Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Ігровий програмний застосунок у жанрі космічного симулятора. Кооператив, механіки персонажа, HUD.
(тема)

Виконав:
студент 4 курсу, групи ПЗПІ-20-6

Яковенко Д.О.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник ст.викл. кафедри ПІ Новіков Ю.С.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Яковенко Дмитру Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи Ігровий програмний застосунок у жанрі космічного симулятора. Кооператив, механіки персонажа, HUD.

Затверджена наказом по університету від 20.05. 2024р. № 471 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 07.06.2024

3. Вихідні дані до роботи Розробити ігровий застосунок у жанрі космічного симулятора, а саме такі елементи гри: HUD, кооператив та механіки персонажа.

4. Перелік питань, що потрібно опрацювати в роботі

Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, впровадження, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	08.04.2024	*виконано*
2	Створення специфікації ПЗ	15.04.2024	*виконано*
3	Проектування ПЗ	25.04.2024	*виконано*
4	Розробка ПЗ	19.05.2024	*виконано*
5	Тестування ПЗ	23.05.2024	*виконано*
6	Оформлення пояснювальної записки	24.05.2024	*виконано*
7	Підготовка презентації та доповіді	25.05.2024	*виконано*
8	Попередній захист	26.05.2024	*виконано*
9	Нормоконтроль, рецензування	02.06.2024	*виконано*
10	Здача роботи у електронний архів	03.06.2024	*виконано*
11	Допуск до захисту у зав. кафедри	07.06.2024	*виконано*

Дата видачі завдання 08 травня 2024р.

Студент _____ Яковенко Д.О.
(підпис)

Керівник роботи _____ ст.викл. кафедри ПІ Новіков Ю.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра: 134с., 110 рис., 12 джерел.

ІГРОВИЙ КОСМІЧНИЙ СИМУЛЯТОР, UNREAL ENGINE 5, BLUEPRINT, КООПЕРАТИВ, МЕХАНІКИ ПЕРСОНАЖА, HUD.

Об’єкт розробки – кооператив, механіки персонажа та HUD ігрового застосунку у жанрі космічного симулятора.

Мета розробки – розробка програмної реалізації кооператива, механік персонажа та HUD.

Метод рішення – середовище розробки Unreal Engine 5 з застосуванням мови програмування C++ та середовища візуального скріптинга Blueprint.

У результаті розробки реалізовано кооператив, механіки персонажа та HUD для ігрового застосунку у жанрі космічного симулятора.

SPACE SIMULATION GAME, UNREAL ENGINE 5, BLUEPRINT, CO-OP, CHARACTER MECHANICS, HUD.

The object of development is co-op, character mechanics and HUD of a space simulation game application.

The purpose of the development is to develop a software implementation of co-op, character mechanics and HUD.

The solution method is the Unreal Engine 5 development environment using the C++ programming language and the Blueprint visual scripting environment.

As a result of the development, a co-op, character mechanics and HUD for a space simulation game application were implemented.

Я, Яковенко Дмитро Олександрович, студент гр. ПЗПІ-20-6, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Ігровий програмний застосунок у

жанрі космічного симулятора. Кооператив, механіки персонажа, HUD», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений із діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

У сучасному світі відеоігри стали невід'ємною частиною культури та розваг, а жанр космічних симуляторів набуває все більшої популярності. Ці ігри пропонують гравцям можливість досліджувати безмежні простори космосу, взаємодіяти з іншими гравцями та створювати власну історію. Одним з ключових аспектів, що сприяє захоплюючому ігровому досвіду, є реалізація кооперативного режиму, який дозволяє гравцям об'єднувати зусилля для досягнення спільних цілей.

Даний проект присвячений розробці ігрового програмного застосунку у жанрі космічного симулятора з акцентом на кооперативну взаємодію гравців. Проект передбачає створення віртуального світу, де гравці зможуть керувати персонажем, лагодити частини корабля, боротися з ворогами та намагатись змусити постійно заламаний корабель долетіти до пункту призначення разом з іншими гравцями.

Особлива увага приділяється розробці механік персонажів, що включають в себе система життєзабезпечення та взаємодія з предметами оточення.

Ключовими аспектами кооперативного режиму є реплікація ігрових об'єктів та мережева інфраструктура, що полягає у створенні ігрових сесій, під'єднання гравців та функціонал ігрового лобі.

Важливим елементом ігрового процесу є HUD, який відображає ключову інформацію про стан персонажа та ігрового світу є індикатори стану персонажа та інформація про інших гравців, а саме відображення їх очки життя та кисню для забезпечення координації дій.

Реалізація цих ключових аспектів дозволить створити захоплюючий та динамічний ігровий процес, що забезпечить гравцям неповторний досвід спільного дослідження космосу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

1.1.1 Аналіз розвитку кооперативних ігор

Космічні симулятори є популярним жанром відеоігор, який дозволяє гравцям відчути себе членом космічного екіпажу, який виконує свою власну роль у механізмі управління кораблем. Цей жанр характеризується широким спектром геймплейних механік, починаючи симуляції системи роботи корабля, до аркадних боїв з ворогами.

Кооперативні ігри, де гравці об'єднуються для досягнення спільної мети, стали одним з найдинамічніших сегментів ігрової індустрії за останні роки. Цей жанр пропонує унікальний досвід, який відрізняється від традиційних змагальних ігор, де гравці борються один з одним.

Походження кооперативних ігор можна прослідкувати ще до часів ранніх настільних ігор, наприклад, у настільній грі "Dungeons & Dragons" 1974 року гравці утворювали команди для досягнення спільної мети. Такі ігри не тільки заохочували співпрацю між гравцями, але й сприяли розвитку соціальних навичок та стратегічного мислення. З розвитком відеоігор жанр еволюціонував, пропонуючи нові та інноваційні механіки геймплею. Сучасні кооперативні відеоігри дозволяють гравцям об'єднуватися в мережеві команди, боротися з ворогами, виконувати завдання і навіть змагатися у глобальних рейтингах.

Одними з ранніх прикладів кооперативних відеоігор є Gauntlet 1985 року (див. рис. 1.1), який пропонували аркадний досвід, де гравці повинні були працювати разом, щоб вижити. У грі кооперативний геймплей був центральним аспектом, який робив гру такою захоплюючою. Чотири гравці могли об'єднати сили, щоб грати за чотирьох різних героїв: Воїна, Мага, Гнома та Ельфа, кожен з яких мав унікальні здібності та зброю. Спільна робота була ключовою для успіху, адже гравцям доводилося спілкуватися, координувати свої дії та ділитися ресурсами, щоб подолати орди ворогів, вирішувати головоломки та знаходити скарби.

Рисунок 1.1 – Ігровий процес Gauntlet 1985 року

У 1990-х роках жанр почав розвиватися з появою ігор, таких як Secret of Mana 1993 року (див. рис. 1.2), які пропонували більш сюжетні та орієнтовані на співпрацю механіки.

Рисунок 1.2 – Ігровий процес Secret of Mana 1993 року

На сьогоднішній день кооперативні ігри представлені в різних жанрах і формах. Відеоігри, такі як "Overcooked 2" (2018), "Minecraft" (2011) та "Sea of Thieves" (2018), демонструють широкий спектр можливостей кооперації – від спільного виживання до креативної співпраці. Хоча і ці ігри вийшли вже відносно давно, завдяки гарно продуманим ігровим механікам та постійним оновленням вони досі є актуальними.

Однією з ключових тенденцій сучасних кооперативних ігор є їхня складність і глибина. Гравці мають виконувати ролі з унікальними здібностями, що вимагає координації та стратегічного планування. Це сприяє розвитку навичок командної роботи та спілкування.

Кооперативні ігри мають позитивний вплив на соціальну взаємодію. Вони сприяють розвитку комунікативних навичок, умінню вирішувати конфлікти та працювати в команді. Особливо це стосується молоді, яка через ігри може навчитися ефективно співпрацювати з іншими.

Існує ряд факторів, які сприяли популярності кооперативних ігор в останні роки. Одним з найважливіших факторів є зростання популярності онлайн-ігор. Інтернет дозволив гравцям з усього світу легко з'єднуватися один з одним і грати разом. Це зробило кооперативні ігри більш доступними для ширшої аудиторії.

Ще одним фактором, який сприяв популярності кооперативних ігор, є зростання попиту на соціальні ігри. Гравці все частіше шукають ігри, в які вони можуть грати зі своїми друзями і сім'єю. Кооперативні ігри пропонують гравцям можливість спілкуватися один з одним і працювати разом для досягнення спільної мети. Це може бути дуже корисним досвідом, який може зміцнити стосунки між гравцями.

Майбутнє кооперативних ігор виглядає багатообіцяючим. Цей жанр продовжує рости і розвиватися, і з'являються все нові та інноваційні ігри. Зростання популярності онлайн-ігор і попит на соціальні ігри ймовірно, призведуть до того, що кооперативні ігри стануть ще більш популярними в найближчі роки.

Існує ряд тенденцій, які, ймовірно, вплинуть на майбутнє кооперативних ігор. Однією з найважливіших тенденцій є зростання популярності асиметричних

кооперативних ігор. У цих іграх у гравців є різні ролі і здібності, і вони повинні працювати разом, щоб досягти успіху. Асиметричні кооперативні ігри можуть бути дуже захоплюючими і складними, і вони стають все більш популярними.

Кооперативні ігри пройшли довгий шлях від простих настільних ігор до складних відеоігор, що об'єднують гравців з усього світу. Вони мають значний вплив на розвиток соціальних навичок та психічне здоров'я, але також стикаються з певними викликами. Майбутнє цього жанру виглядає перспективним завдяки новим технологіям та постійному інтересу з боку гравців.

Ідея кооперативного космічного симулятора на 4 гравців, де команда повинна лагодити корабель під час польоту та боротися з монстрами, які проникають через пробоїни, вписується в цей жанр, пропонуючи унікальний ігровий досвід, що поєднує елементи виживання, командної роботи та управління ресурсами. Можемо виділити наступні ключові та додаткові аспекти галузі:

а) ключові аспекти предметної галузі:
   1) кооперативний геймплей: Гра повинна бути розроблена з урахуванням спільних дій 4 гравців, де кожен гравець має свою роль і відповідальність за успіх місії;
   2) динамічна система пошкоджень: Корабель повинен отримувати пошкодження під час польоту, що створюватиме пробоїни, через які будуть проникати монстри, та завади для гравців, наприклад вимкнення світла чи сповільнення корабля;
   3) система ремонту: Гравці повинні мати можливість лагодити пошкоджені частини корабля, використовуючи необхідні інструменти під різні види поломок;
   4) бойова система: Гравці повинні мати можливість боротися з монстрами, використовуючи різні види зброї та тактики;
   5) управління ресурсами: Гравці повинні купувати та розподіляти ресурси, необхідні для ремонту корабля та боротьби з монстрами;

6) ігрові режими: Можливість вибору різних ігрових режимів, наприклад, одиночна кампанія, режим з розділеним екраном, та онлайн режим;
7) баланс складності: Необхідно забезпечити оптимальний рівень складності, щоб гра була цікавою і водночас доступною для гравців з різним рівнем досвіду;

б) додаткові аспекти:
1) графічний стиль: Вибір графічного стилю, який відповідає тематиці гри. В нашому випадку реалістичний стиль буде ідеально підходити для гри;
2) звуковий дизайн: Створення звукових ефектів та музики, які посилять занурення гравців в ігровий світ;
3) інтерфейс користувача: Розробка зручного та інтуїтивно зрозумілого інтерфейсу, який дозволить гравцям легко керувати кораблем, використовувати інструменти та взаємодіяти один з одним.

Враховуючи усі ці аспекти, використання ігрового рушія Unreal engine 5 полегшить розробку, завдяки наявній в ньому революційної системи реплікації, що забезпечує безперебійну синхронізацію даних між клієнтами в реальному часі. Завдяки цьому, Unreal Engine 5 гарантує плавний ігровий процес без затримок та лагів під час багатокористувацької ігри.

Для забезпечення цікавого геймплею, необхідно продумати багато механік персонажів, які гравці мають використовувати для вирішення ситуацій, які пропонує гра. Механіки персонажа мають відповідати вимогам, які ставить гра.

1.1.2 Аналіз конкурентів

Кооперативні ігри надають гравцям можливість спільної гри, де взаємодія та співпраця є ключовими елементами. Цей тип ігор сприяє соціальній взаємодії та зміцненню командних навичок. У цьому аналізі розглянемо такі популярні кооперативні ігри: Overcooked 2, Sea of Thieves, A Way Out, It Takes Two та Left 4

Dead 2. Аналіз допоможе визначити сильні та слабкі сторони кожної гри та їх вплив на ринок кооперативних ігор.

Overcooked 2 – це кулінарна кооперативна гра, розроблена Ghost Town Games і видана Team17. Гра відома своєю простотою та доступністю: вона має просте управління і зрозумілу механіку, що робить її привабливою для широкої аудиторії. Високий рівень кооперації, що вимагає активної комунікації та координації дій між гравцями, сприяє залученню та взаємодії. Більше того, велика кількість рівнів та доповнень забезпечує довготривалу зацікавленість гравців. Проте гра може стати занадто напруженою, що може відштовхнути гравців, які шукають розслаблення. Також, попри велику кількість рівнів, механіка гри може здатися повторюваною з часом, що обмежує повторний інтерес. На рисунку 1.3 можна побачити ігровий процес Overcooked 2.

Рисунок 1.3 – Ігровий процес Overcooked 2

Sea of Thieves – це піратська кооперативна гра від першої особи, розроблена Rare і видана Microsoft Studios. Гра пропонує масштабний відкритий світ з безліччю можливостей для дослідження та пригод, що приваблює гравців. Високий рівень взаємодії дозволяє гравцям взаємодіяти з багатьма елементами світу та один з

одним, що робить гру дуже соціальною. Постійний розвиток гри, регулярні оновлення та новий контент підтримують інтерес гравців. Однак, деяким гравцям може не вистачати конкретної мети або сюжету, що знижує мотивацію. Крім того, граючи соло, досвід може бути значно менш цікавим і ефективним, що обмежує привабливість для індивідуальних гравців. На рисунку 1.4 можна побачити ігровий процес Sea of Thieves.

Рисунок 1.4 – Ігровий процес Sea of Thieves

A Way Out – це кооперативна пригодницька гра від третьої особи, розроблена Hazelight Studios і видана Electronic Arts. Гра пропонує унікальний кооперативний досвід, де кожен гравець має свою роль і впливає на хід історії. Сильний сюжет гри, який тримає гравців у напрузі до самого кінця, є однією з основних переваг. Крім того, висока якість графіки і постановки сцен додають грі кінематографічного відчуття. Незвичайний підхід до кооперативного геймплею дозволяє гравцям відчувати справжню командну роботу та взаємодію, що є рідкістю у сучасних іграх. Проте, через сюжетну спрямованість, ігровий процес має обмежену повторюваність, що знижує бажання гравців повертатися до неї знову. Крім того, гра вимагає постійної наявності другого гравця, що може бути незручним для деяких гравців. Незважаючи на ці недоліки, A Way Out залишається яскравим

прикладом того, як інноваційні ідеї можуть змінювати звичний підхід до кооперативних ігор. На рисунку 1.5 можна побачити ігровий процес A Way Out.

Рисунок 1.5 – Ігровий процес A Way Out

It Takes Two – це кооперативна пригодницька гра, також розроблена Hazelight Studios. Гра пропонує безліч різних ігрових механік, що тримають гравців у постійному інтересі. Сюжет гри торкається важливих тем стосунків та співпраці, що додає їй глибокого емоційного впливу. Висока якість кооперативного досвіду вимагає тісної співпраці між гравцями, що робить її унікальною. Однак, як і у випадку з A Way Out, гра має обмежену привабливість після одного проходження. До того ж, гра не підтримує одиночний режим, що може відштовхнути деяких гравців. На рисунку 1.6 можна побачити ігровий процес It Takes Two.

Рисунок 1.6 – Ігровий процес It Takes Two

Аналіз цих ігор показує, що успішні кооперативні ігри мають певні спільні риси, такі як висока якість кооперативного геймплею, різноманітність механік та постійний розвиток контенту. Проте, кожна з них має свої слабкі сторони, які потрібно враховувати при розробці нової гри. Важливо забезпечити баланс між доступністю та викликом, а також знайти оптимальний рівень реіграбельності, щоб гравці хотіли повертатися до гри знову і знову.

1.2 Виявлення проблем та актуалізація рішень

Незважаючи на популярність жанру космічних симуляторів, багато ігор стикаються з певними проблемами, які можуть негативно вплинути на ігровий досвід. Однією з таких проблем є одноманітність геймплею, особливо в кооперативному режимі, коли гра надмірно фокусуються на бойових діях, нехтуючи іншими аспектами ігрового процесу. Це призводить до повторюваності та одноманітності, що може швидко набриднути гравцям. Корабель стає просто платформою для зброї, а командна робота зводиться до звичайного знищення ворожих цілей. Часто кооперативні ігри не використовують весь потенціал командної взаємодії, пропонуючи обмежений набір ролей та способів взаємодії між гравцями. Кожен гравець виконує схожі дії, не маючи чіткої спеціалізації та можливості впливати на дії інших гравців.

Для вирішення цих проблем у розробці важливо враховувати різноманітні аспекти геймплею. Окрім бойових дій, гра повинна включати різноманітні активності, такі як ремонт корабля, управління ресурсами, дослідження невідомих планет та взаємодія з космічним оточенням. Для забезпечення різноманітності та динамічності гри можна використовувати систему пошкоджень корабля, яка впливає на геймплей та вимагатиме від гравців швидкого реагування та командної роботи. Крім того необхідно запропонувати гравцям вибір з кількох ролей, кожна з яких матиме свою спеціалізацію та унікальні навички, важливі для виконання певних завдань.

Впровадження нововведень і вирішення проблем, що виникають у жанрі, є ключовими для створення успішного кооперативного космічного симулятора.

1.3 Постановка задачі

Створення захоплюючого кооперативного космічного симулятора вимагає вирішення комплексу завдань, пов’язаних з розробкою ключових ігрових механік та мережевої інфраструктури:

а) реалізація кооперативного режиму є фундаментальною задачею, що вимагає:
   1) розробки системи реплікації ігрових об’єктів для забезпечення синхронізації стану корабля, персонажів, монстрів та інших елементів ігрового світу між усіма гравцями;
   2) створення мережевої інфраструктури, що включатиме функціонал для створення ігрових сесій, підключення гравців, обробки мережевих повідомлень, забезпечення безпеки та стабільності з’єднання;
   3) розробка ігрового лобі, де гравці зможуть створювати та приєднуватись до ігрових сесій, налаштовувати параметри гри та спілкуватись між собою;

б) розробка механік персонажа повинна враховувати;
   1) систему життєзабезпечення, що включає показники кисню, здоров’я та інші параметри, які впливатимуть на стан персонажа та його здатність виконувати дії;
   2) взаємодію з предметами оточення: можливість лагодити частини корабля, використовувати інструменти, взаємодіяти з панелями керування та іншими об’єктами;
   3) бойову систему: реалізацію різних видів зброї, а також механізми отримання пошкоджень та смерті персонажа;

в) розробка HUD вимагає:
   1) відображення індикаторів стану персонажа, таких як рівень кисню, здоров’я, запас ресурсів, стан зброї;

2) відображення інформації про інших гравців, що дозволить координувати дії та контролювати стан команди. Це включатиме відображення рівня здоров’я, кисню та ролі гравця;
3) інтуїтивний та зручний дизайн, який не перевантажуватиме гравця інформацією, але водночас надасть все необхідне для ефективного управління персонажем та участі в кооперативній грі;

г) технічні аспекти:
1) розробка буде проводитись в Unreal Engine 5, використовуючи мову програмування C++ та систему візуального скріптинга Blueprint;
2) для реалізації мережевого функціоналу буде використана Epic Online Services, що надає широкий спектр інструментів для розробки багатокористувацьких ігор[1];
3) необхідно забезпечити оптимізацію продуктивності, враховуючи складність реплікації та обробки великого об’єму даних в реальному часі.

Успішна реалізація поставлених задач дозволить створити унікальний кооперативний космічний симулятор, який запропонує гравцям захоплюючий геймплей, заснований на командній роботі, виживанні та управлінні ресурсами в умовах динамічного та небезпечного космічного середовища.

1.3.1 Цільова аудиторія

Основну аудиторію цієї гри складають люди віком від 12 до 27 років. Ця гра насамперед приверне увагу тих, хто полюбляє грати в команді, а також тих, хто отримує задоволення від спільних ігор з родиною або друзями. Вона також може стати цікавою для тих, хто захоплюється динамічним геймплеєм, де необхідно швидко приймати рішення. Особливо привабливою гра буде для геймерів, які вже мають досвід у іграх подібного жанру, оскільки вони зможуть оцінити усі тонкощі та виклики, які вона пропонує.

Тобто гра має багато чого запропонувати різноманітним категоріям гравців.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Постановка мети

Метою виконання роботи є створення програмного модулю, що реалізує механіки персонажа, кооперативну взаємодію та HUD для рушію Unreal Engine 5. Програмний модуль повинен бути компактним, але охоплювати базову функціональність цих аспектів, які використовуються в кооперативних космічних симуляторах.

Механіка персонажа має містити у собі такі основін аспекти, як: реалізація системи життєзабезпечення, що включатиме показники кисню та здоров’я, які динамічно змінюватимуться в залежності від стану персонажа та його дій, створення системи взаємодії з предметами оточення, такими як інструменти для ремонту корабля, панелі керування, зброя для боротьби з монстрами, розробка бойової системи, що включатиме різні види зброї, механізми отримання пошкоджень та смерті персонажа.

Для успішної імплементації кооперативного режиму, необхідно реалізувати систему реплікації ігрових об'єктів для синхронізації стану корабля, персонажів, монстрів та інших елементів ігрового світу між гравцями, створити мережеву інфраструктуру, що забезпечить функціонал для створення ігрових сесій, підключення гравців, обробки мережевих повідомлень та розробити ігрове лобі, де гравці зможуть створювати та приєднуватись до ігрових сесій, налаштовувати параметри гри та спілкуватись.

На HUD необхідно розмістити відображення індикаторів стану персонажа, таких як рівень кисню, здоров'я, запас ресурсів, стан зброї, відображення інформації про інших гравців, що дозволить координувати дії та контролювати стан команди, включаючи відображення рівня здоров’я, кисню та ролі гравця та індикатор успішності польоту. Необхідно розробити інтуїтивний та зручний дизайн, що не перевантажуватиме гравця інформацією, але надасть все необхідне для управління персонажем та участі в кооперативній грі.

Програмний модуль має бути розроблений з урахуванням можливості подальшого розширення та інтеграції з іншими системами гри.

2.2 Загальний опис

Система кооперативної гри, механік персонажа та HUD представляє собою набір взаємопов'язаних компонентів, що забезпечують спільну гру, керування станом персонажів та відображення ключової інформації для гравців.

Кооперативний режим реалізується за допомогою мережевої інфраструктури, що включає функціонал для створення ігрових сесій, підключення гравців, обробки мережевих повідомлень та синхронізації стану ігрових об'єктів між усіма учасниками. Система реплікації гарантує, що всі гравці бачать однакову інформацію про стан корабля, персонажів, монстрів та інших елементів ігрового світу, забезпечуючи плавний та синхронізований ігровий процес.

Механіки персонажа включають систему життєзабезпечення, що відстежує рівень кисню та здоров’я, впливаючи на здатність персонажа виконувати дії. Взаємодія з оточенням дозволяє гравцям лагодити частини корабля, використовувати інструменти, взаємодіяти з панелями керування та іншими об'єктами. Бойова система передбачає використання різних видів зброї та тактик для боротьби з монстрами, враховуючи отримання пошкоджень та смерть персонажа.

HUD відображає ключову інформацію про стан персонажа та ігровий світ, допомагаючи гравцям приймати рішення та координувати дії. Індикатори стану персонажа, такі як рівень кисню, здоров'я, запас ресурсів та стан зброї, дозволяють контролювати стан персонажа та ресурси. Інформація про інших гравців, включаючи їх рівень здоров'я, кисню та роль, допомагає координувати дії та контролювати стан команди. Інтуїтивний та зручний дизайн HUD забезпечує легкий доступ до важливої інформації, не перевантажуючи гравця зайвими даними.

2.3 Загальні обмеження

Ігровий програмний застосунок повинен відповідати наступним обмеженням:

а) механіки персонажа:
   1) модуль механік персонажа має бути інтегрований до ігрового проекту;
   2) для оцінки працездатності модуля необхідна наявність реалізованої системи пошкоджень корабля та механіки проникнення монстрів;
   3) модуль механік персонажа не відповідає за коректність роботи системи життєзабезпечення та інших систем корабля;
   4) збереження стану персонажа покладено на функціональність рушія;

б) кооператив:
   1) модуль кооперативу має бути інтегрований до ігрового проекту та взаємодіяти з іншими модулями, такими як механіки персонажа, система пошкоджень та інші;
   2) для оцінки працездатності модуля кооперативу необхідна наявність реалізованого ігрового лобі та функціоналу створення/підключення до ігрових сесій;
   3) Модуль кооперативу не відповідає за безпеку та стабільність мережевого з'єднання, це покладено на Epic Online Services;

в) HUD:
   г) Модуль HUD має бути інтегрований до ігрового проекту та взаємодіяти з модулем механік персонажа та кооперативу для отримання необхідних даних;
   д) Для оцінки працездатності модуля HUD необхідна наявність реалізованих механік персонажа та функціоналу кооперативу.

Врахування цих обмежень на стадії проектування та розробки дозволить створити стабільний та масштабований ігровий застосунок, здатний забезпечити гравцям захоплюючий ігровий досвід.

2.4 Припущення та залежності

Під час розробки механік персонажа, кооперативного режиму та HUD розглядаються наступні припущення та залежності:

а) механіки персонажа:
   1) реалізація системи життєзабезпечення може бути ускладнена особливостями рушія Unreal Engine 5, що може потребувати нестандартних рішень для відстеження та зміни показників кисню, здоров'я та інших параметрів;
   2) взаємодія з предметами оточення, такими як лагодження частин корабля, може вимагати додаткових розрахунків для визначення доступності та коректності дій, враховуючи динамічний стан корабля;
   3) бойова система може бути обмежена можливостями рушія Unreal Engine 5 щодо симуляції фізики та реалізації бажаного рівня реалізму бою;

б) кооператив:
   1) система реплікації ігрових об'єктів може зіткнутися з проблемами продуктивності при великій кількості гравців та складних ігрових об'єктах, що потребуватиме ретельної оптимізації;
   2) мережева інфраструктура може бути чутливою до якості інтернет-з'єднання гравців, що може призвести до затримок та розсинхронізації ігрового процесу;
   3) функціонал ігрового лобі може вимагати додаткового тестування для забезпечення стабільності та зручності використання;

в) HUD:
   1) відображення індикаторів стану персонажа може бути ускладнене обмеженнями Unreal Engine 5 щодо розміщення та оновлення візуальних елементів інтерфейсу;

2) відображення інформації про інших гравців може потребувати додаткових ресурсів для обробки та передачі даних, що може вплинути на продуктивність гри;
3) інтуїтивний дизайн HUD може вимагати багаторазового тестування та внесення змін для досягнення оптимального балансу між інформативністю та зручністю використання;

г) загальні припущення:
1) використані значення параметрів персонажів, монстрів та інших ігрових об'єктів можуть конфліктувати з вбудованими механізмами обробки та збереження серіалізованих даних всередині рушія;
2) цільова операційна система, на якій планується запуск гри, може неадекватно реагувати на модуль збереження даних через ймовірність відсутності доступу до файлової системи комп'ютера;
3) використання сторонніх плагінів та бібліотек може призвести до конфліктів з Unreal Engine 5 та потребуватиме додаткових зусиль для інтеграції та налагодження.

Враховуючи ці припущення та залежності, необхідно ретельно планувати розробку, проводити тестування на різних етапах та вносити необхідні зміни для забезпечення стабільності, продуктивності та позитивного ігрового досвіду.

3 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ

3.1 Опис ідеї та створення плану розробки

Ідея гри – це кооперативний космічний симулятор на 4 гравців, де команда повинна лагодити корабель під час польоту, боротися з монстрами, які проникають через пробоїни, та намагатися долетіти до пункту призначення, незважаючи на постійні поломки.

Основна гри складається з:

- гравці опиняються на пошкодженому космічному кораблі, на якому треба втекти від погоні;
- корабель постійно зазнає пошкоджень, що створює пробоїни та поламки в системі корабля;
- гравці повинні розподілити ролі (капітан, інженер, боєць, механік), щоб ефективно лагодити корабель, боротися з монстрами та підтримувати життєзабезпечення;
- успіх місії залежить від злагодженої командної роботи, управління ресурсами та швидкого реагування на критичні ситуації.

В той час, як основні геймплейні елементи – це:

- гравець бачить світ від першої особи, що забезпечує максимальне занурення в роль члена екіпажу;
- кожен гравець має свої унікальні здібності та інструменти, необхідні для виконання його ролі;
- корабель має різні системи (двигун, щити, життєзабезпечення), які потребують постійного ремонту та контролю;
- бойова система передбачає використання різної зброї та тактики для боротьби з монстрами.

Розробка гри охоплює три основні етапи: концепцію та дизайн, програмування та тестування.

Концепція та дизайн ґрунтується на деталізації ігрового світу, його сеттінгу та історії. На цьому етапі розробляється концепція персонажів, їхні ролі та

здібності. Важливо визначити візуальний стиль гри та створити зручний інтерфейс користувача.

Програмування включає реалізацію основних ігрових механік: руху, ремонту, бою та взаємодії з об'єктами. На цьому етапі розробляється система пошкоджень корабля, механіка проникнення монстрів та штучний інтелект для них. Додатково реалізується кооперативний режим та мережева інфраструктура.

Тестування та налагодження мають на меті перевірити ігрові механіки, баланс та кооперативний режим. Під час тестування виявляються та виправляються помилки, а також оптимізується продуктивність гри.

Технології:

- розробка буде проводитись на Unreal Engine 5;
- мова програмування: C++ та Blueprint;
- мережевий функціонал: Epic Online Services.

Рушій Unreal Engine 5 забезпечує високоякісну графіку та реалістичну фізику. Програмування відбуватиметься з використанням мови C++ та візуальних скриптів Blueprint, що дозволить створити складні ігрові механіки та інтерактивні системи[2]. Мережевий функціонал гри буде реалізовано за допомогою Epic Online Services, що забезпечить стабільний кооперативний режим і зручний багатокористувацький досвід для гравців.

3.2 Розробка ігрових механік

Одним з ключових елементів, що формують ігровий досвід, є система персонажів, їх унікальні характеристики та способи взаємодії з оточенням.

У грі кожен персонаж володіє унікальним набором характеристик, що визначають його роль та здібності в команді. Така спеціалізація сприяє розвитку кооперативної стратегії, адже гравцям необхідно координувати свої дії, враховуючи сильні та слабкі сторони кожного персонажа.

Рух персонажів у віртуальному просторі гри реалізовано за допомогою стандартних методів управління: клавіатури або геймпада. Швидкість пересування персонажа динамічно змінюється в залежності від ваги предмета, який він

переносить, додаючи реалізму та тактичної глибини. Взаємодія з оточенням включає можливість маніпулювання дверима - відкривання та закривання - що підкреслює інтерактивність ігрового світу.

Гравці мають доступ до різноманітних інструментів та зброї, які відіграють ключову роль у виконанні поставлених завдань. До таких інструментів належать зварювальний апарат, паяльник та вогнегасник, кожен з яких має своє специфічне застосування. Використання інструментів та зброї пов’язане з витратою ресурсів: заряду, палива або боєприпасів. Вогнегасник використовується для

Окрім інструментів, гравці можуть використовувати аптечки та балони з киснем для відновлення здоров’я та запасів кисню, що є критично важливими показниками для виживання в ігровому світі.

Гра розрахована на кооперативну гру на розділеному екрані або по мережі. Кожен гравець керує одним персонажем, володіючи унікальним набором навичок. Ефективна командна робота вимагає злагодженої координації дій та чіткої розподілу обов’язків.

HUD – це візуальний інтерфейс, що надає гравцям ключову інформацію про стан персонажів та ігровий процес. Він відображає:

- рівень здоров’я (HP), що показує життєздатність персонажа;
- запаси кисню, що визначають здатність персонажа знаходитись в середовищі з обмеженим доступом повітря;
- заряд інструменту або кількість амуніції, що дозволяє контролювати ресурси;
- таймер, що відраховує час до наступної контрольної точки, створюючи відчуття невідкладності;
- стан інших персонажів, сприяючи командній координації.

HUD не тільки надає гравцям ключову інформацію про стан персонажів та ігровий процес, але й виступає критичним інструментом для стратегічного планування та ухвалення рішень в режимі реального часу.

3.3 UML проектування ПЗ

Перед початком розробки ігрового застосунку, були визначені основні функції, які може виконувати гравець. Після аналізу цих функцій, була створена Use-case діаграма (див. рис. 3.1).

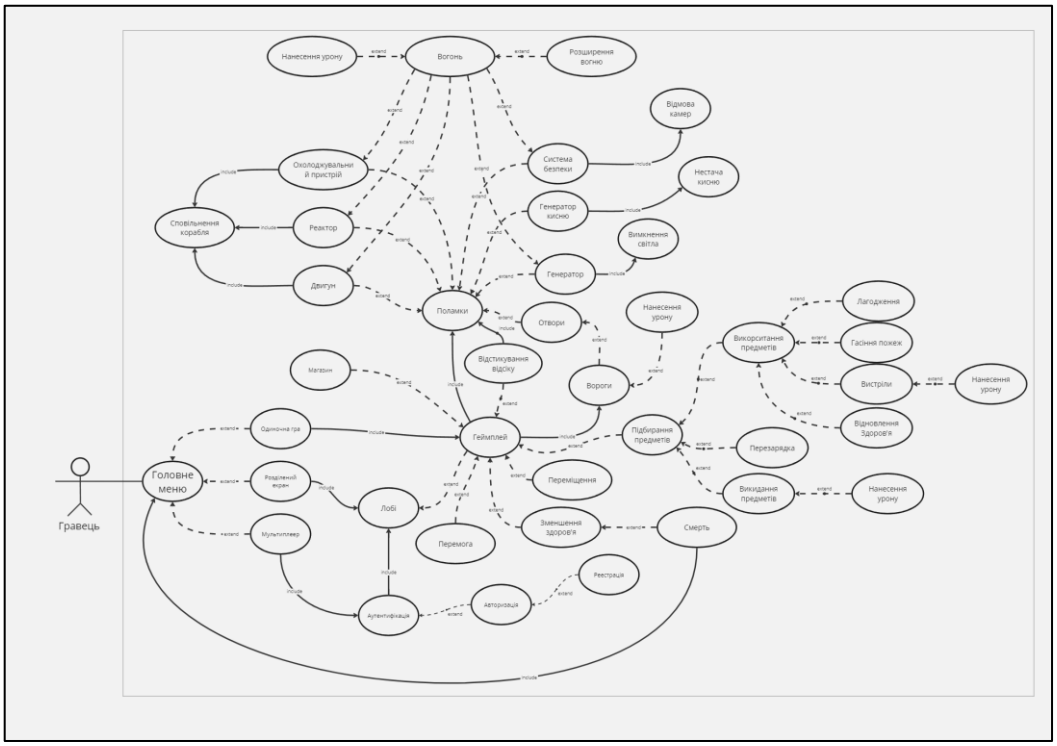


Рисунок 3.1 – Use-case діаграма ігрового застосунку (рисунок виконаний самостійно)

Гравець може розпочати одиночну гру, в режимі розділеного екрану або за допомогою мережевого з’єднання. В усіх трьох випадках, він почне гру, під час якої він має змогу переміщуватися у просторі, підбирати, використовувати та викидати предмети.

З предметів наявні наступні екземпляри:

- пістолет – стріляє одиночними вистрілами;
- полуавтоматична гвинтівка – стріляє по 3 кулі;
- автоматична гвинтівка – стріляє, поки натиснута клавіша;
- вогнегасник – для гасіння пожеж;

- паяльник – для лагодження модулей корабля;
- зварювальний апарат – для лагодження дірок та дверей;
- балон з киснем – дозволяє деякий час не дихати в відсіку без кисню;
- аптечка – відновлю здоров’я персонажу.

Кожен з цих предметів гравець може підібрати та використати на свій власний розсуд. Також при викиданні предмети завдають шкоди ворогам. Також для зброї та інструментів є можливість перезарядити боєзапас у термінала, який знаходиться на кораблі. Поповнити вміст терміналів можна через магазин.

Ігрові персонажі будуть мати наступні характеристики:

- HP – здоров’я персонажа. При зниженні до 0, персонаж помирає, тим самим вибуває з гри повністю, або якщо гра виконувалась у со-ор інший гравець може викупити нового персонажа в магазині для вільного гравця;
- кисень. Під час перебування у секції з пожежею або пробоїною він буде знижуватися (кожны 2 сек на 1 од). Після вичерпання запасів кисню, почне знижуватись HP (кожні 1 сек на 1 од), що може призвести до смерті. Якщо персонаж опускає забрало шолома скафандру, то кисень спочатку буде витрачатися з нього;
- швидкість переміщення. Вона залежить від предмета, який тримає гравець;
- рівень бойових навичок – впливає на якість персонажа у бою - додатковий урон від зброї, розкид стрільби (хитання ствола), відстань відкидання;
- рівень навичок Інженера – впливає на здатність персонажу лагодити складні електричні системи, такі як реактор, двигуни, систему охолодження, систему подачи кисню, систему автопілота, генератори, камери та систему спостереження;
- рівень навичок Механіка – впливає на здатність персонажу лагодити механічні пошкодження корабля, такі як пробоїни, дверей та пожеж.

У грі представлено чотири основні класи персонажів: Офіцер, Механік, Інженер та Капітан. Кожен із цих класів має унікальний набір навичок та

спеціалізацій, що робить їх важливими для успішного виконання різноманітних завдань у грі. Рівень навичок кожного персонажа розподілений відповідно до його класу, що відображає їхні сильні та слабкі сторони, а також їхню роль у команді. Ці навички детально представлені у таблиці 3.1, де можна побачити, як саме розподілені показники навичок для кожного класу, включаючи такі аспекти, як бойові навички, здоров’я, кисень та здатність до лагодження різноманітних частин корабля.

Таблиця 3.1 – Розподілення характеристик персонажа відповідно до класу

	Рівень навичок Механіка	Рівень навичок Інженера	Рівень бойових навичок	Здоров’я	Кисень
Офіцер	1	1	3	200	100
Капітан	2	2	2	100	100
Механік	3	1	1	100	200
Інженер	2	3	1	100	100

Обрати можна буде лише унікальні класи, тобто під час багатокористувацької гри обрати двох капітанів чи трьох офіцерів бути не може.

3.4 Проектування архітектури ПЗ

Гри буде розроблена за допомогою ігрвого рушія Unreal Engine 5, та мови візуального скриптінгу Blueprint, з використанням мови програмування C++[3].

Unreal Engine 5 – це ігровий рушій, розроблений компанією Epic Games. Unreal Engine 5 ґрунтується на потужних технологіях, таких як Lumen та Nanite, й пропонує безпрецедентний рівень реалізму та деталізації, роблячи його ідеальним інструментом для розробників ігор, які прагнуть досягти нових вершин. Одне з

ключових особливостей Unreal Engine 5 – це революційна система реплікації для онлайн ігор, яка забезпечує безперебійну синхронізацію даних між клієнтами в реальному часі. Завдяки цьому, Unreal Engine 5 гарантує плавний ігровий процес без затримок та лагів.

Blueprint – це інноваційна система візуального програмування, яка є невід'ємною складовою частиною Unreal Engine 5. Завдяки Blueprint розробники отримують потужний інструмент для створення складної ігрової логіки та геймплею без необхідності занурюватися у текстове програмування. Ця система дозволяє використовувати візуальні скрипти для реалізації різноманітних функцій, від базових дій персонажів до складних механік ігрового процесу. Це значно прискорює процес розробки, робить його більш зрозумілим та доступним, особливо для тих, хто не має великого досвіду в програмуванні.

Для забезпечення мережевої взаємодії в грі було обрано Epic Online Services. Ця потужна платформа надає широкі можливості для організації багатокористувацьких ігор, дозволяючи гравцям з усього світу збиратися разом і насолоджуватися спільним геймплеєм. Epic Online Services забезпечує надійні інструменти для створення лобі, матчмейкінгу, управління друзями та багатьох інших аспектів онлайн-гри. Використання цієї платформи гарантує стабільне та безперебійне з'єднання між гравцями, що є критично важливим для успішної роботи багатокористувацьких режимів.Epic Online Services – це набір безкоштовних, кроссплатформних онлайн-сервісів, розроблених компанією Epic Games для розробників відеоігор. Epic Online Services покликаний спростити та прискорити процес запуску, експлуатації та масштабування ігор. Цей комплекс послуг охоплює широкий спектр функціональних можливостей, необхідних для створення багатокористувацьких ігрових середовищ, включаючи:

Автентифікація та авторизація: Epic Online Services дозволяє розробникам інтегрувати безпечні та надійні системи автентифікації та авторизації, що дає змогу гравцям створювати облікові записи та отримувати доступ до своїх ігрових профілів на різних платформах.

Соціальні функції: Epic Online Services сприяє створенню динамічних ігрових спільнот, надаючи гравцям можливість спілкуватися з друзями, знаходити нових співгравців та формувати команди.

Мережеві функції: Epic Online Services забезпечує надійну та масштабовану мережеву інфраструктуру, необхідну для підтримки багатокористувацьких ігор, включаючи голосовий чат, текстові повідомлення та синхронізацію даних[4].

Аналітика даних: Epic Online Services надає розробникам доступ до цінних даних про поведінку гравців, що дозволяє їм вдосконалювати свої ігри та пропонувати кращий ігровий досвід.

Для реалізації схеми управління було обрано систему Enhanced Input, яка дозволяє динамічно змінювати схеми управління під час геймплею[5].

Enhance Input — це розширений модуль обробки вводу, призначений для ефективного та гнучкого налаштування різних типів управління в іграх. Ця система дозволяє розробникам легко конфігурувати і налаштовувати обробку вводу з різних джерел, таких як клавіатура, мишка, геймпад, сенсорний екран тощо. Enhance Input забезпечує потужний інструментарій для визначення, обробки та керування подіями, що дозволяє створювати складні та багатофункціональні схеми управління. Основними перевагами цієї системи є гнучкість та масштабованість, що надає розробникам змогу створювати складні конфігурації управління, які можуть легко адаптуватися під різні типи ігор та пристроїв. Це дозволяє забезпечити унікальний досвід для кожного користувача, незалежно від платформи, на якій він грає.

3.5 Проектування HUD

У грі впровадження основних принципів дизайну HUD є критично важливим для забезпечення оптимального ігрового досвіду. Ці принципи включають мінімалізм, чіткість, контекстність та ергономічність, і кожен з них грає унікальну роль у створенні ефективного та інтуїтивного інтерфейсу.

Принцип мінімалізму у дизайні HUD спрямований на те, щоб відображати лише найважливішу інформацію, необхідну для прийняття рішень гравцем.

Перенасичення екрану інформацією може відволікати та ускладнювати процес гри, тому зведення кількості елементів до мінімуму дозволяє гравцю зосередитися на основних аспектах геймплею. Усі непотрібні або вторинні елементи повинні бути усунені або згруповані так, щоб не заважати сприйняттю основної інформації.

Чіткість є ще одним важливим аспектом дизайну HUD. Індикатори та елементи інтерфейсу повинні бути легко розрізнюваними та зрозумілими з першого погляду. Це означає використання чітких шрифтів, контрастних кольорів та зрозумілих іконок, які не викликають сумнівів у їхньому значенні. Важливо, щоб гравець міг миттєво інтерпретувати інформацію, представлену на HUD, без потреби додатково роздумувати над її значенням[6].

Контекстність передбачає, що інформація, яка відображається на HUD, повинна бути актуальною для поточної ситуації в грі. Інтерфейс повинен динамічно змінюватися відповідно до ігрових подій, надаючи гравцеві саме ті дані, які йому потрібні в конкретний момент. Наприклад, під час бою важливо показувати рівень здоров’я та боєприпасів, тоді як під час дослідження світу більше значення мають індикатори напрямку та карти.

Ергономічність стосується зручності розташування елементів HUD для сприйняття гравцем. Важливо, щоб елементи інтерфейсу не заважали огляду ігрового світу та були розміщені так, щоб їх можна було швидко знайти і зчитати. Це передбачає ретельне планування розташування індикаторів та меню, врахування природних рухів очей та загальної композиції екрану. Грамотне розміщення елементів дозволяє гравцеві зосереджуватися на грі, не витрачаючи час на пошук необхідної інформації.

Таким чином, дотримання цих принципів дозволяє створити HUD, який не тільки покращує зручність використання інтерфейсу, але й підвищує загальне задоволення від гри. Мінімалізм, чіткість, контекстність та ергономічність є фундаментальними аспектами, які повинні враховуватися на кожному етапі розробки HUD, щоб забезпечити його ефективність та функціональність.

Важливою частиною інтерфейсу користувача у грі є система індикаторів стану, яка забезпечує гравців необхідною інформацією про стан їхніх персонажів, корабля та інших членів команди.

Основні компоненти HUD (head-up display) включають: індикатори стану персонажа, інших персонажів, корабля та інші інформаційні елементи.

До індикаторів стану персонажа відносяться:

Рівень здоров'я – це параметр, що відображається у вигляді смуги, яка зменшується в міру отримання персонажем пошкоджень. Для зручності гравців і наочності критичного стану, колір смуги динамічно змінюється під час отримання урону.

Аналогічно до індикатора здоров'я, рівень кисню відображається смугою, яка поступово зменшується при перебуванні персонажа в середовищі з недостатньою кількістю кисню. Колір смуги також змінюється під час зниження, попереджаючи гравця про те, що той знаходиться в середовищі без кисню.

Індикатор стану інших персонажів надає інформацію про стан кисню, роль та рівень здоров'я інших гравців. Такий інструмент є важливим для координації дій у командних завданнях, забезпечуючи гравців необхідною інформацією для прийняття стратегічних рішень та підтримки членів команди.

Індикатор успішності польоту відображає відстань до пункту призначення та прогрес. Завдяки цьому гравець може легко стежити за ходом виконання завдання та контролювати процес досягнення цілі.

Індикатор активного інструменту стан предмету, який зараз знаходиться у гравця в руках. Це дозволяє гравцям швидко оцінити готовність інструменту до використання та планувати свої дії відповідно до його заряду.

Система індикаторів у HUD сприяє підвищенню інтерактивності гри, забезпечуючи гравців своєчасною та точною інформацією про різні аспекти ігрового процесу. Це дозволяє їм приймати обґрунтовані рішення, що підвищує їх шанси на успішне завершення завдань та виживання у складних ігрових ситуаціях.

Для візуальної реалізації графічного інтерфейсу користувача в контексті космічного симулятора, пропонується використання футуристичного

мінімалістичного дизайну, що відповідає тематиці дослідження космосу. Графічний стиль буде реалізована в мінімалістичній білій палітрі, але яскраві акценти будуть використовуватись для візуального виділення ключових елементів інтерфейсу.

Графічний інтерфейс користувача буде адаптивним до різних роздільних здатностей екранів та співвідношень сторін. З метою забезпечення оптимального сприйняття інформації, розмір та розташування елементів інтерфейсу автоматично підлаштовуватимуться відповідно до характеристик конкретного відображувального пристрою.

У майбутньому передбачається можливість розширення функціональних можливостей графічного інтерфейсу користувача шляхом впровадження наступних інновацій:

- налаштування HUD: Користувачі матимуть можливість індивідуального налаштування розміру, розташування та прозорості елементів HUD з урахуванням особистих вподобань та потреб;
- додаткові Індикатори: Передбачається можливість відображення додаткових індикаторів, що інформують про стан різних систем корабля, таких як рівень палива, стан щитів, температура двигунів тощо;
- інтерактивна Міні-Карта: Гравцям буде надана можливість взаємодії з інтерактивною міні-картою, включаючи встановлення маркерів для позначення ключових точок та маршрутів.

Впровадження цих інноваційних функцій сприятиме подальшому розвитку та удосконаленню графічного інтерфейсу користувача, підвищуючи зручність та функціональність інтерфейсу та покращуючи загальний ігровий досвід гравців.

Процес розробки графічного інтерфейсу користувача для ігор часто потребує глибокого розуміння потреб гравців та їхньої взаємодії. В даному випадку, ґрунтовне дослідження, враховуючи всі вимоги, дозволило розробити HUD, який не тільки відповідає потребам окремого гравця, але й створює ефективне середовище для командної взаємодії (див. рис. 3.2).

Рисунок 3.2 – Макет HUD

На екрані гравця відображається комплексна інформація, що охоплює його фізичний стан та технічні параметри у віртуальному просторі. У верхньому лівому куті розташований червоний статус-бар, який відображає відносний рівень здоров'я. Поруч із ним знаходиться синій статус-бар, що індикує наявність кисню в балонах. На протилежному верхньому куті розташована інформація про стан здоров'я та рівень кисню команди.

Додатково, у нижній частині екрану знаходиться індикатор залишку заряду поточного предмету, який утримує гравець у руці. Індикатор шляху, який вказує напрямок, який корабель має пройти, та відображає вже пройдений шлях, позначений зеленим статус-баром зверху.

Завдяки комплексному підходу, інтерфейс пропонує гравцям широкий спектр інформації, необхідної для успішного проходження гри. HUD дозволяє гравцям одерживати всю необхідну інформацію про стан ігрового процесу, характеристики героя та окруження. Це дозволяє оптимізувати процес прийняття рішень та сприяє більш успішному проходженню гри.

Але важливим елементом цього інтерфейсу є його здатність підвищити ефективність командної взаємодії. HUD надає інструменти для зв'язку між

гравцями, дозволяючи їм спільно аналізувати ігровий процес, планувати тактику та ефективніше працювати в команді. Це зроблено шляхом впровадження інтуїтивно зрозумілих елементів інтерфейсу, які дозволяють гравцям легко обмінюватися інформацією та координувати свої дії.

Такий інтегрований підхід до розробки HUD створює новий рівень досвіду для гравців, зміцнюючи взаємодію та сприяючи успішному проходженню гри.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Розробка механік персонажа

4.1.1 Управління. Прив’язка кнопок

В Unreal Engine 5 є система Enhanced Input[5], яка дозволяє легко та зручно прив’язувати кнопки з пристрою введення до дій в грі. Enhanced Input має дві основні концепції, які будуть задіяні під час розробки гри:

Input Action – це комунікаційна ланка між системою Enhanced Input та кодом проекту.

Input Mapping Contexts – набір Input Action. Один гравець може мати багато Input Mapping Contexts та динамічно додавати та видаляти їх під час гри.

Для персонажа, потрібно створити наступні Input Action:

IA_Move – відповідає за переміщення персонажа.

IA_Look – відповідає за управління камерою в грі.

IA_Interact – відповідає за взаємодією з оточенням гри, наприклад підбирання предметів чи натискання фізичних кнопок в грі.

IA_Tape – відповідає за нанесення клейкої стрічки на зламані механізми.

IA_Use – відповідає за застосування предмету, який гравець тримає в руці.

IA_Drop – відповідає за викидання предмету, який гравець тримає в руці.

IA_DropOxygenTank – відповідає за скидання балону з киснем з персонажа.

IA_Pause – відповідає за відкриття меню під час гри.

Для того, щоб дозволити гравцеві виконувати усі ці дії, необхідно створити наступні Input Mapping Contexts:

IMC_Default – основний Input Mapping Contexts, що включає в себе IA_Move, IA_Look, IA_ Interact, IA_Tape, IA_Pause.

IMC_HandItem – буде додаватися до розкладки користувача, коли той підбиратиме предмети. Включає у себе IA_Use, IA_Drop.

IMC_OxygenTank – буде додаватися до розкладки користувача, коли тий підбиратиме балон з киснем. Включає в себе IA_ DropOxygenTank.

Вказавши для кожного Input Action певну кнопку, в Input Mapping Contexts, маємо дві схеми управління для нашої гри, а саме: за допомогою миші та клавіатури, та геймпаду (див. рис. 4.1).

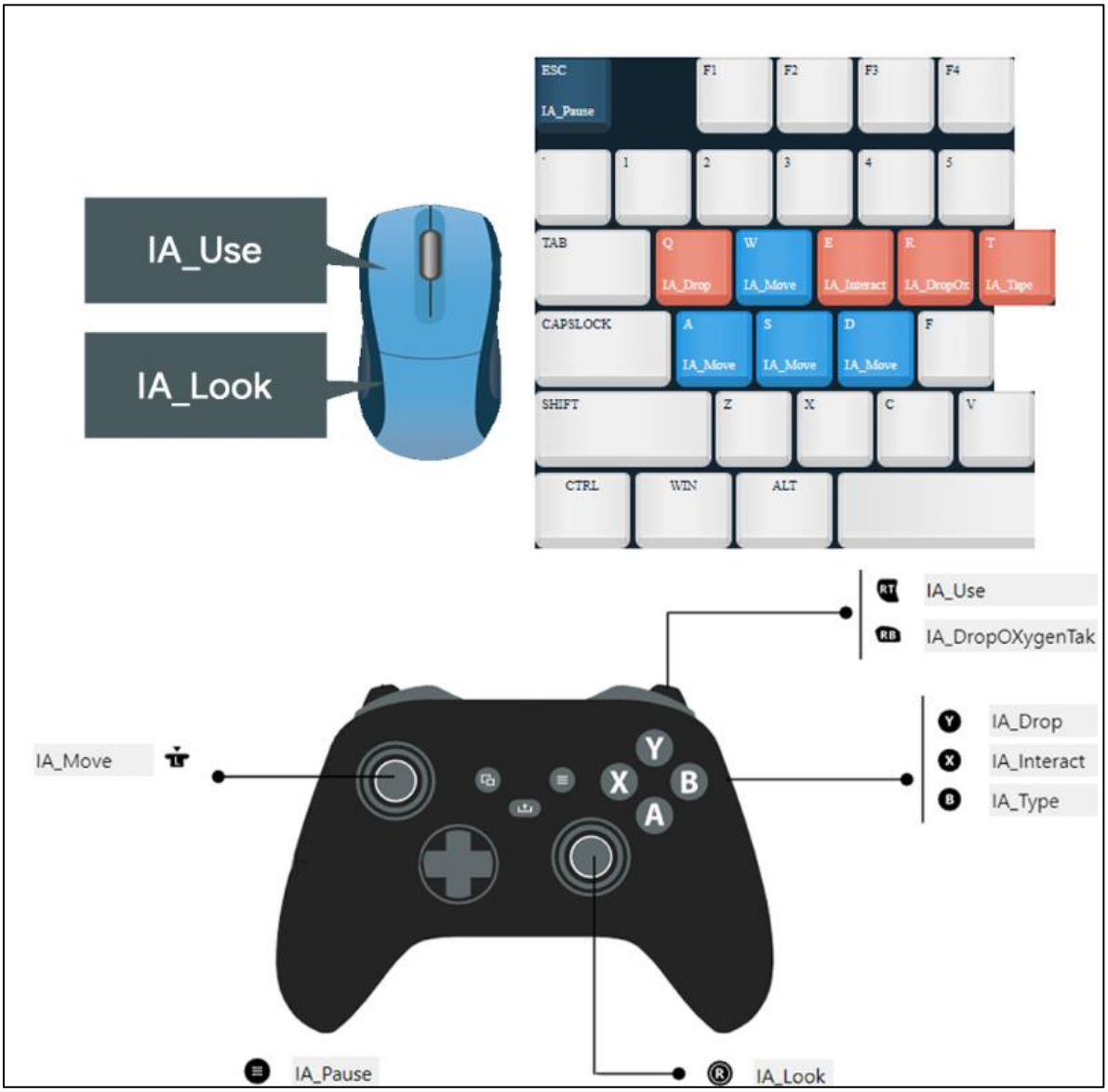


Рисунок 4.1 – Схема керування ігровим персонажем

Для того, щоб мати можливість додавати Input Mapping Contexts до персонажа, створимо функцію в основному Blueprint персонажа (див. рис. 4.2).

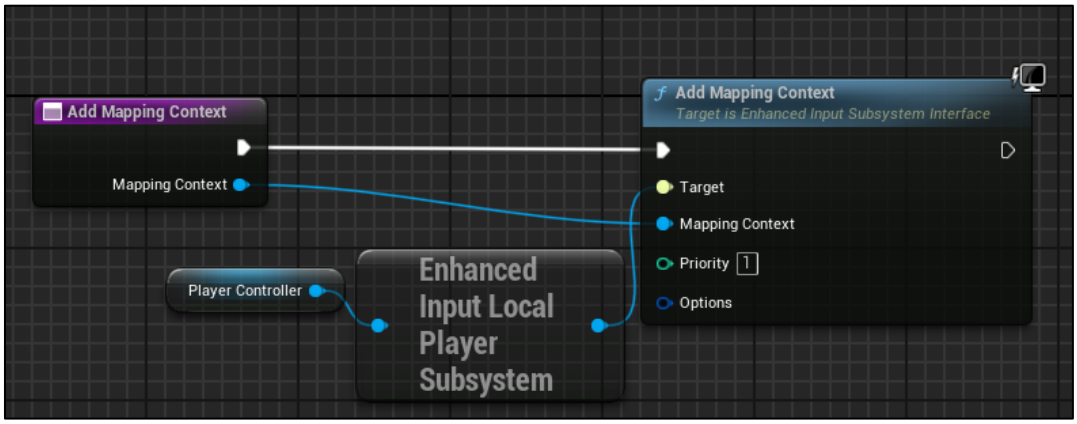


Рисунок 4.2 – Функція для додавання Input Mapping Contexts

Також, для необхідно створити функцію для видалення Input Mapping Contexts у персонажа (див. рис. 4.3).

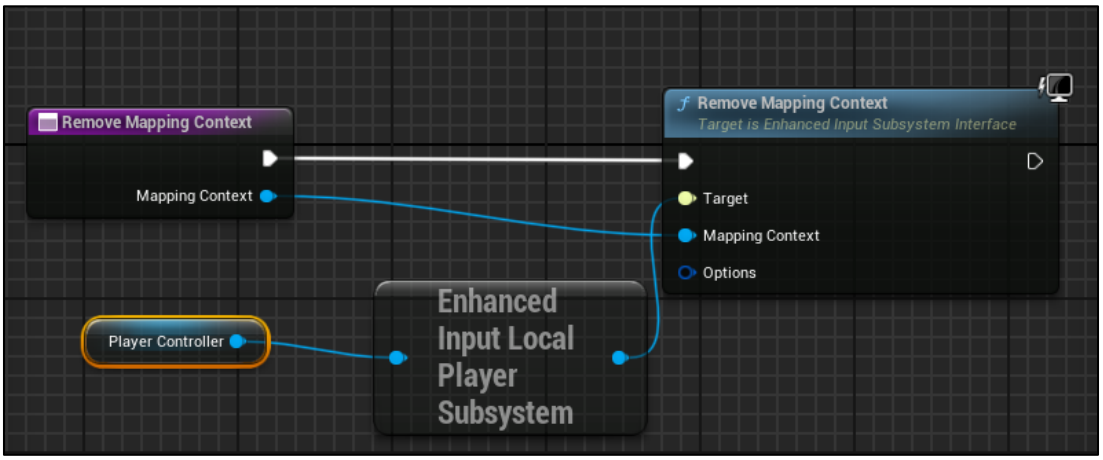


Рисунок 4.3 – Функція для видалення Input Mapping Contexts

Також необхідно створити Blueprint Component для керування методами вводу, щоб можна було перемикатися між клавіатурою та геймпадом, і програма розуміла який з контролерів гравець зараз використовує, та відображала йому підказки використовуючи правильну схему керування. Для цього був створений BPC_Input, який зберігає у собі бульову змінну bIsGamepad, яка встановлюється на true, коли була натиснута кнопка з геймпаду, та на false, коли була натиснута кнопка з клавіатури або миші. Завдяки цьому програма буде знати який з контролерів

використовується, а ми матимемо можливість будувати логіку гри спираючись інформацію, про тип контролера, що використовує гравець.

В результаті маємо можливість персонажа викликати конкретні події, на натискання кнопок на пристрої вводу.

4.1.2 Механіка переміщення

Для реалізації переміщення персонажа, необхідно написати логіку, на подію IA_Move та IA_Look. Для IA_Move будемо використовувати функцію AddMovmentInput, що вбудована в Unreal Engine 5 (див. рис. 4.4).

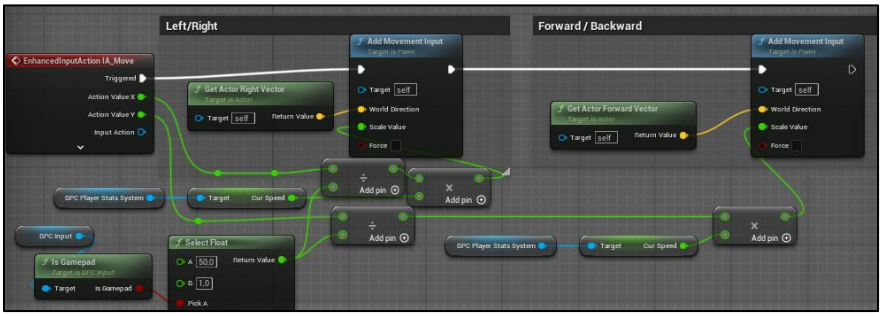


Рисунок 4.4 – Переміщення персонажа

Для IA_Look використовуватимемо вбудовану в Unreal Engine 5 функцію AddControllerInput (див. рис. 4.5).

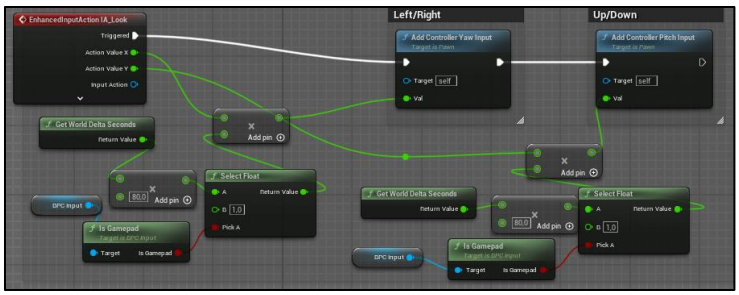


Рисунок 4.5 –Управління камерою персонажа

Оскільки дані, що надходять від геймпаду та клавіатури з мишою, мають різний формат, це ускладнює їхню обробку та реалізацію єдиної системи керування. Для подолання цієї проблеми було розроблено метод, який дозволяє

привести дані з геймпада та клавіатури з мишею до спільного стандарту, що ґрунтується на діленні множника різниці між даними з цих двох пристроїв.

В результаті маємо можливість пересуватися ігровим простором, використовуючи геймпад або клавіатуру з мишею, та перемикатися між ними під час ігрового процесу.

4.1.3 Взаємодія з об’єктами

Для взаємодії з предметами, необхідно створити Trace Chanel для них, та шукати акторів по цьому каналу. Для реалізації цього, було створено компонент BPC_GrabComponent. В якому будуть відбуватися обчислення пов’язанні з пошуком акторів по каналам. Була створена функція TraceForGrabableItem, яка повертає об’єкт для підбирання, що знаходиться в центрі екрану гравця (див. рис. 4.6).

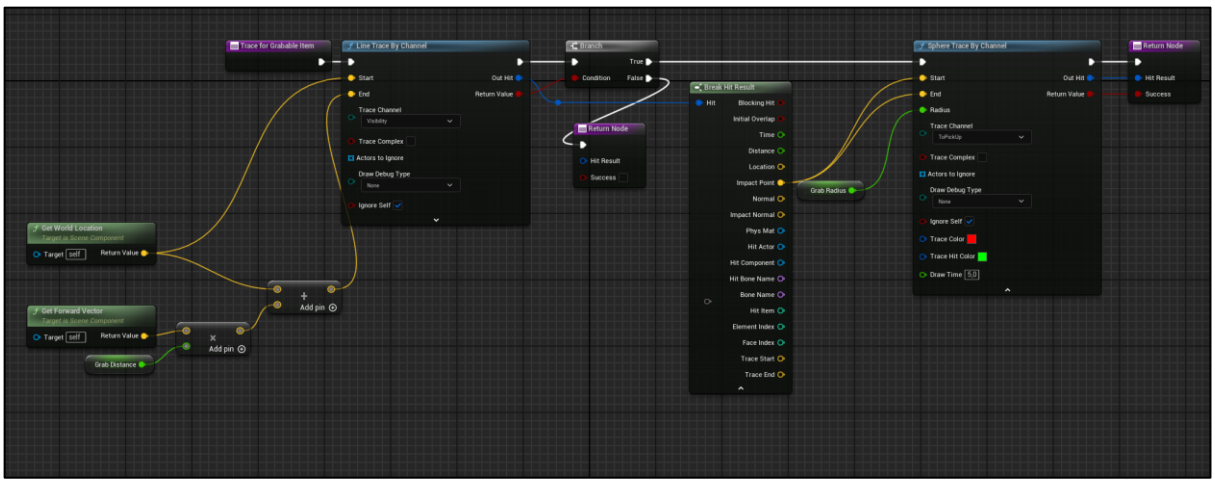


Рисунок 4.6 – Функція пошуку об’єктів для підбирання гравцем

Після отримання інформації про актора для підбирання, необхідно вставити його в руки персонажу та додаємо IMC_HandItem, щоб гравець мав можливість використовувати та викидати цей предмет.

Для викидання предмету на IA_Drop необхідно відкріпити цей предмет від рук персонажа, видалити IMC_HandItem та надати йому імпульс, щоб той полетів вперед(див. рис. 4.7).

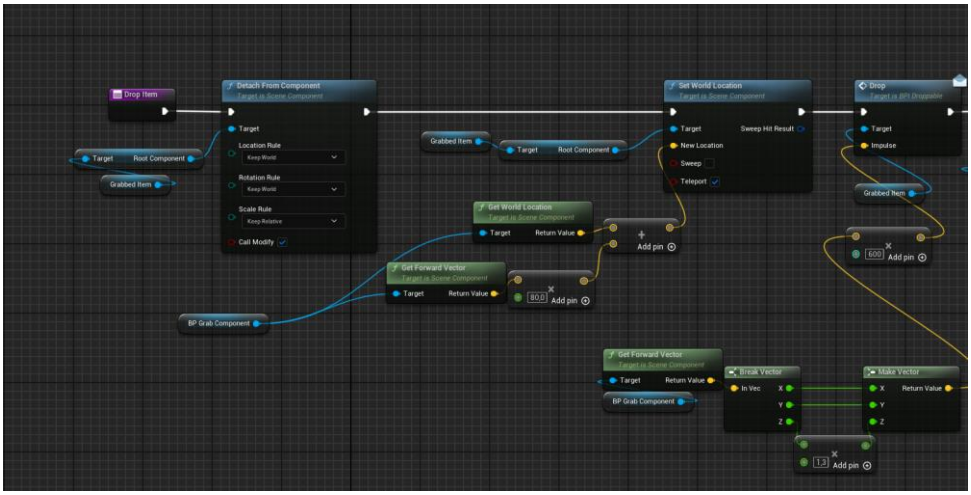


Рисунок 4.7 – Функція викидання предметів з рук гравця

Для використання предметів, що знаходяться в руках персонажа, треба викликати функцію Use у цього предмета(див. рис. 4.8).

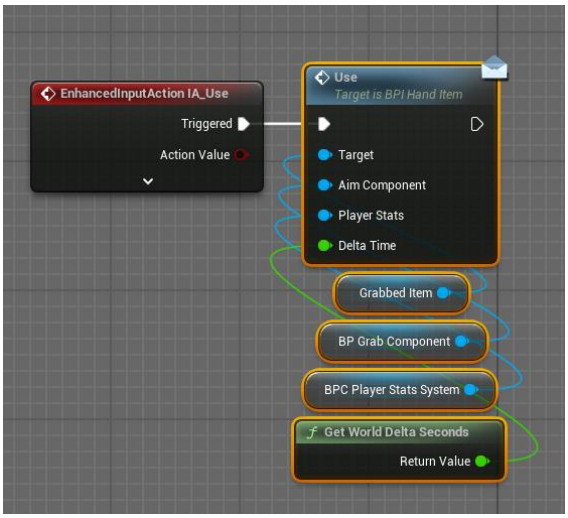


Рисунок 4.8 – Використання предмету

Для взаємодії з кнопками в ігровому просторі необхідно додати до персонажа WidgetInteraction, який дозволяє взаємодіяти з віджетами. Також на подію IA_Interact поставимо натискання та відпускання віртуальної кнопки WidgetInteraction(див. рис. 4.9).

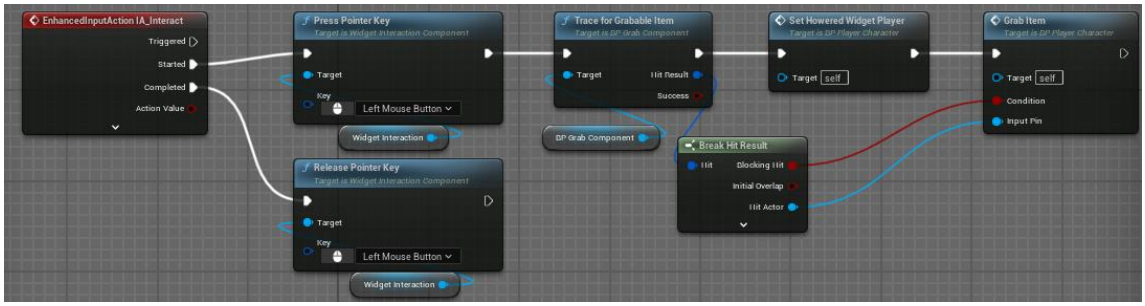


Рисунок 4.9 – Логіка взаємодії з об’єктами

Тепер, гравець має можливість підбирати, використовувати та викидати різноманітні предмети, які він знайде в ігровому просторі. Це можуть бути як корисні речі, що допоможуть йому у виконанні завдань або у боротьбі з ворогами, так і інші елементи, що впливають на ігровий процес. Гравець може підбирати ці предмети для подальшого використання. Крім того, він може вирішити викинути непотрібні речі, звільняючи руки для нових. Ця можливість надає гравцю більше свободи дій і робить ігровий процес більш захоплюючим та інтерактивним. Від того, які предмети гравець вибирає залишити, залежить його стратегія та спосіб проходження гри. Наприклад, деякі предмети можуть мати важливе значення для вирішення головоломок або відкриття нових локацій. Інші ж можуть бути потрібні для торгівлі або обміну з іншими персонажами в грі. Така система інвентарю стимулює гравців до уважного дослідження світу гри та прийняття зважених рішень щодо використання знайдених предметів.

4.1.4 Ролі та характеристики

У грі представлено чотири основні ролі персонажів, кожна з яких виконує унікальну функцію і має свої особливі навички та характеристики. Ці ролі включають: Механік, Інженер, Капітан та Офіцер. Кожен персонаж має свою спеціалізацію, яка визначає його роль у команді та підхід до виконання завдань. Детальна інформація про їхні характеристики, спеціалізації та можливості представлена в таблиці 4.1.

Таблиця 4.1 – Розподілення характеристик персонажа між ролями

	Рівень навичок Механіка	Рівень навичок Інженера	Рівень бойових навичок	Здоров’я	Кисень
Офіцер	1	1	3	200	100
Капітан	2	2	2	100	100
Механік	3	1	1	100	200
Інженер	2	3	1	100	100

Для реалізації цих характеристик, було створено BPC_PlayerStatSystem, де реалізована логіку зменшення кисню й здоров’я, та зберігається інформацію про навички гравця, його роль, та множник швидкості персонажа, який буде змінюватися в залежності від предмету, який він тримає у руках (див. рис. 4.10).

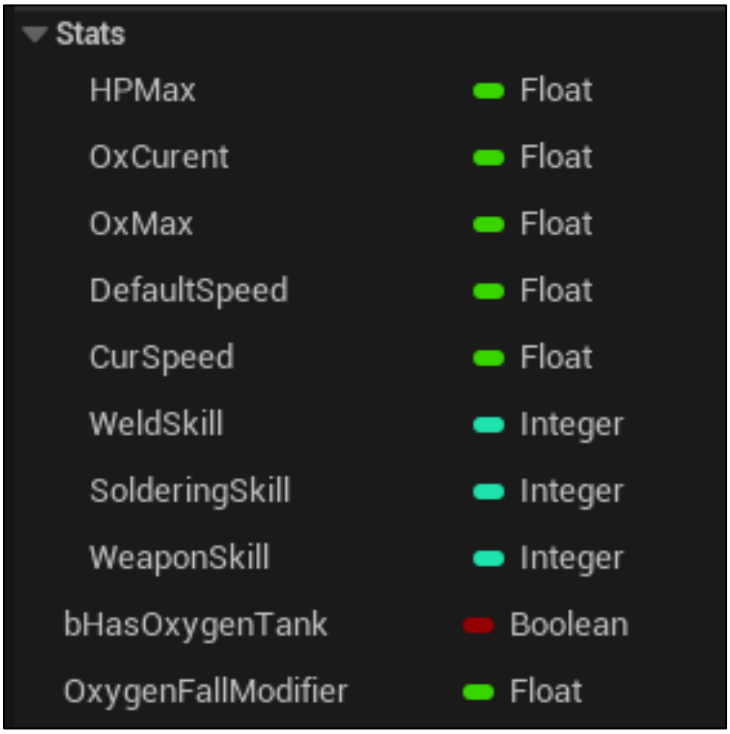


Рисунок 4.10 – Список змінних в BPC_PlayerStatSystem

Була створена DataTable, в якій зберігається розподілення характеристик, які були зазначені в тиблиці 4.1. Також був додан механізм зміни кольору світлових елементів на персонажі в залежності від його ролі (див. рис. 4.11).

Рисунок 4.11 – Різні кольори для ролей

Також була створена BPC_HelathSystem, яка буде відповідати за усі операції зі здоров'ям гравця, а саме зниження, зменшення та зберігання його.

Отже персонажі тепер мають навички, швидкість переміщення, механізм зменшення та збільшення рівню здоров'я та кисню.

4.1.5 Балон з киснем

В грі є механіка балону з киснем, що кріпиться на спину персонажа, та який забезпечує постачання кисню в момент, коли на кораблі його немає через технічні неполадки. Гравець підбирає його як звичайний предмет, але замість того, щоб закріплятися в руках персонажа, він прикріпляє його на спину. Балон також можна зняти, щоб передати його іншому гравцеві.

Логіка зменшення рівня кисню знаходиться в BPC_ PlayerStatSystem, При зниженні рівня кисню до 0, балон зникає

4.1.6 Предмети

В грі наявні три типи предметів, які можна підібрати: зброя, інструменти та аптечка. Кожен з них має як унікальні, так і загальні засоби застосування.

До загальних можна віднести можливість усіх предметів завдавати шкоду ворогу, при викидані їх з рук. Для реалізації цього, треба створити логіку на подію OnHit в загальному класі предметів, яка буде наносити шкоди ворогу, якщо той був вдарений предметом після викидання його з рук.

До унікальних засобів застосування відносяться: стрільба, лагодження, гасіння пожеж, відновлення очок здоров'я.

Для стрільби було додано наступні змінні: поточна кількість куль, максимальна кількість куль, час між вистрілами, поточний рівень перегріву, максимальний рівень перегріву, та урон, що буде наносити одна куля. Також було додано механізми стрільби та перегріву зброї(див. рис. 4.12).

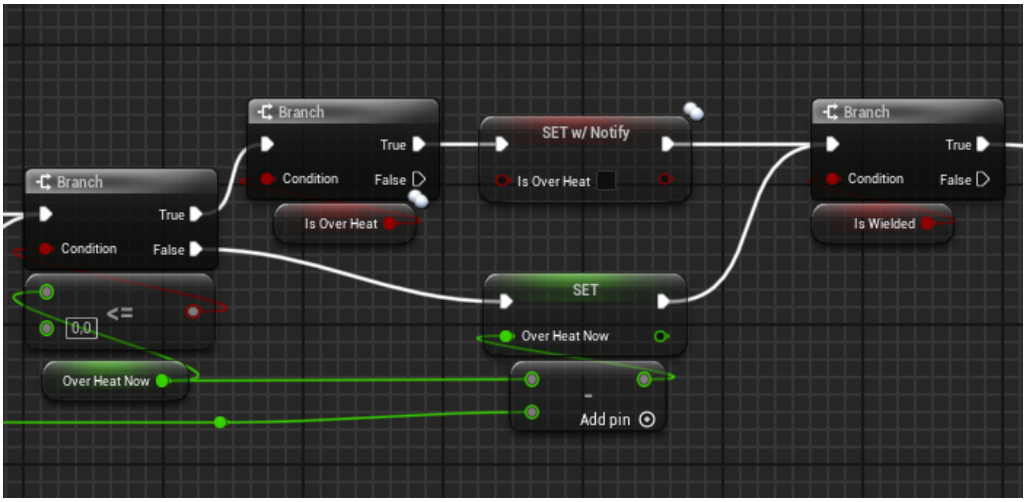


Рисунок 4.12 – Логіка перегріву зброї

Для інструментів було додано змінні, що показують максимальний та поточний запас ресурсів, а також ефективність інструменту в лагодженні певних поломок. Також в BPC_GrabComponent була додана функція, для пошуку акторів для лагодження (див. рис. 4.13).

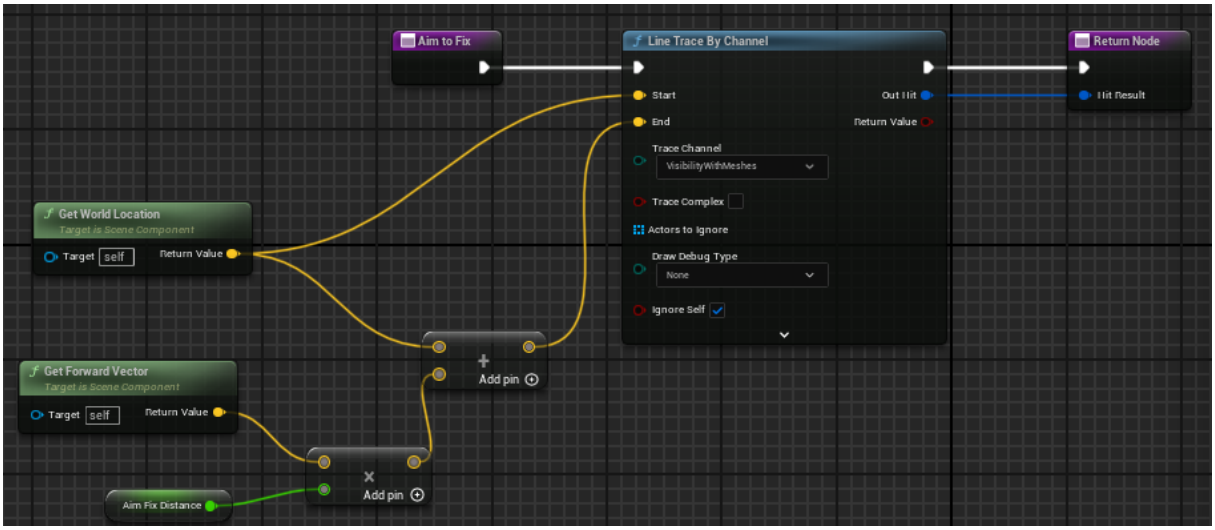


Рисунок 4.13 – Пошук предмету для лагодження

Ця функція викликається на подію Use, та передає інформацію про предмет для лагодження в функцію Fix (див. рис. 4.14), яка вже викликає в цьому предмету функцію інтерфейсу для лагодження.

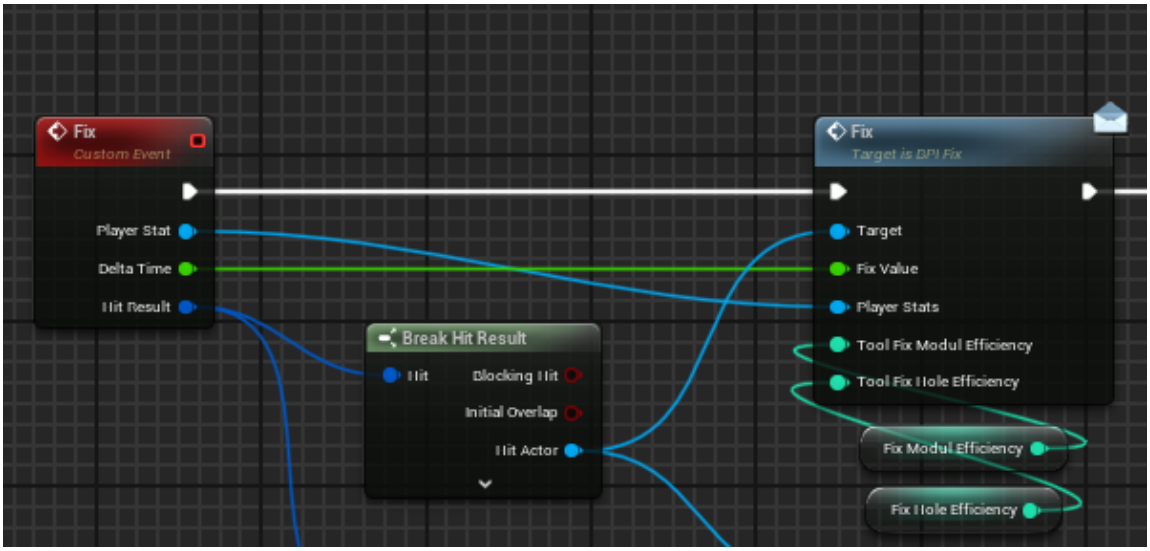


Рисунок 4.14 – Лагодження знайденого предмету

Аналогічним образом зроблена логіка для гасіння фогню, тільки замість одного об’єкту, ми шукаємо кілька. А на подію Use ми встановлюємо на знайдені об’єкти вогню змінну isExtingush на true (див. рим. 4.15). Далі, мої колеги зможуть налаштувати логіку гасіння та розгоряння вогню.

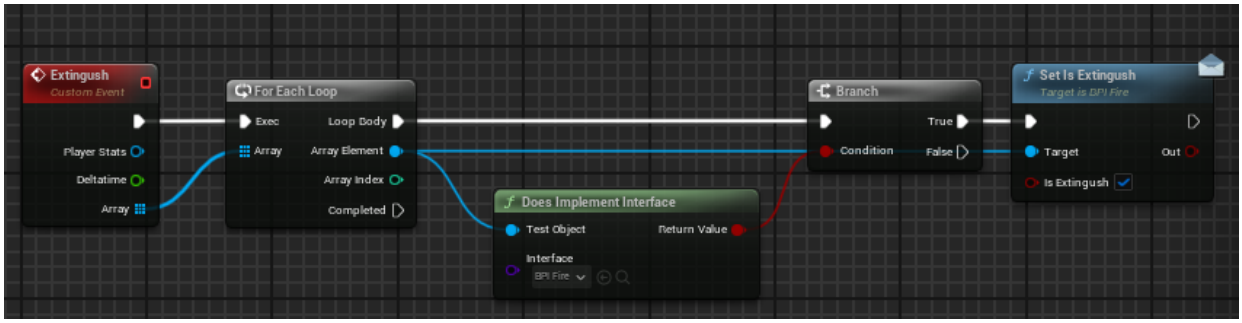


Рисунок 4.15 – Гасіння вогню

Для реалізація аптечки на подію StartUse було додано поступове зростання відсотку застосування, оскільки аптечка має застосовуватись не одразу, а через кілька секунд. А на момент, коли змінна зростає до 1, аптечка застосовується та відновлює гравцеві очки здоров'я (див. рис. 4.16).

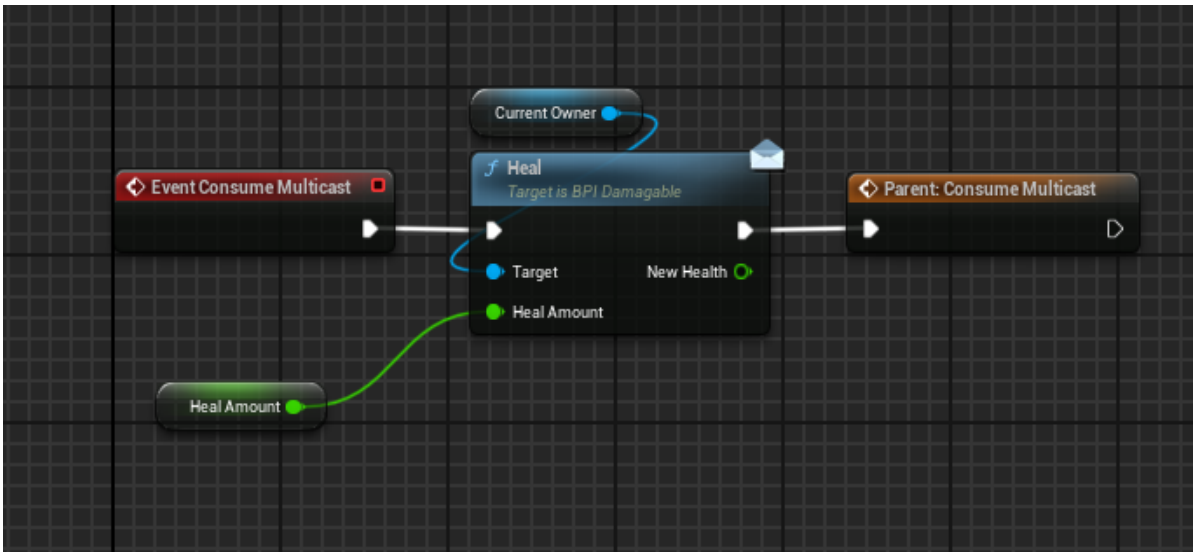


Рисунок 4.16 – Застосування аптечки

Також серед предметів є балон з киснем. Цей балон є надзвичайно важливим для забезпечення дихання у критичних ситуаціях. Він може використовуватися в умовах, де концентрація кисню в атмосфері є недостатньою для нормального дихання. Детальнішу інформацію про нього розписано в розділі 4.1.5, де це питання розглянуто більш ґрунтовно.

4.1.7 Клейка стрічка

Для реалізації клейкої стрічки, яка тимчасово лагодить частини корабля, треба розрахувати кут нахилу, під яким треба заспавнити її. Заради цього, ми з очей гравця випускаємо з трохи різних місць три промені, завдяки яким можна розрахувати рівняння площини, з якої вже можна дізнатись необхідний нахил(див. рис. 4.17).

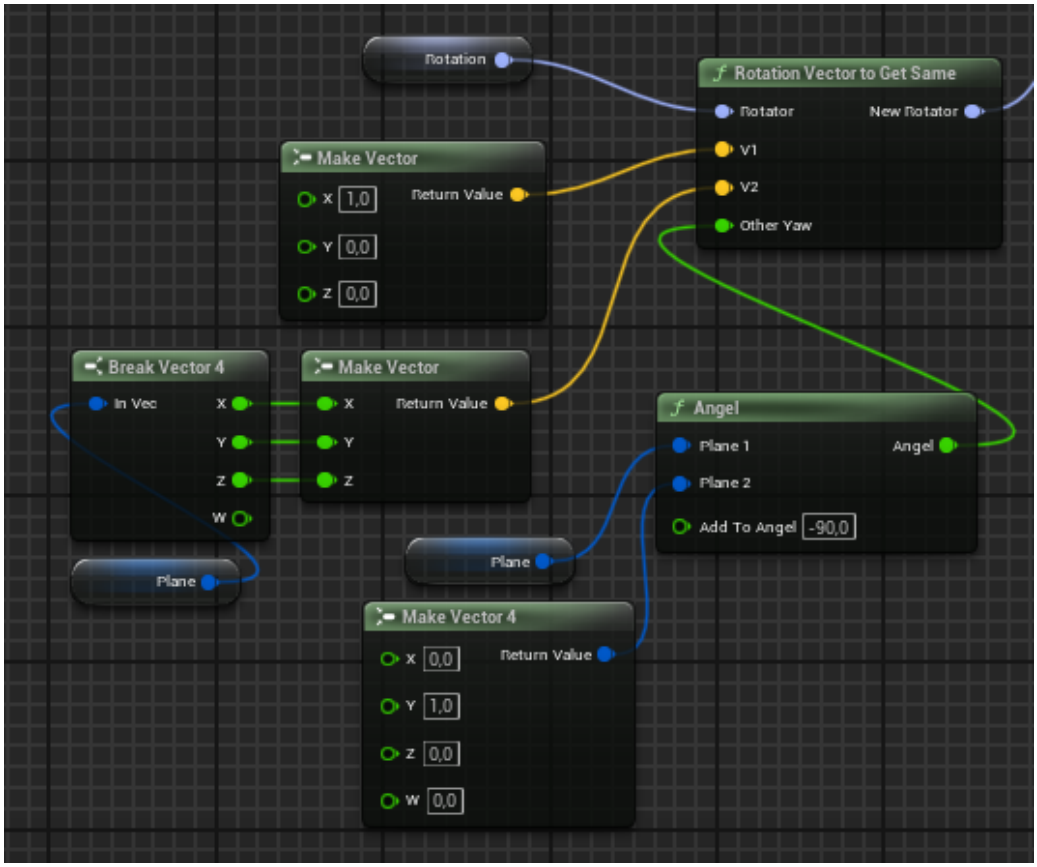


Рисунок 4.17 – Розрахунок кута нахилу клейкої стрічки

Після чого, на подію IA_Type створюємо клейку стрічку з необхідним нам нахилом, та починаємо тимчасове лагодження актора, на якому гравець поставив стрічку (див. рис. 4.18).

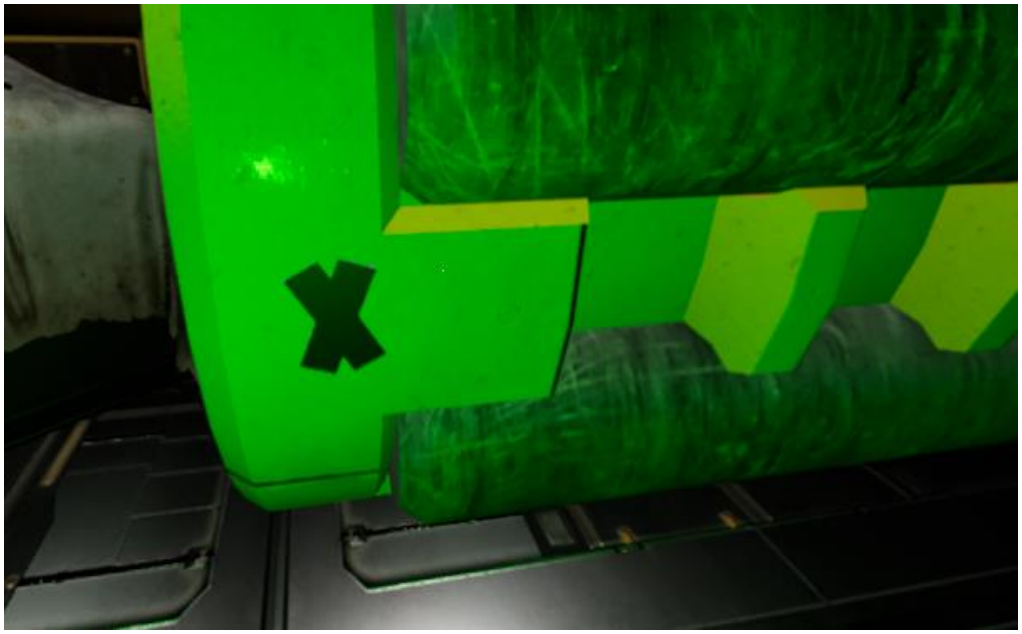


Рисунок 4.18 – Клейка стрічка на модулі

Також був створений BPI_Type, який був доданий до усього, на що можна наліпити стрічку, що дозволить колегам в подальшому налаштувати логіку лагодження клейкою стрічкою

4.1.8 Меню гри

Для реалізації меню гри створюємо віджет UI_Pause. Меню буде виконувати роль меню-паузи, але оскільки гра заточена під кооперативне проходження, то на паузу гра ставитись не буде. В створеному віджеті робимо розмітку меню (див. рис. 4.19). Важливо, щоб меню було інтуїтивно зрозумілим для гравців і мало всі необхідні функції, такі як доступ до налаштувань, вихід з гри та можливість повернутись до основного ігрового процесу. Також необхідно передбачити можливість виклику допомоги або перегляду інструкцій з гри. Додатково, дизайн меню має відповідати загальній стилістиці гри, щоб забезпечити візуальну єдність і комфорт для користувачів. Інтерактивні елементи меню повинні бути зручними для взаємодії, незалежно від того, чи грають гравці на клавіатурі, чи на миші.

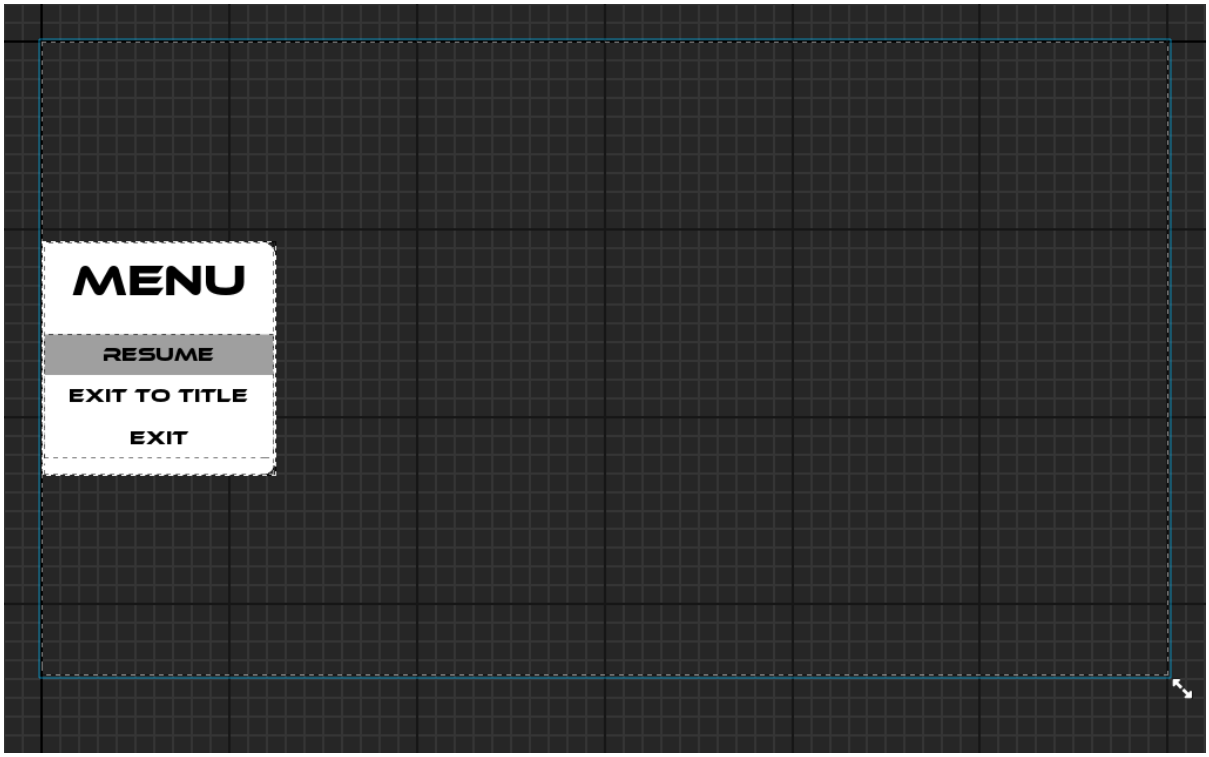


Рисунок 4.19 – Меню гри

В цьому меню є три кнопки: resume, exit to title та exit. Resume – повертає гравця до гри, закриваючи меню, exit title повертає гравця до головного меню, а exit закриває гру.

Для виклику меню, треба на подію IA_Type створити його, додати до гравця та перемкнути режим на віджет (див. рис. 4.20).

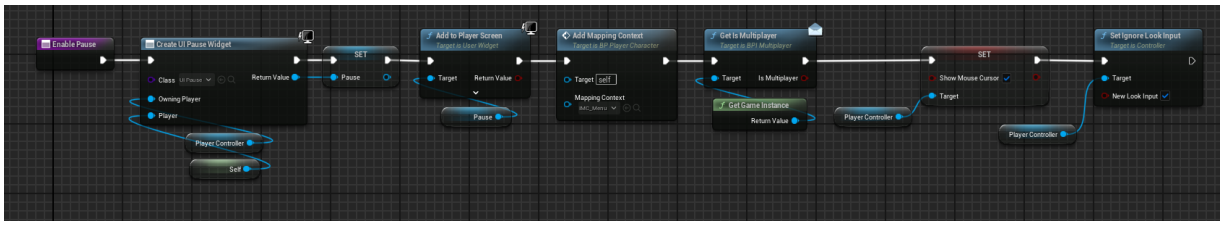


Рисунок 4.20 – Створення меню

В результаті з’являється меню, в якому можна обрати один з трьох елементів меню, а саме: закрити меню, повернутися на головний екран, та вимкнути гру.

4.1.9 Смерть

Для виклику смерті, треба забіндити подію onDeath BPC_HealthSystem в класі персонажа (див. рис. 4.21).

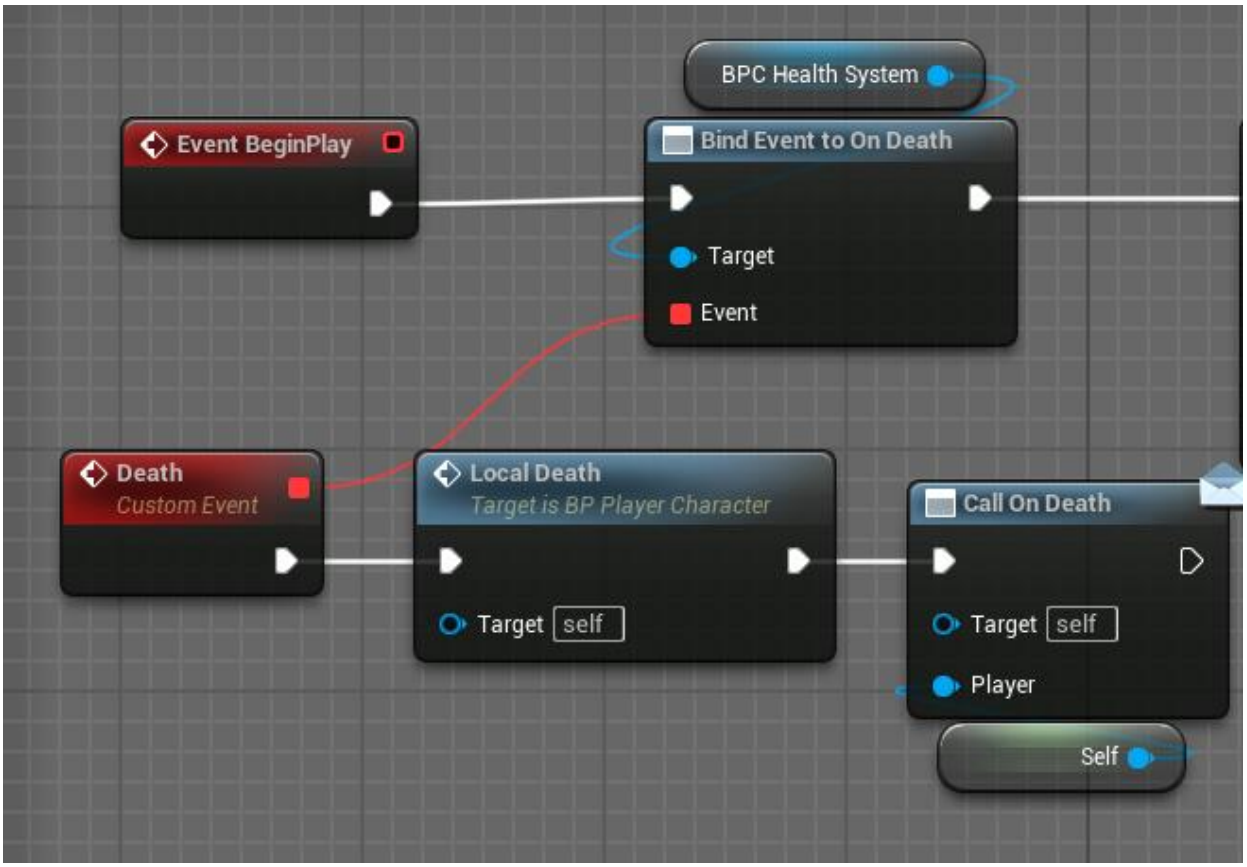


Рисунок 4.21 – Подія на смерть персонажа

Також на цю подію доданий виклик onDeath персонажа, що дозволить в подальшому творити нішу логіку, яка відбувається на смерть. Коли здоров’я персонажа дійде до 0, ця подія буде викликана. При смерті, персонаж має викинути предмет, який тримає, викинути балон з киснем, який він має, сховати HUD персонажа, та стати невидимим та невразливим (див. рис. 4.22).

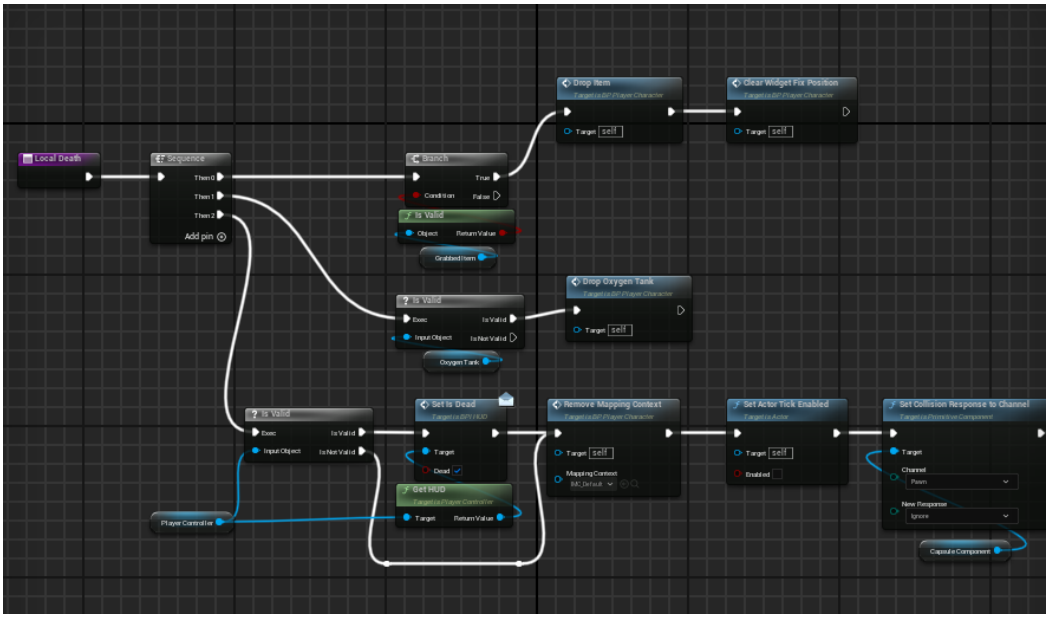


Рисунок 4.22 – Основна логіка смерті

Після смерті, треба вивести на екран напис, що повідомляє про це, та дозволити гравцеві спостерігати за подіями від обличчя свої ще живих товаришів. Для цього створюємо UI_Death (див. рис. 4.23)

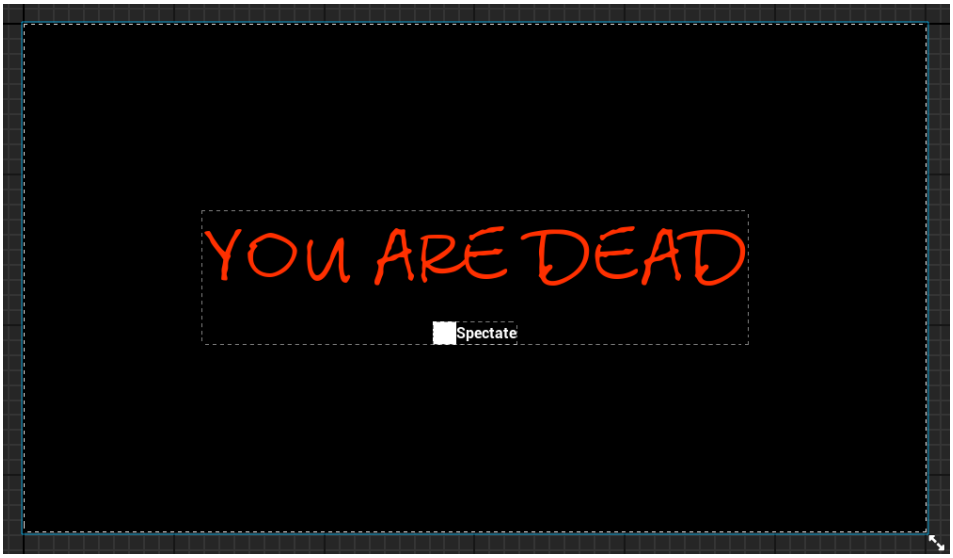


Рисунок 4.22 – Екран смерті

При натисканні на кнопку Spectate, на екрані буде показуватися вигляд з очей живого товариша, та його роль (див. рис. 4.23).

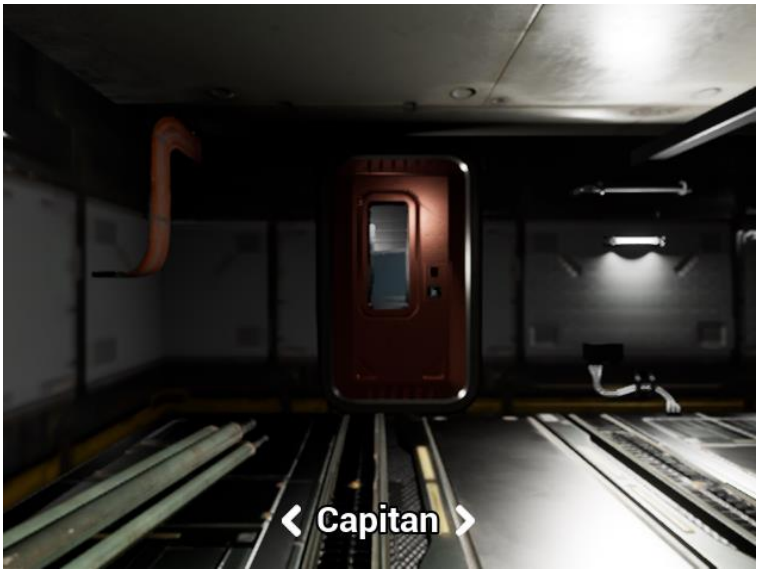


Рисунок 4.23 – Режим спостерігача

Також при смерті іншого гравця, треба перестати транслювати його зображення. Для цього було додавання темного екрану з поміхами, та червоним написом NO SIGNAL. Це доволі цікаве дизайнерське рішення, яке добре працює на створення атмосфети (див. рис. 24).

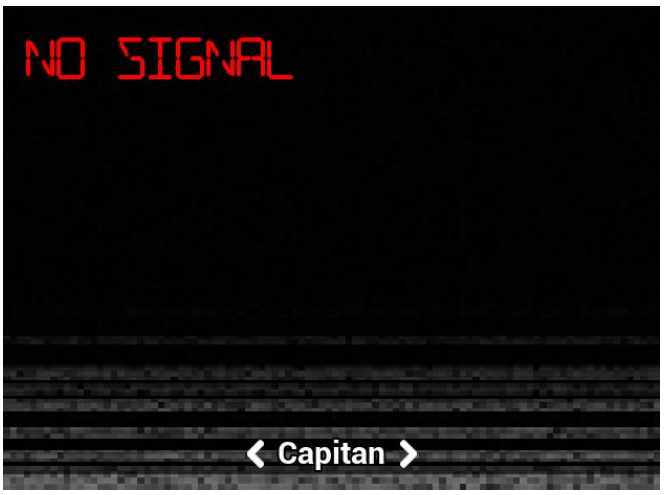


Рисунок 4.24 – Режим спостерігача, коли спостерігаєш за мертвим гравцем

Також, оскільки в нас є смерть, то має бути і оживлення. Для цього просто робимо обернений від смерті процес (див. рис. 4.25).

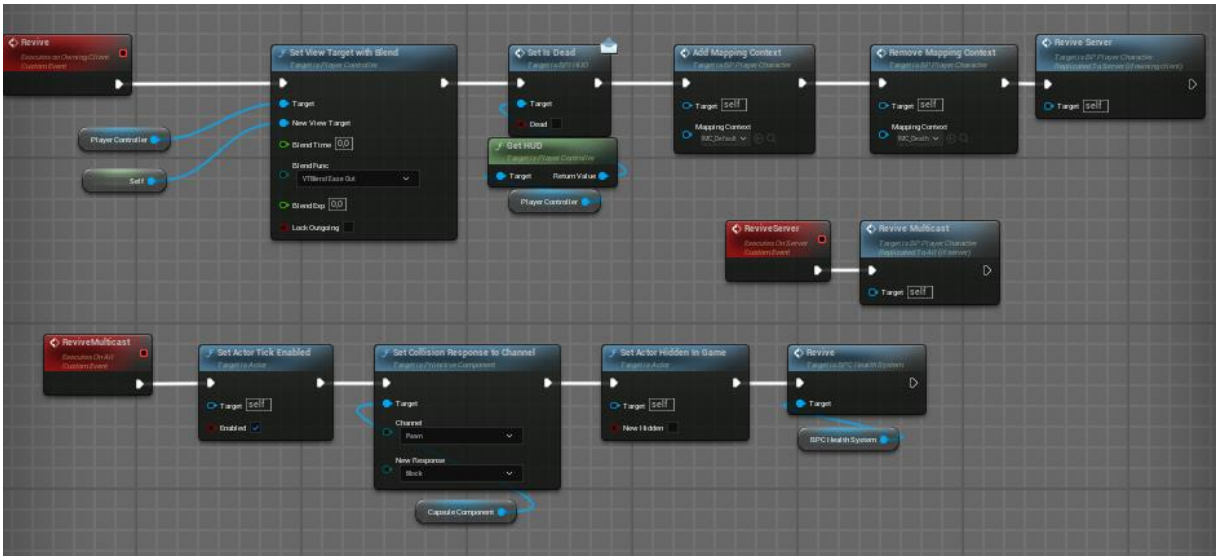


Рисунок 4.25 – Оживлення персонажа

Як бачимо, ми тут знищуємо екран смерті та повертаємо HUD гравця, повертаємо усі колізії та видимість гравця.

4.2 Розробка HUD

Для реалізації HUD був створений віджет UI_BaseHud, в який додаємо статус-бар здоров’я та кисню гравця, а також інформацію про інших гравців та стан подорожі (див. рис. 4.26).

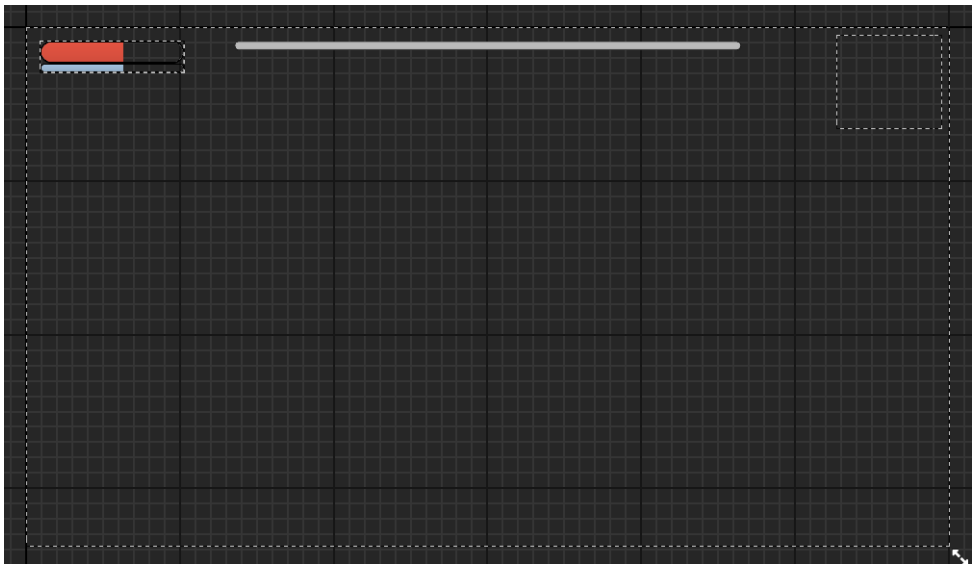


Рисунок 4.26 – HUD гравця

Також були створені додаткові віджети для відображення кількості патронів в зброї, кількості зарядів у інструментів та прогрес застосування аптечки. Ці віджети будуть додаватися на екран гравця, коли він буде підіймати відповідні предмети.

Був створений BP_IcarusHUD. Це спеціальний Blueprint клас в грі, який буде відповідати за усю логіку, що пов’язана з HUD. В ньому вписуємо функцію, яка створює віджет HUD та додає його на екран гравця (див. рис. 4.27).

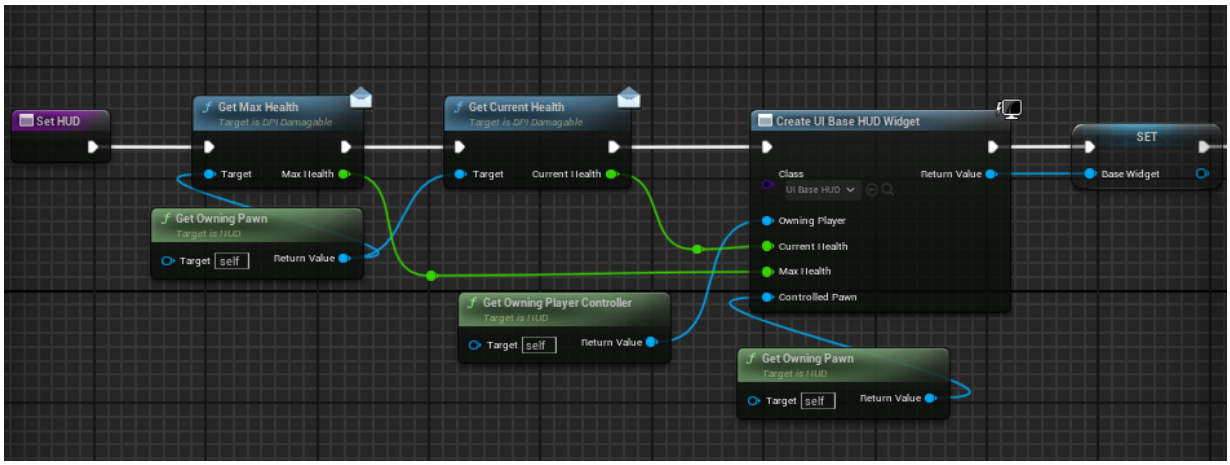


Рисунок 4.27 – Створення HUD

Було створено два додаткових віджети, які будуть відображатися не на екрані гравця, а безпосередньо в ігровому світі, додаючи більше інтерактивності та занурення в гру. Перший з цих віджетів буде з'являтися на предметах, які гравець може підібрати. Цей віджет має виглядати як індикатор, що повідомляє гравця про можливість взаємодії з об'єктом. Індикатор з’являтиметься тоді, коли гравець наводить курсор або спрямовує погляд на відповідний предмет (див. рис. 4.28). Це дозволить гравцю легко ідентифікувати інтерактивні об'єкти, полегшуючи процес їхнього знаходження та підбирання. Віджет також буде інформативним, надаючи гравцю додаткову інформацію про предмет, що дозволить прийняти обґрунтоване рішення щодо взаємодії з ним. Завдяки цьому гравець зможе швидко орієнтуватися в ігровому просторі та ефективніше використовувати свої ресурси.

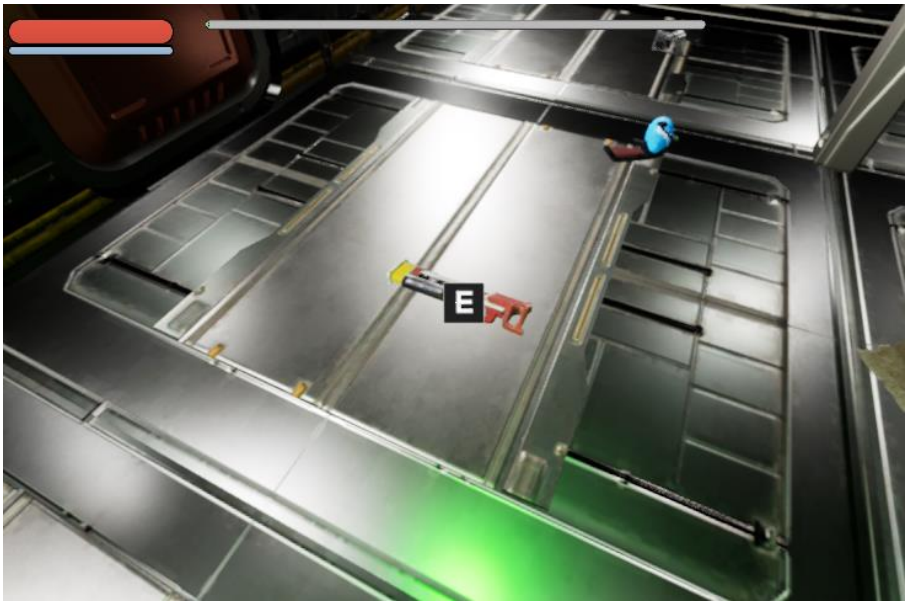


Рисунок 4.28 – Віджет на предмету

Статус лагодження буде відображатись при використанні інструментів для лагодження (див. рис. 4.29).

Рисунок 4.29 – Віджет лагодження

Також до HUD можна віднести екран смерті та режим спостерігача, про який більш детально було написано в розділі 4.1.9.

4.3 Розробка багатокористувацького режиму

4.3.1 Розділений екран

Перед початком реалізації розділеного екрану, треба налаштувати проект. Для цього в налаштуваннях, в меню Maps&Modes в підменю Local Multiplayer виставити необхідні значення (див. рис. 4.30).

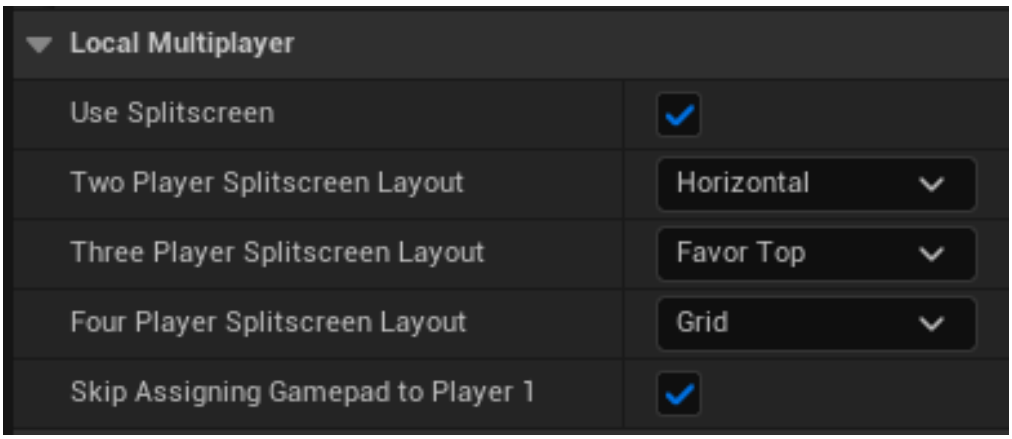


Рисунок 4.30 – Налаштування розділеного екрану

Інформацію про те, який гравець за який клас буде грати, буде зберігатися в GameInstance. Він дозволяє передавати дані між рівнями, що як раз нам необхідно для передачі інформації про класи гравців від головного меню до гри (див. рис. 4.31).

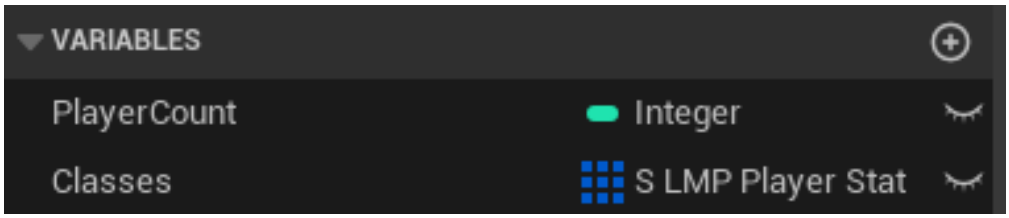


Рисунок 4.31 – Змінні в GameInstance

В GameMode ми отримуємо ці дані, та починаємо створювати гравців. Створюємо локального гравця, створюємо актора персонажа, та пов’язуємо їх (див. рис. 32).

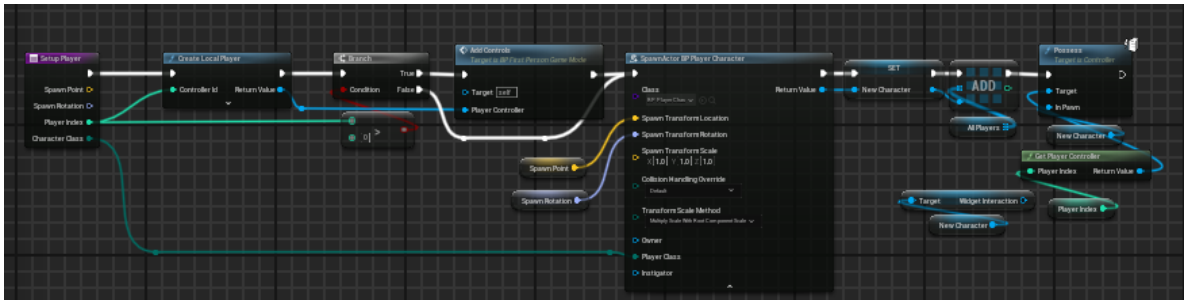


Рисунок 4.32 – Створення персонажа

Далі необхідно застосувати клас до персонажа та налаштувати його до багатокористувацької гри. Для цього з DataTable, в якій зберігається розподілення характеристик отримуємо характеристики та встановлюємо їх на гравця. Зокрема, змінюємо колір костюма завдяки системі динамічних матеріалів (див. рис. 4.33).

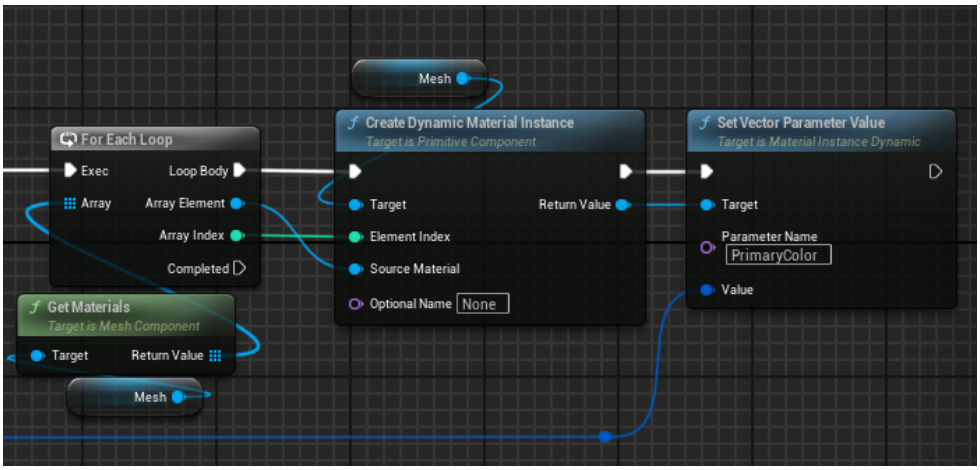


Рисунок 4.33 – Зміна кольору костюма

Для успішної реалізації гри необхідно забезпечити унікальність ідентифікаторів для кожного гравця, що дозволить їм взаємодіяти з віджетами в ігровому світі. Цей процес включає призначення унікального ID для кожного гравця в системі WidgetInteract. Без цього важливого кроку, гравці не зможуть здійснювати взаємодії з віджетами, що значно обмежить їх можливості в грі.

Крім того, для кожного нового гравця має автоматично створюватись балон з киснем. Це критично важливо для забезпечення життєдіяльності гравців у певних зонах ігрового світу, де без такого балону їхній персонаж може втратити життя

через відсутність кисню. Така механіка додає додатковий рівень складності та реалістичності в гру, сприяючи глибшому зануренню гравців у ігровий процес (див. рис. 4.34).

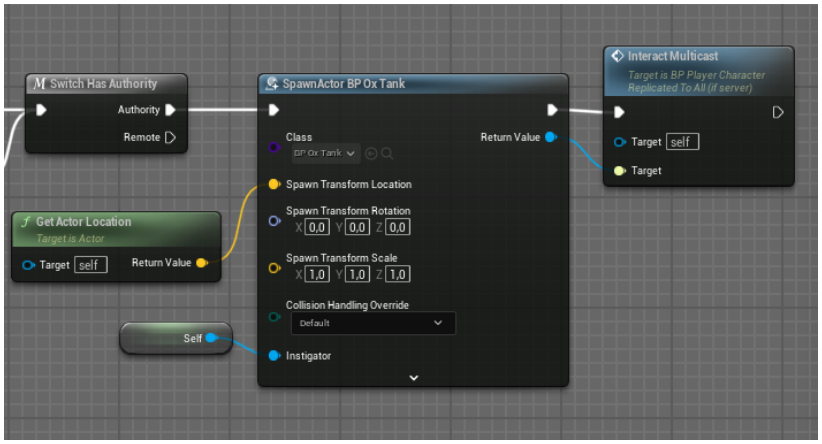


Рисунок 4.33 – Створення балону з киснем

Оскільки розмір ігрового простору при використанні розділеного екрану зменшується, необхідно також зменшити розмір HUD, але треба зменшити лише в тих випадках, коли розмір активного простору гравця зменшується по горизонталі. Тобто лише для 2 та 3 гравця, при 3-х користувацькій грі та для усіх в 4-х користувацькій грі. Для цього зменшимо scale елементів віджетів залежно від типу розділення екрану (див. рис. 4.34).

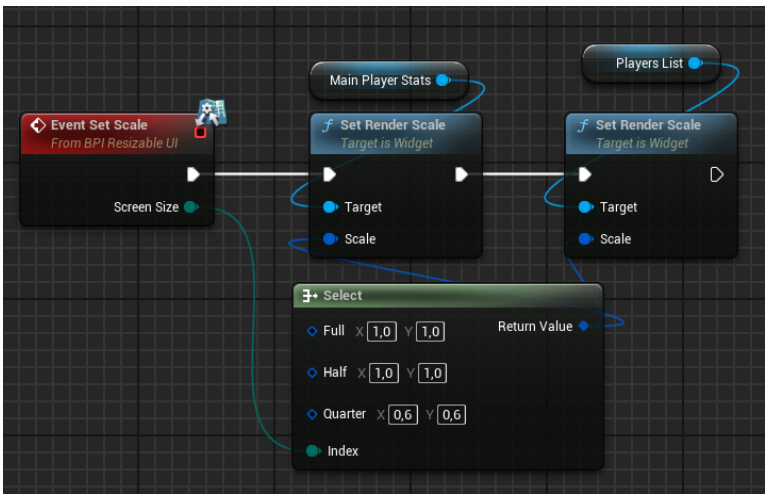


Рисунок 4.34 – Змінення розміру HUD

В результаті маємо можливість грати з чотирма контролерами на одному пристрої з розділеним екраном (див. рис. 4.35).

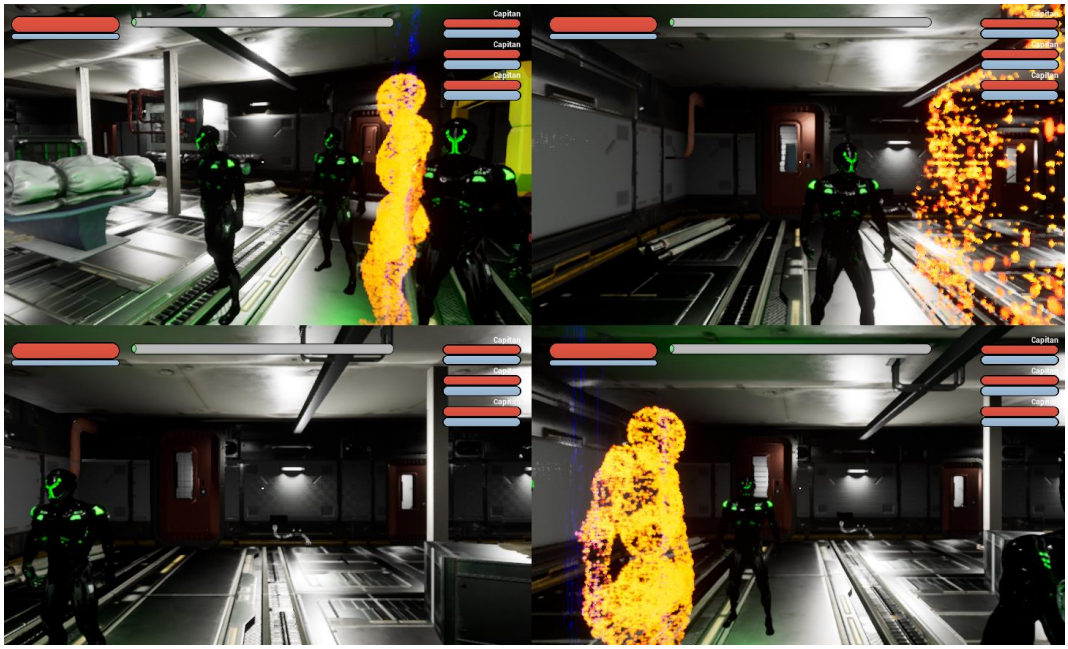


Рисунок 4.35 – Режим розділеного екрану

Для гри в такому режимі, необхідно під’єднати 4 геймпади. Та в головному меню кожен гравець має обрати собі клас персонажа.

4.3.2 Онлайн мультиплеєр

Для реалізації онлайн мультиплеєру необхідно реплікувати всі дані, зокрема положення персонажів, предметів, застосування предметів, поломки та модулі. Це дозволяє забезпечити синхронізацію ігрового процесу між усіма учасниками, щоб вони бачили однаковий стан гри[7]. Важливо, щоб реплікація даних була точною і ефективною, щоб уникнути затримок і десинхронізації, які можуть негативно вплинути на гравців. Для цього необхідно використовувати надійні алгоритми реплікації, які мінімізують обсяг переданих даних і оптимізують мережеве навантаження.

Для точної реплікації положення налаштуємо реплікацію як на рисунку 4.36. Це дозволить системі підтримувати актуальність і точність позицій персонажів та

об'єктів у реальному часі. Окрім цього, слід передбачити механізми для корекції можливих помилок у реплікації, щоб підтримувати цілісність ігрового світу. Регулярне тестування та оптимізація реплікаційних процесів також є важливими для забезпечення стабільної роботи мультиплеєра під час високих навантажень.

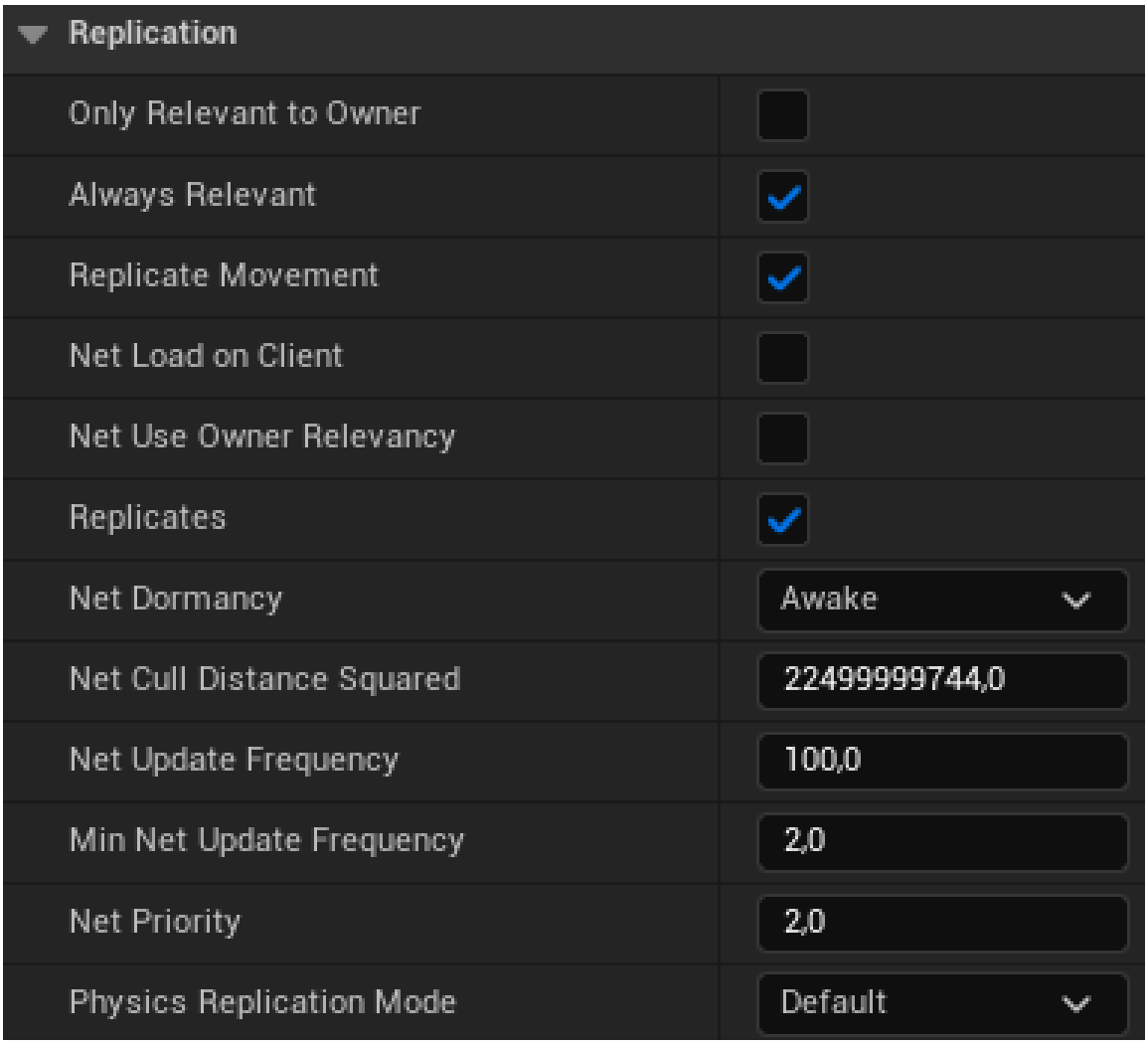


Рисунок 4.36 – Налаштування реплікації акторів

Для оптимізації серверних рішень в Unreal Engine 5 (UE5) необхідно врахувати принцип: чим менше даних потребує реплікації, тим ефективніше працює система. Це означає, що ефективність роботи сервера безпосередньо залежить від кількості даних, які передаються між клієнтами та сервером. Відтак, оптимізація орієнтована на виключення непотрібних для реплікації даних і скорочення обсягу реплікованих даних. Важливо пам'ятати, що кожен байт, який

передається, впливає на продуктивність, тому цю оптимізацію слід розглядати як ключовий фактор для успішної розробки багатокористувацьких ігор в UE5[8].

Розглянемо, як можна зменшити обсяг реплікованих даних на прикладі обертання об’єктів у просторі. В UE5 передбачені два варіанти точності реплікації обертання: Byte (8 біт) і Short (16 біт). У рамках розроблюваного проекту існують чотири актори, які повинні обертатися і синхронізувати своє положення між клієнтами. Результати тестування цих акторів в обох режимах наведені у таблиці 4.2.

Таблиця 4.2 – Тестування різних режимів передачі полодження в просторі

Об'єкти	Short		Byte	
	Avg Bytes	Avg ms	Avg Bytes	Avg ms
1	14,3	0,009	11,2	0,007
2	12,1	0,009	8,7	0,007
3	11,9	0,009	8,7	0,007
4	1,5	0,009	1,5	0,007

Проведене дослідження свідчить, що синхронізація положення з використанням Short вимагає на 25-30% більше часу порівняно з Byte. Враховуючи цю характеристику, Byte є більш економним рішенням у більшості випадків. Проте важливо зазначити, що точність Byte вдвічі нижча порівняно з Short. У певних ситуаціях це може вплинути на візуальну плавність анімації обертання.

В нашому випадку було обраний Byte для системи охолодження, камер та реактра, а Short для двигуна, бо він обертається достатньо повільно, щоб можна було візуально побачити нерівність анімації.

У рамках розроблюваного проекту зброя має механізм перегрівання, синхронізація якого між клієнтами є необхідною. Зважаючи на динамічний характер цієї характеристики, яка змінюється кожен ігровий такт, просте реплікування змінних стає нерентабельним (див. рис. 4.37).

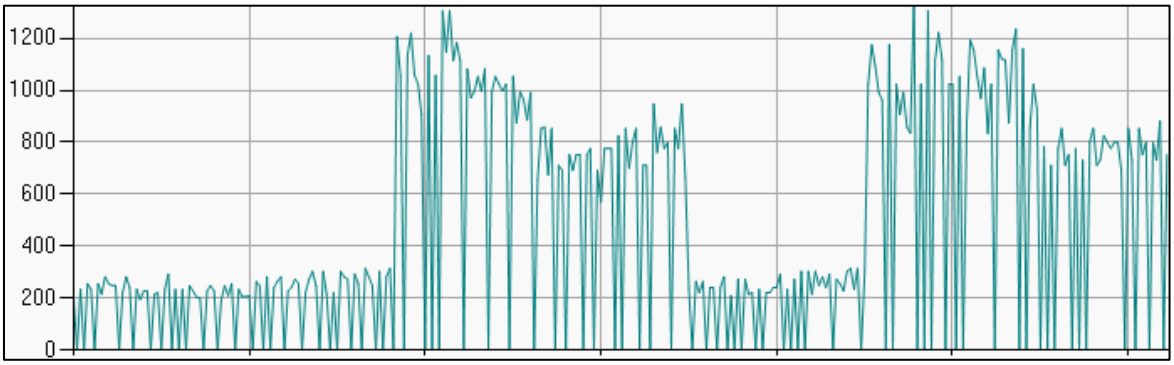


Рисунок 4.37 – Затрати, які спричиняє використання зброї до оптимізації

Як видно на рисунку 4.37, при застосуванні зброї сервер передає значення її нагріву кожен такт, що створює додаткове навантаження в 250 байтів на секунду. Цього можна уникнути, якщо передавати інформацію про нагрівання лише в моменти початку або завершення використання зброї. Оскільки дані про перегрівання потрібні лише під час її використання та тільки клієнту, що її використовує, розрахунки перегріву можна перенести з сервера на клієнт, що тримає зброю. Коли зброю ніхто не використовує, розрахунки виконуватиме сервер. Передавати інформацію від клієнта до сервера слід при зупинці використання зброї, а від сервера до клієнта – при її початку. Це зменшить навантаження на сервер і мережу, як показано на рисунку 4.38.

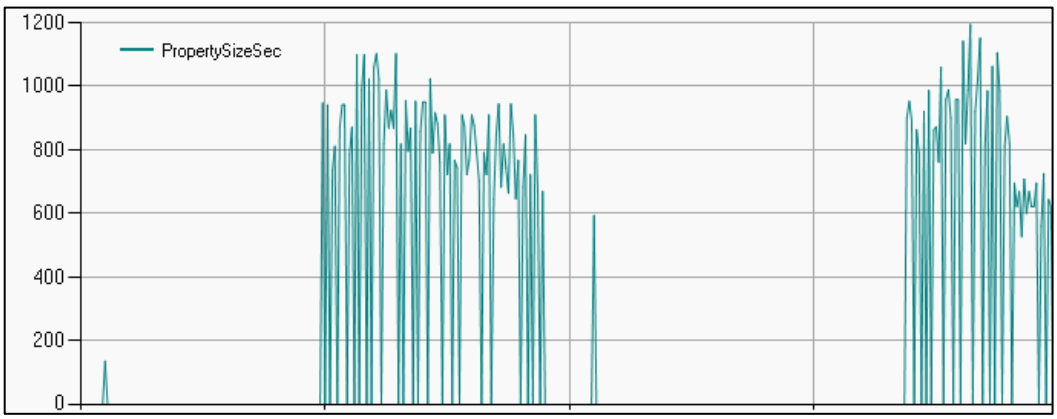


Рисунок 4.38 – Затрати, які спричиняє використання зброї після оптимізації

Як видно, при порівнянні графіків до та після оптимізації, в результаті ми отримали зменшення навантаження на мережу, що спричиняє використання зброї, на 20%.

У якості онлайн сервісу було обрано Epic Online Services[1], який є популярною платформою для розробки багатокористувацьких ігор. Він дозволяє гравцям від'єднуватись один до одного, використовуючи P2P протокол, що значно знижує навантаження на сервери. Це дозволяє розробити онлайн гру, без виділених серверів[9], адже всі комунікації між гравцями відбуваються напряму, без необхідності централізованого сервера. Такий підхід може значно знизити витрати на розробку та обслуговування гри, особливо для невеликих студій.

Створення, пошук та під'єднання до серверу були реалізовані за допомогою плагіну Advance Session[10], що дозволяє використовувати C++ команди для мережевої взаємодії в Blueprint, які розробники рушія не дозволили (див. рис. 4.39). Це значно спростило розробку мережевої функціональності гри, дозволивши використовувати потужні інструменти C++ для виконання складних мережевих завдань. Advance Session забезпечує гнучкість і надійність у реалізації мережевої логіки, що дозволяє створювати високоякісні багатокористувацькі ігри. Використання цього плагіну дає розробникам можливість розширити можливості рушія та створити більш складні та інтерактивні мережеві ігрові досвіди.

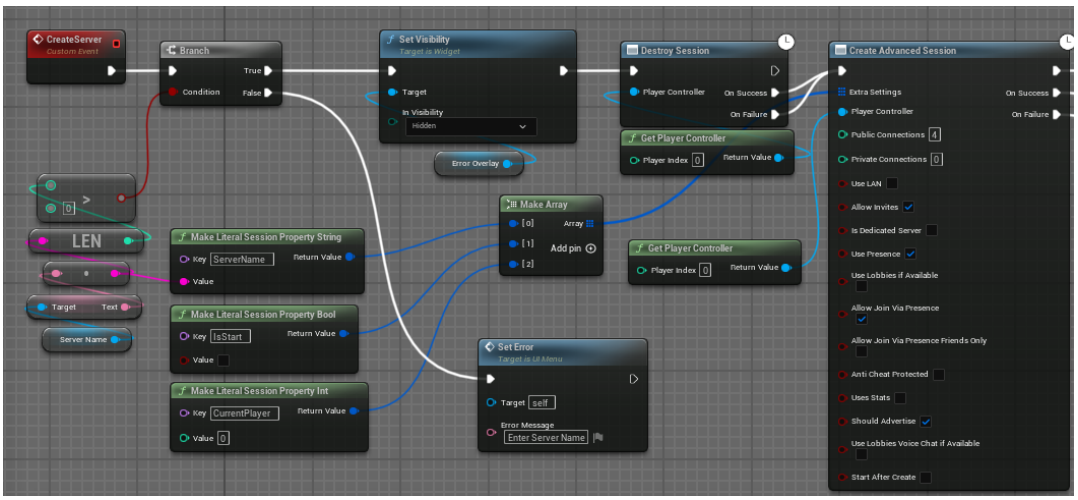


Рисунок 4.38 – Приклад використання Advance Session

Використовуючи плагін Advance Session була реалізована логіка створення, пошуку та під’єднання до сесій. При цьому один гравець, той що створив сесію, буде виступати у ролі сервера, а інші у ролі клієнта.

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розробка мапи думок тестування

Тестування ігрового програмного застосуноку відбувалось відповідно до попередньо розробленого тест-плану, що наведено у додатках. Для формування тест-плану було впроваджено Mind Map, або «діаграму думок» (див. рис. 5.1).

Тестування було проведено за методом «білої скрині», тобто перевірки гри, занурюючись у код, щоб зрозуміти яка частина коду працює некоректно.

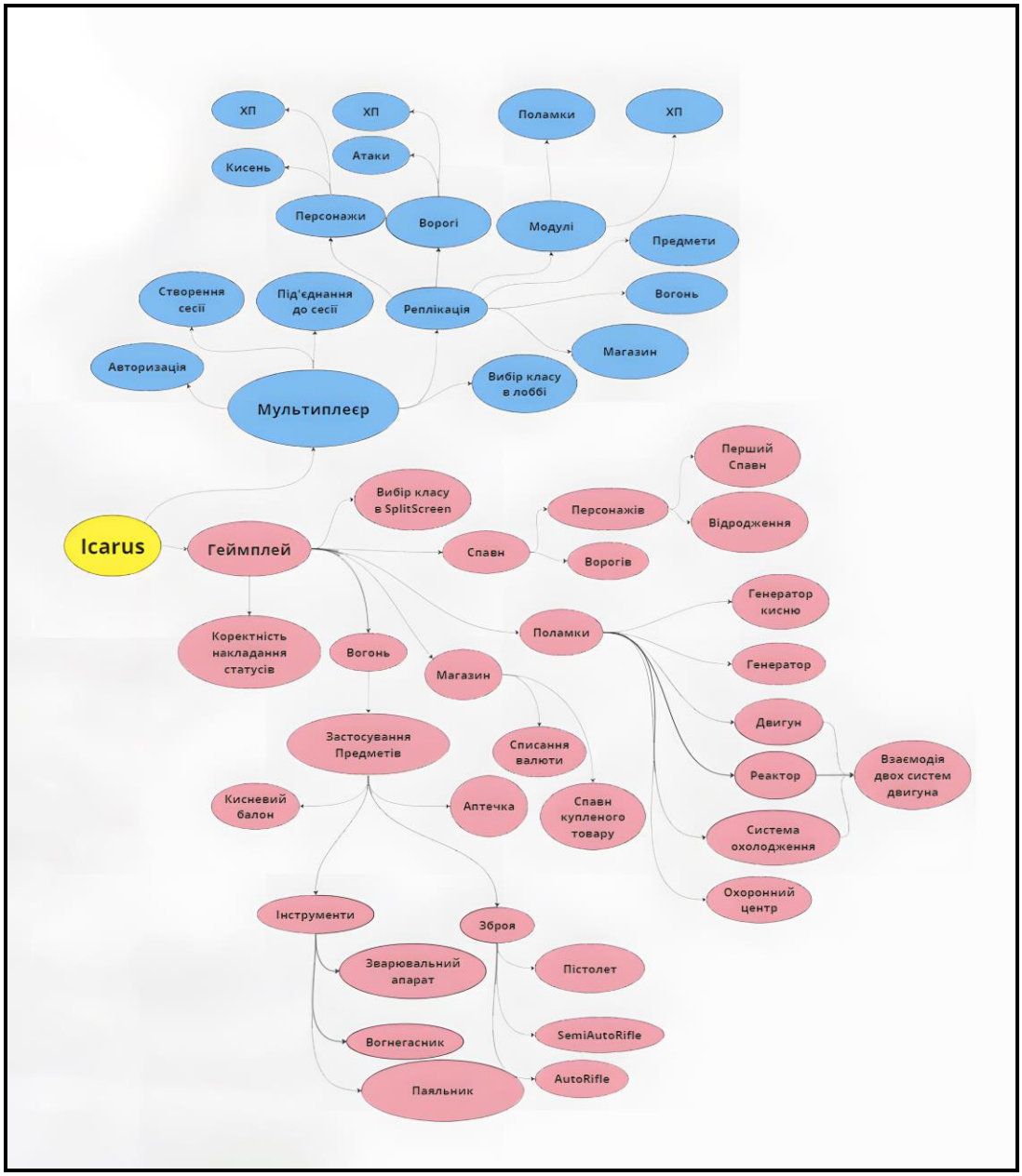


Рисунок 5.1 – Mind Map

Наведена діаграма дозволяє наглядно поділити проект на дві окремі системи для тестування:

– основні механіки: зниження кисню, здоров’я, відновлення кисню та здоров'я, підбирати предмети, використовувати предмети, викидати предмети, коректність використання предметів, пересування;

– мережева взаємодія: Створення сесії одним гравцем, Приєднання до сесії кількома гравцями, Перевірка наявності всіх гравців в сесії, Перевірка наявності коректної інформації про гравців в лобі, реплікація гравців, реплікація предметів, реплікація ворогів, реплікація модулів, Перевірка синхронізації дій гравців, Відключення гравців від сесії.

Перевірка коректності роботи HUD відбувається під час перевірки основних механік, тому виділяти його в окрему систему сенсу немає.

5.2 Розробка тестових випадків

На основі класифікації, отриманої за допомогою розробленої Mind Map-діаграми, було сформовано детальний перелік тестових випадків, необхідних для всебічного та глибокого тестування системи. Кожен тестовий випадок, або тест-кейс, містить детальний опис конкретної тестової ситуації, очікуваний результат, фактично отриманий результат, а також додаткові коментарі, якщо вони необхідні для кращого розуміння тестування.

В таблиці 5.1 представлено повний список тест-кейсів, розроблених у процесі тестування програмного модуля. Пріоритетність кожного тест-кейсу визначається за шкалою від P1 до P3, де P1 відповідає найвищому рівню пріоритету, а P3 – найнижчому. Критичність потенційних багів оцінюється за шкалою від S1 до S5, де S1 означає блокуючий дефект, який унеможливлює подальшу роботу системи, а S5 вказує на тривіальний дефект, що не має значного впливу на функціональність системи.

Таблиця 5.1 – Список тестових випадків

Тест № 1	
Назва тесту:	Передчасне застосування HPTank
Опис тесту:	Клієнт в stendalone mode при вокористанні HPTank застосовує його до того, як статусбар повністю закінчится
Компонент системи:	HPTank
Пріоритет:	P1
Критичність:	S3
Кроки відтворення:	Взяти у руки HPTank на клієнті в standalone mode, та почати його використовувати
Очікуваний результат:	HPTank застосується, коли статус бар повністю заповниться
Фактичний результат:	HPTank застосується на половині статус бару
Результат:	Знайдено баг, виправлено 02.04.2024
Тест № 2	
Назва тесту:	Лагодження дверей в відстикований відсік
Опис тесту:	Користувач може лагодити двері в відстикований відсік
Компонент системи:	Door
Пріоритет:	P1
Критичність:	S3
Кроки відтворення:	Полагодити двері в відстикований відсік

Продовження таблиці 5.1

Очікуваний результат:	Неможна лагодити двері в відстикований відсік
Фактичний результат:	Можна полагодити двері в відстикований відсік, та пройти у них
Результат:	Знайдено баг, виправлено 03.04.2024
Тест № 3	
Назва тесту:	WidgetInteraction на клієнті
Опис тесту:	WidgetInteraction на клієнті при затисканні кнопки її не затискає а натискає та одразу відпускає її
Компонент системи:	MainCharacter
Пріоритет:	P1
Критичність:	S3
Кроки відтворення:	На клієнті спробувати зарядити зброю чи інстурменти
Очікуваний результат:	На клієнті кнопка затискається при затисканні клавіші взіємодії
Фактичний результат:	На клієнті кнопка швидко натискається та відпускається при затисканні клавіші взіємодії
Результат:	Знайдено баг, виправлено 12.04.2024
Тест № 4	
Назва тесту:	Вогонь наносить різний урон
Опис тесту:	При поганому інетрнет з'єднанні, вогонь наносить різний урон на клієнті та сервері

Кінець таблиці 5.1

Компонент системи:	Fire
Пріоритет:	P1
Критичність:	S3
Кроки відтворення:	При поганому інетрнет з'єднанні викликати вогонь та стати у нього
Очікуваний результат:	Вогонь наносить однаковий урон на сервері та клієнті
Фактичний результат:	Вогонь наносить різний урон на сервері та клієнті
Результат:	Знайдено баг, виправлено 16.04.2024
Тест № 5	
Назва тесту:	Від'єднанні гравця під час гри
Опис тесту:	При від'єднанні гравця під час гри, предмети, що він тримає, залишаються весіти в повітрі
Компонент системи:	MainCharacter
Пріоритет:	P1
Критичність:	S1
Кроки відтворення:	Вимкнути гру під час сесії через alt+F4
Очікуваний результат:	Гравець виходить, і його речі викидаються з його рук, щоб їх можна було узяти
Фактичний результат:	Гравець виходить та його речі залишаються в повітрі та їх неможливо узяти
Результат:	Знайдено баг, виправлено 20.04.2024

Усі інші тести було проведено під час написання програмного коду. Подібні тести призначені для оцінки працездатності різноманітних атомарних механік або геймплейних нюансів, пов’язаних із роботою демонстраційного додатку.

6 ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У межах тематики кваліфікаційної роботи була підготовлена наукова стаття у формі тез для XXVIII Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті»[11], який відбувся з 16 по 18 квітня 2024 року в онлайн-форматі на базі Харківського національного університету радіоелектроніки (ХНУРЕ). Представлена робота була опублікована у шостому випуску збірника матеріалів конференції під заголовком «Оптимізація серверних рішень в Unreal Engine 5»[12].

У цій роботі автори презентували концептуальну ідею оптимізації серверних рішень для Unreal Engine 5, спрямовану на покращення продуктивності та ефективності серверних інфраструктур. Основна увага була приділена розробці системи, що дозволяє забезпечити стабільну та надійну роботу серверів, оптимізуючи використання ресурсів та зменшуючи затримки.

Наукова публікація, підготовлена для Молодіжного форуму, є стислим викладом концепцій, детально розроблених у технічній документації. Там були описані методи інтеграції оптимізованих серверних рішень у різні ігрові механіки та системи. Зокрема, було розглянуто питання оптимізації структури даних для підвищення продуктивності та зменшення затримок при обробці запитів.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено унікальний кооперативний космічний симулятор, що враховує всі виявлені під час аналізу предметної галузі проблеми існуючих рішень. Гра повністю відповідає вимогам до онлайн взаємодії, безпеки, масштабування та продуктивності. В процесі роботи було визначено цільову аудиторію, що дозволило створити продукт, орієнтований на конкретних користувачів.

Під час проектування гри було створено детальний план розробки, який включав усі ключові етапи від концепції до реалізації. Розроблено концептуальний дизайн гри, що включає детальні ігрові механіки, систему персонажів із їх характеристиками, можливості взаємодії з оточенням, а також систему використання інструментів та зброї. Особлива увага приділялась проектуванню HUD (Head-Up Display), що забезпечує гравців інтуїтивно зрозумілим інтерфейсом для відображення ключової інформації.

Було виконано UML проектування програмного забезпечення, включаючи створення Use-case діаграми, опис класів персонажів та їх характеристик. Це забезпечило чітку структурну основу для розробки гри та дозволило команді розробників злагоджено працювати над проектом.

Гра була розроблена на базі Unreal Engine 5 із використанням Blueprint та C++, що забезпечило високу продуктивність та можливість масштабування. Було реалізовано всі спроектовані механіки, а також розроблено мережеву інфраструктуру з використанням Epic Online Services, що дозволяє гравцям взаємодіяти в режимі реального часу. Після завершення розробки було проведено ретельне тестування та оптимізацію продуктивності гри.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Epic Online Services (EOS) Overview. [Електронний ресурс] – URL: https://dev.epicgames.com/docs/epic-online-services (дата звернення: 13.05.2024).

2. Introduction to Blueprints | unreal engine documentation. [Електронний ресурс] – URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/introduction-to-blueprints-visual-scripting-in-unreal-engine?application_version=5.3 (дата звернення: 13.05.2024).

3. Programming and Scripting | unreal engine documentation. [Електронний ресурс] – URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-programming-and-scripting?application_version=5.3 (дата звернення: 13.05.2024).

4. Латиш А. С. Модель трафіку онлайн ігор. Science, research, development. 2020. № 28.

5. Enhanced Input | unreal engine documentation. [Електронний ресурс] – URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/enhanced-input-in-unreal-engine?application_version=5.3 (дата звернення: 13.05.2024).

6. Ареф'єв О. О. Інтерфейс користувача. особливості розробки інтерфейсу користувача для ігрових застосунків. Science, research, development. 2021. № 41.

7. Networking and multiplayer | unreal engine documentation. [Електронний ресурс] – URL: https://docs.unrealengine.com/4.26/en-US/InteractiveExperiences/Networking/ (дата звернення: 14.05.2024).

8. Multiplayer game development with unreal engine 5: create compelling multiplayer games with C++, blueprints, and unreal engine's networking features : підручник. Packt Publishing, 2023. 394 с.

9. Даниель Я. Д. Аналіз мережевих протоколів з погляду забезпечення безпеки передавання даних. 20-й Ювілейний Міжнародний молодіжний форум. 2016.

10. Advanced sessions plugin. Epic Developer Community Forums. [Електронний ресурс] – URL: https://forums.unrealengine.com/t/advanced-sessions-plugin/30020 (дата звернення: 13.05.2024).

11. Каталог виставки технічної творчості молоді. [Електронний ресурс] – URL: https://openarchive.nure.ua/handle/document/26355 (дата звернення: 21.05.2024).

12. Т. 6. Конференція "Інформаційні інтелектуальні системи". [Електронний ресурс] – URL: https://openarchive.nure.ua/handle/document/26351 (дата звернення: 21.05.2024).

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

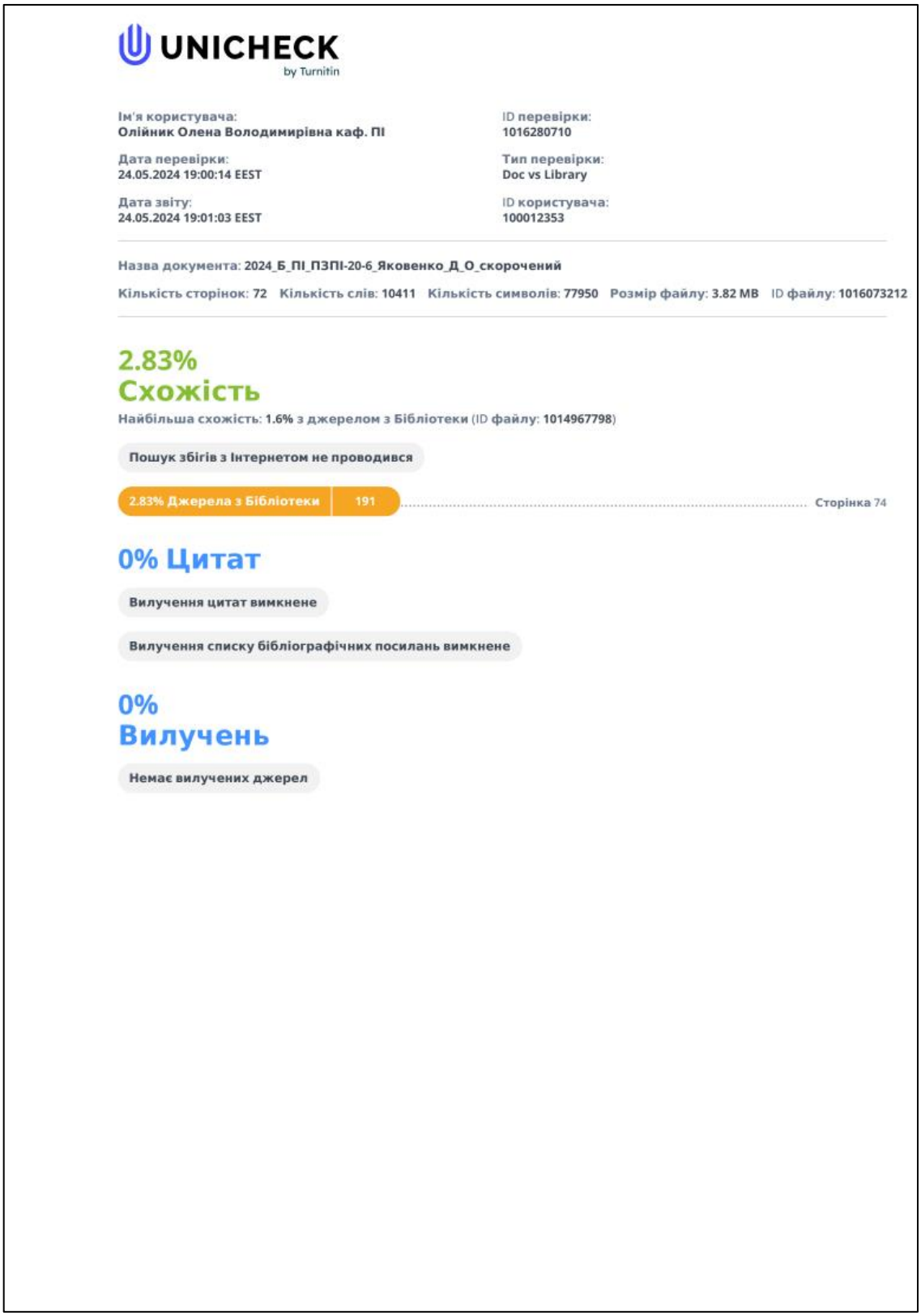


Рисунок А.1 – Результаті перевірки на унікальність тексту в базі ХНУРЕ

ДОДАТОК Б

Слайди презентації

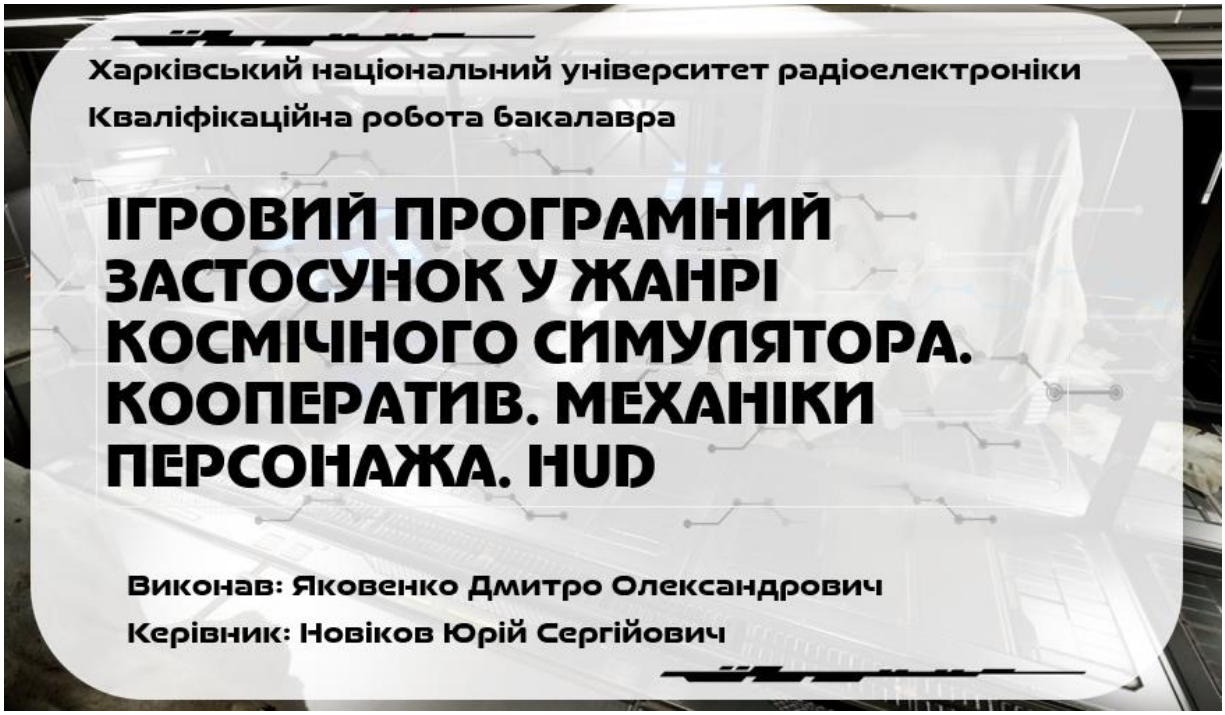


Рисунок Б.1 – Слайд 1

Рисунок Б.2 – Слайд 2

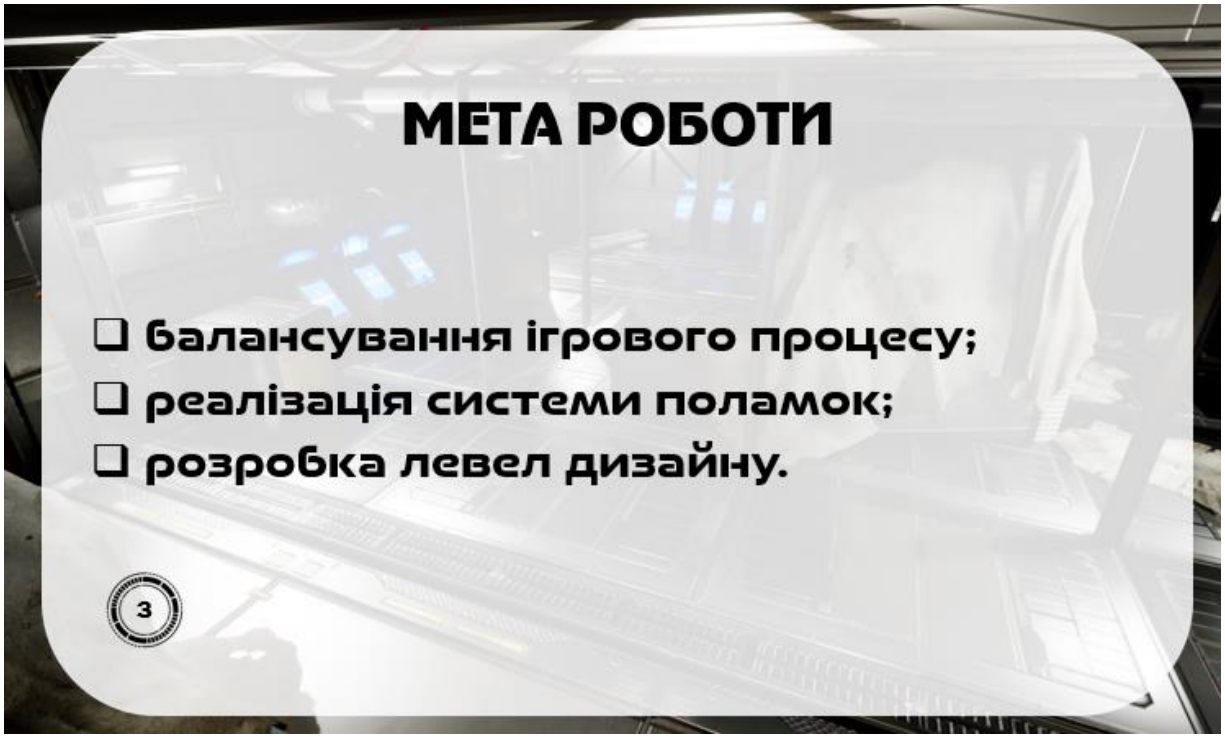


Рисунок Б.3 – Слайд 3

Рисунок Б.4 – Слайд 4

Рисунок Б.5 – Слайд 5

Рисунок Б.6 – Слайд 6

Рисунок Б.7 – Слайд 7

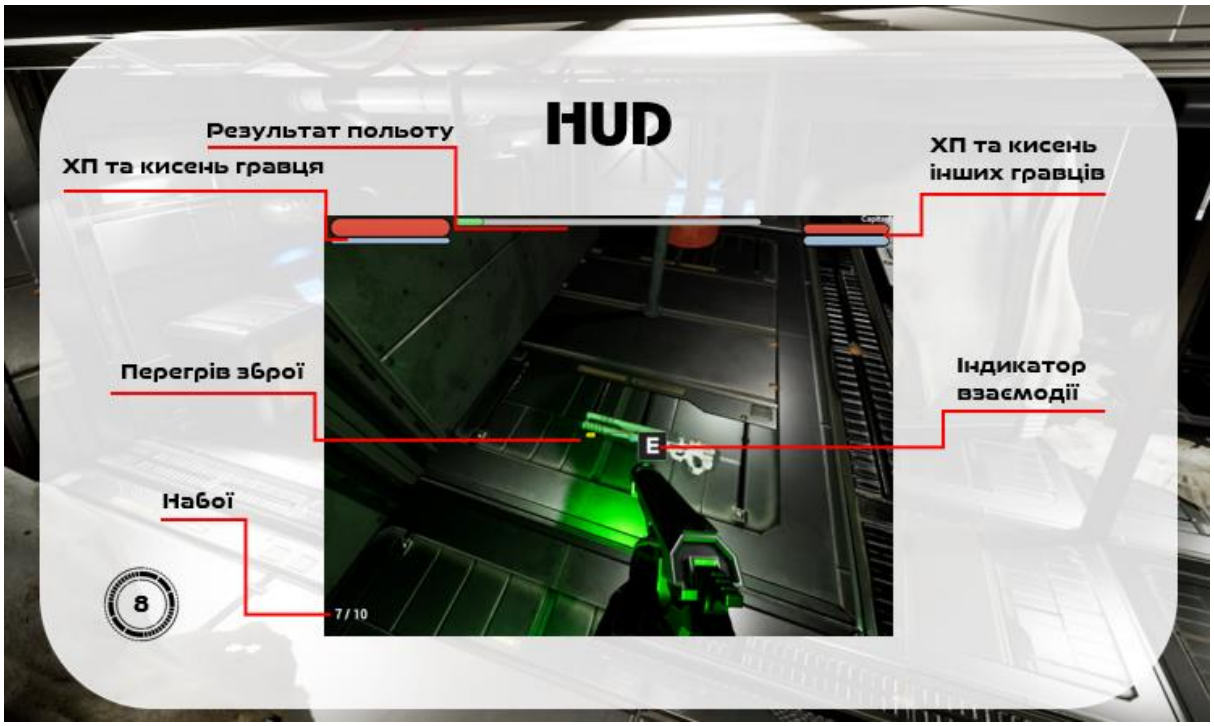


Рисунок Б.8 – Слайд 8

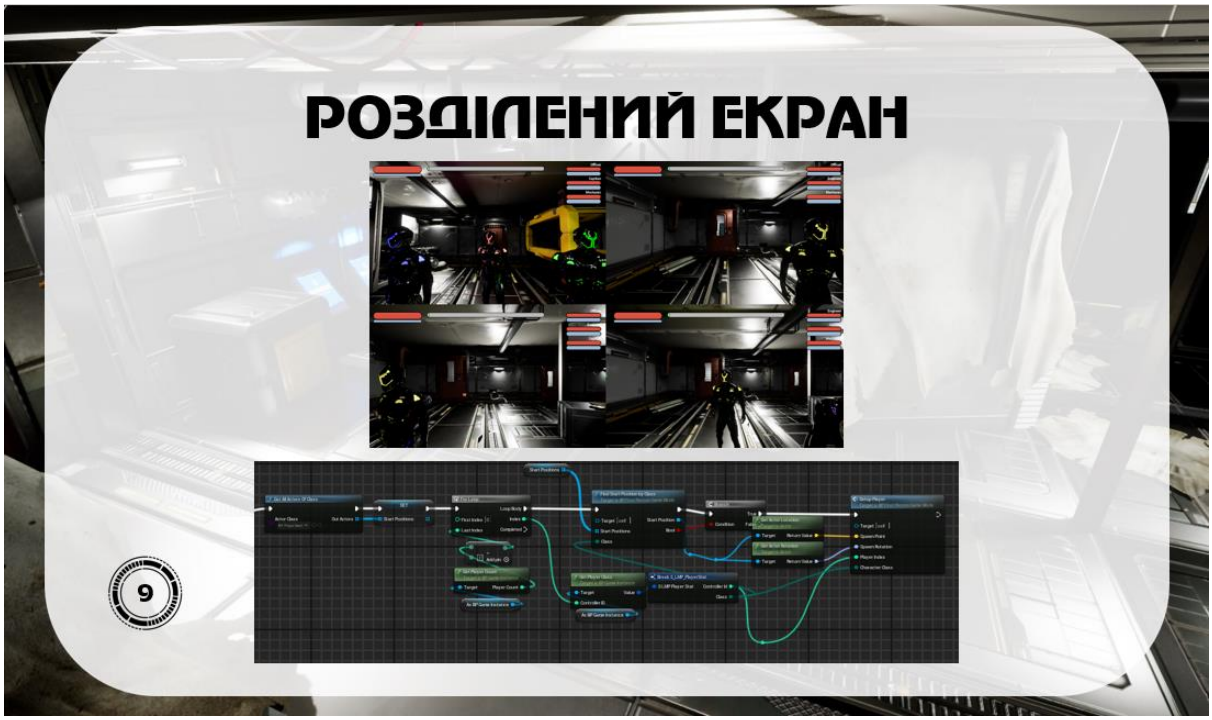


Рисунок Б.9 – Слайд 9

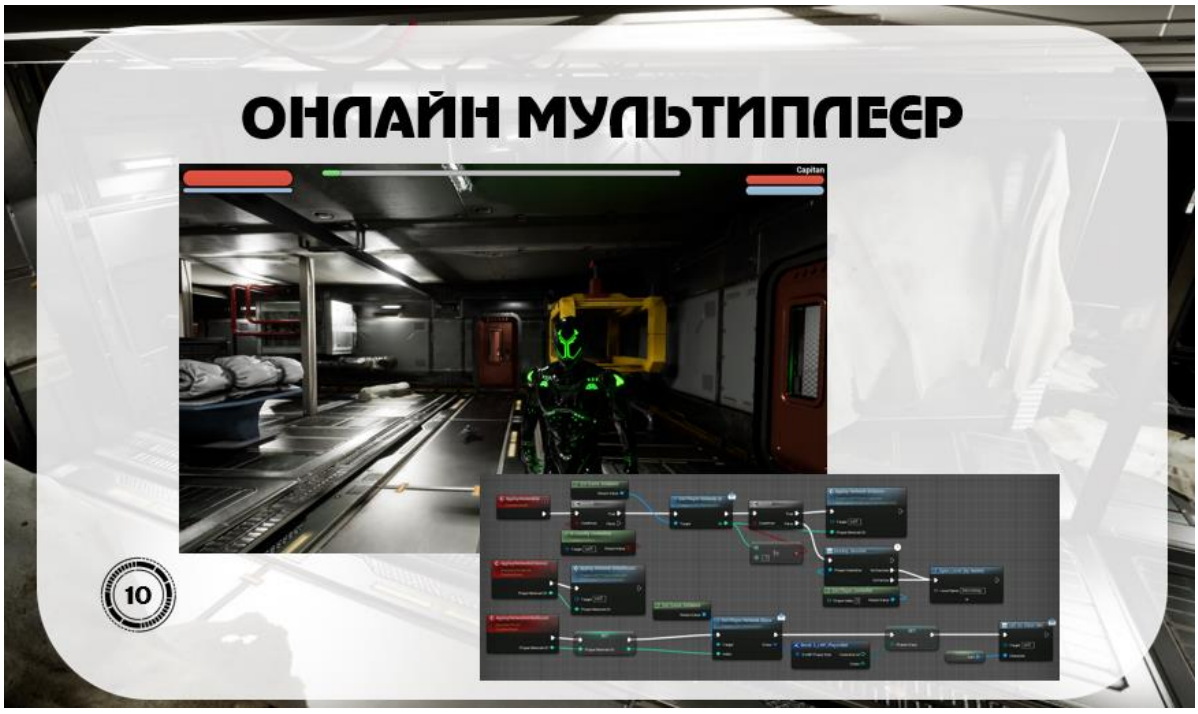


Рисунок Б.10 – Слайд 10

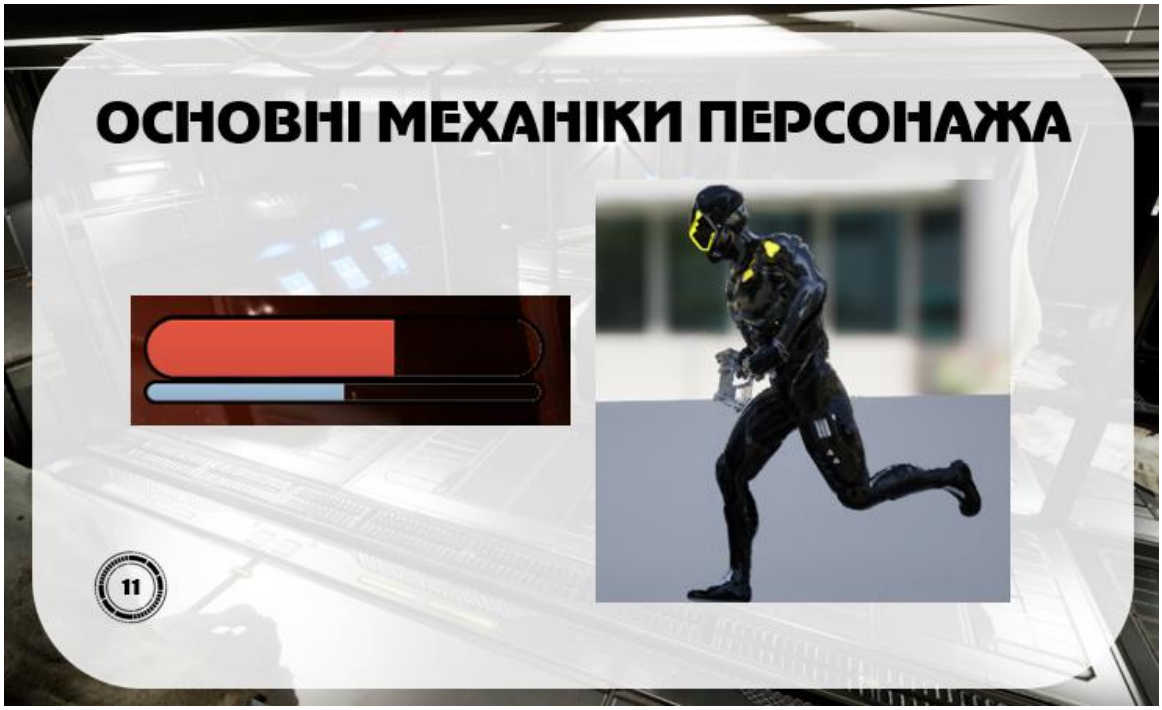


Рисунок Б.11 – Слайд 11

Рисунок Б.12 – Слайд 12

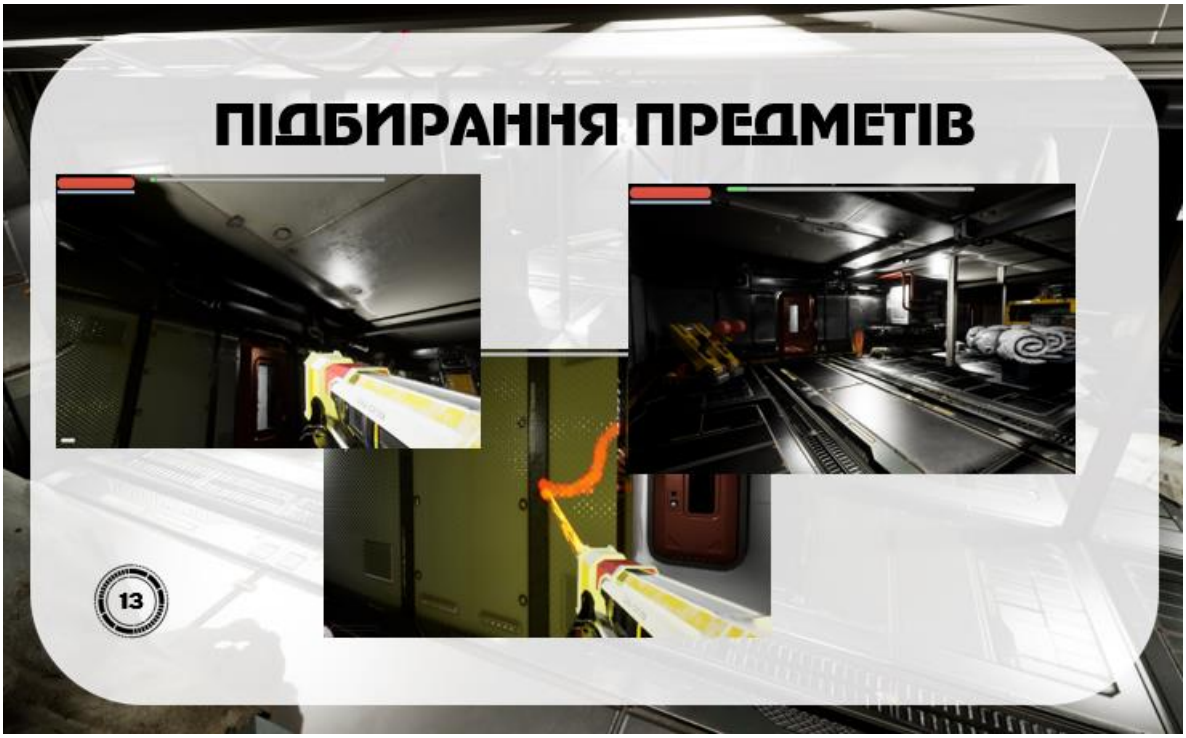


Рисунок Б.13 – Слайд 13

Рисунок Б.14 – Слайд 14

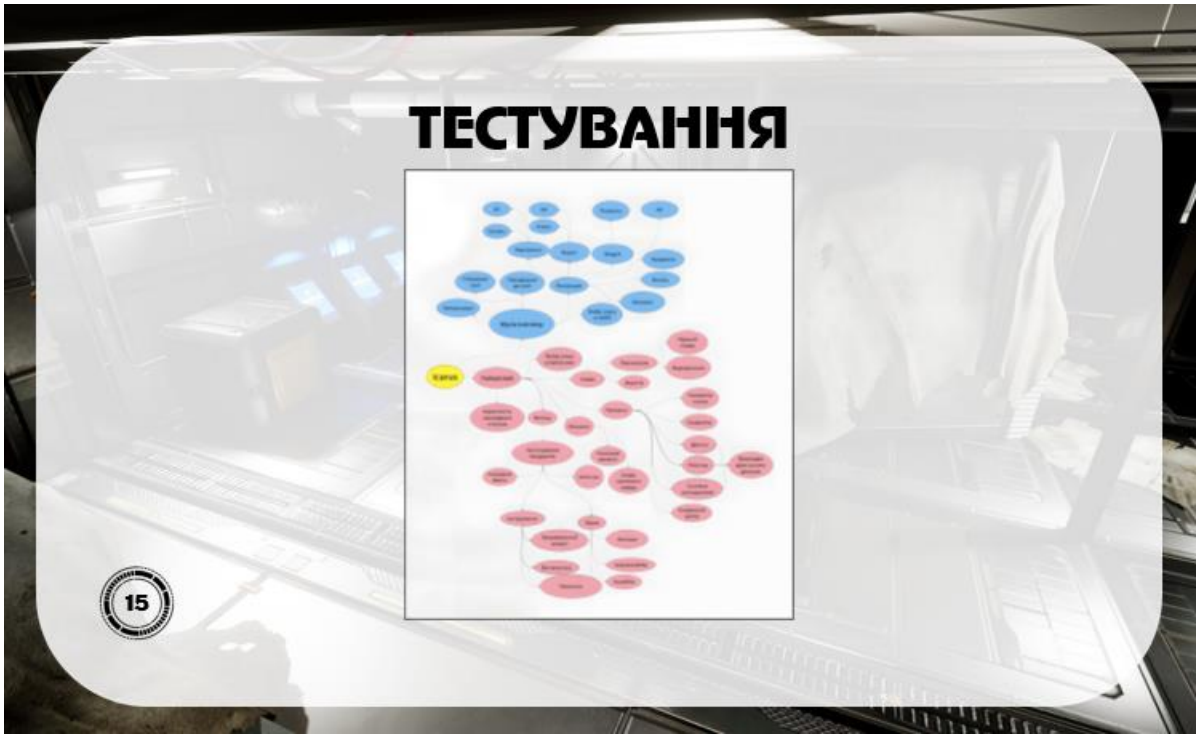


Рисунок Б.15 – Слайд 15

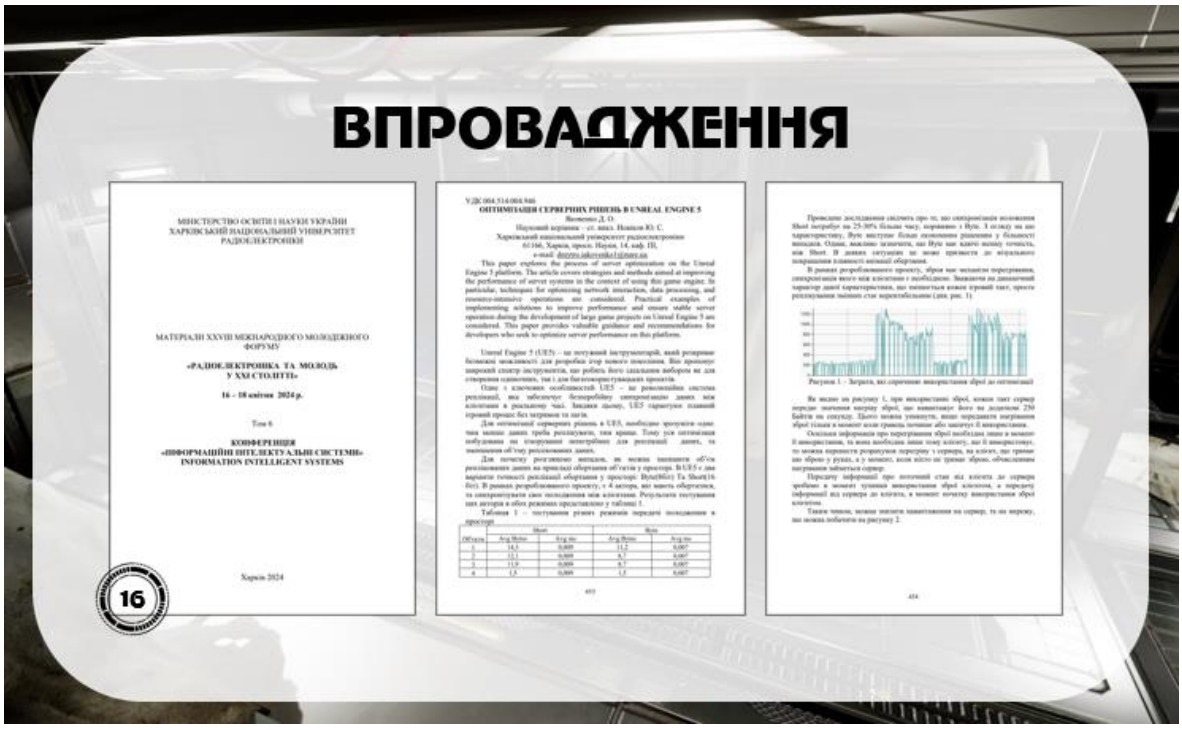


Рисунок Б.16 – Слайд 16

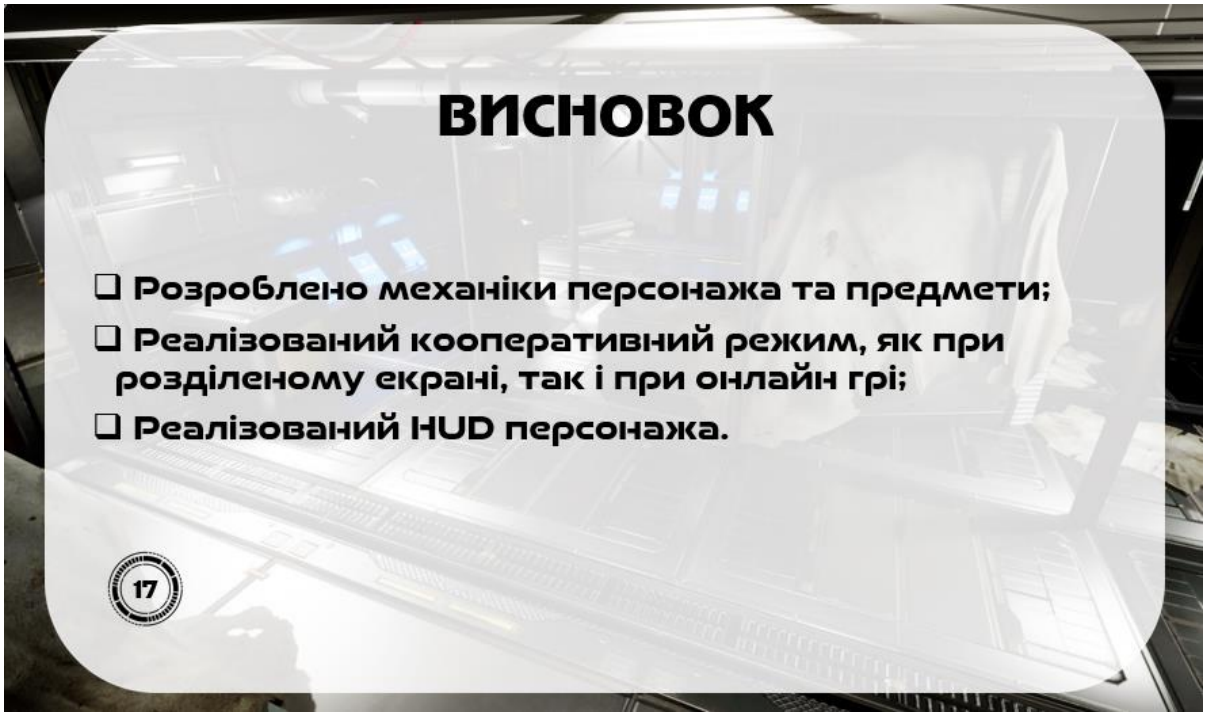


Рисунок Б.17 – Слайд 17

Рисунок Б.18 – Слайд 18

ДОДАТОК В

Геймдизайн-документ

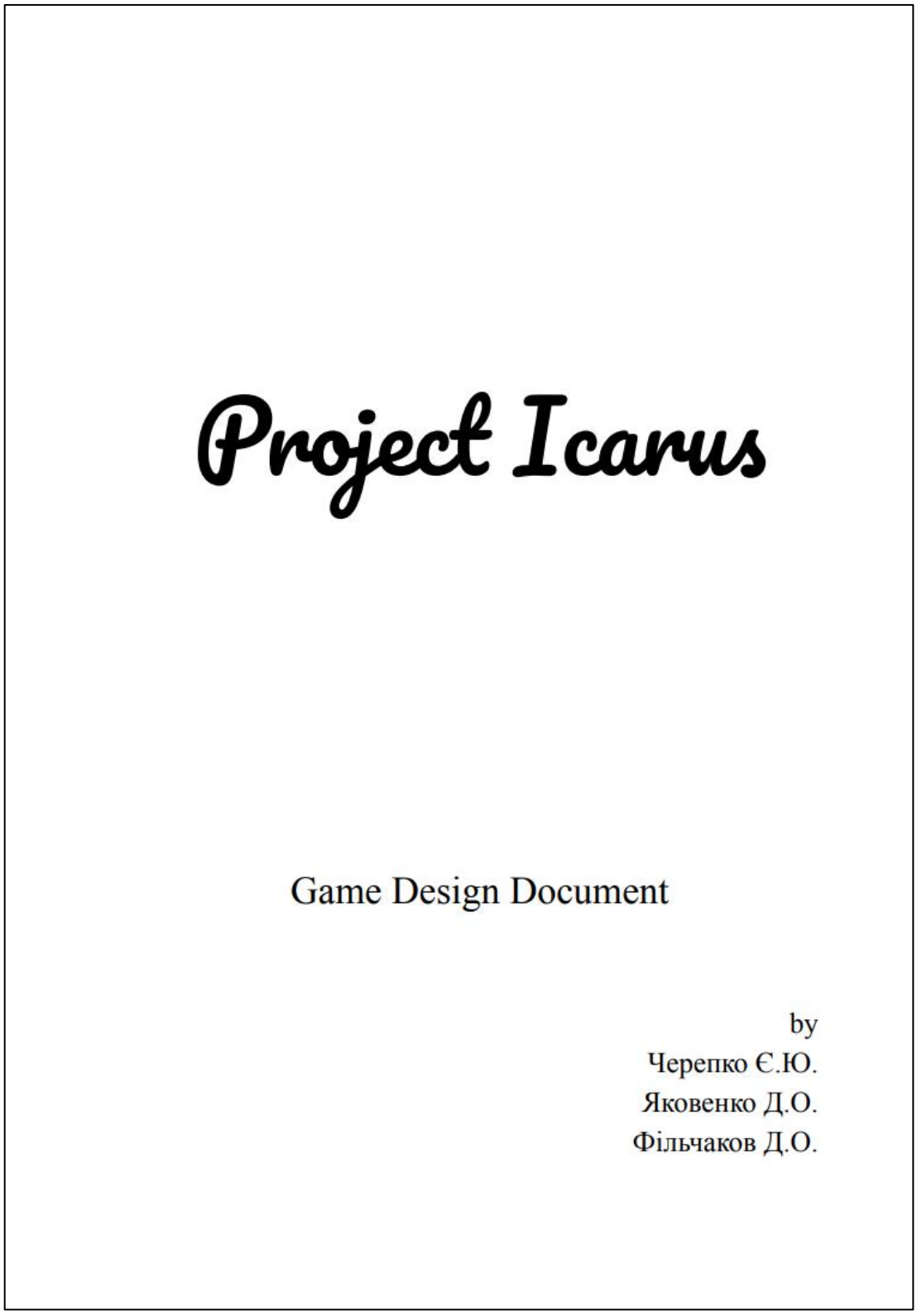


Рисунок В.1 – Сторінка 1 дизайн-документу

Рисунок В.2 – Сторінка 2 дизайн-документу

1. Загальні положення

1.1 Тетра

1) технологія – ПК;
2) механіка – симулятор виживання на космічному кораблі;
3) історія – злочинець;
4) естетика – Sci-fi.

1.2 Введення

Космічні пірати потрапляють у чергову пригоду, під час якої їх корабль зазнає пошкоджень і з кожною секундою починає розвалюватись. Їм треба досягти фінальної точки до того моменту, як від космічного корабля залишиться сама стеля, об'єднавшись з командою, або взявши все у свої руки. Слідкуйте за станом корабля, лагодіть що можливо полагодити, відбивайтеся від загарбників.

1.3 Цільова аудиторія

Основна аудиторія цієї гри це люди віком від 12 до 27 років. В першу чергу гра зацікавить командних гравців, або людей які полюбляють грати з рідними чи друзями. Також гра може залучити людей яким подобається динамічний геймплей зі швидким прийняттям рішень.

1.4 Unique selling proposition

- Проривайтеся крізь космічні простори у спробі втекти від космічної федерації на кораблі, що кожної миті розвалюється на шматочки!
- Зберіть команду спеціалістів у кооперативі на розділеному екрані та організуйте свою роботу, або спробуйте вижити на кораблі наодинці;

Рисунок В.3 – Сторінка 3 дизайн-документу

- Вчасно реагуйте на збої свого космічного судна та відбивайте атаки космічних загарбників.

1.5 Час ігрової сесії

Гра буде поділена на рівні, кожен з яких буде тривати по 10-15 хв. З кожним рівнем складність буде зростати, а в перервах можна буде купити потрібне обладнання та озброєння. Всього буде 3 рівні.

1.6 Технічні характеристики

Мінімальні системні вимоги для ПК:

Windows 10 64-bit, Quad Core 2.4 GHz, 8 GB RAM, GeForce GTX 1050;

Управління: клавіатура та миш, геймпад.

Рисунок В.4 – Сторінка 4 дизайн-документу

2. Ігровий процес

Гра починається на космічному кораблі, який рухається до наступної контрольной точки. Всього контрольних точок 3, включаючи фінальну. Основна задачи гри вижити дійшовши до фінальної точки. Виживанню персонажів буде сприяти стан корабля, а також працюючі модулі на ньому. Під час мандрівки, корабель буде обстрілюватися, що може наносити шкоди корпусу та модулям корабля, екіпажу, а також викликати пожежі (більш детально про загрози на кораблі описано у пункті 2.3.1). Крім того на корабель можуть проникати загарбники, які також несуть загрозу як екіпажу так і кораблю. При досягненні фіналу гри, гравці отримують результат в очках, який буде залежати від кількості грошей, які команда змогла зберегти під час мандрівки, та від цілісності корабля.

2.1 Характеристики персонажів

Ігрові персонажі будуть мати наступні характеристики:

- HP – здоров'я персонажа. При зниженні до 0, персонаж помирає, тим самим вибуває з гри повністю, або якщо гра виконувалась у со-ор інший гравець може викупити нового персонажа в магазині для вільного гравця;
- Кисень. Під час перебування у секції з пожежею або пробоїною він буде знижуватися (кожны 2 сек на 1 од). Після вичерпання запасів кисню, почне знижуватись HP (кожні 1 сек на 1 од), що може призвести до смерті. Якщо персонаж опускає забрало шолома скафандру, то кисень спочатку буде витрачатися з нього.
- Швидкість переміщення. Вона залежить від предмета, який тримає гравець.
- Рівень бойових навичок – впливає на якість персонажа у бою - додатковий урон від зброї, розкид стрільби (хитання ствола), відстань відкидання.

Рисунок В.5 – Сторінка 5 дизайн-документу

- Рівень навичок Інженера – впливає на здатність персонажу лагодити складні електричні системи, такі як реактор, двигуни, систему охолодження, систему подачи кисню, систему автопілота, генератори, камери та систему спостереження.
- Рівень навичок Механіка – впливає на здатність персонажу лагодити механічні пошкодження корабля, такі як пробоїни, дверей та пожеж.

2.2 Класи персонажів

В грі є чотири класи персонажів: Офіцер, Механік, Інженер, Капітан. У кожного рівень навичок розподілений відповідно свого класу:

В одиночній грі

	Рівень навичок Механіка	Рівень навичок Інженера	Рівень бойових навичок	Здоров'я	Кисень
Капітан	3	3	3	200	200

В коопі

	Рівень навичок Механіка	Рівень навичок Інженера	Рівень бойових навичок	Здоров'я	Кисень
Офіцер	1	1	3	200	100
Капітан	2	2	2	100	100
Механік	3	1	1	100	200
Інженер	2	3	1	100	100

Рисунок В.6 – Сторінка 6 дизайн-документу

2.3 Механіки

2.3.1 Поломки

Час від часу секції корабля будуть отримувати поломки. Поломками може бути пошкодження корпусу, викликана пожежа, поломка модуля, або втрата цілого відсіку.

2.3.1.1 Пошкодження корпусу та дверей

При пошкодженні корпусу відповідний відсік втрачає кисень. При трьох і більше пошкоджень в одному відсіку, швидкість переміщення в ньому зменшується.

При пошкодженні дверей їх буде неможливо зачинити, що буде призводити до розповсюдження безкисневості у відкриті відсіки, якщо суміжні з ними відсіки мають пробоїни.

2.3.1.2 Пожежа

При тривалому не ремонтуванні критичних поломок, почнеться пожежа. Коли у модулі почалась пожежа, у ньому закінчиться кисень, і потрібно буде використовувати його запаси зі скафандру. При тривалій пожежі, вона може перекинутись на інші відсіки.

2.3.1.3 Поломка модуля

Модулі корабля напряму впливають на ті чи інші його системи, тому їх пошкодження буде впливати на якість та працездатність цих систем. Кожен модуль має два рівні поломок – звичайний та критичний (про особливості поломок дивись відсіки корабля в яких він розташований у пунктах 2.5.1 - 2.5.9).

Рисунок В.7 – Сторінка 7 дизайн-документу

2.3.1.4 Втрата відсіку

При втраті відсіку, його вже ніяк не можна буде повернути, при цьому втрачаються всі модулі відсіку. Деякі відсіки неможливо втратити, деякі викликають миттєву поразку. Про наслідки для кожного відсіку див. п. 2.5.1 - 2.5.9.

2.3.2 Лагодження систем корабля

Для лагодження пошкоджень потрібні відповідні для цього предмети:

- пошкодження корпусу та дверей лагодяться зварювальним апаратом (3 сек);
- пожежі гасяться вогнегасником (секунда за одиницю вогню);
- пошкодження модулів лагодяться паяльником (звичайне - 4 сек, критичне - 8 сек).

Щоб полагодити пошкодження, треба знайти місце поломку, яка буде виділена vfx ефектами і направити на неї зварювальним апаратом, якщо це пробоїна корпусу, або паяльником, якщо це пошкодження модуля. При цьому знижується заряд відповідного інструмента.

Якщо у гравця в руках немає потрібного обладнання, або він не хоче витрачати заряд відповідного обладнання, можна використати універсальний ресурс (синю ізоленту), щоб відремонтувати будь-яку поломку, але на дуже короткий термін (10 секунд).

Щоб загасити пожежу, треба взяти вогнегасник та почати розпилювати порошок на вогонь.

2.3.3 Розхідники

Всі розхідники у грі можна розділити на 3 категорії:

1) заряди для інструментів – заряди для паяльника та заряди для зварювального апарату.
2) розхідники персонажів – аптечки та балони з киснем;

Рисунок В.8 – Сторінка 8 дизайн-документу

3) бойові розхідники – амуніція для зброї.

2.3.4 Гроші

На початку гри гравці мають фіксовану суму грошей. Гроші витрачаються у магазинах, а також від неї залежить фінальний результат гри.

2.3.5 Магазин

Магазин відкривається на контрольних точках і доступний біля докера. Товари магазину включають наступні:

Товар	**Ціна**
заряди для паяльника	15
заряди для вогнегасника	
заряди для зварювального апарату	10
аптечка	40
найняти нового члена екіпажу(воскресити померлого)	100
балони кисню	25
патрони для пістолету	10
напівавтоматичний автомат	100
патроны напівавтоматичного для автомату	15
гвинтівка	200

Рисунок В.9 – Сторінка 9 дизайн-документу

патроны для гвинтівки	20

заряди для паяльника;

заряди для зварювального апарату;

аптечка;

найняти нового члена екіпажу(воскресити померлого);

балони кисню;

патрони для пістолету;

напівавтоматичний автомат;

патроны напівавтоматичного для автомату;

гвинтівка;

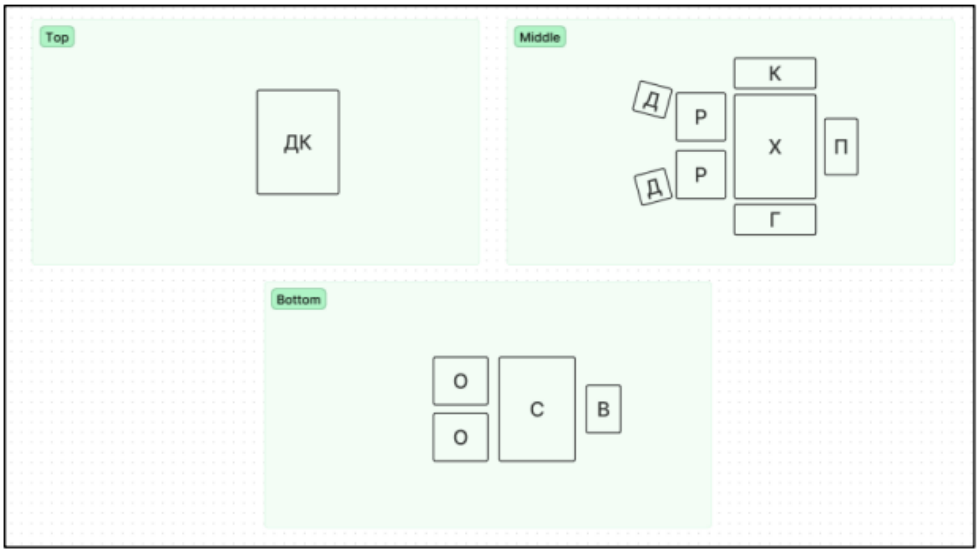
патроны для гвинтівки.

2.4 Корабель

Корабель розділений на секції, які з'єднані між собою шлюзом, або дверима. До основних відсіки корабля належать: палуба, секція реактора, секція двигуна, секція охолодження, склад, секція життєзабезпечення, генераторна, хаб, докер. Більшість з цих секції мають модулі, які можуть бути пошкоджені.

Загальна схема корабля має наступний вигляд:

Рисунок В.10 – Сторінка 10 дизайн-документу



де П - палуба

Р - реактор

Д - секція двигуна

О - секція охолодження

С - склад

К - секція життєзабезпечення

Г - генераторна

Х - хаб

ДК - докер

В - посадковий відсік

2.4.1 Палуба

Секція, на якій будуте розташована система спостереження для аналізу стану корабля та моніторингу вторгнень, інструменти розгерметизування частин корабля та система автопілота.

Система спостереження представлена у вигляді зображень з камер кожного відсіку. Коли виникає поломку у відсіку, індикатор загоряється біля

Рисунок В.11 – Сторінка 11 дизайн-документу

відповідної камери. Дізнатися деталі поломки вже можна через камеру. При поломці, камери перестають працювати, але індикатори працюють. При критичній поломці не працюють як камери, так і індикатори. Також цей відсік неможливо втратити.

Інструмент розгерметизації дозволяє відстикувати модуль корабля, щоб наслідник його поломки цього не перейшли на інші модулі. При його поломці, відстикування модулів корабля буде проводитись повільніше. При критичній поломці, капітан не зможе відстикувати модулі корабля

При поломці автопілота, корабель почне частіше отримувати пошкодження корпусу, тобто пробоїни. В залежності від тривалості поломки, частота появи пробоїн почне збільшуватись.

2.4.2 Секція реактора

На цій секції будуть розташований реактор, який підтримує життєздатність двигуна. При його пошкодженні спалахує пожежа. При виведенні зі строю, зупиняється відповідний двигун. При втраті цієї секції, втрачається секція з відповідним двигуном. При втраті всіх реакторів гра закінчується поразкою.

2.4.3 Секція двигуна

Від двигуна корабля напряму залежить його корабля. Пошкодження двигуна сповільнює корабель, а критична поломка зупинить переміщення корабля. При втраті двигуна, швидкість корабля зменшується вдвічі. Всього на кораблі дві секції з двигунами. При втраті двох двигунів гра закінчується поразкою.

2.4.4 Секція охолодження

На цій секції будуть розташовані системи охолодження реактора. На кораблі дві такі секції, по кожному на реактор. При поломці, реактори час від

Рисунок В.12 – Сторінка 12 дизайн-документу

часу почнуть перегріватися, шо буде викликати їх середню поломку. Якщо ж реактори вже перегріті, то це буде викликати критичну їх поламку.

При критичній поломці, або втраті цієї секції, реактор буде ще частіше перегріватися.

2.4.5 Склад

Секція з усім знаряддям для лагодження, розхідниками та зброєю.

2.4.6 Секція життєзабезпечення

Секція, яка відповідає за подачу кисню на кораблі. При частковій поломці кисень зникає на всіх відсіках корабля, окрім палуби, хабу та самої секція життєзабезпечення. При критичній поломці, або втраті відсіку кисень зникає усюди.

2.4.7 Генераторна

Секція з генератором, який відповідає за освітлення корабля. При поломці, світло на кораблі почне блимати, а при критичній поломці вимкнеться повністю. При від'єднані цієї секції, світло на кораблі вже ніколи не повернеться.

2.4.8 Хаб

Велика основна секція корабля. Звідси можна перейти до інших секцій.

2.4.9 Докер

Невелика секція корабля, призначена для стикування до інших кораблів. Через неї на борт будуть потрапляти загарбники (у більшості випадках). Також через неї виконується торгівля у магазинах. Не може бути розгерметизована.

Рисунок В.13 – Сторінка 13 дизайн-документу

2.4.10 Посадковий відсік

Звичайний відсік, через який команда висаджується на землі.

2.5 Зброя

2.5.1 Пістолет

Урон	10
Магазин	16
Скорострільність	3 пулі/сек
Режим стрільби	По 1 пулі

2.5.2 Напівавтоматичний автомат

Урон	14
Магазин	27
Скорострільність	3 пулі/сек
Режим стрільби	По 3 пулі

Рисунок В.14 – Сторінка 14 дизайн-документу

2.5.3 Гвинтівка

Урон	15
Магазин	30
Скорострільність	5 пулі/сек
Режим стрільби	Автоматичний

Рисунок В.15 – Сторінка 15 дизайн-документу

3. Референси

3.1 Гравці

Кожен гравець матиме свій унікальний колір підсвітки.

Рисунок В.16 – Сторінка 16 дизайн-документу

4. Інтерфейс

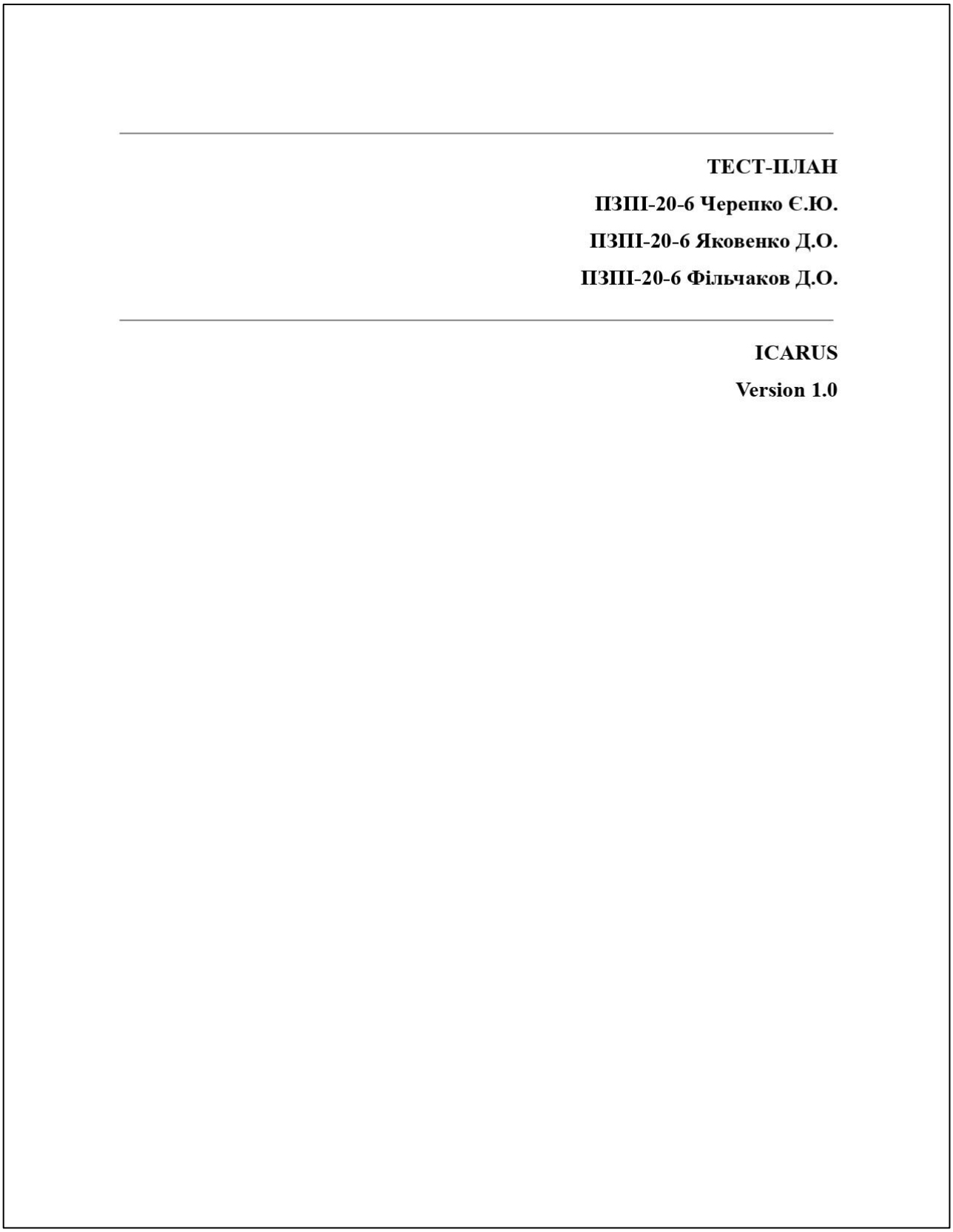
4.1 Основний геймплей

На екрані у гравця відображається інформація про стан його здоров'я (у верхньому лівому куті червоний статус бар), стан балонів з киснем (у верхньому лівому куті синій статус бар), стан здоров'я та балонів з киснем команди (у верхньому правому куті), залишок заряду його поточного предмета в руці предмету, якщо він міє заряди (внизу ліворуч), обраний предмет (внизу праворуч), індикатор шляху, який корабль подала, і який йому ще треба подолати, для успішного проходження рівня (зелений статус бар зверху).

Рисунок В.17 – Сторінка 17 дизайн-документу

ДОДАТОК Г

Тест-план



ТЕСТ-ПЛАН

ПЗПІ-20-6 Черепко Є.Ю.

ПЗПІ-20-6 Яковенко Д.О.

ПЗПІ-20-6 Фільчаков Д.О.

ICARUS

Version 1.0

Рисунок Г.1 – Сторінка 1 тест-плану

Revision History

Date	Version	Description	Author
26/02/2024	1.0	Initial revision	Черепко Є.Ю. Яковенко Д.О. Фільчаков Д.О.

Рисунок Г.2 – Сторінка 2 тест-плану

Table of Contents

Рисунок Г.3 – Сторінка 3 тест-плану

Test Plan

1. Вступ (Introduction)

1.1 Мета (Purpose)

Метою складання Тест Плану є опис процесу тестування комп'ютерної гри Icarus, що є нашою дипломною роботою. Розглянемо визначення існуючої інформації про гру, визначення рекомендованих вимог до тестування, опис стратегій тестування, рекомендацій щодо використання, визначення необхідних ресурсів і забезпечення оцінки випробувань, перелік тестових елементів гри.

1.2 Довідкова інформація (Background)

У якості об’єкта тестування було обрано гра Icarus, що є нашою дипломною роботою.

В ході тестування необхідно буде перевірити найважливіші складові гри . А саме:

1. Графічний інтерфейс;
2. Анімації персонажів;
3. Взаємодія персонажа з оточенням;
4. Ворогів;
5. Геймплей у різних режимах(одиночний\split-screen\multiplayer);

Саме ці функції є основою онлайн гри, що тестується.

Тестування гри буде виконано в ручному режимі. Якщо в результаті тестування буде знайдено 75 багів, то тест можна вважати вдалим.

Рисунок Г.4 – Сторінка 4 тест-плану

1.3 Галузь застосування (Scope)

Повний набір складових частин, що будуть протестовані:

1. Графічний інтерфейс:

– Головне меню;
– Меню налаштувань;
– Меню створення сесії;
– Меню обору ролі в split-screen;
– Меню обору ролі в мультиплеєрі;
– Магазин;
– Екран завантаження гри;
– HUD персонажа;
– Меню паузи.

2. Анімації персонажів:

– Анімація пересування персонажа;
– Анімації персонажа з різними предметами;
– Анімації персонажа при використанні споживаних предметів;
– Анімація пересування ворогів;
– Анімація атак ворогів;

3. Мультиплеер:

– Авторизація;
– Створення сесії;
– Під'єднання до сесії;
– Реплікація пересування;
– Реплікація станів персонажів;
– Реплікація модулів;

Рисунок Г.5 – Сторінка 5 тест-плану

– Реплікація ворогів;
– Реплікація предметів;
– Реплікація магазину;
– Реплікація інтерфейсу в лобі;
– Реплікація поломок;

4. Штучний інтелект ворогів:
– перевірка реакції ворогів на персонажів;
– перевірка реакції ворогів на модулі;
– перевірка коректності пошуку ворогом цілі;
– коректність пересування ворогів по навігаційній сітці;
– перевірка узгодженості атак ворогів між собою;

5. Магазин:
– коректність списання ігрової валюти та начислення предметів, або ресурсу;
– коректність оновлення наявних предметів з переходом на новий рівень;
– коректність відродження загиблих персонажів при їх викупі;

6. VFX
– Коректність відображення частищь.
– Оптимізованість візуальних ефектів.
– Коректність взаємодії частищь з навколишнім середовищем.

7. Баланс
– перевірка балансу характеристик персонажів;
– перевірка балансу характеристик ворогів;

Рисунок Г.6 – Сторінка 6 тест-плану

– перевірка балансу систему токенів атаки;
– перевірка балансу атак ворогів: швидкості, частоти, сили особливих ефектів;
– перевірка балансу характеристик модулів;
– перевірка збалансованості алгоритму поломок;
– перевірка балансу зброї;
– перевірка балансу предметів починки;
– перевірка балансу споживаних предметів;
– перевірка балансу пожеж;
– перевірка балансу витрачання кисню;

8. Геймплей у різних режимах:
– Вибір класу персонажа в split-screen;
– Вибір класу персонажа в мультиплеєрі;
– Геймплей в однокористувацькому режимі;
– Геймплей в split-screen;
– Геймплей в мультиплеєрі;
– Спавн персонажей;
– Винекнення поламок;
– Спавн ворогів;
– Респавн;
– Виграш та програш;

Ризиком, або непередбачуваною обставиною, що може вплинути на тестування може бути відмова роботи Epic Online Services з технічних причин або технічні роботи на сервері.

Рисунок Г.7 – Сторінка 7 тест-плану

1.4 Визначення проекту (Project Identification)

Документ і версія / дата) (Document and version / date)	Створено або Доступно (Created or Available)	Поступило або перегляну то (Received or Reviewed)
Специфікація вимог (Requirements Specification)	Так	Ні
Функціональна специфікація (Functional Specification)	Так	Так
Звіти використання - випадок (Use-Case Reports)	Ні	Ні
План проекту (Project Plan)	Так	Ні
Специфікація дизайну (Design Specifications)	Так	Так
Прототип (Prototype)	Так	Ні
Керівництво користувача (User's Manuals)	Ні	Ні
Бізнес модель (Business Model or Flow)	Ні	Ні
Модель даних (Data Model or Flow)	Ні	Ні
Бізнес-функції (Business Functions and Rules)	Так	Ні

Рисунок Г.8 – Сторінка 8 тест-плану

Оцінка ризиків (Project or Business Risk Assessment)	Ні	Ні

2. Вимоги до тестування (Requirements for Test)

В ході тестування необхідно буде перевірити найважливіші складові гри . А саме:

1. Графічний інтерфейс;
2. Анімації персонажів;
3. Взаємодія персонажа з оточенням;
4. Ворогів;
5. Геймплей у різних режимах(одиночний\split-screen\multiplayer);

Саме ці функції є основою онлайн гри, що тестується.

Тестування гри буде виконано в ручному режимі. Якщо в результаті тестування буде знайдено 75 багів, то тест можна вважати вдалим.

3. Стратегія тестування (Test Strategy)

Для тестування було прийнято рішення використати наступні типи:

- Функціональне тестування;
- Тестування інтерфейсу користувача;
- Тестування продуктивності;
- Тестування відмовостійкості;
- Тестування конфігурації;

Інтеграційне тестування, бізнес-цикл тестування, стресове тестування, тестування безпеки і контролю доступу, тестування навантаження, завантажувальне тестування та тестування інсталяції не будуть проведені з

Рисунок Г.9 – Сторінка 9 тест-плану

ряду причин, що будуть наведені нижче у відповідних пунктах.

3.1 Типи тестування (Testing Types)

3.1.1 *Дані і БД Інтеграційне тестування (Data and Database Integrity Testing)*

Цей тип тестування неможливо провести через відсутність БД. Від тесту відмовляємося, через непотрібність проведення.

3.1.2 *Функціоанльне тестування (Function Testing)*

Було прийнято рішення обрати цей тип тестування з метою перевірки правильності прийняття і обробки даних, а також належного виконання бізнес-правил. Тестування буде проведено за методом «білої скрині», тобто перевірки гри, занурюючись у код, щоб зрозуміти яка частина коду працює некоректно.

Мета випробування (Test Objective):	Виявити баги
Технічний прийом (Technique):	Методом «білої скрині» перевірити кожну функцію, виявити дефекти.
Критерії завершення (Completion Criteria):	Кожна складова гри протестована, виявлені помилки записані у баг-репорт
Спеціальні рекомендації (Special Considerations):	Зафіксувати процес тестування на відео

Рисунок Г.10 – Сторінка 10 тест-плану

3.1.3 *Бізнес-цикл тестування (Business Cycle Testing)*

Бізнес цикл тестування не буде використовуватися, оскільки ми не маємо необхідної документації. Цей тип тестування потребує документації, де визначені бізнес-процеси, що представляють собою сценарії щоденного використання. В нашому випадку такої документації немає у наявності. Від цього тестування відмовляємося.

3.1.4 *Тестування інтерфейсу користувача (User Interface Testing)*

Тестування інтерфейсу користувача було обрано з метою перевірки взаємодії користувача з інтерфейсом гри. В ході тестування необхідно перевірити зручність та правильність роботи інтерфейсу, а також виявити візуальні баги та баги анімацій.

Мета випробування (Test Objective):	Перевірити всі доступні користувачеві сторінки, діалогові вікна та елементи ігрового інтерфейсу під час геймплею.
Технічний прийом (Technique):	Створення та редагування випробувань для кожного елемента, щоб перевірити коректність його роботи та наявність візуальних помилок.
Критерії завершення (Completion Criteria):	Кожен елемент інтерфейсу був перевірений, виявлені візуальні баги занесені у баг-репорт.

Рисунок Г.11 – Сторінка 11 тест-плану

3.1.5 *Тестування продуктивності (Performance Profiling)*

Тестування продуктивності було обрано з метою виміру оцінки часу відгуку, перевірки швидкості транзакції, й інших вимог. Тестування продуктивності реалізується і виконується згідно з профілем, а також розглядається залежно від робочого навантаження або апаратної конфігурації.

Мета випробування (Test Objective):	Перевірка продуктивності різних функцій та час їх відгуку, порівняння з нормою.
Технічний прийом (Technique):	- В ході тестування проводити додаткові перевірки на наявність проблем з продуктивністю та багів, що викликають «лаги»; - Тестування проводити на конфігурації, що відповідає рекомендованим системним потребам, або краще.
Критерії завершення (Completion Criteria):	В ході перевірки всі показники продуктивності знаходяться в межах норми, під час тестування не виникає збоїв.

3.1.6 *Завантажувальне тестування (Load Testing)*

Рисунок Г.12 – Сторінка 12 тест-плану

Завантажувальне тестування не буде проведено, через неможливість залучити до тестування максимально можливу кількість користувачів. Цей вид тестування передбачає перевірку роботи гри при великій кількості користувачів.

3.1.7 *Стресове тестування (Stress Testing)*

Тестування неможливо провести через відсутності виділених серверів, оскільки вся мережева взаємодія відбувається через сервери, які хостят гравці, і стоїть обмеження на 4 гравців на один сервер.

3.1.8 *Навантажувальне тестування (Volume Testing)*

Навантажувальне тестування неможливо для проведення, оскільки гра не передбачає поля вводу де, є можливість ввести велику кількість даних для тестування. Крім того цей вид тестування не є обов'язковим для функцій, що необхідно перевірити.

3.1.9 *Тестування безпеки і контролю доступу (Security and Access Control Testing)*

Даний тип тестування неможливо провести через відсутність різних ролей у грі. Існує лише роль звичайного гравця.

3.1.10 *Тестування відмовостійкості та відновлення (Failover and Recovery Testing)*

Було прийнято рішення обрати цей тип тестування з метою перевірки

Рисунок Г.13 – Сторінка 13 тест-плану

відмовостійкості гри, при втрачанні з'єднання з сервером, або з клієнтом.

Мета випробування (Test Objective):	Перевірити всі етапи гри, на яких можуть виникнути проблеми з мережею.
Технічний прийом (Technique):	Створення та редагування випробувань для кожного етапу, щоб перевірити коректність його роботи.
Критерії завершення (Completion Criteria):	Кожен випадок був перевірений, виявлено баги, та занесено їх в баг-репорт.

3.1.11 *Тестування конфігурації (Configuration Testing)*

Конфігурація тестування перевіряє через тест роботу гри при різних умовах апаратної конфігурації. Цей тип тестування буде проведений з метою перевірки коректності роботи гри на мінімальній та рекомендованій конфігурації.

Рисунок Г.14 – Сторінка 14 тест-плану

Мета випробування (Test Objective):	Тестування проводиться належним чином на мінімальній та рекомендованій конфігурації.
Технічний прийом (Technique):	Проведення тестових випадків на різних конфігураціях та порівняння результатів виконання однакових функцій. Виявлення відмінностей у результатах, що будуть вважатися багами.
Критерії завершення (Completion Criteria):	Робота гри на різних конфігураціях перевірена, виявлені баги занесені в баг-репорт.
Спеціальні рекомендації (Special Considerations):	Для тестування використати наближену до Мінімальна: - Процесор: amd ryzen 5 1600; - Відеокарта: GTX 1650; - Озу: 8гб; А також конфігурацію, що буде вище за рекомендована: - Процесор: amd ryzen 5 3600; - Відеокарта: GTX 2070 - Озу: 16гб;

Рисунок Г.15 – Сторінка 15 тест-плану

3.1.12 *Тестування інсталяції (Installation Testing)*

Тестування інсталяції не потребується для тестування зазначених вище функцій, крім того інсталяція відноситься до тестування цифрового магазину, що не розглядається в тест плані.

3.2 Інструменти (Tools)

The following tools will be employed for this project:

	Tool
Test Management	Microsoft Excel
Defect Tracking	Jira
ASQ Tool for performance testing	FPS Monitor
Створення тест кейсів (Test case creation)	Microsoft Excel
Відстеження тест кейсів(Test case tracking)	Microsoft Excel

4. Ресурси (Resources)

4.1 Ролі (Roles)

Ця таблиця показує, кадрові забезпечення для проекту.

Рисунок Г.16 – Сторінка 16 тест-плану

Людський ресурс (Human Resources)		
Працівник (Worker)	Конкретні обов'язки або Коментарі (Specific Responsibilities or Comments)	Конкретні обов'язки або Коментарі (Specific Responsibilities or Comments)
Тест-менеджер, Менеджер з тестування проекту (Test Manager, Test Project Manager)	1	Забезпечує управління наглядом. Обов'язки: - технічна підтримка, - придбання відповідних ресурсів, - забезпечення управлінської звітності

Рисунок Г.17 – Сторінка 17 тест-плану

Конструктор тестів (Test Designer)	1	Визначення, пріоритетів, і реалізація тестів. Обов'язки: - створення плану тестування, - генерація тестових моделей, - оцінка ефективності тестових зусиль.
Тестувальник (Tester)	1	Виконання тестів. Обов'язки: - виконання тестів, - журнал результатів, - відновлення в журналі реєстрації після помилок, - документ зміни.
Виконавець (Implementer)	1	Реалізує модульні тести і тестові класи Обов'язки: - створює тестові класи і пакети, - виконує тестові моделі

Рисунок Г.18 – Сторінка 18 тест-плану

4.2 Система (System)

Нижче в таблиці представлені системні ресурси для тестування гри.

System Resources	
Resource	Name / Type
Апаратна конфігурація	Мінімальна: - Процесор: amd ryzen 5 1600; - Відеокарта: GTX 1650; - Озу: 8гб;
	Рекомендована: - Процесор: amd ryzen 5 3600; - Відеокарта: GTX 2070 - Озу: 16гб;

Рисунок Г.19 – Сторінка 19 тест-плану

Задачі проекту (Appendix A Project Tasks)

Нижче наведені завдання, пов'язані з тестом:

1. Спланувати тест
– визначити вимоги до тесту;
– розробити стратегію тестування;
– створюємо розклад;
– генеруємо план тестування.
2. Проектування тесту
– визначаємо та описуємо тестові кейси;
3. Реалізація тесту
– записувати або програмувати тестові скрипти;
– визначити специфічну для тестів функціональність в моделі проектування та реалізації;
– створення зовнішніх наборів даних.
4. Виконання тесту
– виконати тестові процедури;
– реєстрація дефектів.
5. Оцінити тест
– аналіз дефектів;
– визначити, чи були досягнуті критерії завершення тесту та критерії успіху.

Рисунок Г.20 – Сторінка 20 тест-плану

ДОДАТОК Ґ

Приклад програмного коду (Header BP_PlayerCharacter)

```
/** Main Character class */
UCLASS(Blueprintable, BlueprintType)
class ABP_PlayerCharacter : public AScifyCharacterBase
{
	GENERATED_BODY()
public:
	/** Function add mapping context to charcater */
	UFUNCTION(BlueprintCallable)
	void Add Mapping Context(UInputMappingContext* Mapping Context);

	/** Function remove mapping context from character */
	UFUNCTION(BlueprintCallable)
	void Remove Mapping Context(UInputMappingContext* Mapping Context);

	/** Function drop hold item */
	UFUNCTION(BlueprintCallable)
	void Drop Item();

	/** Function grab item */
	UFUNCTION(BlueprintCallable)
	void Grab Item(bool Condition, UObject* InputPin);

	/** Function set Local player to Widget */
	UFUNCTION(BlueprintCallable)
	void Set Widget Local Player();

	/** Function round float  */
	UFUNCTION(BlueprintPure)
	void RoundFloat(double Float, double RoundTo, double& RoundedFloat);

	/** Function get current character speed */
	UFUNCTION(BlueprintPure)
	void GetCurrentSpeed(double& Speed);

	/** Function set position WidgetFix */
	UFUNCTION(BlueprintCallable)
	void SetWidgetFixPosition(FVector Point, double Percent);

	/** Function set position WighetFix to default */
	UFUNCTION(BlueprintCallable)
```

```
	void ClearWidgetFixPosition();

	/** Function character death */
	UFUNCTION(BlueprintCallable)
	void LocalDeath();

	/** Function attachActor to chracter */
	UFUNCTION(BlueprintCallable)
	void AttachActor(ABP_AttachableItem_C* ItemToAttach, bool& Success);

	/** Function set location to InteractWidget */
	UFUNCTION(BlueprintCallable)
	void UpdateInteractWidget(TScriptInterface<IInterface> InteractableActor, bool
Detected, FVector ItemLocation, FVector ItemLocation);

	/** Function charcater drop oxygen tank */
	UFUNCTION(BlueprintCallable)
	void Drop Oxygen Tank();

	/** Function get Oxygen Tank reference */
	UFUNCTION(BlueprintPure)
	void GetOxygenTank(ABP_OxTank_C*& OxygenTank);

	/** Function close pause menu */
	UFUNCTION(BlueprintCallable)
	void DisablePause();

	/** Function open pause menu */
	UFUNCTION(BlueprintCallable)
	void Enable Pause();

	/** Function set character to player in widget */
	UFUNCTION(BlueprintCallable)
	void SetHoweredWidgetPlayer();

	/** Function return is first player controller possessed on tish pawn */
	UFUNCTION(BlueprintPure)
	bool IsFirstPlayer();

	/** Function copy players array, but exept one */
	UFUNCTION(BlueprintCallable, Category="Default")
	void Copy Players Array Without One(UPARAM(ref) TArray<ABP_PlayerCharacter_C*>&
Array, ABP_PlayerCharacter_C* PlayerToSkip, TArray<ABP_PlayerCharacter_C*>& Players,
TArray<ABP_PlayerCharacter_C*> Copy);
```

```
	/** Function set class characteristic to character*/
	UFUNCTION(BlueprintCallable, Category="Default")
	void Apply Class();
public:
	/* Blocker Blueprint Component */
	UPROPERTY(BlueprintReadOnly, VisibleAnywhere, Category="Default")
	TObjectPtr<UBPC_BlockerStack_C> DropBlockerStack;

	/** AudioComponent to right steps */
	UPROPERTY(BlueprintReadOnly, VisibleAnywhere, Category="Default")
	TObjectPtr<UAudioComponent> CUE_Steps_Right;

	/** AudioComponent to left steps */
	UPROPERTY(BlueprintReadOnly, VisibleAnywhere, Category="Default")
	TObjectPtr<UAudioComponent> CUE_Steps_Left;

	/** LightComponent to glow in the dark */
	UPROPERTY(BlueprintReadOnly, VisibleAnywhere, Category="Default")
	TObjectPtr<UPointLightComponent> PointLight;

	/** Blueprint Component to Input */
	UPROPERTY(BlueprintReadOnly, VisibleAnywhere, Category="Default")
	TObjectPtr<UBPC_Input_C> BPC_Input;

	/** Blueprint Component to Attack Tokens */
	UPROPERTY(BlueprintReadOnly, VisibleAnywhere, Category="Default")
	TObjectPtr<UBPC_AttackTokensSystem_C> Attack Tokens System;

	/** WidgetFix to show progress while fix */
	UPROPERTY(BlueprintReadOnly, VisibleAnywhere, Category="Default")
	TObjectPtr<UWidgetComponent> WidgetFix;

	/** Get Local Player */
	UPROPERTY(BlueprintReadOnly, VisibleAnywhere, Category="Default")
	TObjectPtr<UGetLocalPlayer> GetLocalPlayer;

	/** Character camera */
	UPROPERTY(BlueprintReadOnly, VisibleAnywhere, Category="Default")
	TObjectPtr<UCameraComponent> FirstPersonCamera;

	/** Blueprint Component to Stats */
	UPROPERTY(BlueprintReadOnly, VisibleAnywhere, Category="Default")
	TObjectPtr<UBPC_Player_StatsSystem_C> BPC_Player_StatsSystem;
```

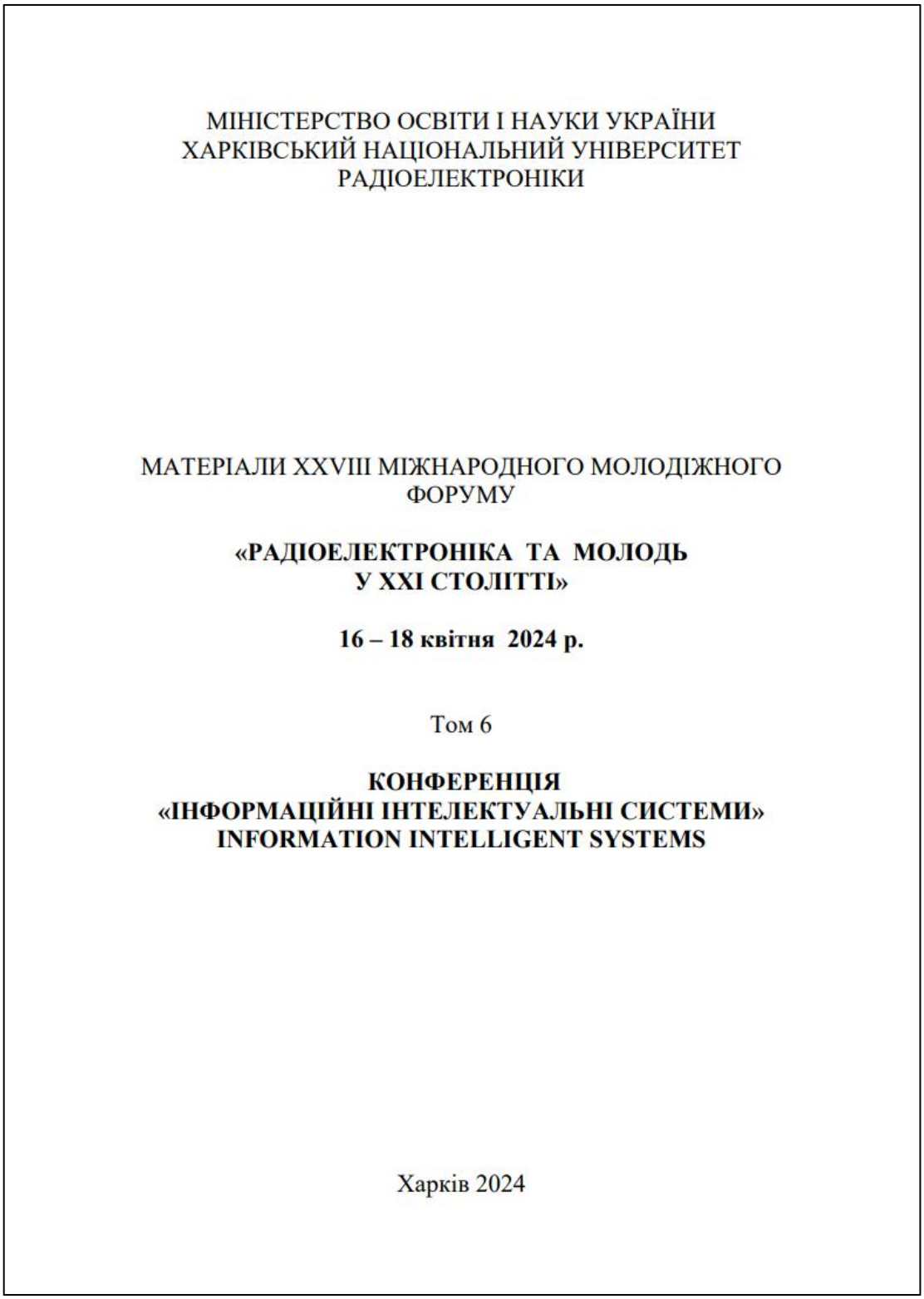
```
	/** Component to interact with widget on level */
	UPROPERTY(BlueprintReadOnly, VisibleAnywhere, Category="Default")
	TObjectPtr<UWidgetInteractionComponent> WidgetInteraction;

	/** Widget to show interact button on items */
	UPROPERTY(BlueprintReadOnly, VisibleAnywhere, Category="Default")
	TObjectPtr<UWidgetComponent> WidgetInteract;

	/** Blueprint Component to menage player health */
	UPROPERTY(BlueprintReadOnly, VisibleAnywhere, Category="Default")
	TObjectPtr<UBPC_HealthSystem_C> BPC_HealthSystem;

	/** Blueprint component to Grab item */
	UPROPERTY(BlueprintReadOnly, VisibleAnywhere, Category="Default")
	TObjectPtr<UBP_Grab_Component_C> BP_Grab_Component;

	/** Property to create timelines */
	UPROPERTY(BlueprintReadOnly, VisibleAnywhere, Category="BP_PlayerCharacter")
	TObjectPtr<UTimelineComponent> Timeline;

	/** Reference property to grabbed item */
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Item")
	TObjectPtr<ABP_GrabbableItem_C> Grabbed Item;

	/** Bool property is character hold item */
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Item")
	bool bHoldsItem;

	/** Reference property to the item the character is looking at */
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Item")
	TObjectPtr<ABP_InteractiveItem_C> Grabb Item In Focus;
private:
	/** Property to animation of character whit different items in hand */
	UPROPERTY(EditDefaultsOnly, Category="Animations")
	TEnumAsByte<E_CharacterOverlay> CurrentOverlay;
public:
	/** Delegate on overlay change */
	DECLARE_DYNAMIC_MULTICAST_DELEGATE_OneParam(FOnOverlayChange,
TEnumAsByte<E_CharacterOverlay>, NewOverlay);
	UPROPERTY(BlueprintAssignable, EditDefaultsOnly, Category="Default")
	FOnOverlayChange OnOverlayChange;
private:
	/** Property to looks up on recoil */
```

```
	UPROPERTY(EditDefaultsOnly, Category="Recoil")
	double BaseLookUpRate;

	/** Property to looks right or left on recoil*/
	UPROPERTY(EditDefaultsOnly, Category="Recoil")
	double BaseTurnRate;
public:
	/** WidgetFix class*/
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="UI")
	TObjectPtr<UUI_FixStatus_C> UI Fix Status;

	/** Timer to enable input on stun damage */
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Default")
	FTimerHandle EnableInputAfterStunTimer;

	/** Player controller */
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Default")
	TObjectPtr<APlayerController> Player Controller;

	/** Delegate on character death */
	DECLARE_DYNAMIC_MULTICAST_DELEGATE_OneParam(FonDeath, ABP_PlayerCharacter_C*,
Player);
	UPROPERTY(BlueprintAssignable, EditDefaultsOnly, Category="Default")
	FonDeath onDeath;
private:
	/** Reference to oxygen tank the character is holding at */
	UPROPERTY(EditDefaultsOnly, Category="Oxygen")
	TObjectPtr<ABP_OxTank_C> OxygenTank;
public:
	/** Reference to widget of pause menu */
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="UI")
	TObjectPtr<UUI_Pause_C> Pause;

	/** Property player class */
	UPROPERTY(BlueprintReadWrite, EditAnywhere, Category="Default",
meta=(ExposeOnSpawn="true"))
	TEnumAsByte<EPlayerClass> PlayerClass;

	/** Property player Id for online multiplayer mode */
	UPROPERTY(BlueprintReadWrite, EditDefaultsOnly, Category="Default")
	int32 PlayerNetworkID;

	/** Delegate on player Disconnect for online multiplayer */
```

```
	DECLARE_DYNAMIC_MULTICAST_DELEGATE_OneParam(FonDisconect,
ABP_PlayerCharacter_C*, Player);
	UPROPERTY(BlueprintAssignable, EditDefaultsOnly, Category="Default")
	FonDisconect onDisconect;
};
```

ДОДАТОК Д

Тези доповіді для науково-практичної інтернет-конференції



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
РАДІОЕЛЕКТРОНІКИ

МАТЕРІАЛИ XXVIII МІЖНАРОДНОГО МОЛОДІЖНОГО ФОРУМУ

«РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ У XXI СТОЛІТТІ»

16 – 18 квітня 2024 р.

Том 6

КОНФЕРЕНЦІЯ
«ІНФОРМАЦІЙНІ ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ»
INFORMATION INTELLIGENT SYSTEMS

Харків 2024

Рисунок Д.1 – Обкладинка збірника

Рисунок Д.2 – Алфавітний покажчик збірника

УДК 004.514:004.946

ОПТИМІЗАЦІЯ СЕРВЕРНИХ РІШЕНЬ В UNREAL ENGINE 5

Яковенко Д. О.
Науковий керівник – ст. викл. Новіков Ю. С.
Харківський національний університет радіоелектроніки
61166, Харків, просп. Науки, 14, каф. ПІ,
e-mail: dmytro.iakovenko1@nure.ua

This paper explores the process of server optimization on the Unreal Engine 5 platform. The article covers strategies and methods aimed at improving the performance of server systems in the context of using this game engine. In particular, techniques for optimizing network interaction, data processing, and resource-intensive operations are considered. Practical examples of implementing solutions to improve performance and ensure stable server operation during the development of large game projects on Unreal Engine 5 are considered. This paper provides valuable guidance and recommendations for developers who seek to optimize server performance on this platform.

Unreal Engine 5 (UE5) – це потужний інструментарій, який розкриває безмежні можливості для розробки ігор нового покоління. Він пропонує широкий спектр інструментів, що робить його ідеальним вибором як для створення одиночних, так і для багатокористувацьких проектів.

Одне з ключових особливостей UE5 – це революційна система реплікації, яка забезпечує безперебійну синхронізацію даних між клієнтами в реальному часі. Завдяки цьому, UE5 гарантує плавний ігровий процес без затримок та лагів.

Для оптимізації серверних рішень в UE5, необхідно зрозуміти одне: чим менше даних треба реплікувати, тим краще. Тому уся оптимізаця побудована на ігноруванні непотрібних для реплікації даних, та зменшення об’єму реплікованих даних.

Для початку розглянемо випадок, як можна зменшити об’єм реплікованих даних на прикладі обертання об’єктів у просторі. В UE5 є два варіанти точності реплікації обертання у просторі: Byte(8біт) Та Short(16 біт). В рамках розроблюваного проекту, є 4 актора, які мають обертатися, та синхронізувати своє полодження між клієнтами. Результати тестування цих акторів в обох режимах представлено у таблиці 1.

Таблиця 1 – тестування різних режимів передачі полодження в просторі

	Short		Byte	
Об'єкти	Avg Bytes	Avg ms	Avg Bytes	Avg ms
1	14,3	0,009	11,2	0,007
2	12,1	0,009	8,7	0,007
3	11,9	0,009	8,7	0,007
4	1,5	0,009	1,5	0,007

Рисунок Д.3 – Матеріал тез (сторінка 1)

Проведене дослідження свідчить про те, що синхронізація положення Short потребує на 25-30% більше часу, порівняно з Byte. З огляду на цю характеристику, Byte виступає більш економним рішенням у більшості випадків. Однак, важливо зазначити, що Byte має вдвічі меншу точність, ніж Short. В деяких ситуаціях це може призвести до візуального покращення плавності анімації обертання.

В рамках розроблюваного проекту, зброя має механізм перегрівання, синхронізація якого між клієнтами є необхідною. Зважаючи на динамічний характер даної характеристики, що змінюється кожен ігровий такт, просте реплікування змінних стає нерентабельним (див. рис. 1).

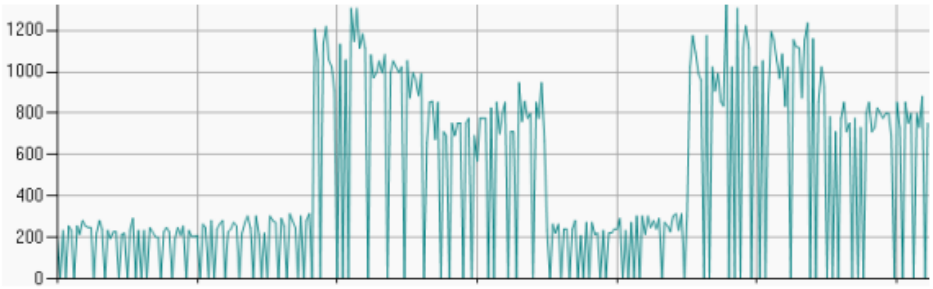


Рисунок 1 – Затрати, які спричиняє використання зброї до оптимізації

Як видно на рисунку 1, при використанні зброї, кожен такт сервер передає значення нагріву зброї, що навантажує його на додаткові 250 Байтів на секунду. Цього можна уникнути, якщо передавати нагрівання зброї тільки в момент коли гравець починає або закінчує її використання.

Оскільки інформація про перегрівання зброї необхідна лише в момент її використання, та вона необхідна лише тому клієнту, що її використовує, то можна перенести розрахунок перегріву з сервера, на клієнт, що тримає цю зброю у руках, а у момент, коли ніхто не тримає зброю, обчисленням нагрівання займеться сервер.

Передачу інформації про поточний стан від клієнта до сервера зробимо в момент зупинки використання зброї клієнтом, а передачу інформації від сервера до клієнта, в момент початку використання зброї клієнтом.

Таким чином, можна знизити навантаження на сервер, та на мережу, що можна побачити на рисунку 2.

Рисунок Д.4 – Матеріал тез (сторінка 2)