив гнучкість, можливість швидкого реагування на зміни у вимогах і постійне підвищення якості продукту. Крім того, це дозволило раціонально розподілити обов’язки між учасниками команди, контролювати хід виконання завдань і зменшити ризики затримок або технічних помилок.

У процесі реалізації проєкту активно застосовувалися принципи **UserCentered Design (UCD)**, що орієнтовані на створення зручної, інтуїтивної та логічної взаємодії користувача з грою. Завдяки цьому вдалося досягти гармонійного поєднання між технічними рішеннями, дизайном і користувацьким досвідом. Інтерфейс і механіки управління були побудовані так, щоб забезпечити комфортну взаємодію, зрозумілу логіку дій і природне занурення в ігровий процес.

Важливим аспектом стало також використання **системного підходу до планування та модульної архітектури**, що дозволило інтегрувати різні елементи проєкту - механіки управління, інвентар, діалогові системи, взаємодію

з об’єктами - в єдину узгоджену структуру. Це не лише спростило тестування та оптимізацію, але й створило основу для подальшого масштабування гри.

У результаті роботи було створено **функціональний прототип квестової гри**, який демонструє ефективність обраних методів управління, архітектурних рішень і технічних засобів. Отримані результати підтверджують доцільність використання гнучких підходів у процесі розробки, підкреслюють важливість поетапного планування, командної взаємодії та безперервного вдосконалення систем.

Таким чином, проєкт став не лише прикладом практичного впровадження сучасних технологій і методологій управління, але й підтвердженням того, що правильна організація процесу, ітераційний розвиток і орієнтація на користувача є ключовими чинниками успішної реалізації інтерактивних ігрових продуктів.

Розроблений прототип продемонстрував, що **Unreal Engine 5** є потужним інструментом для створення квестових ігор, який поєднує гнучкість візуального програмування з високою якістю графіки та продуктивністю.

Отримані результати мають практичну цінність і можуть бути використані як основа для подальшого розвитку повноцінного ігрового продукту. Створений проєкт дозволяє краще зрозуміти процес побудови інтерактивних систем, логіку роботи рушія та підходи до організації ігрової взаємодії.

У перспективі розширення проєкту передбачає інтеграцію нових механік, розробку додаткових рівнів, удосконалення діалогових систем, а також розробку VR/ARверсії для більш глибокого занурення користувача. Таким чином, виконана робота не лише підтвердила ефективність обраних технологій, але й заклала основу для подальших досліджень і розробок у сфері сучасних інтерактивних ігор.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. «The Art of Game Design: A Book of Lenses» (2020). URL: https://lnk.ua/aVp9GmkND (дата звернення: 05.02.2025).
2. A Theory of Fun for Game Design" (2013). URL: https://www.theoryoffun.com/theoryoffun.pdf (дата звернення: 13.02.2025).
3. "Designing Virtual Worlds" (2003). URL:https://lnk.ua/MVwoEvqVz (дата звернення: 20.02.2025).
4. Квест(відеоігри).URL:https://lnk.ua/YNg5Gg5eZ (дата звернення: 22.02.2025).
5. Epic Games URL: https://www.epicgames.com/site/enUS/home (дата звернення: 15.02.2025).
6. Unity Technologies URL: https://unity.com/ (дата звернення: 17.02.2025).
7. Game Development Trends. URL: https://lnk.ua/xNKQlmme8 (дата звернення: 25.02.2025).
8. Godot Engine URL: https://godotengine.org/ (дата звернення: 25.02.2025).
9. Indie Game Development (дата звернення: 25.02.2025).
10. Unreal Engine 5.5 Documentation. URL: https://lnk.ua/R4aMGrA4J (дата звернення: 08.02.2025).
11. Unity UI Guide. URL : https://docs.unity3d.com/Manual/UIToolkits.html (дата звернення: 15.02.2025).
12. Open Source Game Development Report URL: https://lnk.ua/aVp9GmQND (дата звернення: 13.02.2025).
13. Blueprint Scripting Guide. URL: https://lnk.ua/Men0Gn9Ng (дата звернення: 21.02.2025).
14. Game Design Workshop: A Playcentric Approach to Creating Innovative Games (дата звернення: 20.02.2025).
15. MDA: A Formal Approach to Game Design and Game Research" (Hunicke et al., 2004). (дата звернення: 20.02.2025).

16. Game Design: Theory and Practice. URL: https://lnk.ua/QV0k8W3eg (дата звернення: 18.02.2025).
17. (Adams & Dormans, 2012). Game Mechanics. (дата звернення: 25.02.2025).
18. Game Design Workshop: A Playcentric Approach to Creating Innovative Games. (дата звернення: 17.02.2025).
19. Unreal Engine. URL: https://en.wikipedia.org/wiki/Unreal_Engine (дата звернення: 25.02.2025).
20. Unity. URL: https://en.wikipedia.org/wiki/Unity_(game_engine) (дата звернення: 25.02.2025).
21. Unreal Engine VR. URL https://lnk.ua/5V1RZ3dNd (дата звернення: 25.02.2025).
22. Functional Testing. URL: https://lnk.ua/PeRl5vdNY (дата звернення: 25.02.2025).
23. Jira Documentation URL: https://confluence.atlassian.com/jira (дата звернення: 25.02.2025).
24. Методології управління проєктами, або Що таке Waterfall, Agile та Scrum URL: https://lnk.ua/be8Ay7yV5 (дата звернення: 25.02.2025).
25. Epic Games. *Unreal Engine 5 Documentation* [Електронний ресурс]. – Режим доступу: https://docs.unrealengine.com/ (дата звернення: 03.04.2025).
26. Unreal Engine Community. *Best Practices for Blueprint Visual Scripting in UE5* [Електронний ресурс]. – Epic Games, 2024. – Режим доступу: https://dev.epicgames.com/community (дата звернення: 03.04.2025).
27. Schell, J. *The Art of Game Design: A Book of Lenses*. – 3rd ed. – Boca Raton: CRC Press, 2019. – 600 с. (дата звернення: 03.04.2025).
28. Adams, E. *Fundamentals of Game Design*. – 4th ed. – New Riders, 2020. – 576 с. (дата звернення: 03.04.2025).

29. Fullerton, T. *Game Design Workshop: A Playcentric Approach to Creating Innovative Games*. – 4th ed. – CRC Press, 2018. – 560 с. (дата звернення: 24.04.2025).
30. Robins, N., Poland, J. *Learning C++ by Creating Games with Unreal Engine 5*. – Packt Publishing, 2022. – 500 с. (дата звернення: 24.04.2025).
31. Sharma, A. *Blueprints Visual Scripting for Unreal Engine 5*. – Packt Publishing, 2023. – 420 с. (дата звернення: 24.04.2025).
32. McCaffrey, J. *Beginning Game Development with Unreal Engine 5*. – Apress, 2023. – 382 с. (дата звернення: 24.04.2025).
33. Freeman, D. *Creating Emotion in Games: The Craft and Art of Emotioneering*. – New Riders, 2004. – 352 с. (дата звернення: 10.05.2025).
34. Novak, J. *Game Development Essentials: An Introduction*. – Delmar Cengage Learning, 2012. – 480 с. (дата звернення: 10.05.2025).
35. Pressman, R. S., Maxim, B. R. *Software Engineering: A Practitioner's Approach*. – 9th ed. – New York: McGrawHill, 2020. – 912 с. (дата звернення: 18.05.2025).
36. Schwaber, K., Sutherland, J. *The Scrum Guide* [Електронний ресурс]. – Scrum.org, 2020. – Режим доступу: https://scrumguides.org (дата звернення: 01.06.2025).
37. Highsmith, J. *Agile Project Management: Creating Innovative Products*. – 2nd ed. – AddisonWesley, 2013. – 272 с. (дата звернення: 01.06.2025).
38. ISO/IEC/IEEE 12207:2017. *Systems and Software Engineering - Software Life Cycle Processes*. – Geneva: ISO, 2017. – 78 с. (дата звернення: 01.06.2025).
39. Norman, D. A. *The Design of Everyday Things*. – Revised and expanded edition. – Cambridge: MIT Press, 2013. – 368 с. (дата звернення: 08.06.2025).
40. Nielsen, J. *Usability Engineering*. – San Francisco: Morgan Kaufmann, 2020. – 362 с. (дата звернення: 08.06.2025).
41. Salen, K., Zimmerman, E. *Rules of Play: Game Design Fundamentals*. – Cambridge: MIT Press, 2003. – 688 с. (дата звернення: 11.06.2025).

42. Alexander, C. *Notes on the Synthesis of Form*. – Harvard University Press, 2011. – 216 с. (дата звернення: 11.06.2025).
43. Epic Games. *Performance and Optimization Guidelines for Unreal Engine 5* [Електронний ресурс]. – Epic Games, 2024. – Режим доступу: https://dev.epicgames.com/documentation/enus/unrealengine/performance (дата звернення: 11.06.2025).
44. Totten, C. *An Architectural Approach to Level Design*. – Boca Raton: CRC Press, 2019. – 320 с. (дата звернення: 28.06.2025).
45. Rollings, A., Adams, E. *Game Architecture and Design*. – New Riders, 2006. – 896 с. (дата звернення: 10.09.2025).
46. Keith, C. *Agile Game Development with Scrum*. – 2nd ed. – AddisonWesley, 2020. – 464 с. (дата звернення: 10.09.2025).
47. O'Hara, K. *Developing Immersive Storytelling in Unreal Engine 5*. – Apress, 2023. – 305 с. (дата звернення: 10.09.2025).
48. Unity Technologies. *Game Design Patterns: Lessons for Interaction Design* [Електронний ресурс]. – Unity Technologies, 2022. – Режим доступу: https://unity.com/learn (дата звернення: 10.09.2025).
49. Lewis, J. R., Sauro, J. *Quantifying the User Experience: Practical Statistics for User Research*. – 3rd ed. – Morgan Kaufmann, 2020. – 348 с. (дата звернення: 10.09.2025).