Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА КООПЕРАТИВНОЇ ГРИ «MECHANICAL RUNAWAY» В ЖАНРІ ADVENTURE PUZZLE З ВИКОРИСТАННЯМ UNREAL ENGINE 5

DEVELOPMENT OF THE COOPERATIVE GAME «MECHANICAL RUNAWAY» IN THE ADVENTURE PUZZLE GENRE USING UNREAL ENGINE 5

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗс-21
Каліщук Станіслав Вікторович

Керівник:
Суринович Олена Миколаївна

Кваліфікаційну роботу
допущено до захисту
«10» 06 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *бакалавр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«20» 12 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Каліщуку Станіславу Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *«Розробка кооперативної гри «Mechanical Runaway» в жанрі Adventure Puzzle з використанням Unreal Engine 5»*
Керівник роботи: *к.т.н., доцент Суринович О. М.*
затверджені наказом закладу вищої освіти від *«19» грудня 2024 р.* № *474/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра *«10» червня 2025 р.*
3. Вихідні дані до роботи: *Unreal Engine 5, C++, Blueprint, UMG, Advanced Sessions Plugin, Steam, методичні вказівки до виконання кваліфікаційної роботи бакалавра.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *формулювання вимог до кооперативної гри в жанрі Adventure Puzzle, проєктування архітектури ігрової системи з урахуванням асиметричної взаємодії між гравцями, реалізація клієнтської та серверної частин із використанням Unreal Engine 5, Blueprint та C++, розробка мережевого функціоналу для обміну даними в реальному часі, створення геймплейних механік з урахуванням різних ролей персонажів, тестування основних сценаріїв взаємодії, підготовка інсталяційного пакета, створення супровідної документації, а також економічне обґрунтування розробки з оцінкою витрат і потенційної практичної цінності проєкту.*
5. Перелік графічного матеріалу: *20 рисунків.*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Суринович О. М.*		
Специфікація вимог до розробленої системи	*Суринович О. М.*		
Розробка об'єкта проектування	*Суринович О. М.*		
Нормоконтроль	*Вознюк А. В.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	1,40 %		
Академічна доброчесність	*Суринович О. М.*		

7. Дата видачі завдання *«20» грудня 2024 р.*

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 04.02.2025 р.	вик.
2	Аналіз проблеми розробки та впровадження об'єкту проектування	до 01.03.2025 р.	вик.
3	Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 15.03.2025 р.	вик.
4	Розробка функціонально-структурної схеми роботи об'єкта проектування	до 29.03.2025 р.	вик.
5	Практична реалізація об'єкта проектування	до 26.04.2025 р.	вик.
6	Тестування та налагодження об'єкта проектування	до 03.05.2025 р.	вик.
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру	до 10.06.2025 р.	вик.

Здобувач вищої освіти _____ Станіслав КАЛІЩУК

Керівник кваліфікаційної роботи _____ Олена СУРИНОВИЧ

АНОТАЦІЯ

Каліщук С. В. Розробка кооперативної гри «Mechanical Runaway» в жанрі Adventure Puzzle з використанням Unreal Engine 5. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності 121 «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків та списку використаних джерел.

У першому розділі проведено аналіз предметної області кооперативних ігор жанру Adventure Puzzle, розглянуто приклади реалізації подібних проєктів, а також сформульовано технічні та геймплейні вимоги до майбутнього прототипу. У другому розділі спроєктовано архітектуру ігрової системи, розроблено діаграми взаємодії та обґрунтовано вибір технологій, зокрема рушія Unreal Engine 5 та використання мережевої взаємодії. У третьому розділі реалізовано ігровий прототип з підтримкою мережевого мультиплеєра, впроваджено основні геймплейні механіки, протестовано ключові сценарії взаємодії, підготовлено супровідну документацію, інсталяційний пакет і здійснено економічне обґрунтування розробки. У висновках підбито підсумки виконаної роботи та запропоновано напрямки подальшого вдосконалення гри.

Ключові слова: кооперативна гра, Unreal Engine 5, C++, Adventure Puzzle, Blueprints, мережевий мультиплеєр, асиметрична взаємодія.

ABSTRACT

Kalishchuk S. Development of the Cooperative Game "Mechanical Runaway" in the Adventure Puzzle Genre Using Unreal Engine 5. Manuscript. Qualification work of the bachelor's program "Software engineering" in the specialty "Software engineering". Lutsk National Technical University. Lutsk, 2025.

The bachelor's qualification work consists of an introduction, three chapters, conclusions and a list of references.

The first chapter analyzes the subject area of cooperative games of the Adventure Puzzle genre, considers examples of similar projects, and formulates technical and gameplay requirements for the future prototype. The second section designs the architecture of the game system, develops interaction diagrams, and justifies the choice of technologies, including the Unreal Engine 5 engine and the use of network interaction. In the third section, a game prototype with support for online multiplayer is implemented, the main gameplay mechanics are implemented, key interaction scenarios are tested, accompanying documentation and an installation package are prepared, and an economic justification for the development is carried out. The conclusions summarize the results of the work done and suggest areas for further improvement of the game.

Keywords: cooperative game, Unreal Engine 5, C++, Adventure Puzzle, Blueprints, online multiplayer, asymmetric interaction.

ЗМІСТ

ВСТУП

Актуальність теми. Індустрія відеоігор є однією з найдинамічніших галузей цифрових технологій, що постійно розвивається і вдосконалюється. Особливу популярність останніми роками здобули кооперативні ігри, які пропонують гравцям досвід спільного проходження, взаємодії та розв’язання завдань у команді. Поєднання жанрів adventure та puzzle у межах кооперативного середовища створює унікальні виклики як для користувачів, так і для розробників: асиметрична взаємодія між персонажами, необхідність синхронної роботи в режимі реального часу та балансування геймплею вимагають глибокого проектування як на рівні дизайну, так і архітектури програмного коду.

Сучасні ігрові рушії, зокрема Unreal Engine 5, забезпечують широкі можливості для реалізації складних механік, як-от фізичний рух персонажів, передача подій по мережі, розподіл ролей і гнучке налаштування середовища розробки. У таких умовах особливої уваги потребує питання побудови мережевої взаємодії – це фундамент для стабільної кооперативної гри, що залежить від синхронізації даних, швидкості обробки подій та злагодженості дій усіх гравців. Актуальність розробки полягає в створенні гнучкої архітектури, яка дозволяє не лише швидко прототипувати геймплей, а й забезпечує масштабованість та адаптацію під нові сценарії.

Кваліфікаційна робота присвячена розробці ігрового прототипу «Mechanical Runaway» – кооперативної гри в жанрі puzzle adventure, де гравці виконують ролі двох різнопланових персонажів з унікальними здібностями. У процесі реалізації було вирішено низку технічних задач, зокрема розроблено базову архітектуру системи, реалізовано передачу даних по мережі, впроваджено асиметричні ігрові ролі, а також створено рівень з логічними задачами для перевірки командної взаємодії гравців.

Об’єктом є інформаційно-комп’ютерна система кооперативної гри.

Предметом, виступають архітектурні рішення, геймплейні механіки та технології реалізації мережевої взаємодії у рамках проєкту на базі Unreal Engine

Отже, основною метою цієї кваліфікаційної роботи є розробка ігрового прототипу з базовими можливостями онлайн кооперації.

Для досягнення поставленої мети було визначено такі завдання:

– провести аналіз сучасних тенденцій у жанрі кооперативних adventure/puzzle ігор;

– спроєктувати архітектуру ігрової системи з урахуванням асиметричної взаємодії гравців;

– реалізувати прототип рівня з інтегрованими механіками для обох типів персонажів;

– налаштувати та протестувати мережеву взаємодію між гравцями;

– підготувати супровідну документацію, інсталяційний пакет і провести тестування прототипу;

– визначити витрати та можливості комерціалізації.

Таким чином, розробка цього проєкту дозволяє поєднати як теоретичні аспекти інженерії програмного забезпечення, так і практичну реалізацію сучасних геймдев-технологій для створення функціонального, ефективного ігрового прототипу.

РОЗДІЛ 1
АНАЛІЗ РОЗРОБКИ ІГОР І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

Сфера відеоігор перебуває в постійному динамічному розвитку, охоплюючи все ширший спектр жанрів та ігрових механік. Одним із напрямів, що набирає популярності, є жанр Adventure Puzzle, особливо в поєднанні з кооперативною взаємодією. Цей жанр поєднує елементи дослідження, вирішення головоломок і сюжетної лінії, занурюючи гравця в наративне середовище, де взаємодія між учасниками є ключовою для досягнення спільної мети.

Особливістю кооперативних puzzle-ігор – назва жанру відеоігор, метою яких є вирішення логічних завдань, що вимагають від гравця задіяння логіки, стратегії, інтуїції та іноді ерудиції й уважності [1], є асиметричний геймплей, де кожен гравець виконує унікальну роль, що вимагає тісної співпраці, обміну інформацією, координації дій та прийняття рішень у реальному часі.

На формування концепції гри «Mechanical Runaway» значний вплив мали такі представники індустрії, як It Takes Two, Portal 2 (режим Co-op) та серія We Were Here. Вони продемонстрували ефективність використання обов’язкової кооперації, різнопланових механік і асиметричного дизайну рівнів. Ці ігри відзначаються оригінальністю завдань, якісною візуалізацією, продуманою режисурою та високим рівнем взаємодії між гравцями. Проте, попри очевидні переваги, у багатьох з них простежуються недоліки, як-от обмежена кастомізація персонажів, складність впровадження багатокористувацької гри – тип відеоігор або їхня складова, в яку одночасно грають від двох і більше гравців [2], без затримок або прив’язки до окремих платформ.

Одним із технологічних рішень, що відкриває широкі можливості для розробки сучасних кооперативних ігор, є Unreal Engine 5. Unreal Engine – ігровий рушій, який розробляється та підтримується компанією Epic Games [3]. Цей

рушій вирізняється високим рівнем деталізації графіки, підтримкою динамічного освітлення (завдяки технології Lumen – це повністю динамічна система глобального освітлення та відбиттів Unreal Engine 5 [4]), а також надає розробникам інструменти для швидкої реалізації логіки гри через Blueprint – це повноцінна система скриптів ігрового процесу, заснована на концепції використання інтерфейсу на основі вузлів для створення елементів ігрового процесу в Unreal Editor [5]. Компоненти мережевої взаємодії дозволяють реалізовувати як локальну кооперацію (LAN – є об'єднанням певного числа комп'ютерів на відносно невеликій території [6]), так і потенційне розширення до інтеграції з платформами на кшталт Steam – сервіс компанії Valve, відомого розробника відеоігор, який надає послуги цифрової дистрибуції, багатокористувацьких ігор і спілкування гравців [7]. Безкоштовна ліцензія для інді-розробників, активна спільнота та велика кількість плагінів додатково підвищують привабливість рушія.

Аналіз відкритих наукових джерел у сфері ігрового дизайну підтверджує, що більшість досліджень зосереджені на аспектах психології гравців, побудові рівнів, балансі між учасниками та механізмах комунікації. Водночас, майже жоден із розглянутих прикладів не поєднує саме ті технічні та дизайнерські рішення, які закладені у «Mechanical Runaway». Зокрема, ідея двох гравців із радикально різними фізичними моделями поведінки – сферичний наземний робот та літаючий дрон – створює новий виклик у побудові головоломок і архітектурі рівнів. Додатково, низка інтерактивних об'єктів у грі активуються лише за умов синхронної присутності обох гравців, що суттєво ускладнює як розробку, так і проходження.

Попри існування успішних зразків, більшість ігор цього напряму належать до категорій AAA – умовна підмножина відеоігор, створюваних й розповсюджуваних середніми й великими видавництвами, що зазвичай мають більше коштів на розробку й рекламу [8] або AA проєктів і вимагають значних фінансових і людських ресурсів. У межах цієї кваліфікаційної роботи ставиться за мету створення прототипу, який демонструє функціональність ключових

механік, їхню взаємодію та потенціал для подальшого масштабування. Такий підхід дозволяє перевірити життєздатність ігрової ідеї на практиці, з урахуванням сучасних інструментів розробки та актуальних вимог до якості користувацького досвіду.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Метою кваліфікаційної роботи є створення прототипу кооперативної пригодницької гри «Mechanical Runaway» у жанрі Adventure Puzzle на базі рушія Unreal Engine 5.

Для досягнення поставленої мети було визначено такі завдання:

– провести аналіз сучасних тенденцій у жанрі кооперативних adventure/puzzle ігор;

– спроєктувати архітектуру ігрової системи з урахуванням асиметричної взаємодії гравців;

– реалізувати прототип рівня з інтегрованими механіками для обох типів персонажів;

– налаштувати та протестувати мережеву взаємодію між гравцями;

– підготувати супровідну документацію, інсталяційний пакет і провести тестування прототипу;

– визначити витрати та можливості комерціалізації.

На першому етапі передбачається проведення аналітичного дослідження предметної області. Слід вивчити сучасний стан індустрії кооперативних ігор в жанрі Puzzle Adventure, проаналізувати успішні реалізації ігрових механік, виявити сильні сторони конкурентних проєктів та їхні недоліки. Окрему увагу буде приділено питанням побудови геймплею в умовах асиметричної взаємодії між гравцями та формуванням дизайну рівнів для двох різних типів персонажів.

Другим етапом передбачено проектування архітектури ігрової системи. У межах цього етапу необхідно побудувати UML діаграми – уніфікована мова

моделювання, використовується у парадигмі об’єктно-орієнтованого програмування [9], які відображають структуру основних класів, їхні взаємозв’язки, а також сценарії взаємодії гравців із внутрішніми механіками гри. Додатково, на цьому етапі слід обґрунтувати вибір інструментів і технологій, що будуть використані у проєкті – зокрема, обрання рушія Unreal Engine 5, застосування Blueprint для логіки, можливість використання C++ – мова програмування загального призначення з підтримкою кількох парадигм програмування: об’єктно-орієнтованої, узагальненої, процедурної та інших [10], для розширення функціональності, а також визначення підходів до реалізації мережевої взаємодії між гравцями. Це забезпечить технічну стабільність, масштабованість і адаптивність майбутнього ігрового прототипу.

Наступним кроком є безпосередня реалізація ігрового прототипу «Mechanical Runaway». Потрібно розробити мережевий код на базі можливостей Unreal Engine 5, який забезпечить стабільне з’єднання між клієнтами, передачу та синхронізацію ігрових даних у режимі реального часу. Важливо впровадити механізми обробки мережевих подій та оновлення ігрового інтерфейсу, що дозволить гравцям отримувати актуальну інформацію про стан гри і дії партнерів. Паралельно з цим буде здійснюватися розробка геймплейної логіки, що включає реалізацію основних механік кооперативної гри в жанрі adventure puzzle та забезпечення взаємодії між персонажами з різними ролями.

На етапі тестування необхідно перевірити всі ключові сценарії кооперативної взаємодії гравців, зокрема підключення до мережевої сесії, передачу ігрових подій у реальному часі, синхронізацію об’єктів та їхніх станів між клієнтами. Увага приділяється і перевірці стійкості мережевого коду до типових збоїв, таких як затримки зв’язку, нестабільне підключення або втрата даних.

У межах наступного етапу необхідно створити супровідну документацію, яка забезпечить зрозумілість використання та підтримки ігрового прототипу «Mechanical Runaway».

На цьому етапі необхідно створити інсталяційний пакет для розгортання ігрового прототипу «Mechanical Runaway» на кінцевих системах. Основним завданням є підготовка повноцінної збірки гри з усіма необхідними файлами та ресурсами, зібраної за допомогою засобів пакування Unreal Engine 5.

Завершальним етапом кваліфікаційної роботи є визначення витрат та можливості комерціалізації. На цьому етапі здійснюється оцінка витрат ресурсів, необхідних для реалізації проєкту, а також аналіз доцільності та ефективності використання отриманого результату з урахуванням потенційної практичної або комерційної цінності.

Висновки до розділу 1

У результаті аналізу сучасного стану індустрії відеоігор у жанрі Adventure Puzzle з кооперативним геймплеєм було встановлено, що попит на подібні проєкти залишається стабільно високим, особливо серед ігор, які пропонують нестандартну взаємодію між гравцями, асиметричний геймплей та логічні головоломки.

Розглянуто найбільш відомі аналоги (It Takes Two, Portal 2, We Were Here), визначено їх сильні сторони та обмеження. Зроблено висновок про актуальність створення власного проєкту, що поєднує унікальні ігрові механіки, оригінальний дизайн персонажів і взаємодію, засновану на фізичних властивостях об’єктів.

Проаналізовано можливості сучасного рушія Unreal Engine 5, який надає всі необхідні інструменти для реалізації поставлених завдань, зокрема візуальне програмування, системи мережевої взаємодії та фізичної симуляції.

У підсумку сформульовано основні технічні та геймдизайнерські завдання, що мають бути реалізовані в межах кваліфікаційної роботи. Результатом стане прототип вступного рівня гри «Mechanical Runaway», який відображає концепцію майбутньої гри та демонструє ключові аспекти кооперативного геймплею.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

Для розробки ігрової системи було обрано ітераційну (інкрементну) модель розробки, яка дозволяє поступово додавати функціональність до прототипу гри, тестуючи і вдосконалюючи її на кожному етапі. Такий підхід є ефективним у розробці ігор, де важливо тестувати геймплейні механіки ще до повного завершення проєкту. У рамках цієї моделі спочатку розробляється ядро гри – головне меню, підключення гравців, базова логіка одного рівня, після чого до нього поетапно додаються функціональні можливості.

Для визначення вимог до програмного забезпечення використовувалися наступні методи:

1) аналіз ігрових аналогів та конкурентів (It Takes Two, Portal 2, We Were Here);

2) сценарний підхід (use-case) – опис поведінки системи, як вона відповідає на зовнішні запити [11];

3) опитування потенційних користувачів – неформальні консультації з іншими геймерами-розробниками щодо очікувань від кооперативної гри;

4) особистий досвід – як гравця і розробника.

У процесі розробки ігрового прототипу «Mechanical Runaway» висувається низка функціональних, нефункціональних вимог і обмежень, які визначають рамки реалізації проєкту.

Функціональні вимоги:

– гравець повинен мати змогу створити власний профіль, який включатиме нікнейм і аватар;

– у грі має бути реалізована система головного меню для навігації між основними розділами;

– необхідно забезпечити підтримку мережевої кооперативної гри через локальну мережу (LAN), з можливістю подальшої інтеграції зі Steam;

– гравці повинні мати змогу обирати між двома персонажами: літаючим дроном і наземним роботом-кулькою;

– у грі має бути реалізований внутрішньоігровий текстовий чат для комунікації між гравцями;

– обов’язковим є створення вступного рівня, що містить кооперативні головоломки, які потребують взаємодії обох гравців;

– повинна бути реалізована система виявлення взаємодії з елементами рівня, такими як натискні плити, кнопки, декалі, а також переміщувані об’єкти (наприклад, куби).

Нефункціональні вимоги:

– гра повинна стабільно функціонувати в середовищі Unreal Engine 5;

– інтерфейс має бути простим і інтуїтивно зрозумілим для користувача;

– мережева взаємодія повинна мати мінімальну затримку для забезпечення комфортної гри;

– кросплатформеність не є обов’язковою вимогою, головною платформою для запуску проєкту є Windows – група сімейств комерційних пропрієтарних операційних систем корпорації Microsoft, орієнтованих управління за допомогою графічного інтерфейсу [12].

Обмеження:

– у межах проєкту реалізується лише один рівень, який демонструє основні ігрові механіки та кооперативну взаємодію;

– складні анімації, кінематографічні сцени (катсцени) та повноцінна сюжетна лінія в межах поточної реалізації не передбачені.

У процесі проєктування програмної системи для демонстраційного рівня гри Mechanical Runaway було прийнято рішення використовувати уніфіковану мову моделювання UML (Unified Modeling Language). Такий вибір обумовлений прагненням до структурованості, наочності та формалізованого підходу до опису

системи, що є особливо важливим у розробці навіть невеликого ігрового проєкту, орієнтованого на кооперативну взаємодію.

UML дозволяє зручно подати як статичні, так і динамічні аспекти програмної системи, виявити потенційні слабкі місця ще до етапу реалізації, узгодити архітектуру між різними компонентами проєкту, а також служить корисним інструментом у майбутньому – при масштабуванні гри чи залученні інших розробників. Загалом у межах проєкту було використано три UML-діаграми.

Першим кроком у моделюванні взаємодії гравців у межах проєкту «Mechanical Runaway» стала побудова UML-діаграми варіантів використання (рис. 2.1). Ця діаграма дозволяє наочно представити основні сценарії взаємодії кінцевих користувачів із системою ще до початку реалізації геймплейної логіки. Основними «акторами» на діаграмі виступають два гравці: Player 1 (Host) – ініціатор ігрової сесії, який виконує роль сервера, та Player 2 (Client) – гравець, який приєднується до створеної сесії як клієнт. Така структура відповідає типовій архітектурі Listen Server у ігрових мережах.

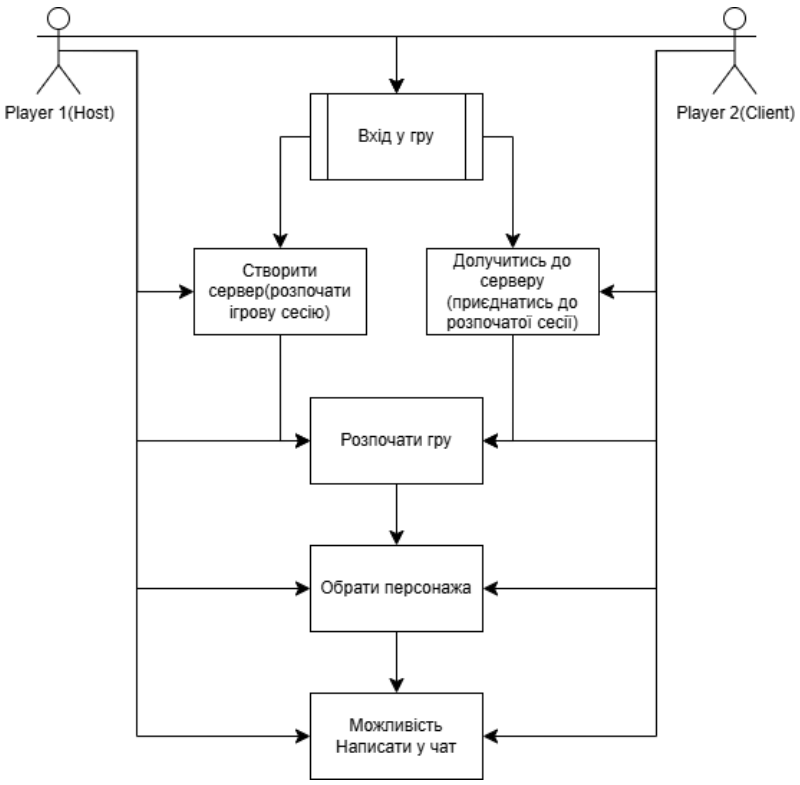

Рисунок 2.1 – UML діаграма варіантів використання

Процес взаємодії починається з запуску гри кожним з гравців. Якщо користувач перед цим не авторизувався на платформі Steam, система запропонує ввести власне ім’я гравця (нікнейм) та вибрати аватар у налаштуваннях профілю. У випадку попередньої авторизації в Steam, дані будуть автоматично імпортовані з акаунта – це спрощує початкову взаємодію та покращує персоналізацію.

Наступним кроком є вибір ролі – гравець має вирішити, чи він бажає створити нову сесію (стати хостом), чи приєднатися до існуючої (стати клієнтом). У головному меню гри передбачено окремі кнопки для обох сценаріїв, що ведуть до відповідних підменю. Якщо обирається варіант створення сесії, гравець повинен: вибрати тип мережевого з’єднання, а також ввести назву сесії, що дозволить іншим гравцям знайти її.

Якщо ж гравець обирає приєднання до сесії, йому потрібно буде лише знайти наявний сервер. Такий підхід забезпечує гнучкість і зручність мережевої взаємодії без складного введення адрес або технічних даних.

Після встановлення з’єднання між двома гравцями, кожен з них переходить до етапу вибору персонажа. В грі передбачено дві ролі: літаючий робот та сферичний робот, кожен з яких має унікальні фізичні характеристики та взаємодії з ігровим світом. Ігрова логіка побудована таким чином, що вибір персонажа є унікальним – один і той самий персонаж не може бути обраний обома гравцями одночасно. Такий механізм виключає можливість конфліктів або помилок під час розподілу ролей і забезпечує збалансований кооперативний геймплей.

Окрім функціональних дій, пов’язаних із з’єднанням та налаштуванням, в систему також інтегровано внутрішньо ігровий чат. Гравці можуть користуватись ним як до початку рівня, так і під час гри. Це дозволяє краще координувати дії, обговорювати вибір персонажа або вирішувати технічні моменти, не покидаючи ігровий клієнт. Така функціональність є важливою частиною кооперативного UX, оскільки створює комфортне комунікаційне середовище, що сприяє ефективнішій грі та зниженню стресу під час спільного проходження рівня.

Дана діаграма надає можливість мені як розробнику бачити більш чітку картину що до необхідного функціоналу, який потрібно реалізувати в першу чергу.

Другим кроком у процесі моделювання стала розробка UML діаграми класів – статичне представлення структури моделі в UML [13], яка відображає структуру головного меню гри Mechanical Runaway та пов’язаних з ним компонентів (рис. 2.2). Цей тип діаграми дозволяє детально проаналізувати взаємозв’язки між класами, їхню внутрішню будову (методи, поля, атрибути) та логіку взаємодії між модулями системи. У контексті реалізації інтерфейсу користувача така діаграма є надзвичайно цінною, оскільки вона демонструє, як саме компоненти GUI пов’язані між собою та з ігровою логікою.

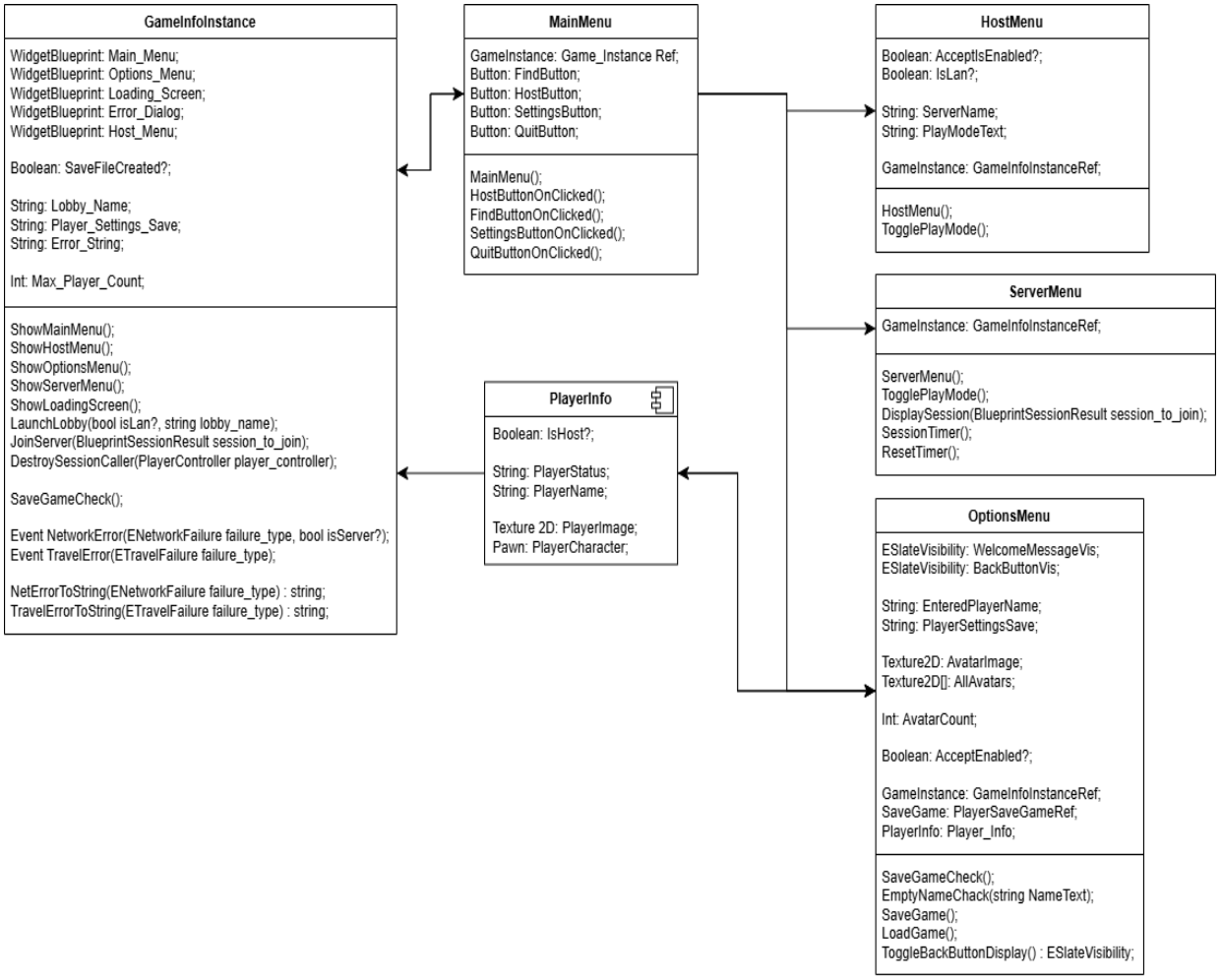

Рисунок 2.2 – UML-діаграми класів

Центральне місце в даній діаграмі займає клас GameInfoInstance, який наслідується від базового класу GameInstance – стандартного для архітектури Unreal Engine 5. Саме GameInfoInstance виступає головним системним менеджером проєкту. Його життєвий цикл охоплює всю тривалість роботи гри – від моменту її запуску до повного завершення сесії. У цьому класі зберігається критично важлива інформація:

- статус підключення гравця до сервера;
- обраний нікнейм та аватар;
- роль користувача (host/client);
- параметри з’єднання;
- структура даних профілю тощо.

Цей клас виступає посередником між меню та логікою з’єднання, об’єднуючи UI-рівень з мережевими ігровими механіками.

Наступним елементом діаграми є клас MainMenu – основне головне меню гри. Воно містить у собі посилання на GameInfoInstance, з якого отримує необхідні дані для побудови інтерфейсу. Клас MainMenu також містить змінні, які представляють графічні елементи – Button-об’єкти. У кожної кнопки є прив’язаний метод обробки подій (event handler), який викликається при натисканні: наприклад, запуск гри, перехід до меню налаштувань, вихід з програми тощо. Всі ці дії ініціюються відповідними callback-функціями, прописаними у методах класу.

Клас HostMenu є логічним продовженням MainMenu, однак виконує більш специфічну функцію. Він відповідає за налаштування створення нової сесії, зокрема:

- введення імені майбутнього сервера;
- вибір типу з’єднання;
- збереження параметрів сесії до моменту запуску гри.

У зв’язку з тим, що HostMenu є переважно автономним модулем, він не взаємодіє безпосередньо з іншими класами діаграми, окрім GameInfoInstance, через який здійснюється обмін параметрами.

Клас ServerMenu виконує протилежну функцію – він призначений для пошуку вже існуючих сесій, створених іншими гравцями (у режимі клієнта). Його основна задача – запустити пошук гри та відобразити статус можливості приєднання гравця до гри. Структурно він є простішим за HostMenu і містить менше змінних, оскільки не потребує конфігурації параметрів сесії.

Найцікавішим з представлених у діаграмі є клас ProfileSettingsMenu. Він містить найбільшу кількість функцій і виконує декілька важливих ролей у системі. Його логіка викликається у двох ключових випадках:

– коли користувач самостійно заходить у меню налаштувань гри;

– коли система не змогла автоматично ідентифікувати гравця – наприклад, якщо Steam неактивний, і дані профілю (нік, аватар) не були підвантажені.

У такому випадку ProfileSettingsMenu автоматично активується та дозволяє користувачеві вручну ввести свою інформацію, що є важливою умовою переходу до гри.

Усі перераховані класи об’єднуються загальною логікою з використанням структури даних PlayerInfo. Вона є універсальною моделлю профілю гравця, що містить базові поля: ім’я, аватар, роль, статус підключення. PlayerInfo використовується на всіх етапах – від запуску гри до створення або приєднання до мережевої сесії, а також передається між меню та ігровим середовищем.

Таким чином, UML-діаграма класів не лише відображає структуру меню та його взаємодію з іншими елементами системи, але й слугує наочним шаблоном для реалізації ключових функцій початкової навігації в грі. Вона дозволяє уникнути дублювання логіки, забезпечити гнучкість інтерфейсу та закласти фундамент для подальшої розширюваності.

Третьою складовою моделювання стало створення UML діаграми діяльності – в UML та SysML, візуальне представлення графу діяльностей [14], яка відображає послідовність внутрішніх подій у грі Mechanical Runaway до початку геймплейної частини. Дана діаграма є важливою частиною функціонального проєктування, оскільки дозволяє змоделювати початкову

логіку поведінки гри залежно від виборів, які здійснює гравець у головному меню (рис. 2.3).

Основною метою цієї діаграми є демонстрація умовного потоку рішень, що впливають на початкову взаємодію гравця із системою: від перевірки наявності профілю – до підключення до мережевої гри.

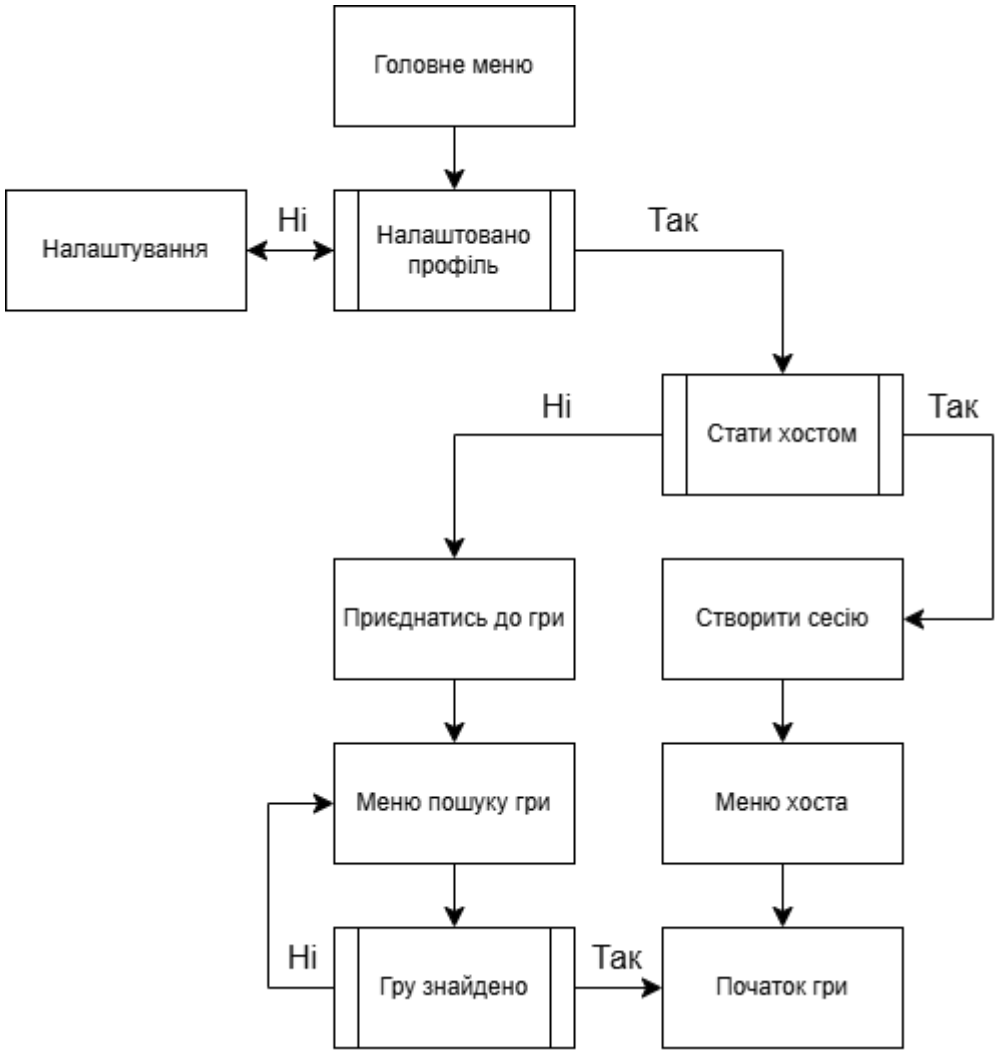

Рисунок 2.3 – UML-діаграма діяльності

Після запуску гри користувач потрапляє до головного меню. На цьому етапі система перевіряє, чи існують у користувача необхідні ідентифікатори: ім’я та аватар. Якщо ці дані були отримані з ігрової платформи (наприклад, Steam), або введені раніше – гравець може одразу перейти до вибору між двома опціями: створити новий сервер або приєднатися до вже існуючої сесії.

Якщо ж профіль гравця ще не налаштований, або авторизація не відбулася, система автоматично перенаправляє його до меню налаштувань профілю. Тут користувач має ввести нікнейм, обрати аватар – і лише після цього йому стають доступні подальші функції.

Подальша логіка розгалужується залежно від вибору ролі. Якщо гравець вирішує стати хостом, відбувається перехід до меню створення сесії. У цьому вікні гравець має змогу:

– задати назву майбутнього сервера;

– обрати тип мережевого з’єднання;

– розпочати сесію, після чого гра переходить у стан очікування клієнта.

У цей момент система працює як сервер, а гравець – як ініціатор сеансу. Програма постійно перевіряє, чи підключився другий гравець. Як тільки це відбувається, можна перейти до вибору персонажів та початку рівня.

У разі, якщо гравець обирає підключення до сесії, відбувається перехід до меню пошуку сесій. Тут користувач може ініціювати сканування активних серверів. Якщо гра знаходить принаймні одну активну сесію – вона виводиться. Гравець може:

– приєднатись до знайденої сесії;

– відхилити з’єднання і продовжити пошук;

– повторно запустити пошук, якщо сесій не знайдено.

Після успішного з’єднання з хостом клієнт також переходить до меню вибору персонажа, після чого обидва гравці завантажують рівень.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Для реалізації демонстраційного рівня кооперативної гри «Mechanical Runaway» було обрано набір сучасних інструментів, які дозволяють ефективно

реалізувати як геймплейну логіку, так і візуальну складову проєкту. Основним рушієм, на базі якого побудована уся система, є Unreal Engine 5 – один із найпотужніших інструментів для створення інтерактивного 3D-контенту, який активно використовується як у ігровій індустрії, так і в суміжних сферах (віртуальне виробництво, архітектура, симуляції тощо). Разом із UE5, у роботі використовувались допоміжні програми: Blender – програмний пакет для створення тривимірної комп'ютерної графіки, що включає засоби моделювання, анімації, рендерингу, після-обробки відео [15] та Substance Painter – гнучкість та універсальність цього інструменту ініціювала появу більш експресивних візуальних стилістик на ринку та, подекуди, деякий шифт у взаємодії не тільки між художниками, а й технічними спеціалістами [16].

Вибір Unreal Engine 5 був обумовлений насамперед його багатофункціональністю та наявністю вбудованих рішень для більшості потреб сучасної гри. Однією з ключових переваг рушія є можливість працювати з візуальним програмуванням через Blueprints, що дає змогу розробляти ігрову логіку без написання великої кількості коду. У межах дипломного проєкту це стало вирішальним чинником, адже дозволило сфокусуватися на реалізації механік без необхідності створювати складні системи з нуля.

Ще однією перевагою UE5 є система User Widget Blueprint, яка забезпечує повноцінну підтримку розробки інтерфейсу користувача. Завдяки цьому можна створювати адаптивні меню, чат, екрани налаштувань, вікна вибору персонажа та інші елементи UI, які не лише візуально привабливі, але й функціонально гнучкі. У межах демонстраційного рівня ця можливість була активно використана для створення головного меню, інтерфейсу вибору ролі, системи авторизації гравця (нік, аватар), а також внутрішньоігрового чату.

Окремо варто згадати мережеву архітектуру UE5, яка дозволяє реалізовувати мультиплеєрні ігри на основі моделі «сервер-клієнт» з використанням внутрішніх засобів реплікації. У грі «Mechanical Runaway» ця система забезпечує обмін даними між двома гравцями в режимі реального часу: переміщення, активація тригерів, стан головоломок – усе це синхронізується між

клієнтом і хостом без значних затримок. Завдяки механізму реплікації Unreal Engine автоматично керує передачею змін між гравцями, дозволяючи уникнути дублювання логіки або неузгодженостей під час гри.

Окрім логіки, рушій UE5 забезпечує надзвичайно потужні графічні можливості. Візуальна складова гри є однією з ключових частин гравецького досвіду, і завдяки таким технологіям, як Nanite (віртуалізована геометрія) та Lumen (глобальне освітлення в реальному часі), навіть невеликі сцени можуть виглядати професійно. У роботі ці технології використовувалися для створення динамічного освітлення кімнат, світлових ефектів під час активації головоломок та реалістичного відображення матеріалів. Гравець бачить результат своїх дій у візуально переконливому середовищі, що підвищує занурення в гру.

Для створення моделей персонажів та об’єктів було використано Blender – вільний інструмент 3D-моделювання, що має підтримку імпорту в Unreal Engine. У рамках демонстраційного рівня в Blender були змодельовані як декоративні, так і ігрові об’єкти: елементи інтер’єру, натискні плити, фізичні платформи та частини механізмів головоломок. Усі елементи, що належать до моделей, були спроєктовані та відтворені самостійно в даному 3D-редакторі.

Для досягнення високої деталізації текстур було використано Substance Painter. Цей інструмент дозволив наносити на 3D-моделі текстури з підтримкою PBR (physically-based rendering), що забезпечує матеріалам максимально наближене до реальності відображення при освітленні. Зокрема, за його допомогою були створені текстури металевих поверхонь, елементів керування, ґрат, світлових індикаторів тощо. У поєднанні з можливостями Lumen це дозволило отримати дійсно насичену й привабливу картинку навіть у рамках невеликого прототипу.

З точки зору реалізації алгоритмів, у грі використано як подійно-орієнтовану, так і стано-залежну логіку. Наприклад, відкриття дверей реалізується через послідовність активації тригерів від двох різних гравців, а система вибору персонажа використовує контроль доступності для запобігання вибору одного й того ж героя обома гравцями одночасно. Чат синхронізується

через репліковані змінні, а введення даних гравця (нік, аватар) зберігається через спеціальний об’єкт GameInstance, який має доступ до глобальних змінних під час усієї сесії гри.

Також важливо зазначити, що всі компоненти гри були організовані з урахуванням принципів модульності. Усі меню, логіка мережевого підключення, персонажі та головоломки винесені в окремі Blueprint або C++ класи, що дозволяє за потреби легко масштабувати проєкт і додавати нові рівні чи функціональність.

Таким чином, поєднання Unreal Engine 5, Blender та Substance Painter забезпечило повноцінний цикл розробки ігрової сцени: від створення 3D-моделей і текстурування до побудови логіки взаємодії гравців і мережевої синхронізації. Обрані інструменти дозволили не лише швидко реалізувати прототип демонстраційного рівня, але й сформувати технологічну основу для можливого продовження розробки повноцінної гри в майбутньому. Вони також дали змогу сконцентруватися на створенні геймплейної цінності – ключового компонента будь-якого успішного ігрового проєкту.

Висновки до розділу 2

У другому розділі було здійснено комплексний аналіз вимог до проєктованої системи та обґрунтовано вибір інструментів, засобів і методів, які використовувалися під час її реалізації. На основі поставлених завдань і геймдизайнерських рішень було побудовано специфікацію функціоналу майбутньої гри, охоплюючи як загальні принципи взаємодії гравця із системою, так і технічні аспекти реалізації інтерфейсу, мережевих механік та ігрової логіки.

У процесі розробки було застосовано методи об’єктно-орієнтованого моделювання із використанням UML-діаграм. Зокрема, діаграми варіантів використання, класів і діяльності дозволили структурувати взаємодію між компонентами системи, виявити основні сценарії поведінки гравців та оптимізувати архітектуру меню і логіку мережевої взаємодії. Це сприяло більш

точному плануванню роботи, а також забезпечило узгодженість усіх елементів проєкту.

Для реалізації проєкту було обрано потужний технологічний стек, основою якого став рушій Unreal Engine 5, що забезпечив повноцінну підтримку розробки ігрової логіки, графіки, а також інтерфейсних рішень. Інструменти Blender і Substance Painter дозволили реалізувати 3D-моделі та текстури власного виробництва, що забезпечує індивідуальність стилю гри та її візуальну цілісність.

У підсумку, розділ заклав міцну теоретико-практичну основу для безпосередньої реалізації демонстраційного рівня гри Mechanical Runaway, а також підтвердив доцільність обраних рішень з погляду функціональності, гнучкості та відповідності сучасним підходам у розробці інтерактивних систем.

РОЗДІЛ 3
РОЗРОБКА КООПЕРАТИВНОЇ ГРИ «MECHANICAL RUNAWAY» В ЖАНРІ ADVENTURE PUZZLE З ВИКОРИСТАННЯМ UNREAL ENGINE 5

3.1 Практична реалізація об’єкта проєктування

Unreal Engine 5 забезпечує широкий спектр інструментів і технологій, що дозволяють ефективно створювати сучасні кооперативні ігри. Зокрема, рушій підтримує систему динамічного освітлення Lumen, яка значно підвищує візуальну достовірність ігрового середовища, але це потребує значних ресурсних витрат. Для побудови логіки проєкту активно використовується система візуального програмування Blueprints, що дозволяє швидко розробляти та тестувати геймплейні механіки. Інтерфейс користувача реалізується за допомогою фреймворку UMG (Unreal Motion Graphics [17]), який забезпечує гнучкість і адаптивність UI-компонентів. Крім того, наявність численних плагінів, зокрема «Advanced Sessions», дає змогу розширити можливості мережевої взаємодії, покращити стабільність підключень та полегшити реалізацію сесійної архітектури гри, з його допомогою ми будемо створювати та приєднуватись до сесії через Steam.

Уся початкова логіка реалізується безпосередньо у Level Blueprint стартового рівня. Level Blueprint для всіх нових карт будуть створені за допомогою цього класу, що дозволить додавати специфічні для гри доповнення та функціональність, або робити початкові налаштування рівню та гри. Спершу відбувається завантаження налаштувань гри з локального конфігураційного файлу, що зберігається в директорії проєкту. Це дає змогу відразу підлаштувати гру під збережені параметри користувача. Далі ініціалізується клас GameInstance, який існує протягом усього життєвого циклу гри й відповідає за управління глобальними станами, такими як перехід між рівнями, збереження інформації про гравця, ініціалізація мережевих сесій тощо. Після цього у Level

Blueprint виконується перевірка на наявність локального файлу користувача (рис. 3.1) – у разі його відсутності буде створено новий.

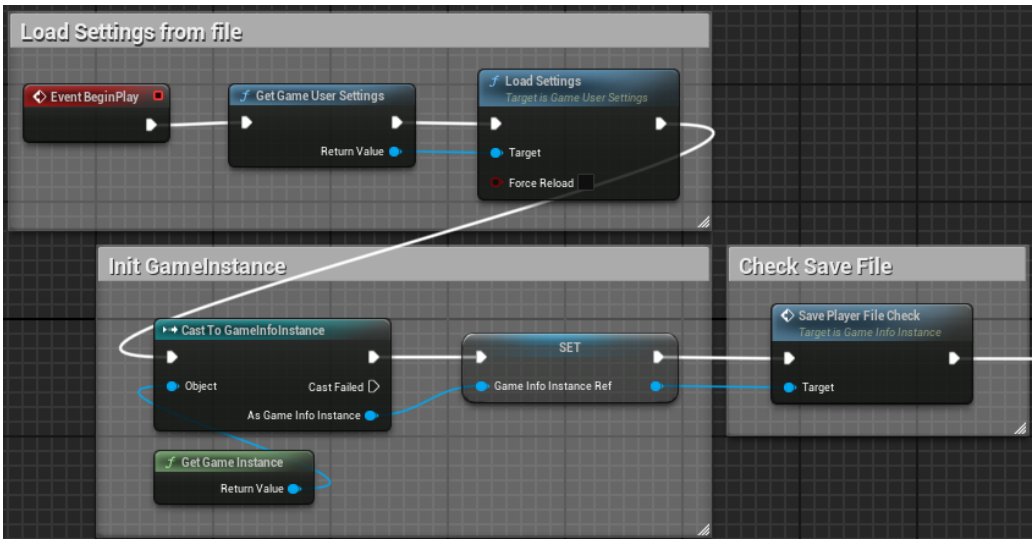

Рисунок 3.1 – Початкове налаштування гри

Лише після виконання всіх підготовчих дій у тому ж Level Blueprint встановлюється нова камера для PlayerController локального гравця, що забезпечує правильне відображення головного меню та стартової сцени (рис. 3.2). Така послідовність гарантує, що інтерфейс та внутрішня логіка гри будуть налаштовані коректно ще до початку взаємодії з користувачем.

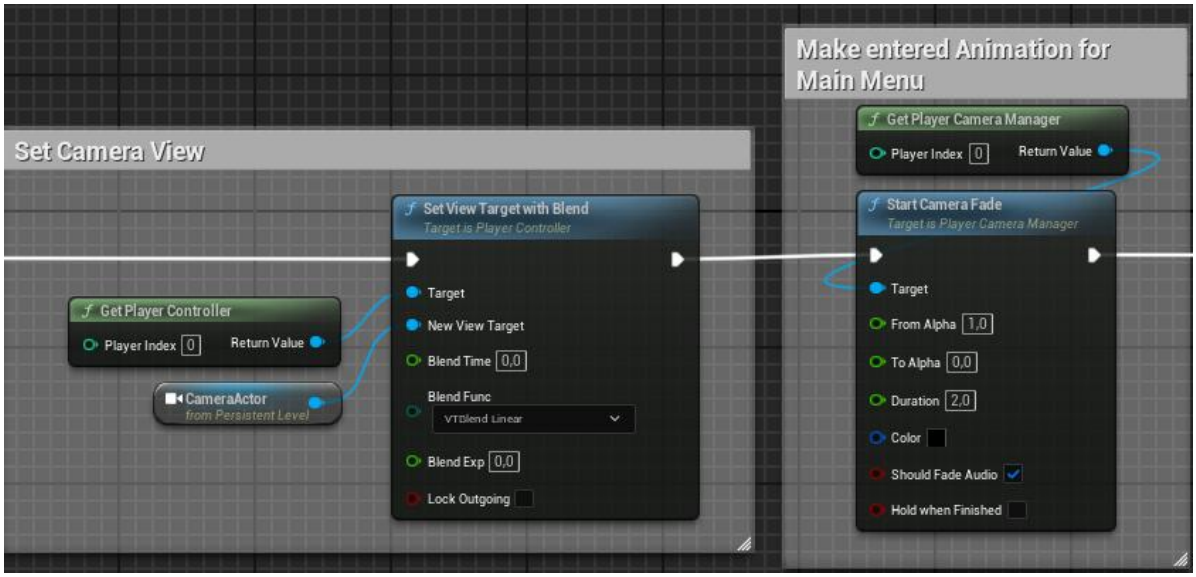

Рисунок 3.2 – Постановка позиції камери у головному меню

Останнім кроком у логіці Level Blueprint є встановлення статусу гравця, що перебуває в головному меню (рис. 3.3). Це необхідно для коректного функціонування подальших сценаріїв взаємодії, зокрема активації інтерфейсу, обмеження ігрових дій до моменту початку сесії, а також забезпечення відповідної поведінки інших систем гри, які залежать від поточного стану гравця.

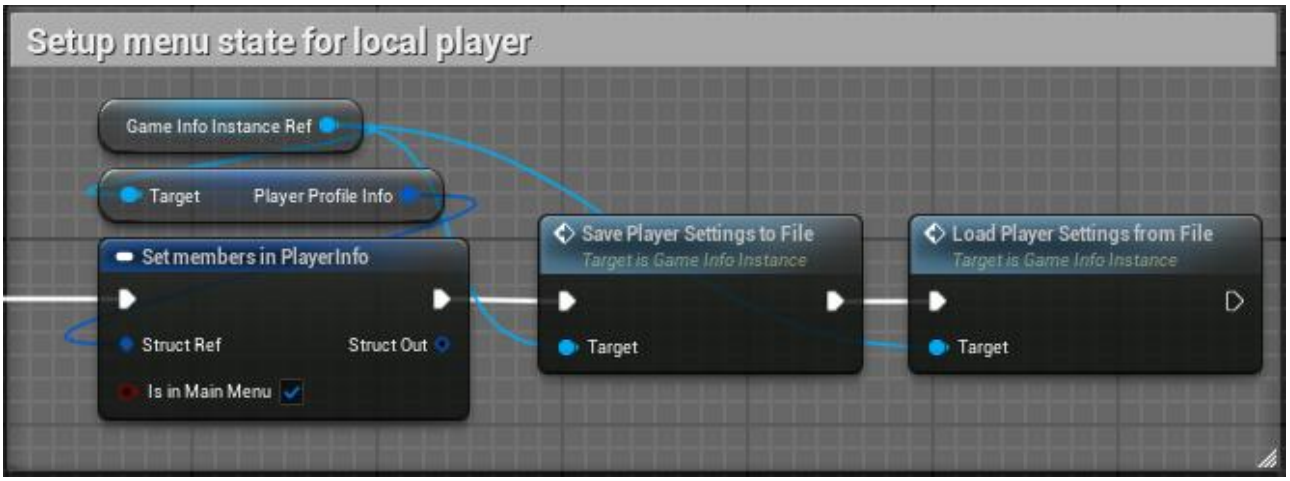

Рисунок 3.3 – Встановлення статусу гравця

Наступним етапом у роботі програми є забезпечення користувача розумінням того, як саме відбувається його ідентифікація в системі. Для реалізації цього функціоналу було використано Unreal Motion Graphics (UMG) – систему побудови інтерфейсів в Unreal Engine.

На основі UMG було створено віджет профілю гравця, який відображає ключову інформацію: нікнейм, аватар та статус з’єднання зі Steam, зчитані з локального конфігураційного файлу (рис. 3.4).

Рисунок 3.4 – Профіль користувача

Усередині логіки цього віджета реалізовано доступ до GameInstance, через який здійснюється зчитування збережених параметрів гравця та їх динамічне відображення на екрані (рис. 3.5). Такий підхід дозволяє миттєво оновлювати вміст віджета при зміні користувацьких даних, забезпечуючи зручність і прозорість ідентифікації для самого гравця.

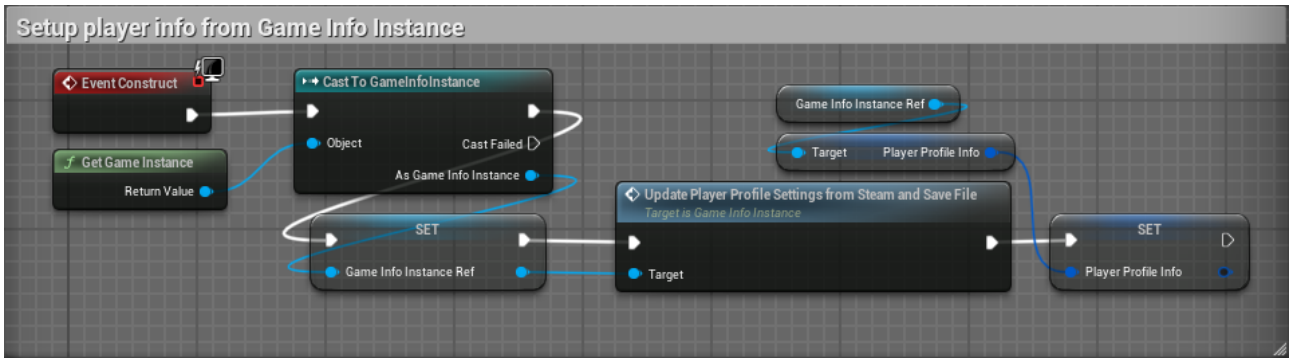

Рисунок 3.5 – Встановлення інформації про гравця

Для налаштування профілю гравця було створено окремий інтерфейсний віджет (рис. 3.6), який надає користувачу можливість самостійно сформувати власну ігрову ідентичність, незалежно від інтеграції зі сторонніми сервісами, такими як Steam.

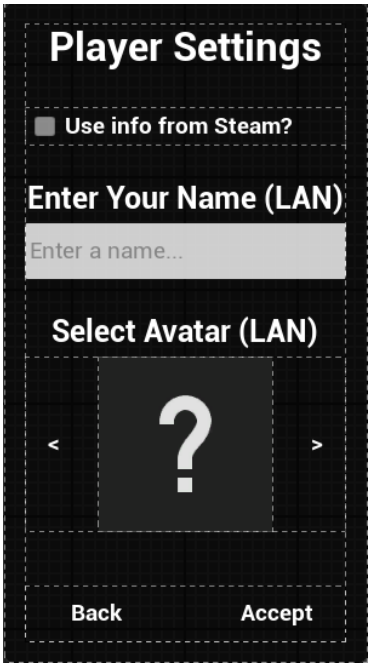

Рисунок 3.6 – Налаштування профілю гравця

У межах цього віджета реалізовано функціонал вибору одного з доступних аватарів, що надаються внутрішньою бібліотекою зображень, а також поле для введення довільного нікнейму. Ці дані зберігаються локально у вигляді конфігураційного файлу та використовуються для подальшої персоналізації інтерфейсу, а також для відображення в мультиплеєрній грі.

Для збереження змін у профілі гравця віджет використовує уніфіковану логіку оновлення даних. Після того як користувач обирає нові налаштування (аватар та нікнейм) і натискає кнопку «Apply», система виконує послідовну перевірку:

– перевіряється наявність локального файлу конфігурації профілю;

– якщо файл не існує, він створюється, і нові дані записуються до нього;

– якщо файл вже існує, його вміст перезаписується оновленими даними.

Цей механізм забезпечує надійне збереження користувацької інформації та дозволяє зберігати персоналізацію профілю між сесіями гри. Таким чином, навіть без доступу до зовнішніх серверів чи акаунтів, гравець отримує індивідуальний досвід, який зберігається локально.

Меню налаштувань гри реалізоване у вигляді UMG-віджету, який створено модульним, що дозволяє використовувати його як у головному меню, так і безпосередньо під час ігрового процесу. Такий підхід підвищує зручність для користувача та спрощує інтеграцію в різні частини інтерфейсу.

У межах цього віджета реалізовано основні параметри конфігурації, серед яких:

1) вибір режиму відображення вікна (Windowed, Windowed Fullscreen, Fullscreen);

2) зміна роздільної здатності екрану;

3) налаштування графічної якості (Low, Medium, High, Epic, Ultra);

4) можливість увімкнення або вимкнення вертикальної синхронізації.

Таке меню (рис. 3.7) забезпечує гнучке налаштування продуктивності й зовнішнього вигляду гри відповідно до можливостей системи гравця, що особливо важливо в контексті інді-розробки, орієнтованої на широкий діапазон конфігурацій.

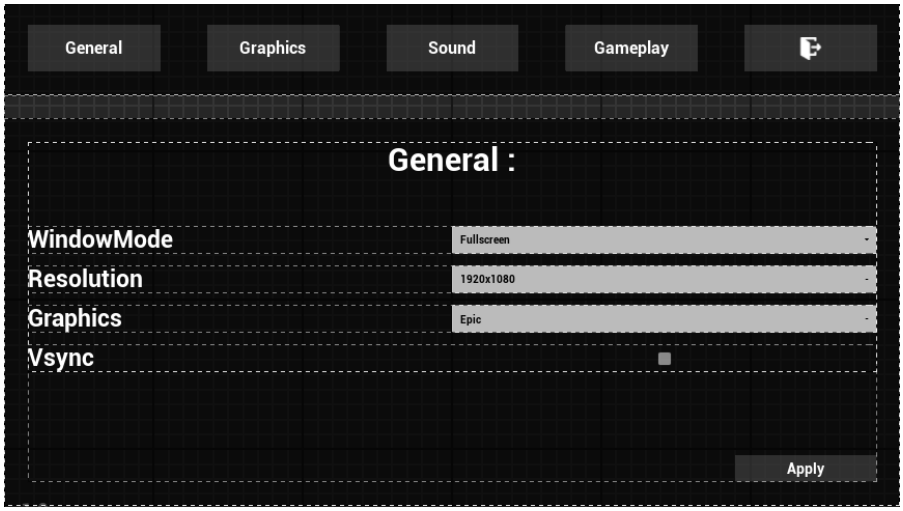

Рисунок 3.7 – Налаштування гри

Цей віджет за своєю логікою роботи схожий на систему збереження профілю гравця: він також оперує даними, які потрібно зберегти локально. Проте, на відміну від кастомного файлу, що зберігає дані користувача (нікнейм, аватар тощо), віджет налаштувань гри використовує стандартний механізм збереження Unreal Engine – вбудований файл конфігурації, який автоматично генерується рушієм.

Після внесення змін користувачем, віджет перезаписує відповідні параметри у цей файл, забезпечуючи збереження вибраних налаштувань навіть після перезапуску гри. Такий підхід дозволяє ефективно керувати технічними параметрами без потреби створювати окрему систему збереження для конфігурацій, використовуючи можливості рушія для автоматичного оброблення системних налаштувань.

Розглянемо віджет, який відповідає за створення ігрового серверу гравцем-хостом. Цей UMG-віджет дозволяє користувачеві ініціювати створення сесії, обравши спосіб з’єднання: через Steam або по локальній мережі (LAN).

Інтерфейс віджета містить також поле для введення назви серверу, яка згодом буде відображатися у списку доступних сесій для інших гравців (рис. 3.8).

Рисунок 3.8 – Віджет для створення сесії

Це забезпечує зручність у розпізнаванні хостів та сприяє більш організованому підключенню до гри. Логіка, що стоїть за віджетом, використовує функціонал плагіна Advanced Sessions, що розширює стандартні можливості створення і реєстрації сесій в Unreal Engine, дозволяючи легко реалізувати гнучке налаштування типу з’єднання та параметрів хостингу.

Після натискання кнопки «Accept», у головному класі GameInstance викликається функція LaunchLobby, якій передаються два основні параметри – тип з’єднання та назва серверу. На цьому етапі назва лобі зберігається у змінну для подальшого використання, а гравцю демонструється екран завантаження, поки триває ініціалізація сесії.

Далі відбувається основний етап – виклик функції Create Advanced Session з плагіна Advanced Sessions, який використовується для створення Steam з’єднання. У неї передається кілька критично важливих параметрів:

1) масив налаштувань сесії – зокрема, назва гри. Оскільки Steam-сервери можуть об’єднувати всі ігри, що використовують стандартний Spacewar

AppID, необхідна фільтрація саме нашої сесії, тож назва гри вказується окремо як умовний «тег», за яким інші гравці можуть її знайти;

2) PlayerController гравця – коли створюється сесія, рушій пов’язує гравця з конкретним контролером і зберігає дані про його інтернет-з’єднання. Без цього ідентифікація в сесії буде некоректною;

3) кількість гравців – задається максимальна кількість учасників, які можуть підключитися до сесії одночасно;

4) тип з’єднання – LAN або Steam, в залежності від вибору користувача.

Використання ноди «Create Advanced Session» наведено нижче на рисунку 3.9.

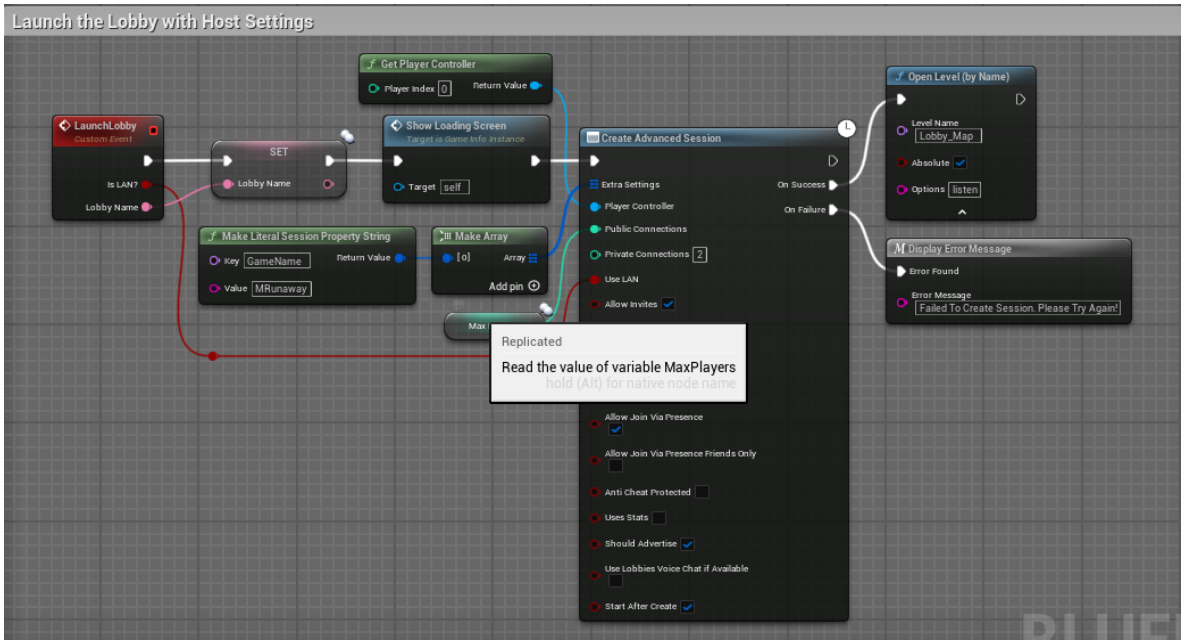

Рисунок 3.9 – Створення сесії

Після успішного створення сесії, система викликає функцію Open Level із встановленим прапорцем Absolute, назвою рівня, який відповідає за лоббі для вибору персонажа, а також з параметром «Listen», що дозволяє хосту приймати підключення інших гравців. Завдяки цій конфігурації рушій автоматично ініціює прослуховування підключень на цьому рівні, що робить гравця-хоста центральним вузлом мережевої сесії.

У випадку, якщо сесію не вдалося створити, система викликає попередньо налаштований макрос, який виводить на екран гравця повідомлення про помилку, вказуючи тип збою (наприклад, проблема з підключенням до Steam, неправильні параметри сесії тощо).

Наступним етапом у реалізації мережевої взаємодії є віджет з’єднання з сервером (рис. 3.10). Його інтерфейс надає користувачу можливість вибрати тип підключення (Internet або LAN). Після вибору типу з’єднання гравець може перейти до пошуку доступних ігор.

Рисунок 3.10 – Віджет пошуку серверу

Перед початком пошуку попередньо очищується масив знайдених сесій, щоб уникнути дублювання або конфліктів. Далі викликається функція з плагіна Advanced Sessions – Find Session Advanced (рис. 3.11).

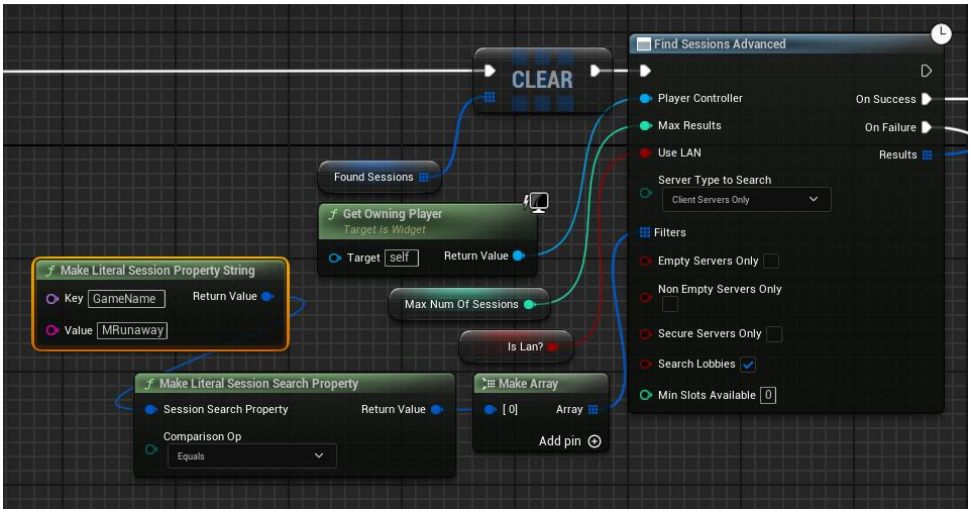

Рисунок 3.11 – Пошук серверів

Після виконання запиту функція повертає масив знайдених серверів, який обробляється циклічно. Для кожної сесії перевіряється, чи є вільне місце для приєднання. Якщо така сесія знайдена:

– пошук припиняється;

– користувачу виводиться інформація про знайдену сесію;

– гравцю пропонується приєднатися до неї.

У разі згоди, на екрані відображається екран завантаження, а система ініціює процес підключення до сервера, забезпечуючи плавний перехід до ігрового рівня (рис. 3.12).

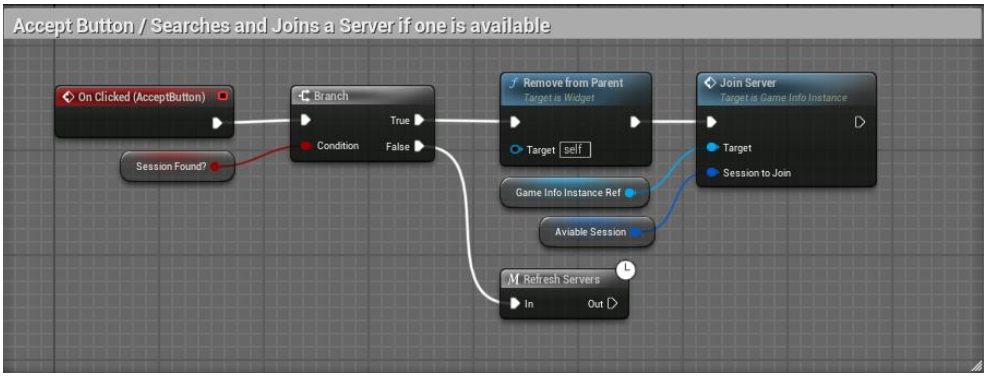

Рисунок 3.12 – З’єднання з сервером

Для подальшого розуміння логіки мережевої гри в Unreal Engine необхідно пояснити специфіку реплікації об’єктів, оскільки не всі ігрові елементи існують або поводяться однаково на сервері й клієнтах.

В Unreal Engine можна умовно розділити об’єкти за їх областю існування:

– об’єкти, які існують лише на сервері – ці об’єкти повністю контролюються сервером і недоступні для прямих змін з боку клієнтів. Наприклад: логіка створення сесій, авторитетне керування станами гри;

– об’єкти, які існують на сервері і на всіх клієнтах – це зазвичай об’єкти з увімкненою реплікацією (наприклад, персонажі, що мають мережеву синхронізацію). Їхня поведінка синхронізується між усіма гравцями;

– об’єкти, які існують на сервері та тільки на конкретному клієнті – це ситуація, коли сервер володіє логікою, а конкретний клієнт, локальним контролем (наприклад, PlayerController);

– об’єкти, які існують лише на клієнті – наприклад, локальні віджети інтерфейсу або тимчасові ефекти, які не потребують синхронізації з сервером.

На рисунку 3.13 буде візуалізовано, до якої з цих груп належать основні мережеві елементи гри, які ми використовуємо. Це допоможе точніше пояснити, чому певні дії виконуються саме на сервері або лише на клієнтах, і як це впливає на загальну архітектуру гри.

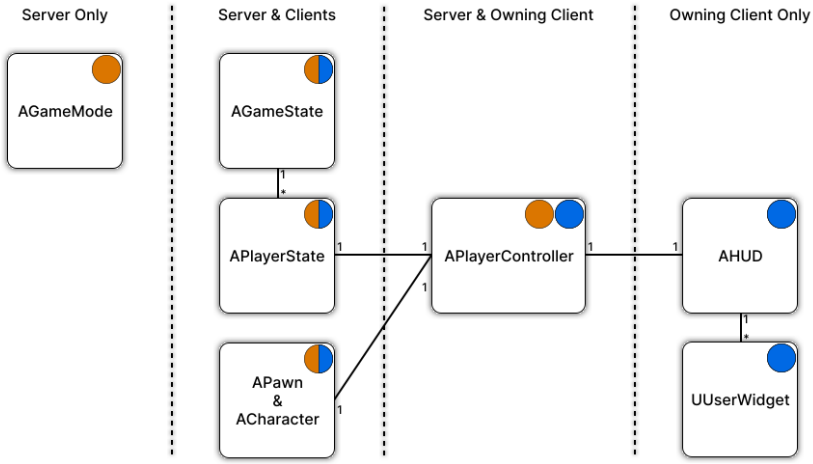

Рисунок 3.13 – Візуалізація груп на мережеві елементи

Розглянемо детально процес вибору персонажа, у якому якраз використовується раніше згадана логіка розподілу об’єктів за контекстом (сервер/клієнт) в Unreal Engine. У цій системі саме сервер відповідає за створення, володіння та синхронізацію об’єктів, а клієнт лише ініціює запит на зміну або створення персонажа, після чого сервер приймає рішення, виконує потрібну логіку та передає результат клієнту. Такий підхід дозволяє гарантувати узгодженість ігрового стану між усіма учасниками мережевої гри, запобігає конфліктам через одночасне звернення кількох клієнтів та підтримує належну безпеку обробки запитів. Сам механізм реалізовано через виклики RPC-функцій

(Remote Procedure Call), які чітко розділяють відповідальність між клієнтами та сервером, забезпечуючи стабільність і передбачуваність роботи системи вибору персонажа.

Процес вибору персонажа реалізовано через віджет лобі, де кожен гравець бачить свою карточку і може зробити вибір одного з доступних персонажів, де вибір персонажа залежить від позиції картки. Також у цьому віджеті реалізовано внутрішньо-ігровий чат для онлайн спілкування, кнопка меню, назва сервера та таблиця з підключеними гравцями. Віджет лобі буде наведено нижче на рисунку 3.14.

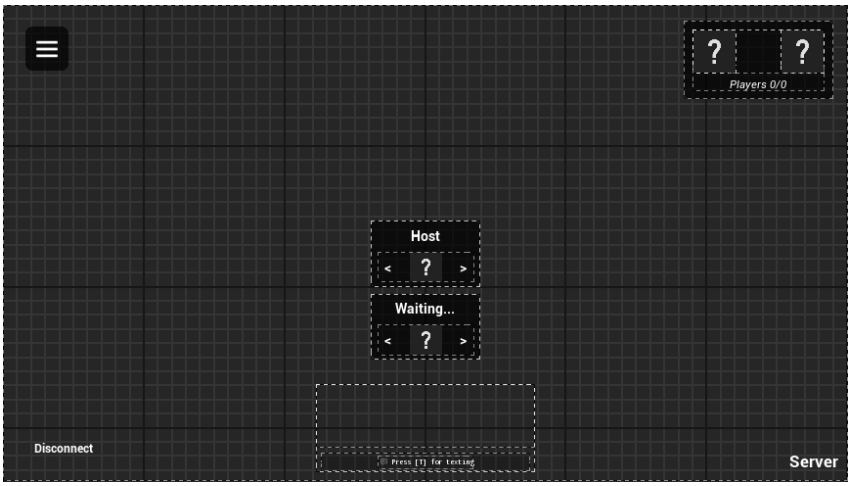

Рисунок 3.14 – Віджет лобі

Важливо зазначити, що саме позиція карточки гравця у списку визначає, який саме гравець робить вибір – це дозволяє коректно асоціювати дані у подальшому (рис. 3.15).

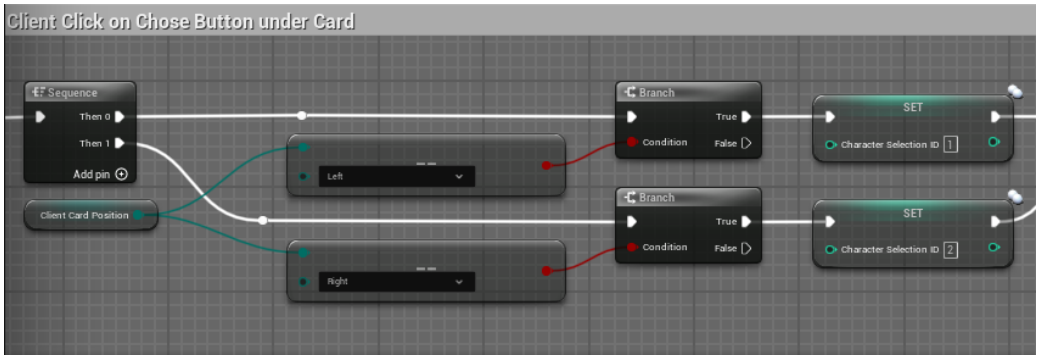

Рисунок 3.15 – Вибір персонажа в залежності від позиції картки

Після відкриття віджету вибору персонажа гравець може обрати бажаного героя. У результаті цього у віджеті лобі зберігається ID обраного персонажа. Після підтвердження вибору ця інформація передається до класу PlayerController.

У PlayerController відбувається валідація вибору. Контролер перевіряє, чи вже не було обрано цього персонажа іншим гравцем, чи сам користувач ще не зробив вибір, а також перевіряє коректність переданих даних. Лише після успішного проходження всіх цих перевірок вибір вважається дійсним.

Після цього ID обраного персонажа зберігається у локальний сейв-файл гравця. Це дозволяє зчитати інформацію при наступному підключенні або у випадку перезапуску гри.

На завершальному етапі контролер передає дані до GameMode, повідомляючи про вибір (рис. 3.16). GameMode, який існує виключно на сервері, оновлює загальну таблицю або масив з інформацією про обраних персонажів та асоціює конкретного гравця з відповідним героєм. Завдяки тому, що GameMode функціонує лише на сервері, він має авторитет у прийнятті рішень, і таким чином забезпечується унеможливлення повторного вибору одного і того ж персонажа кількома гравцями.

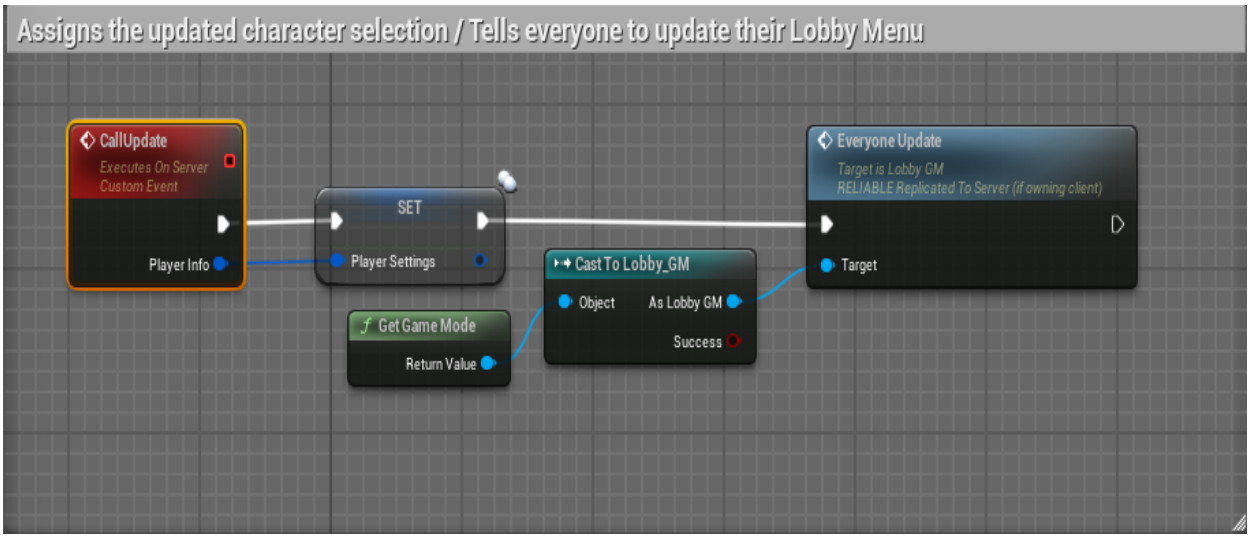

Рисунок 3.16 – Оновлення вибору персонажа на сервері

Такий підхід гарантує, що:

– персонаж не буде обраний кількома гравцями одночасно;
– вибір синхронізується між клієнтами через реплікацію;
– всі дії проходять через центральну перевірку на сервері;
– кожен гравець має локальне збереження обраного персонажа.

Усе це забезпечує чітку і безпечну логіку мультиплеєрного вибору персонажів, яка працює стабільно навіть у випадках затримок чи втрати з'єднання.

Слід також відзначити реалізацію внутрішньоігрового чату, яка базується на аналогічному принципі обробки, що й механіка вибору персонажа. Спочатку повідомлення, яке вводить гравець, обробляється локально на клієнті – власнику повідомлення. Після цього ця інформація надсилається на сервер, де відбувається перевірка та розповсюдження повідомлення до всіх інших гравців у сесії. Такий підхід забезпечує як ефективну синхронізацію, так і контроль над інформацією, яка поширюється між клієнтами.

Далі процес логічно переходить до етапу геймплею. Коли обидва гравці завершили вибір персонажів, і всі вибори пройшли успішну валідацію на сервері, у хоста з'являється активна кнопка «Start Match». Натискання цієї кнопки запускає серверну процедуру, яка перевіряє готовність усіх учасників сесії, після чого ініціює синхронний перехід до гри. Зокрема, сервер надсилає команду всім підключеним клієнтам про початок завантаження основного ігрового рівня, що супроводжується появою відповідного екрану завантаження. Такий підхід дозволяє уникнути ситуацій, коли один із клієнтів не встиг завантажити рівень, а інші вже розпочали гру, забезпечуючи тим самим коректну синхронізацію та одночасний старт геймплею для всіх гравців.

Після завантаження нової карти першим компонентом, що активується, є GameMode. Саме він перехоплює всі PlayerController-и, що приєдналися до рівня, та ініціює спавн персонажів для кожного з гравців. Після цього сервер запитує актуальну інформацію про кожного гравця, зокрема дані збережених виборів персонажів, і розподіляє контроль між відповідними гравцями та їхніми

персонажами (рис. 3.17). Такий підхід дозволяє чітко синхронізувати стан гри між усіма учасниками сесії, а також гарантує, що кожен гравець отримає саме того персонажа, якого обрав раніше.

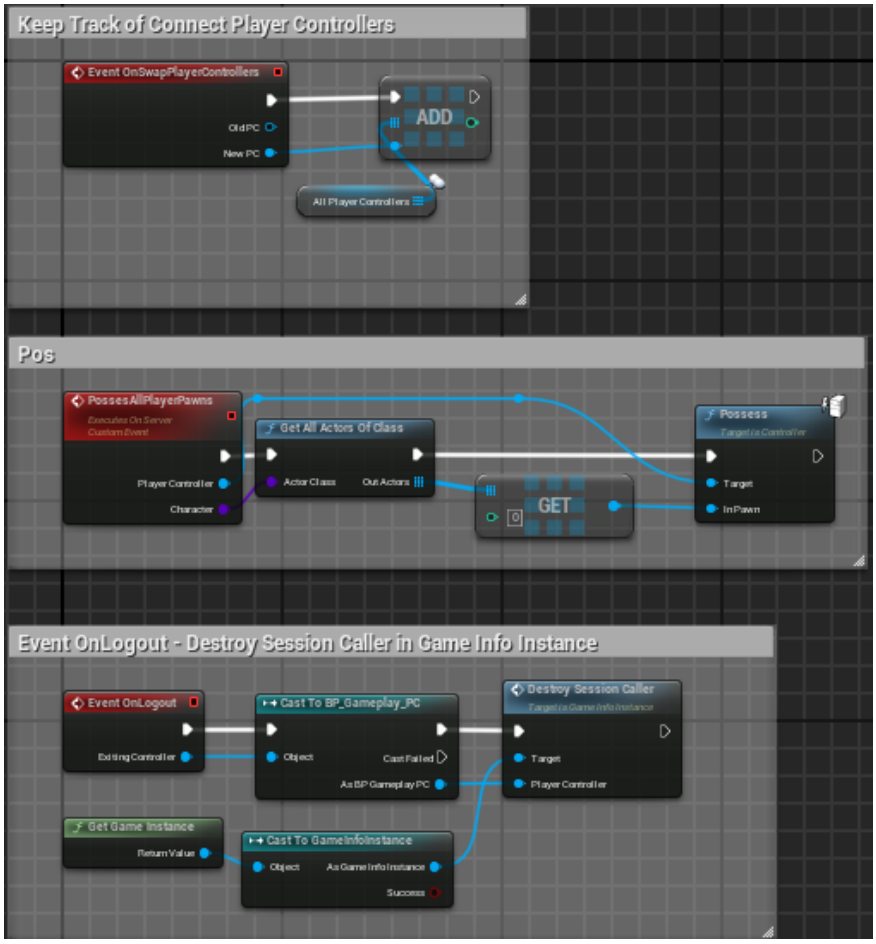

Рисунок 3.17 – Blueprint ігрового режиму

Останнім ключовим елементом у реалізації концептуально важливих механік гри, необхідних для побудови вступного рівня, є безпосередньо самі персонажі. Центральним аспектом у цьому контексті виступає система руху (мувмент), яка формує базову взаємодію гравця з ігровим світом і закладає основу для всіх подальших дій у межах геймплею. У кооперативних іграх, особливо з асиметричною взаємодією між гравцями, кожен персонаж має унікальний стиль пересування, набір можливостей та механік, що впливають не лише на індивідуальний досвід гравця.

У грі реалізовано двох унікальних персонажів з різною логікою пересування, що задає фундамент кооперативної взаємодії. Літаючий робот використовує стандартний компонент Character Movement, модифікований для плавного переміщення у повітрі з можливістю регулювання висоти польоту, приклад його пересування буде продемонстровано у лістингу 3.1.

Лістинг 3.1 – Демонстрація способу пересування літаючого робота

```
FVector direction =
      (is_forward_backward
      ? Camera_Spring_Arm->GetForwardVector()
      : Camera_Spring_Arm->GetRightVector())
      * Movement_Speed
      * GetWorld()->DeltaTimeSeconds
      * (is_negative_axis ? -1 : 1);
direction.Z = 0.0f;

if (GetLocalRole() < ROLE_Authority)
{
      AddMovementInput(direction);

      Server_Move_For_Axis_Triggered
      (Target_Angle_Of_Body_Lean_Forward_Backward,
      Target_Angle_Of_Body_Lean_Left_Right);
}
else
{
      AddMovementInput(direction);
}
```

кінець лістингу 3.1

У наведених лістингах використано конструкцію «if (GetLocalRole() < ROLE_Authority)», яка дозволяє визначити, чи виконується код на клієнтській стороні. Це порівняння відіграє важливу роль у побудові мережевої логіки, оскільки дозволяє розмежувати виконання дій між клієнтом і сервером. Зокрема, якщо локальна роль об’єкта нижча за ROLE_Authority, то виконання відбувається на клієнті, а не на authoritative-сервері, що забезпечує коректну маршрутизацію викликів ігрової логіки.

Цей підхід дозволяє досягти більш реалістичної фізики руху для кулькового персонажа, підкреслюючи його відмінність і забезпечуючи унікальне

ігрове відчуття. Обидва типи руху синхронізовані для коректної роботи в мережевому середовищі: всі дії гравців узгоджуються між сервером і клієнтами, що критично важливо для стабільного кооперативного проходження головоломок.

Натомість робот-кулька реалізований як простий Pawn: у нього немає Movement Component, просто на Mesh увімкнено фізичну симуляцію, а для руху застосовуються сили, що імітують котіння, що забезпечує більш реалістичну фізичну поведінку об’єкта у просторі гри, приклад його пересування буде наведено у лістингу 3.2.

Лістинг 3.2 – Демонстрація способу пересування робота-кульки

```
FVector camera_vector =
      is_forward_backward
      ? Camera->GetForwardVector()
      : Camera->GetRightVector();
FVector body_vector =
      is_forward_backward
      ? Body_Mesh->GetForwardVector()
      : Body_Mesh->GetRightVector();

FVector completed_vector =
      FVector(camera_vector.X, camera_vector.Y, body_vector.Z);

if (GetLocalRole() < ROLE_Authority)
{
      completed_vector *= is_negative_axis
      ? -Movement_Speed
      : Movement_Speed;

      Server_Move_For_Axis_Triggered(completed_vector,
      Target_Angle_Of_Head_Lean_Forward_Backward,
      Target_Angle_Of_Head_Lean_Left_Right);
}
else
{
      Body_Mesh->AddForce(
      completed_vector * (is_negative_axis
      ? -Movement_Speed
      : Movement_Speed));
}
```

кінець лістингу 3.2

3.2 Тестування та налагодження інформаційно-комп’ютерної системи

Після завершення реалізації основних функціональних модулів гри було проведено комплексне тестування інформаційно-комп’ютерної системи для забезпечення її стабільної, надійної та передбачуваної роботи в різних умовах використання. Основна мета полягала у виявленні та усуненні логічних, технічних і мережевих помилок, перевірці внутрішньої взаємодії між компонентами (UI, контролери, серверна логіка) та оптимізації процесів обробки даних.

Початкове тестування проводилося локально у середовищі Unreal Engine 5 з використанням інструментів трасування, візуального дебагу, логів та відлагоджувальних повідомлень. Окремо перевірялися елементи інтерфейсу (UMG-віджети), зокрема створення профілю гравця, налаштування, створення сесій, приєднання до гри, вибір персонажа та внутрішньоігровий чат. Під час налагодження було виявлено ряд типових помилок, пов’язаних із синхронізацією даних, порядком ініціалізації, неправильними посиланнями між об’єктами та випадковими конфліктами у логіці взаємодії між клієнтом і сервером. Ці недоліки були усунені шляхом послідовного оновлення логіки віджетів, структури GameInstance та контролю за станами гри.

Окрему увагу було приділено мережевому тестуванню. Для реалізації мультиплеєру використовувався плагін Advanced Sessions, який дозволяє створювати, знаходити та приєднуватися до мережевих сесій через LAN або Steam. Було протестовано сценарії з кількома гравцями, які одночасно входили до лобі, вибирали персонажів, надсилали повідомлення у чат і переходили до геймплейного рівня. Важливою частиною тестування стала перевірка коректності обробки конфліктів вибору персонажа: система надійно блокує повторний вибір, проводить валідацію даних на стороні сервера та забезпечує передачу інформації тільки після підтвердження на рівні GameMode.

Тестування також охопило поведінку гри під час переходу між рівнями. Після натискання кнопки «Start match» лише сервер ініціює завантаження

наступного рівня, забезпечуючи при цьому коректну передачу контролерів, спавн обраних персонажів та активацію HUD у кожного гравця. На цьому етапі перевірялася стабільність з’єднання, затримка між командами, збереження стану гри та цілісність даних.

На завершальному етапі розробки було визначено мінімальні системні вимоги та рекомендовані параметри, необхідні для повноцінної роботи гри.

Мінімальні системні вимоги:

– ОС: Windows 10 (64-bit);
– процесор: Intel Core i5-7400 або AMD Ryzen 3 1200;
– оперативна пам’ять: 8 ГБ;
– відеокарта: NVIDIA GTX 1050 Ti або AMD RX 560 (2-4 ГБ VRAM);
– мережа: стабільне LAN або інтернет-з’єднання;
– місце на диску: 2 ГБ.

Рекомендовані системні вимоги:

– ОС: Windows 10/11 (64-bit);
– процесор: Intel Core i7-9700 або AMD Ryzen 5 3600;
– оперативна пам’ять: 16 ГБ;
– відеокарта: NVIDIA GTX 1660 Super / RTX 2060 або AMD RX 5700;
– SSD для встановлення гри;
– інтернет-з’єднання з низькою затримкою (ping < 50ms).

Таким чином, проведене тестування та налагодження дозволили сформувати цілісну, керовану інформаційно-комп’ютерну систему, що демонструє високу стабільність у середовищі Unreal Engine 5, здатну працювати у мультиплеєрному середовищі та забезпечувати комфортний досвід для гравців.

3.3 Розробка документації для програмного продукту

Особливу увагу під час розробки було приділено створенню документації, яка супроводжує кожен етап налаштування та запуску ігрового прототипу. Документація охоплює інструкції з встановлення необхідного програмного

забезпечення, опис структури проєкту, налаштування мережевого середовища, а також детальний порядок дій для коректного запуску гри на кінцевих системах. Це дозволяє не лише забезпечити зрозумілість процесу для кінцевого користувача, а й спрощує підтримку та подальший розвиток проєкту іншими розробниками або командами.

У репозиторій програмного продукту додано README-файли, що є основним джерелом інструкцій для кінцевого користувача та технічного персоналу. Вони містять детальну інформацію про те, як розгорнути проєкт у середовищі Unreal Engine, які налаштування необхідно виконати для роботи мережевого підключення, а також які компоненти слід встановити до початку роботи.

README-файл детально описує порядок дій:

– завантаження та відкриття проєкту в Unreal Engine з урахуванням використаних плагінів (зокрема Advanced Sessions);

– інструкції щодо увімкнення Steam-сервісу, налаштування AppID через файл steam_appid.txt, де за замовчуванням використовується ID гри Spacewar для тестування;

– покроковий опис збірки гри під локальний запуск та мережевий мультиплеєр;

– налаштування конфігураційних файлів у проєкті (наприклад, DefaultEngine.ini та DefaultGame.ini) для коректної роботи мережевих функцій, системи збережень, віконного режиму, підключення до сесій тощо;

– пояснення структури проєкту: як організовано логіку головного меню, профілю гравця, системи сесій, вибору персонажа та ігрового режиму;

– керівництво щодо збирання фінального білду: які параметри використовуються, які файли включаються до складу дистрибутиву, та як виконується упаковка гри для Windows;

– налаштування Steam Overlay, у разі потреби, для тестування взаємодії через Steam Friends;

– відомості про можливі помилки під час запуску і способи їх усунення (наприклад, помилки створення сесії, з’єднання, відсутність Steam API тощо).

Окремий підрозділ у документації присвячений локальним файлам збереження, які створюються системою для профілю гравця, графічних налаштувань і вибору персонажа. У README зазначено, де саме ці файли розміщуються, як відновити їх при пошкодженні або перенести між машинами.

Таким чином, підготовлена документація в повному обсязі охоплює не лише технічну сторону розгортання, але й спрямована на зручність використання продукту кінцевим користувачем. Вона дозволяє швидко адаптувати проєкт до нових умов запуску, провести самостійне тестування і забезпечити стабільну роботу як у локальній, так і у мережевій багатокористувацькій грі.

3.4 Розробка інсталяційного пакета

З метою забезпечення зручного розповсюдження та встановлення програмного продукту було створено інсталяційний пакет, що містить усі необхідні файли для запуску гри на кінцевому пристрої користувача. Формування інсталяційного пакета здійснюється за допомогою стандартного інструментарію Unreal Engine – функції Packaging Project, яка дозволяє зібрати повністю готовий до запуску білд гри.

У процесі створення білду враховуються особливості проєкту: підключення зовнішніх плагінів (зокрема, Advanced Sessions для мультиплеєру), використання Steam API, а також наявність користувацьких налаштувань та локальних файлів збереження. Усі ці компоненти автоматично включаються до складу збірки, щоб уникнути помилок під час запуску на інших системах.

Після упаковки проєкту формується окрема директорія з виконуваним файлом гри (.exe), усіма необхідними динамічними бібліотеками, контентом та конфігураційними файлами. Для спрощення процесу інсталяції на комп’ютері користувача цей комплект було додатково обгорнуто у архівований інсталяційний пакет (ZIP). За потреби, можливе також використання

спеціалізованого інсталяторного ПЗ (наприклад, Inno Setup або NSIS) для створення повноцінного інсталятору з графічним інтерфейсом, ярликами, налаштуванням директорії встановлення тощо.

Встановлення гри зводиться до розпакування архіву в зручне для користувача місце та запуску виконуваного файлу.

Інсталяційний пакет протестовано на декількох конфігураціях системи з різними дозволами екрану, типами підключення та параметрами запуску, що дозволяє гарантувати його стабільну роботу на більшості сучасних ПК. У README-файлі надається інструкція з встановлення, запуску та усунення можливих проблем.

Таким чином, підготовлений інсталяційний пакет забезпечує зручний доступ до продукту без необхідності додаткових налаштувань, що суттєво спрощує його поширення серед цільової аудиторії.

3.5 Визначення витрат і можливостей комерціалізації

Економічне обґрунтування розробки інформаційно-комп’ютерної системи гри полягає в аналізі доцільності витрат на етапі створення проєкту та оцінці потенційної вигоди від його впровадження. Оскільки розробка відбувалась як частина кваліфікаційної роботи, прямі витрати на оплату праці не здійснювались, однак у реальних умовах вони становили б основну статтю бюджету.

У проєкті використано переважно безкоштовні інструменти й технології з відкритим доступом, зокрема Unreal Engine (на умовах роялті після комерційного використання), а також плагін Advanced Sessions, який також є безоплатним. Для мережевої взаємодії застосовано Steam API, що не потребує ліцензійних витрат у межах відкритого тестування. Це дозволило мінімізувати вартість розробки на інфраструктурному рівні.

З технічного боку, запуск і тестування проєкту здійснювались на стандартному персональному обладнанні, без потреби в оренді серверів чи

купівлі ліцензійного ПЗ. Завдяки використанню Blueprint-системи в Unreal Engine розробка не вимагала платного IDE, що також скорочує витрати у комерційних умовах.

У разі комерційного запуску потенційне джерело прибутку може включати продаж гри, мікротранзакції, або інші бізнес-моделі монетизації, як-от передплата чи платний доступ до розширених функцій. У свою чергу, інвестиції, необхідні для комерційної адаптації (серверна інфраструктура, маркетинг, технічна підтримка), мають бути прораховані окремо на основі обраної бізнес-стратегії.

Таким чином, розробка продемонструвала економічну доцільність завдяки використанню доступних технологій, що дозволяє надалі адаптувати її для комерційного застосування з мінімальними початковими витратами.

Висновки до розділу 3

У межах третього розділу було реалізовано повноцінну інформаційно-комп’ютерну систему, що охоплює усі основні етапи життєвого циклу програмного продукту – від розробки клієнтського та серверного функціоналу до підготовки документації та інсталяційного пакета. Було створено набір користувацьких інтерфейсів на основі UMG, що забезпечують взаємодію з гравцем: автентифікація, налаштування профілю, створення і приєднання до сесій, вибір персонажа, внутрішньоігровий чат і безпосередньо запуск геймплею.

Особливу увагу було приділено мережевій взаємодії через плагін Advanced Sessions, що забезпечує підтримку мультиплеєру як через LAN, так і через Steam. Реалізовано складну логіку валідації і синхронізації дій користувача між клієнтами та сервером, що гарантує коректну роботу гри в умовах мережевого середовища.

Система була протестована в умовах, наближених до продуктивних, що дозволило виявити та усунути критичні помилки ще на етапі розробки. Окремо сформовано README-документацію з покроковими інструкціями для

розгортання, а також створено інсталяційний пакет, що дозволяє швидко і зручно встановити гру на цільовий пристрій.

Таким чином, у цьому розділі було реалізовано як функціональні, так і технічні аспекти програмного продукту, що забезпечує його повноцінне функціонування, підтримку користувача і можливість подальшого масштабування.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи досягнуто поставлену мету – розроблено інформаційно-комп’ютерну систему, що забезпечує повноцінний цикл взаємодії користувача з ігровим середовищем. Реалізовано механізми створення та налаштування ігрового профілю, організацію мережевих сесій через LAN або Steam, вибір персонажів з валідацією на рівні сервера, інтеграцію внутрішньоігрового чату, запуск геймплею та управління логікою взаємодії між клієнтами і сервером. Усі ключові компоненти було реалізовано з використанням Blueprints та C++ в Unreal Engine та плагіна Advanced Sessions, що забезпечило високу гнучкість і стабільність системи в умовах мультиплеєрного середовища.

На першому етапі було проведено аналітичне дослідження предметної області, вивчено сучасний стан індустрії кооперативних ігор у жанрі Puzzle Adventure, проаналізовано вдалі приклади реалізації ігрових механік, а також визначено сильні та слабкі сторони конкурентних проєктів. Особливу увагу було приділено побудові геймплею в умовах асиметричної взаємодії між гравцями та особливостям дизайну рівнів для різних типів персонажів.

На другому етапі було виконано проєктування архітектури ігрової системи, побудовані UML-діаграми, які відображали структуру класів і взаємозв’язки між ними. Також був обґрунтований вибір основних інструментів і технологій, серед яких рушій Unreal Engine 5, система Blueprint для створення логіки, можливість використання C++ та підходи до реалізації мережевої взаємодії. Це забезпечило фундамент для стабільного та масштабованого прототипу.

Наступним кроком була реалізація ігрового прототипу «Mechanical Runaway», розроблено мережеву логіку на базі Unreal Engine 5, яка забезпечувала стабільну синхронізацію даних між клієнтами в режимі реального часу. Також були реалізовані основні геймплейні механіки, включаючи кооперативну взаємодію між двома асиметричними персонажами, та впроваджено логіку обробки подій і відображення інформації в інтерфейсі.

На етапі тестування було перевірено основні сценарії взаємодії в багатокористувацькому режимі, протестовані механізми підключення до сесії, передача подій, синхронізація станів об’єктів та стійкість до збоїв мережі. В результаті було виявлено і усунуто низку помилок, що підвищило стабільність системи.

У межах наступного етапу була створена супровідна документація. До кожного з компонентів системи – клієнтського та серверного – було додано README-файли з покроковими інструкціями для розгортання, а також шаблони конфігураційних файлів для налаштування середовища. Це забезпечило простоту встановлення та використання прототипу. Було підготовлено інсталяційний пакет, який включав усі необхідні файли для запуску гри на кінцевих системах. За допомогою вбудованих засобів Unreal Engine 5 була зібрана повноцінна збірка проєкту, що дозволила швидко інсталювати гру без додаткових налаштувань.

Завершальним етапом було визначення витрат і можливостей комерціалізації, оцінено обсяг витрачених ресурсів, проаналізовано доцільність реалізації проєкту та визначено його потенційну цінність для подальшого розвитку, з урахуванням практичних і комерційних перспектив.

Отже, усі поставлені завдання було виконано в повному обсязі. Розроблений ігровий прототип є прикладом практичного застосування сучасних ігрових технологій на базі Unreal Engine 5 і може слугувати основою для створення повноцінного кооперативного проєкту або бути розширений відповідно до нових ігрових чи технічних вимог.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Учасники проектів Вікімедіа. Головоломка (жанр відеоігор) – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Головоломка_(жанр_відеоігор) (дата звернення: 04.04.2025).

2. Учасники проектів Вікімедіа. Багатокористувацька гра – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Багатокористувацька_гра (дата звернення: 05.04.2025).

3. Учасники проектів Вікімедіа. Unreal Engine – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Unreal_Engine (дата звернення: 08.04.2025).

4. Epic Games. Lumen Global Illumination and Reflections. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/lumen-global-illumination-and-reflections-in-unreal-engine (date of access: 08.04.2025).

5. Epic Games. Blueprints Visual Scripting. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/blueprints-visual-scripting-in-unreal-engine (date of access: 09.04.2025).

6. Учасники проектів Вікімедіа. Локальна мережа – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Локальна_мережа (дата звернення: 11.04.2025).

7. Учасники проектів Вікімедіа. Steam – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Steam (дата звернення: 11.04.2025).

8. Учасники проектів Вікімедіа. AAA (індустрія відеоігор) – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/AAA_(індустрія_відеоігор) (дата звернення: 17.04.2025).

9. Учасники проектів Вікімедіа. Unified Modeling Language – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Unified_Modeling_Language (дата звернення: 24.04.2025).

10. Учасники проектів Вікімедіа. C++ – Вікіпедія. URL: https://uk.wikipedia.org/wiki/C++ (дата звернення: 02.05.2025).

11. Учасники проектів Вікімедіа. Сценарій використання – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Сценарій_використання (дата звернення: 05.05.2025).

12. Учасники проектів Вікімедіа. Microsoft Windows – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Microsoft_Windows (дата звернення: 05.05.2025).

13. Учасники проектів Вікімедіа. Діаграма класів – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Діаграма_класів (дата звернення: 09.05.2025).

14. Учасники проектів Вікімедіа. Діаграма діяльності – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Діаграма_діяльності (дата звернення: 12.05.2025).

15. Учасники проектів Вікімедіа. Blender – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Blender (дата звернення: 17.05.2025).

16. DOU. Чим можна замінити Substance Painter. URL: https://gamedev.dou.ua/articles/imitation-of-substance-painter (дата звернення: 18.05.2025).

17. Epic Games. UMG Best Practices. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/umg-best-practices-in-unreal-engine (date of access: 18.05.2025).