НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___» _____ 2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

спеціальності 121 Інженерія програмного забезпечення

на тему: «Програмне забезпечення для симуляції віртуальної багатокористувацької рольової взаємодії»

Виконав:
студент IV курсу, групи КП-93
Деркач Станіслав Дмитрович _____

Керівник:
Доцент кафедри ПЗКС, к. ф.-м. н, доцент,
Нещадим Олександр Михайлович _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Доцент кафедри СПіСКС, к.т.н.,
Боярінова Юлія Євгеніївна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2023 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___»_____2022 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Деркач Станіслав Дмитрович

1. Тема проєкту Програмне забезпечення для симуляції віртуальної багатокористувацької рольової взаємодії, керівник проєкту Нещадим Олександр Михайлович, доцент, к.ф.м.н, затверджені наказом по університету від «31» травня 2023 р. № 2107-с

2. Термін подання студентом проєкту «16» червня 2023 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - аналіз галузі;
   - аналіз платформи та технологій розробки;
   - архітектура розробленого продукту;
   - аналіз розробленого застосунку.

5. Перелік обов’язкового графічного матеріалу:
- рівень авторизації (креслення);
- рівень головного меню (креслення);
- система автоматизованого налаштування серверів (плакат);
- інфраструктура для передачі чутливих даних (плакат).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2022 р.

Календарний план

№ з/п	'Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	09.11.2022	
2.	Розроблення та узгодження технічного завдання	26.11.2022	
3.	Розроблення структури програми	15.12.2022	
4.	Підготовка матеріалів першого розділу дипломного проєкту	30.12.2022	
5.	Розроблення дизайну, псевдокоду алгоритмів.	05.02.2023	
6.	Підготовка матеріалів другого розділу дипломного проєкту	18.02.2023	
7.	Програмна реалізація	10.03.2023	
8.	Тестування програми	17.03.2023	
9.	Підготовка матеріалів третього розділу проєкту	30.03.2023	
10.	Підготовка графічної частини дипломного проєкту	21.04.2023	
11.	Оформлення документації дипломного проєкту	28.05.2023	

Студент Станіслав ДЕРКАЧ

Керівник проєкту Олександр НЕЩАДИМ

АНОТАЦІЯ

Даний дипломний проєкт присвячений розробленню програмного забезпечення для симуляції віртуальної багатокористувацької рольової взаємодії.

У роботі виконано порівняльний аналіз існуючих рішень для симуляції віртуальної багатокористувацької рольової взаємодії, проаналізовано інструменти розроблення подібних симуляцій, такі як ігрові двигуни та мови програмування, види зв'язку сервера та бази даних, систем управління базами даних та методи збереження безпеки особистих даних. На основі аналізу обґрунтовується вибір відповідних технологій під час розробки.

Розроблений застосунок надає користувачам можливість приєднатися до симуляції, обрати свою роль та взаємодіяти з ботами, іншими користувачами, об'єднуватись в коаліції, виконувати кооперативні завдання, змагатися один з одним та впливати своїми діями на оточуючий світ.

Даний проєкт дозволяє задовольнити соціальні потреби користувачів, за допомогою надання можливості об'єднуватись в команди з іншими користувачами, будувати сумісні плани, змагатися з іншими користувачами та обмінюватись досвідом між собою.

В даному проєкті розроблено та досліджено: архітектуру серверної та клієнтської частини симуляції, сервер-абстракцію над базою даних, асиметричні алгоритми шифрування для безпечної передачі особистих даних між клієнтом та сервером, симетричні алгоритми шифрування для безпечного зберігання особистих даних в базі даних.

ABSTRACT

This diploma project is dedicated to the development of software for simulating virtual multi-user role-playing interaction.

The work includes a comparative analysis of existing solutions for simulating virtual multi-user role-playing interaction, analysis of tools for developing suchlike simulations, such as game engines and programming languages, types of server and database communication, database management systems, and methods for maintaining the security of personal data. The analysis is used to justify the choice of appropriate technologies during development.

The developed application provides users with the opportunity to join the simulation, choose their role and interact with bots or other users, form coalitions, perform cooperative tasks, compete with each other and influence the world around them with their actions.

This project allows us to meet the social needs of users by providing the opportunity to form teams with other users, build joint plans, compete with other users and share experiences with each other.

In this project, next things were developed and researched: the architecture of the server and client parts of the simulation, database abstraction layer), asymmetric encryption algorithms for secure transmission of personal data between the client and the server, symmetric encryption algorithms for secure storage of personal data in the database.

ДП.045480-01-90 Програмне забезпечення для симуляції віртуальної багатокористувацької рольової взаємодії. Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045480-02-91	Програмне забезпечення для	7	
	симуляції віртуальної		
	багатокористувацької		
	рольової взаємодії. Технічне		
	завдання		
ДП.045480-03-81	Програмне забезпечення для	62	
	симуляції віртуальної		
	багатокористувацької		
	рольової взаємодії.		
	Пояснювальна записка		
ДП.045480-04-51	Програмне забезпечення для	4	
	симуляції віртуальної		
	багатокористувацької		
	рольової взаємодії. Програма		
	та методика тестування		
ДП.045480-05-34	Програмне забезпечення для	12	
	симуляції віртуальної		
	багатокористувацької		
	рольової взаємодії.		
	Керівництво користувача		
ДП.045480-06-99	Програмне забезпечення для	1	
	симуляції віртуальної		
	багатокористувацької		
	рольової взаємодії. Рівень		
	авторизації. UML-діаграма		
	класів		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045480-07-99	Програмне забезпечення для	1	
	симуляції віртуальної		
	багатокористувацької		
	рольової взаємодії. Рівень		
	головного меню.		
	UML-діаграма класів		
ДП.045480-08-98	Програмне забезпечення для	1	
	симуляції віртуальної		
	багатокористувацької		
	рольової взаємодії.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2022 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ СИМУЛЯЦІЇ ВІРТУАЛЬНОЇ БАГАТОКОРИСТУВАЦЬКОЇ РОЛЬОВОЇ ВЗАЄМОДІЇ

Технічне завдання

ДП.045430-02-91

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Олександр НЕЩАДИМ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Станіслав ДЕРКАЧ

2022

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: програмне забезпечення для симуляції віртуальної багатокористувацької рольової взаємодії.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проєктування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання з метою симуляції різних соціальних рольових взаємодій.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Програмний продукт має підтримувати дві ролі: Гість, Гравець.

Програмний продукт забезпечує наступні функції для ролі Гість:

1. Реєстрація нового Гравця.
2. Авторизація.

Програмний продукт забезпечує наступні функції для ролі Гравець перед долученням до гри:

1. Створення нового героя.
2. Вибір героя.
3. Видалення героя.
4. Долучення до гри.

Програмний продукт забезпечує наступний функціонал для ролі

Гравець після долучення до гри:

1. Приєднання до сервера з іншими гравцями.
2. Керування персонажем:
   - пересування персонажа;
   - керування камерою;
   - взаємодія з іншими об'єктами всесвіту;
   - керування віджетами.
3. Підтримка міні-карти та карти всесвіту:
   - відображення приблизного рельєфу;
   - відображення поточного положення гравця;
   - відображення ботів та їх типів;
   - відображення магазинів;
   - відображення інших гравців;
   - можливість масштабування міні-карти.
4. Керування інвентарем:
   - огляд власних предметів;
   - огляд іншого сховища;
   - переміщення предметів між сховищ;
   - позбавлення від предмету;
   - встановлення предмета в екіпіровку;
   - перегляд поточного власного балансу ігрової валюти;
   - перегляд детальної інформації предмета.
5. Користування магазинами:
   - купівля предметів;
   - продаж предметів;
   - вибір кількості предметів для купівлі/продажу.
6. Керування екіпіровкою:
   - слоти для екіпіровки наступних типів: зброя, шолом, перчатки, нагрудник, аксесуари;
   - вплив екіпіровкою на характеристики героя;

- можливість одягнути та зняти екіпіровку;
- валідація предмету та слоту.

7. Спілкування з деякими ботами:
   - початок діалогу;
   - вибір варіанту відповіді із запропонованих;
   - вплив обраної відповіді на подальший хід діалогу;
   - вплив ходу діалогу на поведінку інших ботів;
   - можливість отримати предмет через діалог;
   - можливість отримати гроші через діалог;
   - можливість отримати квест через діалог.
8. Користування квестовою системою:
   - перегляд поточних квестів та їх прогресу;
   - кожен квест має складатись із декількох підквестів;
   - підтримка підквестів наступних видів: знаходження або позбавлення предмета, вбивство заданої кількості ботів заданого типу, пошук особливої інформації;
   - нагородження після завершення квесту.
9. Перегляд власних поточних характеристик:
   - запас здоров'я;
   - рівень герою та його прогрес;
   - атака;
   - магічна могутність;
   - швидкість атаки;
   - точність;
   - магічна точність;
   - шанс критичного удару;
   - сила критичного удару;
   - придкість.

Системні вимоги:

1. Розробку виконати на мові програмування C++ та Blueprints з використанням Unreal Ungine версії 4 або вище.
2. Клієнт гри має працювати на операційній системі Windows із мінімальним апаратним забезпеченням: 8 гб оперативної пам'яті, процесор intel core i5, графічним процессором GeForce 1050 ti.
3. Зв'язок з базою даних мовою SQL.

Додаткові вимоги:

1. Додаткове шифрування даних для запобігання втрати даних.
2. Гнучка архітектура для подальшого масштабування проєкту.
3. Обведення інтерактивних об'єктів.
4. Слоти для швидкого використання предметів.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проєкту повинна бути розроблена наступна документація:

1. Пояснювальна записка.
2. Програма та методика тестування.
3. Керівництво користувача.
4. Креслення:
   - «Рівень авторизації. UML-діаграма класів».
   - «Рівень головного меню. UML-діаграма класів».

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою роботи.....16.11.2022

Розроблення та узгодження технічного завдання.....27.11.2022

Розроблення структури додатка.....15.12.2022

Розроблення дизайну графічних елементів.....03.02.2023

Програмна реалізація системи.....15.03.2023

Тестування системи.....07.04.2023

Підготовка матеріалів текстової частини проєкту.....28.04.2023

Підготовка матеріалів графічної частини проєкту.....12.05.2023

Оформлення технічної документації проєкту.....25.05.2023

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до «Програми та методики тестування».

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___» _____ 2023 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ СИМУЛЯЦІЇ ВІРТУАЛЬНОЇ БАГАТОКОРИСТУВАЦЬКОЇ РОЛЬОВОЇ ВЗАЄМОДІЇ

Пояснювальна записка

ДП.045480-03-81

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олександр НЕЩАДИМ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Станіслав ДЕРКАЧ

2023

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

RPG (role play game) – жанр відеоігор, де гравці приймають ролі фіктивних персонажів та приймають рішення від імені цього герою.

MMORPG – тип RPG, який дозволяє великій кількості гравців взаємодіяти між собою у віртуальному світі.

UE (unreal engine) – ігровий рушій, створений компанією Epic Games на базі мови C++.

PVP (player vs player) – режим геймплею, де гравці змагаються проти один одного.

PVE (player vs environment) – режим геймплею, де гравці співпрацюють, щоб протистояти згенерованим перешкодам.

MMO (massively multiplayer online) – жанр ігр, який дозволяє багатьом гравцям одночасно взаємодіяти у віртуальному світі онлайн.

Down Casting – процес перетворення об'єкту до типу, який є нижчим ніж початковий за ієрархією наслідування.

Ігрова студія – компанія або команда, яка спеціалізується на розробці відеоігор.

Полінаслідування – концепція в програмуванні, коли клас може успадковувати властивості та методи від декількох класів одразу.

БД (база даних) – система організації та зберігання даних, яка дозволяє ефективно управляти інформацією.

Інстанс – окремий екземпляр класу в програмуванні.

Персонажний об'єкт – об'єкт, яким можливо керувати та який може поводиться як персонаж.

САБ – сервер-абстракція над базою даних.

СУБД (система управління базами даних) – спеціалізоване програмне забезпечення, що дозволяє організувати, зберігати та управляти базами даних.

ПК (персональний комп'ютер) – електронний пристрій, призначений для особистого використання, який здатний виконувати різноманітні обчислення та завдання.

Факторизація – Математична операція, яка полягає у розкладі числа або виразу на множники.

Чит – у контексті відеоігор, це термін, що використовується для опису нелегального або не дозволеного до використання програмного забезпечення з метою отримання переваги у грі.

Спавн – місце у відеогрі, де гравець або об'єкти з'являються або сам процес появлення там.

Віджет – елемент графічного інтерфейсу, що використовується для відображення певної інформації на екрані.

Фрейм – одна ітерація ігрового циклу.

Інсталяція – процес встановлення або налаштування програмного забезпечення на комп'ютер або інший пристрій для його використання.

Лут – у відеоіграх, це відноситься до предметів або нагород, які гравець отримує після успішного виконання завдання, перемоги над ворогами або знаходження в спеціальних місцях гри.

Слот – відведений простір або місце, яке призначене для збереження або розміщення певних об'єктів, таких як предмети інвентаря, зброя, спеціальні вміння тощо.

ВСТУП

Нестача соціальної взаємодії завжди був значною проблемою в житті людини. Люди стикаються з нею з різних причин: переїзд, хвороба, тощо. За останні кілька років ця проблема стала ще більш актуальною через низку подій, таких як карантин та війна в Україні. Багато людей почули себе відокремленими та втратили можливість активно взаємодіяти з іншими людьми. Війна також спричинила розлуку і роз'єднання багатьох родин та друзів, що поглибило проблему недостатньої соціального взаємодії.

Нестача соціального взаємодії має негативні наслідки для людини, її продуктивності та якості життя в цілому. Також, якщо соціальні потреби залишаються незадовільними достатньо довгий час, це може мати наслідки на здоров'ї.

Одна з найважливіших соціальних потреб людини – це почуття спільноти з іншими. Ми, як соціальні істоти, маємо потребу в спілкуванні, співпраці та взаємодії з оточуючими нас людьми. Цю потребу можна задовольнити за допомогою багатокористувацької системи, яка ставить перед користувачами перешкоди, спонукаючи їх до кооперації задля вирішення спільних завдань. Така система є основою для ігр у жанрі масових багатокористувацьких онлайн-рольових ігор (ММОРПГ), де гравці змушені співпрацювати та взаємодіяти між собою для досягнення спільних цілей.

Метою цієї дипломної роботи є аналіз жанру ММОРПГ, розробка системи, яка реалізує керування користувачами системи, безпечну передачу їх особистих даними та базові механіки ММОРПГ, тестування та аналіз розробленого застосунку.

1. АНАЛІЗ ГАЛУЗІ

1.1. Загальний аналіз ігрової галузі

Потрібність у іграх з'явилась у той самий час, коли широкому споживачеві стали доступні перші обчислювальні машини. У ті часи програмісти створювали ігри у текстовій консолі. З технологічним прогресом та збільшенням складності комп'ютерів, ця потреба не зникла. Про це свідчить кількість товарів та фінансів в обороті у цій галузі [1]. А аналіз галузі від Statista за 2021 рік показав, що ці обсяги будуть лише збільшуватись [2].

Геймінгова індустрія охоплює велику кількість жанрів. Різні жанри задовольняють різні потреби споживачів: від додатків, які дозволяють просто розважатись однотипними, але яскравими діями, до систем, які дозволяють їх користувачам реалізовувати свої стратегічні та управлінські таланти. І всі жанри у цьому спектрі користуються попитом.

Жанр ММОРПГ базується на створенні світу, який дозволяє користувачам зіткнутися з перешкодами, що вимагають від гравців спільної роботи. Цей жанр є одним з тих, що здатні задовольняти комплексні та складні соціальні потреби гравців.

1.2. Загальний огляд жанру ММОРПГ

ММОРПГ (Massively Multiplayer Online Role-Playing Game) – це надзвичайно популярний жанр комп'ютерних ігор [3], у якому гравці взаємодіють у віртуальному світі, виконують ролі своїх персонажів та взаємодіють з іншими гравцями спільного віртуального світу. Жанр ММОРПГ поєднує в собі елементи рольових ігор (RPG) та масових онлайн-ігор (ММО).

У цьому жанрі гравці можуть створювати героїв, визначати їхні атрибути, вигляд, вибирати класи або професії та розвивати їхні навички та вміння під час гри. Вони можуть виконувати квести та досліджувати світ, прогресуючи у грі та отримуючи нові рівні, предмети та нагороди. Гравець

має можливість обрати власний стиль гри: бути одиноким дослідником чи захисником свого клану; прагнути миру або чинити хаос.

Основною особливістю ігор цього жанру є можливість взаємодії з іншими гравцями у віртуальному світі. Гравці можуть спілкуватися, формувати команди або гільдії, спільно виконувати завдання та битися з монстрами чи іншими гравцями. У цьому жанрі висока комунікація з іншими гравцями, тому там можна легко знайти собі нових друзів, що є важливим та проблемним аспектом сьогодення. Деякі ММОРПГ можуть запропонувати битви гравець проти гравця, або ж великі масові події та економічні аспекти гри, такі як торгівля предметами та ресурсами.

1.3. Відмінності представників жанру ММОРПГ

Цей жанр ігор пропонує глибоку систему розвитку персонажа, де гравці можуть вибирати класи, навички, атрибути та вдосконалювати їх під час гри. Цей прогрес дозволяє гравцям відчувати постійне досягнення цілей та розвиток свого персонажа.

ММОРПГ зазвичай пропонують обширні та деталізовані віртуальні світи, які гравці можуть досліджувати. Ці світи часто мають різноманітні локації, міста, дикі землі та підземелля, що запрошують гравців відкривати нові області та випробовувати свої навички.

Багато ігор цього жанру пропонують регулярне оновлення та додавання нового контенту, такого як квести, розширення, нові предмети або режими гри. Це дозволяє гравцям постійно вирішувати нові виклики та завдання, уникати монотонності та зберігати інтерес до гри впродовж тривалого часу.

Також, перевагою жанру є можливість гравцям грати разом зі своїми друзями або зустрічати нових людей у віртуальному світі. Великий фокус на соціальній взаємодії створює можливість для спільних пригод, співпраці та суворих битв, що підвищує захоплення грою та стимулює гравців до тривалої участі.

Деякі ММОРПГ включають економічні системи, в яких гравці можуть торгувати предметами, ресурсами та послугами з іншими гравцями. Це може створювати складні економічні взаємини та стимулювати гравців до розвитку економічних стратегій та імітації виробництва.

Багато, проте не всі, ігри цього жанру пропонують епічні події, такі як великі битви, вторгнення монстрів або осади фортець, в яких беруть участь сотні або навіть тисячі гравців одночасно. Крім того, багато ММОРПГ мають PvP режими, де гравці можуть змагатися один з одним у битвах або конкурувати за ресурси та території.

Економіка в ММОРПГ може варіюватися від гри до гри. Деякі ігри мають складну економічну систему з торгівлею гравців та виготовленням предметів. Інші можуть мати простішу систему, де гравці отримують предмети через квести або ворожіння. Економічна система може впливати на гру і соціальну взаємодію гравців.

Різні ігри цього жанру можуть мати різні системи бою. Деякі ігри використовують активний бій в реальному часі, де гравці повинні активно керувати своїми персонажами під час бою. Інші ігри можуть мати більш традиційну систему хід за ходом або комбінацію обох підходів. Кожна гра може мати свої унікальні механіки бою. Вони можуть вимагати використання навичок, стратегій або реакцій гравців.

1.4. Висновок щодо актуальності галузі

Жанр ММОРПГ залишається актуальним вже довгий час, створюючи віртуальні світи, де гравці можуть поглибитися на десятки, а іноді й сотні годин геймплею, взаємодіяти з іншими гравцями та відчути себе частиною великої спільноти.

Серед популярних представників жанру можна виділити такі ігри: World of Warcraft, Final Fantasy XIV, Guild Wars 2, The Elder Scrolls Online, Black Desert Online та інші. Жанр ММОРПГ забезпечує якісне дозвілля

завдяки тому, що задовольняє соціальні потреби, дозволяє приміряти на себе незвичайні ролі, експериментувати зі стилем гри і долати різноманітні випробування.

1.5. Аналіз існуючих рішень

1.5.1. *Гра «World of Warcraft»*

Для аналізу жанру та реалізованих у ньому рішень, гарно підійде гра "World of Warcraft" [4]. Це, безумовно, найвідоміший та найуспішніший проєкт цього жанру, згідно з кількістю користувачів (як зареєстрованих, так і активних).

Гра пропонує широкий вибір класів та рас, що дає можливість створити унікального персонажа. Кожен клас має свої особливі навички, стиль гри та роль у команді. Раси також мають свої особливі розуміння світу та унікальні характеристики, що додає глибини в створенні персонажів.

Гравцям доступно багато завдань, таких як: квести, рейди, підземелля та експедиції. Гравці можуть досліджувати великі світи, виконувати різноманітні завдання, битися з монстрами та босами, отримувати нагороди та розвивати свій персонаж. Рейди є особливо складними випробуваннями, де команди гравців спільно працюють, щоб подолати потужних противників. Це все можна віднести до PvE контенту.
У грі також присутній PvP контент, де гравці можуть боротися один з одним. Це включає битви на аренах, полях бою та відкритих бойових зонах. Гравці змагаються в одиночному чи груповому PvP, показуючи свої навички бойової майстерності та стратегії.

World of Warcraft має розвинену систему професій, де гравці можуть вибрати ремесло та створювати різні предмети. Професії можуть бути поділені на дві категорії: збирання ресурсів та виробництво предметів. Гравці можуть отримувати рецепти, розробляти свої навички та створювати корисні предмети для себе або для продажу на ринку.

WoW пропонує різні можливості для соціальної взаємодії між гравцями. Гравці можуть об'єднуватися в гільдії, щоб спільно розкривати світ гри, виконувати великі завдання та підтримувати командний дух. Також існують місця збору, такі як міста і таверни, де гравці можуть спілкуватися, торгувати та зустрічатися з іншими гравцями.

Також гра має систему покращення персонажа, де гравці можуть збирати досвід і отримувати нові рівні, що відкривають доступ до нових навичок, умінь та обладунків. Гравці можуть спеціалізуватися в різних напрямках, вдосконалюючи свої здібності у вибраних класах та розширюючи свій стиль гри.

Реалізація проєкту має яскравий та кольоровий світ з деталізованою графікою та вражаючим дизайном. Кожна локація має свою унікальну атмосферу, що дозволяє гравцям поглинутися в ігровий світ і відчути його магію.

Загалом, World of Warcraft пропонує різноманітні рішення та функції, які дозволяють гравцям насолоджуватися багатогранністю ігрового досвіду. Доступні класи та раси дозволяють гравцям створювати унікальних персонажів, а PvE та PvP контент надають можливість розкрити свої навички бойових сутичок як індивідуально, так і в команді. Розвиток персонажа і система професій дозволяють гравцям гнучко налаштувати свій персонаж і створювати корисні предмети.

Однією з сильних сторін цієї гри та жару в цілому є соціальна взаємодія. Гравці мають змогу об'єднуватися в гільдії, де вони можуть спільно розкривати світ гри, виконувати складні завдання і підтримувати командний дух. Місця збору, такі як міста і таверни, створюють можливості для спілкування, торгівлі та взаємодії з іншими гравцями.

1.5.2. *Гра «The Elder Scrolls Online»*

Другим об'єктом порівняння буде The Elder Scrolls Online – це гра того ж жанру, що базується на популярній серії рольових ігор "The Elder Scrolls" [5].

Безумовною перевагою є відкритий світ та історія, адже гра пропонує користувачам великий відкритий світ, який можна вільно досліджувати. Гра базується на багатообіцяючих локаціях, таких як провінції Тамріеля, де гравці можуть подорожувати через різноманітні ландшафти, міста, селища та дику природу. Кожна провінція має свою унікальну історію, персонажів та завдання, що додає глибину захоплення процесом.

The Elder Scrolls Online пропонує різні класи персонажів, включаючи драконорідних, вояків, чарівників, некромантів та інших. Кожен клас має свої особливі навички та можливості, які гравці можуть розвивати під час гри. Крім того, гравці можуть спеціалізуватися в різних навичках, таких як бойові мистецтва, магія, лікування, крадіжки та багато інших.

Також гра пропонує розмаїття PvE контенту для гравців. Це включає квести з унікальними сюжетними лініями, підземелля, рейди та світові боси. Квести розповідають захоплюючі історії та дають можливість гравцям впливати на події у світі гри. Рейди і підземелля представляють складні випробування, які можна виконувати в групі разом з іншими гравцями. Вони вимагають координації, стратегії та співпраці, щоб перемогти потужних ворогів і отримати цінні нагороди. Світові боси – це великі монстри, які з'являються від часу до часу в різних локаціях світу гри і вимагають міцного командного зусилля для перемоги над ними.

Не обходиться й без PvP контенту, де гравці можуть змагатися один з одним. Це включає бойові зони, де гравці можуть зустрічатися і боротися, а також війни альянсів, де групи гравців з різних фракцій змагаються за контроль над територіями. Бої у грі можуть бути інтенсивними та вимагати від гравців вміння використовувати свої навички в битвах проти інших гравців.

Гра має унікальну систему професій та крафтингу, яка дозволяє гравцям створювати різні предмети і розвивати свої навички. Гравці можуть вибрати ремесло, таке як ковальство, шкіряні ремесла, алхімія та інші, й виробляти предмети, збільшувати свій рівень майстерності та створювати унікальні предмети з різними властивостями.
Є також можливості для соціальної взаємодії між гравцями. Проте вони не відрізняються від описаних у попередній грі.

Узагальнюючи, це гра з різноманітним ігровим контентом, який пропонує гравцям багато можливостей для розвитку свого персонажа та взаємодії з іншими гравцями. Незалежно від того, чи ви хочете досліджувати світ гри, виконувати квести, битися з ворогами або взаємодіяти з іншими гравцями. Для кожного є щось своє.

1.5.3. *Гра «Guild Wars 2»*

Також проаналізуємо гру цього жанру "Guild Wars 2". Гра відображає світ фантастичної Тіриї, який насичений пригодами, битвами та соціальною взаємодією між гравцями [6].

Гра пропонує систему динамічних подій, які відбуваються в усьому світі гри. Ці події можуть змінюватися від дії гравців і реагувати на їхні дії. Вони включають битви з монстрами, оборону фортець, пошук і рятування, збір ресурсів та багато іншого. На відміну від попередніх, події розвиваються дуже динамічно та роблять світ живим і непередбачуваним, забезпечуючи постійний потік пригод для гравців.

Guild Wars 2 (GW2) пропонує широкий вибір PvE контенту. Гравці можуть виконувати квести, досліджувати підземелля та рейди, боротися з монстрами та виконувати завдання. Квести в GW2 мають нелінійні сюжети, які можуть впливати на світ історії гри. Рейди – це великі і вимогливі випробування, де групи гравців спільно працюють над подоланням потужних босів.

Гра має розвинуту систему боїв, включаючи битви на аренах та війни серверів серед усіх гравців. Для цього вона має систему майстерності, яка дозволяє гравцям розвивати свої навички та отримувати нові можливості. Гравці можуть, під час виконання завдань, заробляти очки майстерності, що дозволяє їм розблоковувати нові способи подорожей, покращувати бойові навички та впливати на світ навколо себе. Система майстерності додає глибину і прогрес до геймплею, надаючи гравцям додаткові цілі для досягнення.

GW2 дозволяє гравцям створювати та налаштовувати своїх персонажів. Гравці можуть вибирати з різних рас, класів та налаштувань зовнішності, що дозволяє створити унікальний персонаж, відповідного їхньому стилю гри. Крім того, гра пропонує широкий вибір костюмів, аксесуарів та зброї, які можна зібрати та налаштувати.

GW2 підтримує соціальну взаємодію між гравцями, аналогічну двом попереднім іграм, без жодних змін.

Загалом, Guild Wars 2 пропонує широкий вибір інноваційних рішень та функцій, які забезпечують захоплюючий геймплей та різноманітність досвіду для гравців. Гра акцентує на взаємодії гравців, спільній грі та спільноті, що дозволяє зблизитися та спілкуватися з іншими гравцями з усього світу. Крім того, система майстерності додає прогресу до геймплею та забезпечує додаткові цілі для досягнення.

1.6. Відмінності та особливості розробленого продукту

Під час розроблення проєкту, особливу увагу буде звернено на гнучку систему діалогів та підвищену надійність зберігання особистих даних. Діалоги можуть бути ключовим аспектом інтерактивності гри, дозволяючи гравцям взаємодіяти з NPC та впливати на події в грі.

Гнучка система діалогів може допомогти гравцям відчувати більшу свободу та контроль над своєю стратегією, оскільки вони зможуть вибирати різні варіанти діалогів та дій, що впливатимуть на подальші події

в грі. Наприклад, гравець може обрати мирну або бойову стратегію взаємодії з NPC, в залежності від своїх власних цілей та стилів гри.

Що стосується підвищеної надійності зберігання особистих даних, це може бути дуже важливим аспектом для гравців, які переживають за безпеку своїх особистих даних під час використання онлайн-ігор. Використання асиметричних алгоритмів шифрування може забезпечити додатковий рівень захисту під час відправлення особистих даних гравцями, а хешування паролів та використання симетричних криптосистем дозволить захистити особисті дані користувачів навіть у випадку витоку даних.

Однак, варто зазначити, що реалізація такої системи може бути складною і вимагати значних ресурсів. Тому потрібно забезпечити адекватну інфраструктуру та розглянути питання відповідальності за зберігання та обробку особистих даних гравців. А в майбутніх версіях розглядається можливість використання блокчейн-технологій для забезпечення безпеки та прозорості обробки особистих даних гравців.

Також в майбутньому не виключається використання нейронних мереж в деяких елементах діалогової системи. Таку можливість важливо розглянути для автоматизації діалогів та аналізу поведінки гравців. Штучний інтелект може допомогти підібрати оптимальні варіанти діалогів залежно від поведінки гравців та взаємодії з NPC, що може покращити ігровий досвід гравців та спростити розробку нових дерев діалогів та розширення існуючих.

Узагалі, ідея створення нової гри у жанрі ММОРПГ з акцентом на гнучку систему діалогів та підвищену надійність зберігання особистих даних може мати великий потенціал для привернення уваги гравців, які цінують інтерактивність та захист своїх особистих даних. Однак, варто ретельно розглянути всі аспекти розробки та впровадження системи, щоб забезпечити якість гри та безпеку гравців.

1.7. Освітній елемент проєкту

Існуюча проблема браку досвіду програмістів на ринку праці є досить поширеною, особливо для тих, хто щойно закінчив університет або курси з програмування. Багато роботодавців вимагають досвіду від кандидатів на роботу, і це може бути складно для тих, хто тільки починає свою кар'єру.

Одним із способів розв'язання цієї проблеми може бути отримання досвіду на практиці шляхом стажування або роботи над проєктами. Це може допомогти набратися досвіду та отримати реальні навички програмування. Також можна звернутися до менторів або спільнот програмістів, щоб отримати поради та допомогу у покращенні своїх навичок, але такий варіант не є безкоштовним.

Тому другорядною задачею цього проєкту буде отримання досвіду та ключових навичок програмування повноцінних 3-вимірних онлайн ігор. При створенні гри у жанрі ММОРПГ можна отримати ряд корисних навичок, таких як:

- розробка клієнт-серверної взаємодії;
- робота з базами даних;
- розробка штучного інтелекту;
- розробка ігрових механік;
- тестування та оптимізація;
- керування проєктом.

Загалом, розробка гри у жанрі ММОРПГ може допомогти програмістам отримати досвід роботи зі складними мережевими системами, базами даних, штучним інтелектом, графікою та анімацією, ігровою механікою, тестуванням та оптимізацією, а також керуванням проєктом.

При створенні нової гри розробники також повинні вирішувати різні технічні та дизайнерські проблеми, такі як оптимізація гри для різних пристроїв, створення високоякісних графічних ефектів, підвищення

швидкості та стабільності гри, тестування та поліпшення гри на основі отриманого фідбеку від користувачів.

Підсумовуючи, при створенні нової гри у жанрі ММОРПГ програмісти можуть отримати значну кількість корисних навичок, таких як розробка складних мережевих систем, робота з базами даних, розробка штучного інтелекту, розробка графіки та анімації, а також розробка ігрової механіки. Всі ці навички можуть бути корисними в інших проєктах та допоможуть розробникам стати більш кваліфікованими та відомими в своїй галузі. Отже, розробка гри у цьому жанрі буде гарною можливістю для власного розвитку як спеціаліста.

2. АНАЛІЗ ПЛАТФОРМИ ТА ТЕХНОЛОГІЙ РОЗРОБКИ

2.1. Аналіз платформ

В даному підрозділі буде проведений аналіз трьох основних платформ – Linux, MacOS та Windows, з метою вибору оптимальної платформи для розробки нової гри. Кожна з цих платформ має свої особливості, переваги та недоліки, які необхідно врахувати при виборі. В ході аналізу будуть розглянуті такі аспекти, як популярність серед користувачів, кількість доступних ігор, безпека, а також можливості для розробки та потенційної аудиторії.

2.1.1. *Операційна система Linux*

Одним з основних недоліків розробки ігор на платформі Linux є те, що кількість користувачів цієї ОС суттєво менша, ніж у інших операційних систем [7]. Це означає, що потенційна аудиторія ігор на Linux обмежена, що може негативно вплинути на прибутки розробників.

Крім того, серед користувачів Linux малий відсоток потенційних гравців, оскільки найчастіше цю систему використовують у професійних цілях або для дуже слабких комп'ютерів. В результаті, розробка ігор для Linux може виявитися невигідною для студій, що їх розробляють.

Однак, для Linux існує досить мало ігор, що може бути перевагою для студій, готових зайняти свою нішу на ринку. Тому користувачі, які з якихось причин змушені користуватися Linux, але які зацікавлені в іграх, будуть готові платити більше за якісні ігри, які доступні на їхній операційній системі.

Крім того, розробка ігор для Linux може виявитися вигідною у плані сумісності та безпеки. Linux є відкритою операційною системою, що дозволяє розробникам повністю контролювати процес створення гри. Крім того, Linux більш стійкий до вірусів та інших шкідливих програм, що знижує ризики для користувачів та підвищує рівень безпеки ігор.

2.1.2. *Операційна система MacOS*

Розробка ігор на macOS є однією з перспективних галузей в індустрії геймдеву. Хоча ця операційна система не так популярна, як Windows, вона все ж є більш популярною вибором серед користувачів порівняно з Linux [8]. До того ж, той факт, що ця операційна система прив'язана до пристроїв Apple при їх придбанні, робить заміну macOS складною задачею. Це означає, що серед користувачів macOS вищий відсоток потенційних гравців порівняно з Linux.

Завдяки цим особливостям, розробники ігор для macOS мають більшу можливість залучити аудиторію та отримати комерційний успіх. Втім, незважаючи на це, наразі на ринку все ще відносно мало повноцінних ігор, розроблених спеціально для macOS. Це створює можливості для розробників пропонувати свої продукти за вищу ціну, з урахуванням меншої конкуренції.

2.1.3. *Операційна система Windows*

Windows є найпопулярнішою операційною системою для персональних комп'ютерів і має широку базу користувачів у всьому світі [9]. Це дає розробникам ігор великий потенціал для досягнення широкої аудиторії та комерційного успіху.

Windows також має найвищий відсоток геймерів серед усіх операційних систем. Багато геймерів вибирають Windows як свою основну платформу для гри через доступність і широкий вибір ігор, які розроблені спеціально для цієї операційної системи. Це дає можливість розробникам ігор залучити велику кількість гравців та отримати широке визнання.

Однак, через свою популярність, Windows також є ціллю для кіберзлочинців. Вона вважається однією з найуразливіших операційних систем щодо впливу вредоносного програмного забезпечення. Розробники ігор на Windows повинні приділяти особливу увагу заходам безпеки, щоб захистити свої ігри та гравців від потенційних загроз.

Розробка ігор на Windows також вимагає роботи в умовах високої конкуренції. Завдяки великій кількості розробників і видавництв, ринок ігор для Windows є дуже насиченим. Це означає, що розробники повинні вкладати багато зусиль у створення унікальних, цікавих продуктів, щоб виділитися серед конкурентів та привернути увагу гравців.

2.1.4. *Остаточний вибір платформи*

В сучасній ігровій індустрії ігри на персональних комп'ютерах (ПК) залишаються надзвичайно популярними серед геймерської аудиторії. Однак найпопулярнішою платформою для ПК є операційна система Windows. Розробка масової мультиплеєрної онлайн-рольової гри (ММОРПГ) під керуванням цієї операційної системи може мати значний комерційний успіх, оскільки велика частина геймерської спільноти використовує саме Windows.

Додатковою перевагою розробки гри під Windows є набуття практичного досвіду роботи з цією операційною системою. Оскільки Windows є широко використовуваною платформою у галузі геймдеву, тому практичний досвід у розробці під цю платформу може бути корисною при пошуку роботи в індустрії розробки подібних систем.

Тому даний проєкт вирішено розробляти під операційну систему Windows.

2.2. Аналіз ігрових рушіїв

У цьому розділі буде проведений аналіз двох провідних ігрових рушіїв: Unity та Unreal Engine (UE), з метою вибору оптимального рушія для розробки нашого власного проєкту. Кожен з цих рушіїв має свої особливості, функціональність, переваги та недоліки, які варто врахувати при прийнятті рішення. Під час аналізу будуть розглянуті такі аспекти, як зручність використання, гнучкість, наявність інструментів для створення графіки та анімації, підтримка різних платформ, продуктивність спільноти розробників, мови програмування.

2.2.1. *Ігровий рушій «Unity»*

Unity – це потужний і популярний ігровий движок, який використовується для розробки ігор, віртуальної реальності та інтерактивних додатків [10]. Він надає розробникам зручні інструменти і можливості для створення високоякісних ігрових проєктів.

Рушій Unity є кросплатформним і підтримує розробку ігор для різних платформ, таких як Windows, macOS, Linux, iOS, Android, Xbox, PlayStation і багатьох інших. Це дозволяє розробникам створювати одну версію гри і випускати її на різних платформах без значних модифікацій.

Також Unity має потужний вбудований візуальний редактор, що дозволяє розробникам візуально створювати сцени, об'єкти, анімацію, ефекти та інше. Візуальний редактор дозволяє швидше прототипувати та відлагоджувати гру без необхідності написання великої кількості коду.

Розробка ігор на цьому рушії відбувається на кількох мовах програмування, включаючи C# і UnityScript (подібна до JavaScript), але C# є основною рекомендованою мовою для розробки в Unity, і саме вона має використовується для створення складних ігрових систем.

Найчастіше рушій Unity використовується для розробки віртуальної реальності (VR) та доповненої (AR). Також він є найпопулярнішим для розробки казуальних ігр та ігр-платформерів.

2.2.2. *Ігровий рушій «Unreal Engine»*

Unreal Engine (UE) – це потужний інтегрований ігровий рушіїв, який використовується для розробки високоякісних ігор різних жанрів [11]. Даний рушій надає широкі можливості для створення ігрових проєктів з вражаючою графікою, реалістичною фізикою та захоплюючою геймплейною механікою.

Unreal Engine забезпечує широкі можливості для розробки різних жанрів ігор, включаючи екшн, рольові ігри, шутери, аркади та ігри

віртуальної реальності. Він може використовуватися для створення як масштабних відкритих світів, так і невеликих інді-проєктів.

Також цей рушій має найпотужніші інструменти для реалістичної обробки графіки, включаючи високу деталізацію моделей, реалістичне освітлення, воду, частинки та інші ефекти. Це дозволяє розробникам створювати вражаючі візуальні ефекти та віртуальні світи, наближуючи графіку до фотореалістичної.

Ще однією особливістю UE є вбудована фізична система, яка дозволяє створювати реалістичну поведінку об'єктів, симулювати фізичні ефекти, такі як гравітація, колізії, руйнування та інші. Це додає реалізму та динаміки до ігрового світу.

Як і Unity, Unreal Engine постачається з потужним інтегрованим середовищем розробки (IDE), яке надає зручний інтерфейс для створення, редагування та налагодження ігрових об'єктів, сцен та скриптів. Це спрощує розробку та прискорює процес створення ігор.

Також, як і Unity, Unreal Engine дозволяє розробляти ігри для різних платформ, включаючи ПК, консолі, мобільні пристрої та віртуальну реальність. Це надає розробникам можливість досягати широкої аудиторії та розповсюджувати свої ігри на різних пристроях.

Unreal Engine – це потужний інструмент для розробки ігор, який надає розробникам гнучкість та можливості для створення захоплюючих ігрових проєктів з високоякісною графікою та реалістичною фізикою.

Через це все, багато провідних студій ігрової галузі вибирають Unreal Engine для розробки своїх найамбітніших проэктів, завдяки його потужності та можливостям, що задовольняють високі вимоги AAA-ігор та глобальних ігрових проєктів.

2.2.3. *Остаточний вибір рушія*

При виборі ігрового рушія для розробки даного програмного продукту було прийняте рішення використовувати Unreal Engine.

Основною причиною стало те, що цей движок в комбінації з мовою C++ надає розробникам більший контроль над ресурсами комп'ютера. Це важливо при створенні програмного забезпечення, яке вимагає великих обчислювальних потужностей. Використання Blueprint, графічного інтерфейсу розробки, дозволяє прискорити процес розробки та компіляції, спрощуючи роботу зі скриптами та взаємодію з компонентами гри.

Крім того, UE широко використовується для розробки великих ігрових проєктів в індустрії. Тому отримання досвіду роботи з UE підвищує конкурентоспроможність розробника і відкриває широкі можливості для працевлаштування в провідних компаніях галузі.

2.3. Аналіз мови розробки продукта

2.3.1. *Мова програмування C++*

Мова програмування C++ в Unreal Engine (UE) надає розробникам безмежні можливості для створення ігор з усією гнучкістю повноцінних мов програмування. C++ дозволяє створювати власні класи та функції, розширювати функціональність рушія та взаємодіяти з його компонентами [12].

Особливістю C++ в UE є його прекрасна інтеграція з системою Blueprint. Класи, реалізовані на C++, можна легко використовувати в Blueprint, що дозволяє комбінувати високошвидкісний C++ код з візуальною логікою Blueprint. Це дає змогу розробникам максимально ефективно використовувати обидва підходи для створення складних ігрових систем.

Один із головних плюсів використання C++ в UE – це висока швидкодія виконання коду. C++ код працює значно швидше, ніж код на базі Blueprint, оскільки використовує компіляцію в машинний код. Це особливо важливо для обчислювально важливих операцій та ігрових систем, де кожна мілісекунда має значення.

Однак, варто відзначити, що після кожної зміни в C++ коді потрібно повністю перекомпілювати весь проєкт. Цей процес може займати значний час, особливо при великих проєктах. Втім, це питання оптимізації та компромісу між швидкодією розробки та швидкодією виконання.

Загалом, використання C++ в Unreal Engine надає розробникам можливість втілити найскладніші ідеї та досягти високої продуктивності у створенні ігрових проєктів.

2.3.2. *Візуальний скриптинг «Blueprints»*

Blueprint в Unreal Engine (UE) є візуальною системою програмування, яка дозволяє розробникам створювати функціональність гри за допомогою вузлів та з'єднань, замість традиційного коду C++ [13]. Blueprint пропонує інтуїтивно зрозумілий і графічний підхід до розробки, що робить його більш доступним для новачків та художників, які не мають досвіду в програмуванні.

Один з головних плюсів Blueprint – це його візуальна природа, яка дозволяє швидко створювати і змінювати логіку гри без потреби писати код. Крім того, однією з переваг Blueprint є можливість перекомпілювати лише конкретний модуль Blueprint під час змін, що значно прискорює процес розробки.

Проте, варто зазначити, що Blueprint є менш оптимізованим, ніж код на C++. Це означає, що в деяких випадках важкі або обчислювально інтенсивні функції можуть працювати повільніше в порівнянні з аналогічними реалізаціями на C++. Крім того, Blueprint не має всіх можливостей об'єктно-орієнтованого програмування, таких як полінаслідування та чисті методи.

Blueprint часто використовується для швидкої прототипізації, створення простих ігор та інтерактивних демонстрацій, а також для реалізації геймплею та логіки гри, які не вимагають високої швидкодії та

оптимізації. Для складніших систем або випадків, коли потрібна максимальна продуктивність, використовується код C++.

2.3.3. *Остаточний вибір мови розробки продукту*

Прийнято рішення застосовувати C++ для управління мережевими запитами та серверним менеджментом, для користування можливостями ООП та мережевих з'єднань.

Також під час розробки буде використовуватись Blueprint для реалізації окремих механік і невеликої логіки. Це дозволить спростити процес розробки та забезпечити швидке прототипування

Така комбінація є оптимальним рішенням і активно використовується розробниками комерційних проєктів.

2.4. Аналіз зв'язку сервера та баз даних

2.4.1. *Плагіни Unreal Engine*

В Unreal Engine (UE) існують різні плагіни, які дозволяють взаємодіяти з базами даних та полегшують процес розробки [14]. Один з таких плагінів – "MySQL Connector" – надає можливість підключення та взаємодії з базами даних MySQL [15].

Плагін "MySQL Connector" дозволяє розробникам UE здійснювати зчитування та запис даних в базу даних MySQL безпосередньо з внутрішньої логіки гри. Він надає доступ до різних функцій та методів для виконання запитів, вставки, оновлення та видалення даних в базі даних.

Використання плагіна "MySQL Connector" значно спрощує взаємодію з базами даних під час розробки у Unreal Engine. Розробники можуть звертатися до бази даних, отримувати та змінювати дані безпосередньо зі свого коду гри, що дуже зручно та ефективно.

Важливо також відзначити, що всі плагіни для взаємодії з базами даних, включаючи "MySQL Connector", є платними.

2.4.2. *Через стороній сервер*

Один з ефективних способів взаємодії з базами даних на ігровому сервері – це через використання окремого сервера, який виступає абстракцією над базою даних. Всі модулі управління базами даних можуть бути розміщені в окремому проєкті, що забезпечує зручність при масштабуванні та розробці.

Використання окремого сервера як проміжного зв'язку між ігровим сервером та базою даних має свої переваги. Він надає зручний інтерфейс та методи взаємодії для зчитування, запису та оновлення даних в базі даних. Такий сервер може бути розроблений з використанням різних технологій та мов програмування, незалежно від інструментів розробки самої гри.

Окрім того, той самий сервер, як абстракція над базою даних, може бути використаний і в інших ресурсах всесвіту гри. Це забезпечує однообразність та спільне використання даних, що є дуже зручним та економить час при розробці різних компонентів гри.

Такий підхід дозволяє зберігати централізовану систему управління базами даних та дозволяє змінювати, додавати або видаляти модулі безпосередньо на сервері. Це полегшує розробку, підтримку та масштабування ігрового сервера та бази даних, забезпечуючи більш гнучкий та контрольований процес розробки.

2.4.3. *Остаточний вибір зв'язку сервера та бази даних*

З урахуванням майбутнього масштабування проєкту та потенційного ускладнення запитів до бази даних, більш оптимальним варіантом є створення сервера-абстракції над базою даних (САБ).

Також на поточний час немає можливості купувати додаткові плагіни до Unreal Engine. Тому проєктування САБ є однозначним варіантом для даного проєкту.

2.5. Аналіз мови розробки САБ

Так як прийнято рішення використовувати САБ для зв'язку з БД,

доречно провести аналіз інструментів для розробки подібної системи.

2.5.1. *Мова програмування C++*

Однією з переваг такого підходу є швидка робота коду. C++ відомий своєю ефективністю та швидкодією, що дозволяє оптимізувати обробку запитів та забезпечити швидку відповідь сервера на клієнтські запити [16].

Проте, варто враховувати, що розробка веб-сервера на C++ може вимагати багато коду для реалізації навіть простих операцій. Оскільки C++ є мовою з низькорівневим доступом до системних ресурсів, ви маєте більшу контроль над деталями реалізації, але це також означає, що потрібно більше коду для роботи з мережевими протоколами, маршрутизацією запитів, обробкою даних та іншими аспектами роботи сервера.

2.5.2. *Мова програмування JavaScript*

Розробка REST-API сервера-абстракції над БД на мові JavaScript пропонує безліч можливостей та інструментів, які значно спрощують цей процес. JavaScript широко використовується для створення серверної частини вебдодатків та пропонує безліч фреймворків, які спрощують написання коду та забезпечують готову інфраструктуру для розробки REST-API [17].

Один із найпопулярніших фреймворків для розробки REST-API на JavaScript – це Express.js. Express.js є мінімалістичним та гнучким фреймворком, який надає інструменти для створення маршрутів (роутерів) та обробки запитів. Він забезпечує простоту в налаштуванні сервера та обробці різних типів запитів.

Під час розробки JavaScript для створення REST-API сервера необхідно налаштувати роутери та контролери. Роутери визначають маршрути та їх обробники, які приймають запити та викликають відповідні контролери. Контролери містять логіку обробки запитів, включаючи взаємодію Космосу з базою даних чи іншими зовнішніми сервісами.

Швидкість роботи JavaScript-коду може залежати від різних факторів, включаючи ефективність написаного коду, оптимізацію та характеристики оточення виконання. JavaScript є мовою, що інтерпретується, але сучасні движки JavaScript, такі як V8 (використовується в Node.js), забезпечують високу продуктивність. Крім того, використання асинхронних операцій та обробка запитів у неблокувальному режимі може значно підвищити швидкість роботи сервера на JavaScript.

2.5.3. *Мова програмування PHP*

Сервер REST-API, побудований на PHP, вимагає мінімальної настройки, оскільки це просто набір скриптів, розташованих у папках таким чином, щоб до них можна було звертатися через URL [18]. Це означає, що під час розробки відпадає необхідність налаштовувати окремо роутери та контролери, та не потрібно встановлювати складні середовища або конфігурувати серверні налаштування для роботи з REST-API. Достатньо розмістити скрипти в потрібній структурі папок, і сервер буде готов до роботи.

Також PHP має власний графічний інтерфейс для роботи з MySQL, що спрощує взаємодію з базою даних PHPMyAdmin. Такий інтерфейс спрощує налаштування бази даних та виконання операцій напряму в БД і забезпечує зручність і продуктивність обслуговування сервера.

Інтеграція бібліотеки OpenSSL у PHP значно спрощує роботу з криптографічними функціями. OpenSSL надає набір інструментів і функцій для шифрування, розшифрування, підпису та перевірки цифрових підписів, а також для забезпечення безпеки комунікації за допомогою SSL/TLS протоколу. Завдяки цій бібліотеці, PHP-розробники можуть легко використовувати криптографічні функції для захисту даних і забезпечення безпечних з'єднань.

Також PHP є однією з найшвидших інтерпретованих мов програмування. Завдяки своєму компілятору та оптимізаціям, PHP може швидко виконувати скрипти, що дозволяє забезпечити високу продуктивність REST-API сервера-абстракції над базою даних. Крім того, PHP має широкі можливості кешування даних та оптимізації виконання коду, що ще більше покращує швидкодію PHP-застосунків. Однак PHP-скрипти все одно працюють повільніше за JavaScript та значно повільніше ніж будь-яка компільована мова.

2.5.4. *Остаточний вибір мови програмування*

У даному проєкту взаємодія з базою даних не є основним компонентом. Запити до САБ будуть відбуватися тільки на початку запуску продукту для авторизації та синхронізації даних, і після завершення роботи з програмою для того щоб зберегти останній стан персонажу користувача. Тому швидкість системи не є головним фактором для прийняття рішення, а основним критерієм буде зручність та простота розробки сервера.

Через це для розробки САБ була обрана мова PHP. Вона є оптимальним варіантом завдяки простоті налаштування доступу до скриптів та інтегрованій бібліотеці OpenSSL.

2.6. Аналіз баз даних

2.6.1. *СУБД PostgreSQL*

PostgreSQL є потужною системою управління базами даних (СУБД), яка може бути успішно використано при розробці мультиплеєрних ігор. PostgreSQL відомий своєю надійністю, масштабованістю та розширюваністю, що робить його привабливим вибором для розробників.

Основні переваги PostgreSQL полягають у його високій продуктивності та ефективності [19]. Він пропонує оптимізовані механізми обробки транзакцій, що забезпечують цілісність даних та можливість відновлення у разі виникнення помилок. Багатофункціональність

PostgreSQL дає можливість виконувати складні запити до бази даних, забезпечуючи гнучкість інструментів для розробки мультиплеєрних ігор.

Однак PostgreSQL є важкою системою, яка включає велику кількість функцій, які не будуть використовуватись під час розробки конкретно нашого дипломного проєкту. Наприклад, робота з геоданими або повнотекстовим пошуком, вбудована мова програмування PL/pgSQL та інші не будуть використані, але присутні в PostgreSQL. Це може займати додатковий простір на диску та ресурси під час виконання операцій.

Проте, загальна характеристика PostgreSQL залишається дуже позитивною для розробки мультиплеєрних ігор. Він надає надійну та масштабовану інфраструктуру для зберігання даних, забезпечує швидкий доступ до інформації та дозволяє здійснювати операції з базою даних у реальному часі. Крім того, PostgreSQL підтримує розширення, що дозволяє розробникам адаптувати його до власних потреб та вимог гри.

2.6.2. *СУБД MongoDB*

MongoDB є документоорієнтованою базою даних, яка може бути використана при розробці багатокористувацьких ігор. Вона відноситься до класу NoSQL (нереляційних) СУБД і надає гнучку структуру документів у форматі JSON, що сприяє зберіганню та обробці даних [20].

MongoDB має гнучку схему даних, що дозволяє зберігати дані без жорсткої фіксованої схеми. Це особливо корисно у випадках, коли структура даних може змінюватися протягом розвитку гри. MongoDB дозволяє легко додавати нові поля або змінювати існуючі без переконфігурування бази даних.

Також MongoDB дозволяє розподіляти навантаження даних між кількома серверами, що дозволяє покращити продуктивність і масштабованість гри зі зростанням числа користувачів. Це особливо важливо для багатокористувацьких ігор з великою кількістю активних гравців.

До того ж ця СУБД надає високу продуктивність при операціях зчитування та запису даних. Вона добре підходить для ігор з великим обсягом активності користувачів та частим оновленням даних. MongoDB також підтримує можливості кешування даних, що сприяє швидкому доступу до часто використовуваних даних.

До того ж MongoDB надає власну потужну мову запитів, що дозволяє виконувати складні запити та агрегувати дані.

Проте MongoDB може вимагати більшого обсягу пам'яті порівняно з традиційними реляційними базами даних. Це може стати проблемою при роботі з великими обсягами даних на обмежених ресурсах сервера.

2.6.3. *СУБД MySQL*

MySQL є однією з найпопулярніших реляційних баз даних, яка широко використовується при розробці серверних додатків [21]. Вона надає надійну та ефективну зберігання даних і забезпечує широкий набір функціональних можливостей для роботи з ними.

MySQL відома своєю високою надійністю та стабільністю. Вона має розширені можливості для контролю цілісності даних та забезпечення безпеки. Це особливо важливо при розробці серверних додатків, де надійність та захист даних є важливими аспектами.

Також MySQL має простий у використанні синтаксис SQL, що дозволяє легко створювати, зчитувати, оновлювати та видаляти дані. PHP має добру підтримку для роботи з MySQL, що спрощує взаємодію з базою даних у вебпроєктах.

До того ж ця СУБД підтримується багатьма веб-хостинг-провайдерами та має велику спільноту розробників. Це забезпечує доступність допомоги, документації та різноманітних розширень для розробки серверних додатків.

Ще серед переваг можна виділити підтримку масштабування. Тож MySQL може працювати з великими обсягами даних та обробляти велику

кількість запитів. Вона має можливості горизонтального та вертикального масштабування, що дозволяє розширювати базу даних зі зростанням потреб.

Але MySQL може мати проблеми з продуктивністю при великій кількості одночасних запитів, особливо при використанні поганих або неоптимальних запитів. Тому при роботі з цією СУБД, варто приділити більше уваги до оптимізації SQL-запитів.

2.6.4. *СУБД Neo4j*

Neo4j є потужною графовою базою даних. Вона забезпечує зберігання та оптимізовану обробку даних у вигляді графів, що дозволяє ефективно моделювати та взаємодіяти зі складними взаємозв'язками між об'єктами [22]. Графова модель дозволяє ефективно виконувати складні запити та аналізувати структуру гри.

Головної перевагою цієї СУБД є її швидкість та масштабованість: Neo4j відомий своєю високою продуктивністю та здатністю обробляти великі обсяги даних. Вона дозволяє швидко знаходити та працювати зі зв'язками між об'єктами. Крім того, Neo4j має можливості горизонтального масштабування, що дозволяє розширювати базу даних за необхідністю.

Також Neo4j має розширені аналітичні можливості. Вона надає розширені можливості для аналізу графових даних. Це дозволяє розробникам проводити складний аналіз грального процесу, виявляти патерни та зв'язки, що може бути корисним для удосконалення геймплею.

Проте Neo4j є комерційним продуктом, і використання його в ігровій розробці може бути дорогим. Це особливо важливо для незалежних розробників та невеликих студій з обмеженими бюджетами.

Також Neo4j не підходить для всіх типів даних. Він найбільш ефективний у випадках, коли важливі зв'язки між об'єктами та їх аналіз. Однак, якщо гра базується на простій структурі даних або має значну

кількість незалежних об'єктів, Neo4j може бути зайвим або недоцільним рішенням.

2.6.5. *Остаточний вибір бази даних*

У даному проєкті немає необхідності у глибокому аналізу даних або збереженню даних специфічного типу. Для цього проєкту необхідна база даних, яка задовольнить основні потреби в роботі з даними: збереження, вилучення, фільтрація, масштабування.

Через це в якості СУБД було обрано MySQL, бо ця система є легкою, підтримує масштабування та є достатньо гнучкою для майбутнього вдосконалення структури гри.

2.7. Аналіз криптографічних алгоритмів

У розділі 2.7 буде проведений аналіз криптографічних функцій для забезпечення безпечної доставки та зберігання особистих даних і паролів під час користування даним продуктом. Метою аналізу є вибір оптимальних криптографічних рішень, що забезпечать надійний захист даних та відповідатимуть стандартам безпеки й ефективності.

2.7.1. *Аналіз алгоритмів хешування паролів*

Алгоритм хешування обирається між MD5 та SHA256.

Хеширование MD5 є одним з найвідоміших хеш-алгоритмів, які використовуються для генерації хеш-суми з вхідного повідомлення або даних [23]. Основна особливість MD5 полягає у його здатності швидко обчислювати хеш-суму та представляти її як 128-бітний хеш. Проте, варто зазначити, що MD5 має кілька недоліків, які обмежують його використання в деяких сферах.

Одним з основних переваг MD5 є швидкість обчислення хеш-суми. Алгоритм може ефективно працювати з великими обсягами даних, що робить його привабливим для використання в багатьох застосунках. Крім

того, згенерований хеш має фіксовану довжину і може бути легко перевірений на цілісність.

Однак, перш за все, вважається, що MD5 не є безпечним для застосування в криптографічних цілях. Зараз відомо багато вразливостей алгоритму, завдяки яким можна відновити початковий текст з використанням сучасних обчислювальних ресурсів. Це означає, що хеш, згенерований MD5, є вразливим до пошуку колізій, тобто пошуку двох різних повідомлень які мають однаковий хеш.

Узагалі, MD5 не рекомендується для використання в нових системах або протоколах, оскільки існують більш сильні і безпечні альтернативи, такі як SHA-256 або SHA-3. Ці алгоритми надійніші і володіють криптографічною стійкістю, що робить їх більш підходящими для сучасних вимог забезпечення безпеки даних.

В свою чергу, SHA-256 (Secure Hash Algorithm 256-bit) є одним з найбільш популярних хеш-алгоритмів, які використовуються для забезпечення безпеки даних [24]. Особливістю SHA-256 є його здатність створювати унікальні, нерозривні хеш-суми фіксованої довжини 256 бітів. Цей алгоритм має деякі переваги та недоліки, які важливо враховувати при його використанні.

Однією з основних переваг SHA-256 є його криптографічна стійкість. Цей алгоритм добре вивчений, пройшовши ретельні криптографічні аналізи та тестування. Він вважається безпечним для застосування в багатьох областях, включаючи криптовалюти, цифрові підписи, захист паролів та інші системи забезпечення безпеки.

Ще однією перевагою SHA-256 є його висока надійність. Хеш-суми, створені за допомогою SHA-256, мають дуже малий шанс колізій. Це робить SHA-256 надійним інструментом для перевірки цілісності даних і виявлення незаконних змін.

Тож в якості алгоритму хешування однозначно обираємо SHA-256, оскільки MD5 вже вважається ненадійним.

2.7.2. *Вибір симетричного алгоритму шифрування*

Безпрецедентним кандидатом на роль симетричного алгоритму шифрування є Advanced Encryption Standard (AES). Він є одним з найпоширеніших і надійних алгоритмів, який використовується для забезпечення безпечного зберігання даних [25].

AES є блочним алгоритмом шифрування і потребує ключ фіксованого розміру, довжиною 128, 192 або 256 біт. Кожен блок повинен мати таку саму довжину, як і ключ. Для збільшення надійності, цей алгоритм комбінує основні криптографічні операції: заміну байтів, зсув рядків, змішування стовпців і додавання ключів.

AES відомий своєю популярністю та використовується в різних сферах, включаючи фінансові установи, урядові організації та комерційні підприємства. Його популярність зумовлена великою мірою його високою ефективністю та надійністю. AES є рекомендованим стандартом криптографічного захисту урядових даних США та багатьма іншими країнами.

Також для нашого проєкту важливо те, що AES є стійким до великої кількості видів криптоатак, включаючи криптоатаки на основі відкритого тексту. Це особливо важливо для баз даних, які зберігають особисті дані користувачів, тому що зловмисник без перешкод може попередньо відправити свої дані на зберігання для подальшої криптоатаки злитої бази даних.

2.7.3. *Вибір асиметричного алгоритму шифрування*

Для захищеної передачі даних між клієнтом і сервером буде використовуватись асиметричний алгоритм шифрування RSA [26], що базується на складності факторизації великих простих чисел. Незважаючи на наявність більш ефективних альтернатив, наприклад, ECC (Elliptic Curve Cryptography) [27], було вирішено застосувати RSA через досвід його використання у минулому.

RSA здатний надійно зашифрувати дані та забезпечити цілісність і автентичність інформації. Цей алгоритм використовує пари ключів: публічний ключ для шифрування та приватний ключ для розшифрування. Клієнт та сервер обмінюються публічними ключами перед передачею даних. Клієнт зашифровує дані за допомогою публічного ключа, а сервер розшифровує їх використовуючи свій приватний ключ. Цей процес гарантує, що лише сервер може прочитати дані, а небажані особи, які можуть перехопити дані під час передачі, не зможуть їх розшифрувати.

Хоча RSA використовується на даний момент, в майбутньому планується перехід на ECC. ECC має кілька переваг, зокрема більшу криптографічну міцність при використанні ключів меншої довжини. Крім того, ECC вимагає меншої обчислювальної потужності та займає менше простору для зберігання ключів, що є важливими факторами для систем з обмеженими ресурсами при великих об'ємах даних.

3. АРХІТЕКТУРА РОЗРОБЛЕНОГО ПРОДУКТУ

3.1. Загальна ідея фреймворку

Перед тим як розглянути власні архітектурні та алгоритмічні рішення, необхідно розглянути загальну ідею фреймворку, яка виступає скелетом для даного проєкту.

Рушій Unreal Engine пропонує вбудований фреймворк для розробки проєктів. Розділимо її на дві частини та розглянемо їх окремо: готові до наслідування класи та підтримка клієнт-серверної взаємодії.

3.1.1. *Готові до наслідування класи*

Рушій Unreal Engine пропонує велику кількість класів, кожен з яких містить реалізовані механізми взаємодії з іншими класами. Наслідування від цих класів дозволяють використовувати вже готову оптимізовану взаємодію між різними компонентами проєкту, не обмежуючи потенціал до розширення функціональності відповідних класів. Такий підхід значно спрощує розробку механік та їх взаємодії. Серед основних класів, які використовувались під час розробки можна виділити наступні:

- Actor (актор) – це базовий клас у рушіях для створення об'єктів у грі. Актор виступає як фізичний або логічний об'єкт, який може взаємодіяти з іншими об'єктами у грі та може бути розміщений на сцені. Він містить в собі інформацію про положення в просторі, поведінку та взаємодію з іншими акторами. В даному проєкті клас Actor використовувався для створення декорацій та інтерактивних предметів на сцені.
- ActorComponent (Компонент актора) – це розширення для класу Actor, яке дозволяє додавати додаткову функціональність до актора. Компоненти актора можуть включати графічну модель, фізичне тіло, звукові ефекти або будь-яку іншу специфічну логіку, яка пов'язана з актором. Вони дозволяють розділити функціональність актора на окремі модулі для більшої гнучкості та

повторного використання. В даному проєкті цей клас використовувався для створення модулів взаємодії з САБ, інвентарем, керування віджетами та іншим.

- GameMode (Режим гри) – це клас, який керує логікою гри та встановлює правила та параметри для поточного рівня. Він задає початкові дані для рівня, проводить ініціалізацію, визначає кінець гри та багато іншого. В даному проєкті кожен рівень має свій GameMode.
- PlayerController (контролер гравця) – це клас, який дозволяє гравцеві керувати актором у грі. Він приймає вхідні дані від гравця (такі як натискання клавіш або рух контролера) і перетворює їх на дії для актора. Контролер гравця також відповідає за керування камерою, обробку взаємодії з іншими акторами та взаємодію з режимом гри. В даному проєкті PlayerController є основним класом, в якому реалізована обробка майже усіх подій та сигналів від користувача.
- UserWidget (віджет користувача) – це графічний елемент інтерфейсу користувача, який можна використовувати у грі. Він дозволяє відображати текст, зображення, кнопки та інші елементи інтерфейсу для спілкування з гравцем. Вони дозволяють створювати інтерактивний та зручний інтерфейс для гравців. В даному проєкті UserWidget активно використовувався для побудови системи графічних повідомлень.
- Character (персонаж) – це актор, який представляє головного героя або інших ігрових персонажів у грі. Він містить в собі модель персонажа, анімацію, фізичне тіло та інші характеристики, які визначають зовнішній вигляд та поведінку персонажа у грі. Персонажі можуть взаємодіяти з оточуючими об'єктами, виконувати дії, рухатися та брати участь у битвах або інших подіях

гри. В даному проєкті цей клас використаний для створення іграбельного героя всередині симуляції.

3.1.2. *Підтримка клієнт-серверного з'єднання*

В даному проєкті передбачена окрема робота клієнта та виділеного ігрового сервера. Тож ми розглянемо роботу рушія тільки для такої розділеної роботи.

За замовчуванням в Unreal Engine клієнт-серверна взаємодія працює наступним чином. Одразу після підключення клієнта до сервера, сервер створює всі необхідні об'єкти для нового користувача, дублює їх на клієнт. Після цього створенні об'єкті працюють кожен на своїй стороні незалежно один від одного.

Тож за замовчуванням клієнт-серверна взаємодія майже не відбувається. Проте розробник може це виправити за допомогою наступних інструментів:

- RPC (remote procedure calling). Завдяки цьому протоколу можна створити методи, які будуть виконуватись на стороні сервера, але викликатись клієнтом, або навпаки. Таким чином сервер може контролювати поведінку клієнтів, а клієнти можуть впливати на глобальний світ.
- Реплікація. Реплікація дозволяє синхронізувати значення полів на сервері та клієнті, або навпаки створювати функціонал, закритий від іншої сторони (від клієнта або сервера). Налаштовуючи параметри реплікації, можна уникнути використанню читів, завдяки контролю стану героя на клієнті з боку сервера.

Під час розробки для реалізації безпечної клієнт-серверної взаємодії активно використовувались обидва інструменти.

3.2. Менеджмент серверів

3.2.1. *Особливість переходу між рівнями в UE*

На поточний час в проєкті реалізовано три рівні: для авторизації, вибору героя та самої гри. Кожен з них потребує зв'язок із сервером. Проте виділений сервер може одночасно працювати лише на одному рівні. Тож необхідно зробити таке налаштування сервера, яке дозволить забезпечити обробку запитів для клієнтів на кожному рівні.

3.2.2. *Можливі шляхи вирішення проблеми*

Перший і найпростіший варіант рішення поставленої задачі може бути перехід на потрібні рівні синхронно і сервера, і клієнта. Проте така схема є можливою тільки для обробки одного клієнта, так як авторизація, вибір героя та сама симуляція мають працювати на кожному клієнті незалежно один від одного, а перехід на інший рівень на сервері неминуче вплине на роботу всіх приєднаних клієнтів.

Інший варіант вирішення такої проблеми є робота на одному великому рівні, де в різних місцях одночасно існують локації для авторизації, вибору героя та основна карта. Такий підхід дійсно може забезпечити зв'язок із сервером кожного клієнта на кожній стадії роботи з продуктом, однак він робить архітектуру монолітною та передбачає роботу одного сервера, який обслуговує всі види запитів та приєднань.

Останній варіант – це створити по виділеному серверу для кожного рівня, між яких будуть перемикатися клієнти в залежності від поточної їх задачі. Такий підхід вимагає налаштування кожного сервера окремо, або створення спеціальної системи для автоматизації такого процесу. Проте в такому разі буде знижено навантаження на сервер, який обробляє рівень з основною грою, що є найважливішим для забезпечення якісного досвіду користування системою. Також розроблена система спростить майбутнє розгортання серверів для інших карт для особливих подій, завдань або принципово нових функцій.

3.2.3. *Розроблене рішення*

В даному випадку впроваджено третий варіант роботи з рівнями. Тож було розроблено систему для налаштування серверів, яку зображено на рис. 3.1.

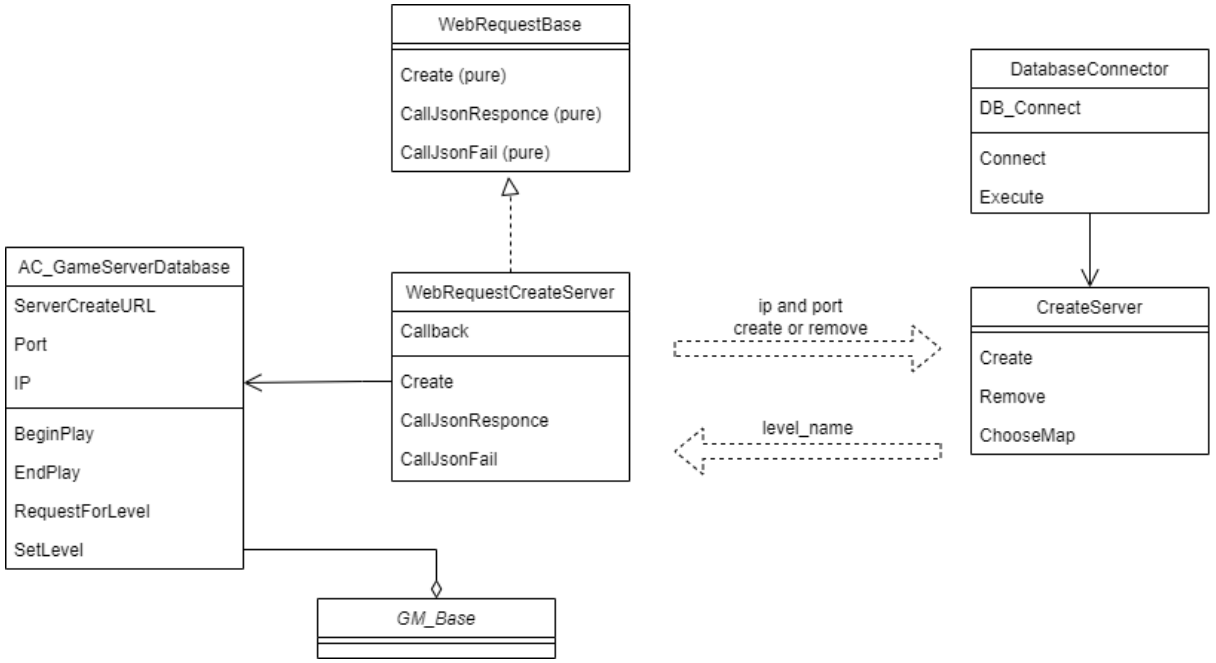


Рис. 3.1. Система автоматизованого налаштування серверів

WebRequestBase – базовий клас для обміну даними в форматі Json за допомогою http(s) запитів.

WebRequestCreateServer – дочірній від WebRequestBase клас, який створює запити на створення або видалення серверів в базі даних.

AC_GameServerDatabase – дочірній від ActorComponent клас, який забезпечує автоматизоване налаштування серверами. Даний компонент створюється та існує тільки на стороні сервера. При першому запуску цей компонент відправляє запит в САБ з адресою роботи сервера.

GM_Base – базовий для всього проєкту GameMode клас. Кожен GameMode для кожного рівня наслідується від нього.

DatabaseConnector – компонент на стороні САБ, який надає функціонал для прямої взаємодії з базою даних. Модуль написаний на PHP.

CreateServer – компонент на стороні САБ, який приймає за протоколом http у формати Json адресу і порт сервера та тип запиту (створення або видалення). Він перевіряє, для якого рівня ще не створено сервера та повертає назву рівня, який буде прив'язаний до заданого сервера. Модуль написаний на PHP.

3.3. Реєстрація та авторизація

Вся функціональність авторизації реалізована в компоненті AC_ClientServerTransfer, який приєднаний до PC_Login – контролер, який керує подіями під час автентифікації. Відповідна частина UML-діаграми зображено на рис. 3.2.

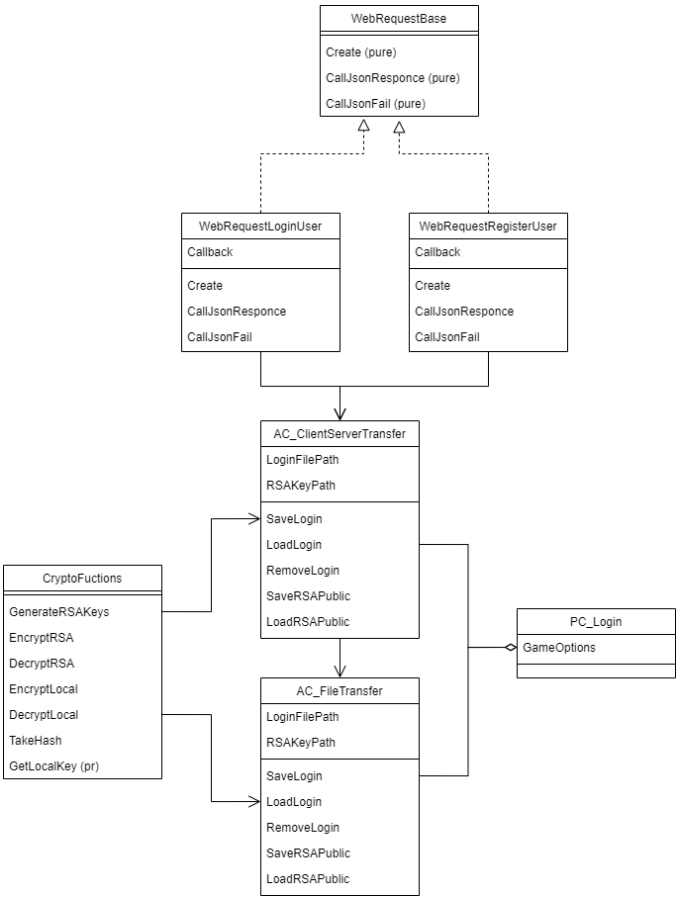


Рис. 3.2. UML-діаграма рівня авторизації

На поточний момент передбачено, що САБ та ігровий сервер мають знаходитись в межах однієї мережі. Це означає, що на поточний момент в

системі існує два небезпечних етапи опрацювання чутливих даних: відправлення повідомлення від клієнта до сервера та зберігання даних в БД.

Для забезпечення безпечної передачі та зберігання даних, було впроваджено інфраструктуру, схема якої зображена на рис. 3.3.

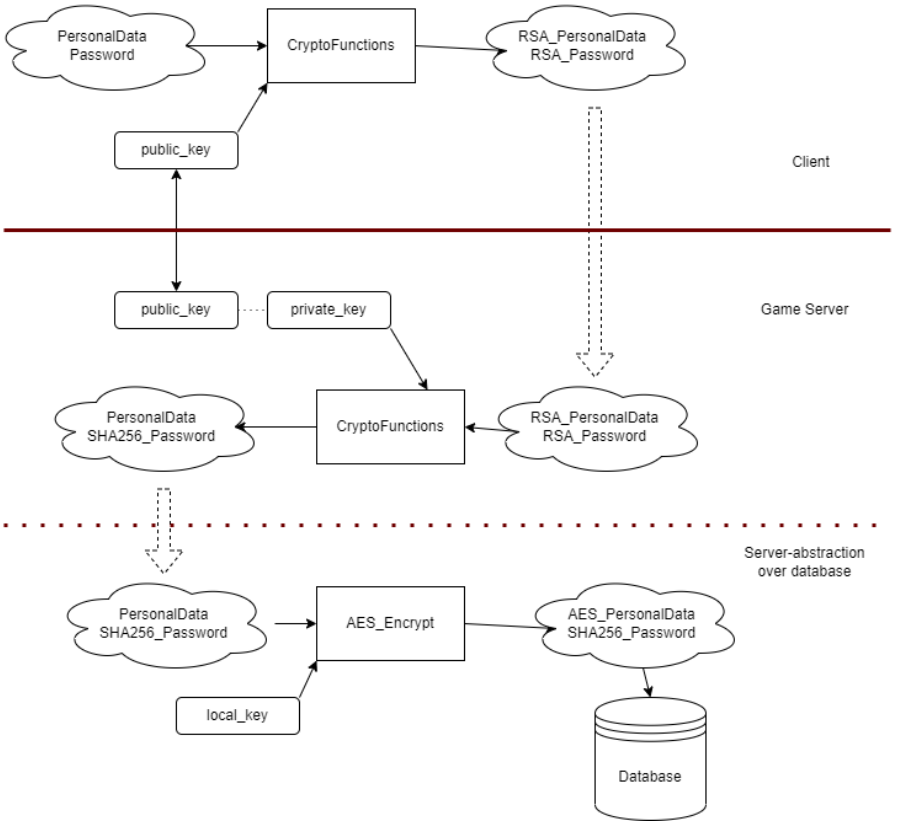


Рис. 3.3. Інфраструктура для передачі чутливих даних

CryptoFunctions – модуль, який відповідає за криптографічні операції, такі як генерація ключів, шифрування, дешифрування, хешування. Після відкриття рівня з автентифікацією, на стороні ігрового сервера компонент AC_ClientServerTransfer створює та зберігає пару ключів RSA. Публічний ключ реплікується на клієнт. Цей модуль являє собою статичний клас.

AC_FileTransfer – компонент, який впроваджує взаємодію з файловою систему.

AES_Encrypt – модуль, який виконує шифрування алгоритмом AES локальним ключем, який генерується один раз на початку роботи САБ.

Завдяки такій схемі, чутливі дані не можуть бути перехоплені під час передачі між клієнтом та сервером завдяки асиметричному шифруванню, а у випадку витоку даних з БД, зловмисник не зможе ними скористатися без знання ключа.

У майбутньому, САБ може бути розміщено в окремій від ігрового сервера мережі. У такому випадку з'єднання ігрового сервера із САБ можна реалізувати через протокол https, для покриття захистом таких публічних транзакцій.

3.4. Вибір персонажа

Перед залученням до гри, користувач має обрати героя. Схематичне зображення архітектури цього модуля зображено на рис. 3.4. у вигляді UML-діаграми класів.

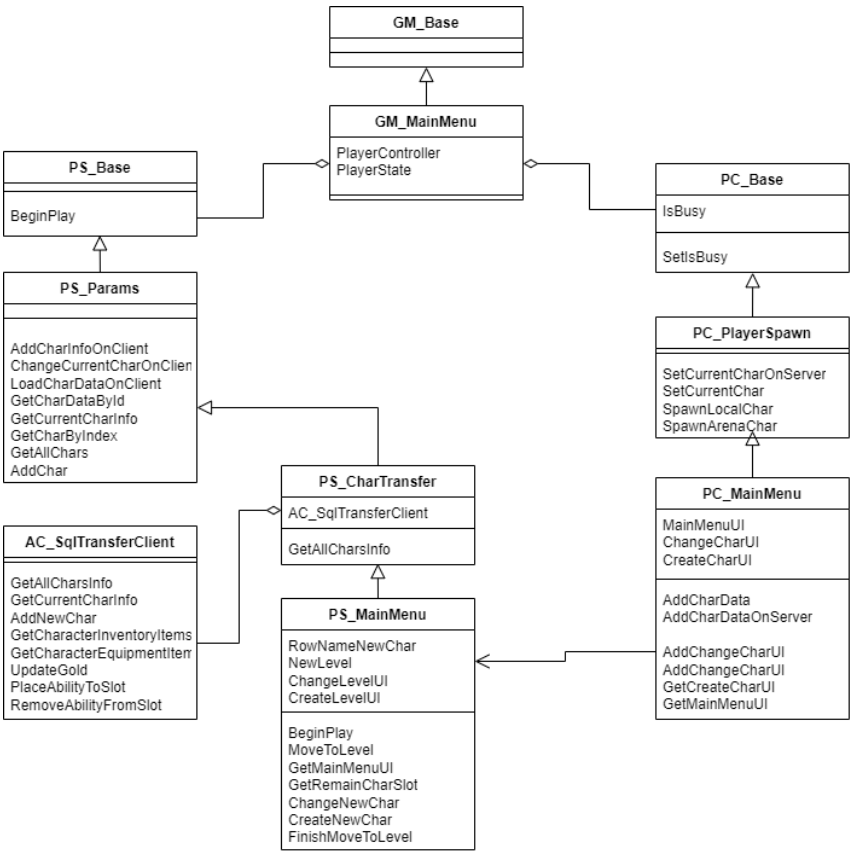


Рис. 3.4. UML-діаграма рівня головного меню

GM_Base – клас, який описує загальні правила усіх рівнів. На цьому рівні в цьому модулі немає особливостей та виконує роль провідника для фреймворку UE.

GM_MainManu – клас, який описує загальні правила рівня вибору персонажа. На цьому рівні в ньому є важливими посилання на створені PlayerState та PlayerController.

PS_Base – клас, який зберігає загальну інформацію про гравця, яка не залежить від рівнів. На поточний час такої інформації не має, але цей клас було створено як батьківський клас для всіх інших PlayerState, для спрощення введення спільного для всіх PlayerState функціоналу.

PS_Params – клас, який реалізує роботу з даними героїв користувача. Він потребує масив структур героїв користувача, запам'ятовує його та забезпечує зручний інтерфейс для їх обробки.

PS_CharTransfer – клас, який забезпечує взаємодію клієнта з сервером для отримання актуальних даних про героїв користувача. Для цього PS_CharTransfer користується компонентом AC_SqlTransferClient.

AC_SqlTransferClient – клас, який існує на стороні сервера та який впроваджує функціонал для взаємодії з базою даних для отримання існуючих героїв, додавання нових та встановлення їх актуального статусу. В цьому рівні цей компонент використовується для отримання існуючих та додавання нових героїв.

PS_MainMenu – клас, який зберігає тимчасову інформацію про героя, який знаходиться в процесі створення, а також керує локаціями, на який буде відображений поточний герой.

PC_Base – цей клас є дочірнім від PlayerController та є базовим для всіх контролерів в проєкті. Він зберігає статус виконання поточної операції: закінчилась вона чи ні.

PC_PlayerSpawn – контроллер, який забезпечує зручний інструментарій для спавну героїв на різних рівнях, включаючи рівень головного меню.

PC_MainMenu – цей контролер є головним модулем на цьому рівні. Саме він напряму взаємодіє з віджетами, обробляє події натискання на відповідні кнопки, координує роботу PS_MainMenu.

3.5. Основний світ симуляції

Переважна більшість механік реалізована та використовується на рівні самого світу симуляції. На поточний час, кількість модулей, які використовуються у цьому рівні перевищує 100, не рахуючи такі елементи як таблиці, структури, текстури і т.д. Тому для демонстрації архітектури головного рівня в пояснювальній записці було вирішено продемонструвати архітектурний центр цього рівня та механіку середньої складності – мапу світу та міні-мапу. Відповідна UML-діаграма зображена на рис. 3.5.

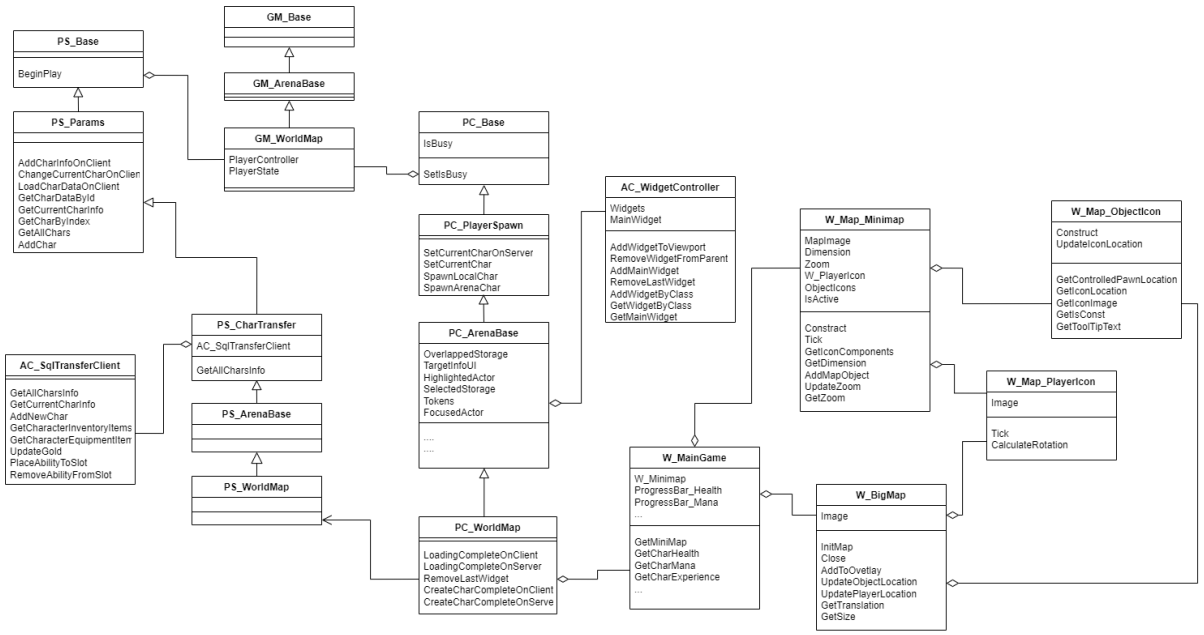


Рис. 3.5. UML-діаграма центральних модулей головного рівня

PS_Base – виконує ту саму роль, як і в рівні головного меню. Проте на цьому рівні буде корисним його метод BeginPlay, який викликається на початку гри і буде корисним в дочірніх класах.

PS_Params та PS_CharTransfer виконують таку саму роль, як в головному меню.

PS_ArenaBase – клас, який є базовим для всіх ігрових PlayerState. На поточний час він не реалізовує нових функцій, але наявність цього класу спрощує розширення проєкту та додавання нових рівнів.

PS_WorldMap – цей клас описує всю необхідну на рівні OpenWorld інформацію про гравця. На поточний час для цього достатньо функціоналу в його батьківських класах, проте створення PS_WorldMap необхідно для спрощення впровадження майбутніх оновлень на рівні.

PC_Base та PC_PlayerSpawn не змінили своєї поведінки відносно рівня головного меню.

PC_ArenaBase – цей контролер реалізує функціональність, яка є необхідною для роботи механік на всіх ігрових рівнях (відкритий світ, підземелля, тощо). Цей модуль є найоб‘ємнішим модулем в усьому проєкті через те, що переважна більшість ігрових механік зберігається на різних ігрових рівнях. Через нього обробляються події керування героєм, змін в інвентарі, екіпіровці, підвищення рівня, взаємодія з акторами та інше. При створенні нових рівнів, які являють собою іграбельні мапи, контролери цих рівнів будуть дочірніми до класу PC_ArenaBase.

PC_WorldMap – цей клас являє собою контролер, який реалізує специфічні механіки для основної відкритої мапи. Він створює потрібного персонажа в потрібному місці, налаштовує відповідний до мапи графічний інтерфейс та відслідковує початок гри на клієнті та сервері.

AC_WidgetController – цей компонент впроваджує менеджмент віджетів під час ігрового сеансу. Він встановлює головний віджет, спрощує додавання нових, видалення непотрібних та допомагає відслідковувати який із них є активним.

W_MainGame – цей клас наслідується від UserWidget та являє собою головний віджет для мапи OpenWorld, який налаштовує відображення основної ігрової інформації, такої як: досвід героя, його здоров'я, інформацію про поточного супротивника, міні-мапи та інше.

W_MapObjectIcon – цей віджет спрощує роботу з іконками на мапі. Він зберігає своє зображення та посилання на об'єкт, якому він належить, завдяки чому він спрощує обчислення відносних координат для віджетів W_Map_MiniMap та W_BigMap. Іконки об'єкт можуть бути константними або ні. Константні іконки не зникають, коли виходять за межі міні-мапи та допомагають гравцю зрозуміти, в якому напрямку розташовані їх об'єкти. Всі інші іконки у випадку виходу за межі міні-мапи зникають.

W_MapPlayerIcon – цей клас являє собою віджет, який налаштовує зовнішній вигляд героя на мапі та самостійно синхронізує поворот героя у просторі та поворот іконки на мапі.

W_Map_MiniMap – цей віджет збирає інформацію про навколишнє середовище, знаходить об‘єкти, які мають бути позначені на мапі, визначає, яку іконку потрібно для цього використовувати, обчислює відносне положення об‘єкту на мапі та розміщує іконку у потрібному місці віджета. Розрахунки координатів кожної іконки відштовхується від розміру оригінального зображення мапи рівня, поточного положення героя та поточного масштабу міні-мапи. На міні-мапі відображається тільки частина мапи та об‘єкти, розташовані на ній.

W_BigMap – це віджет, який відповідає за відображення мапи всього світу. Його принцип дії схожий на W_Map_MiniMap, але на відміну від нього відображає мапу всього світу тв точне положення константних іконок.

3.6. Загальні архітектурні рішення

З метою прискорення роботи алгоритмів, вирішено застосувати наступні два рішення.

- В проєкті активно використовується виклик функцій дочірніх класів через посилання на їх батьківський клас. Найпростіший шлях для цього це використати Down Casting, але ця операція є дуже повільною. Замість цього в проєкті активно

використовуються інтерфейси, які дозволяють викликати методи дочірніх класів без приведення типів.

- В даному проєкті активно використовуються віджети, які можуть самостійно виконувати обчислення. Деякі з них проводять обчислення кожен фрейм. З метою оптимізації, для всіх віджетів зі складною логікою було зроблено розділення на графічну частину та логічну, де перша наслідується від другою. Це спростить переклад логіки цих віджетів на C++, який працює значно швидше, ніж Blueprint.

Такі рішення дозволили оптимізувати роботу проєкту.

4. АНАЛІЗ РОЗРОБЛЕНОГО ЗАСТОСУНКУ

4.1. Технічні характеристики застосунку

Розроблений застосунок поділяється на три частини: клієнт, ігровий сервер та сервер-абстракція над базою даних.

Клієнт являє собою архів, який після розпакування займає 3.3 ГБ памяті. Клієнт є найоб'ємнішою компонентою системи, тому що клієнт має зберігати всі моделі, аудіо- або відео- ресурси та візуальні компоненти віджетів. Докладні властивості клієнтської частини зображений на рис. 4.1.

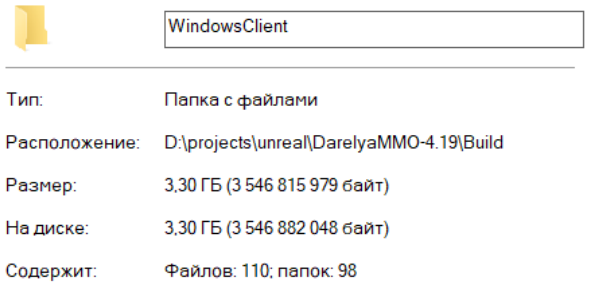


Рис. 4.1. Властивості клієнтської частини

Сервер проводить обчислення тільки з числовими та текстовими даними, а їх резільтати відправляє клієнтам, які на основі отриманих обчислень виконують рендер сцени. Завдяки цьому серверна частина застосунку містить тільки систему папок та бінарних файлів, які визначають алгоритми, якими керується кожен окремий сервер. Докладний перелік властивостей серверної частини зображений на рис. 4.2.

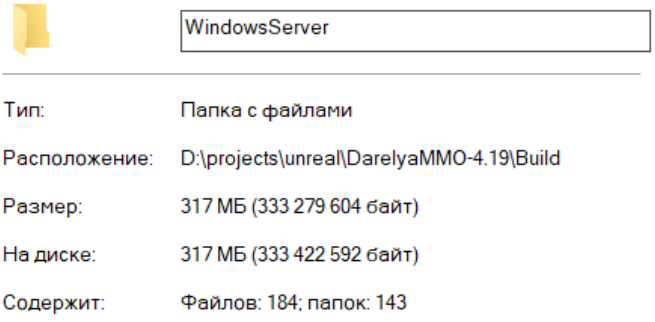


Рис. 4.2. Властивості серверної частини

Сервер-абстракція над базою даних являє собою архів із скриптами для взаємодії з базою даних, шифрування та дешифрування даних. Перелік властивостей цієї частини системи зображений на рис. 4.3.

Рис. 4.3. Властивості САБ-частини

4.2. Список функцій

На поточний час система підтримує 2 ролі: Гість, Гравець.

4.2.1. *Можливості для ролі Гість*

Користувач вважається Гостем до того моменту, доки він не авторизується в системі. Гість має можливість створити новий обліковий запис або авторизуватись як вже існуючий користувач. Також Гість може вказати, чи бажає він запам'ятати дані аутентифікації у разі успішної авторизації.

4.2.1. *Можливості для ролі Гравець*

Після успішної авторизації користувач отримує роль Гравець. Функції, які система забезпечує Гравцю поділяються на ті, які надаються до приєднання до гри та після.

До приєднання до гри, система надає Гравецю можливість керувати своїми героями та створювати нових. Під час створення Гравець може обрати клас героя із дво доступних: маг або воїн, та дати йому ім'я. Перед тим, як приєднатись до гри, Гравець має обрати героя, яким він буде керувати у самій грі.

Після приєднання Гравця до гри, він отримує повний доступ до всіх механік самої симуляції. Серед них є керування інвентарем, відкритими віджетами, екіпіровкою, перегляд карти світу, фокусування на ботах, атакування обраних цілей, обшук сховищ, торгівля в магазині в обидві сторони.

4.3. Розгортання сервера

Для налаштування серверної інфраструктури, достатньо зробити наступні кроки.

Перший крок – це налаштування PHP-сервера на пристрої, який буде виконувати роль сервера. Один із способів швидко налаштувати такий сервер це WampServer.

Після інсталяції, в папці програми буде папка "www". В нього необхідно вставити усі скрипти сервера-абстракції над БД (рис. 4.4).

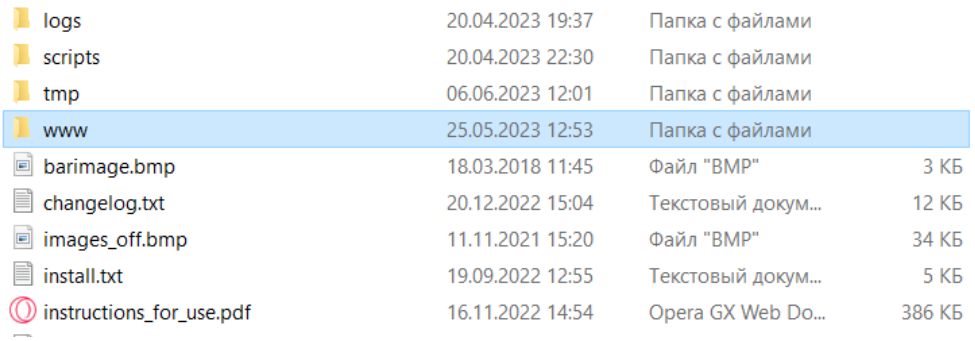


Рис. 4.4. Коренева папка для скриптів у WampServer

Після запуску сервера, необхідно запустити файл сервера стільки разів, скільки розроблено рівнів. На поточний час це 3 рази. Якщо налаштування пройшло успішно, в базі даних можна буде побачити інформацію про робочі рівні та адреса їх роботи. Якщо в базі даних є записи про всі необхідні рівні, то серверна сторона готова до приєднання клієнтів. Проте важливо, щоб адреса рівня LoginMenu була такою самою, яка вказана на клієнті за замовчуванням, тому що всі інші сервера клієнт

знаходить через нього. Приклад запису в базі даних про поточні робочі сервери наведено на рис. 4.5.

		id	level_name	address
☐	Изменить Копировать Удалить	750	Darelya	127.0.0.1:7779
☐	Изменить Копировать Удалить	748	LoginMenu	127.0.0.1:7777
☐	Изменить Копировать Удалить	749	MainMenu	127.0.0.1:7778

Рис. 4.5. Приклад записів у БД про поточні робочі сервери

4.4. Оцінка якості продукту

Для оцінки якості розробленого продукту виконаємо функціональний тест застосунку. Цей вид тестування спрямований на перевірку функцій продукту з метою виявлення потенційних помилок чи недоліків. Під час функціонального тестування будуть перевірені основні механіки, такі як авторизація, вибір героя, створення героя, керування інвентарем, керування екіпіровкою, торгівля тощо.

Сценарій 1 – перевірка рівня авторизації:

- *Реєстрація*: користувач переходить в меню реєстрації, заповню поле логіну, паролю та підтвердження паролю. Очікуваний результат – повідомлення про успішну реєстрацію (рис. 4.6).

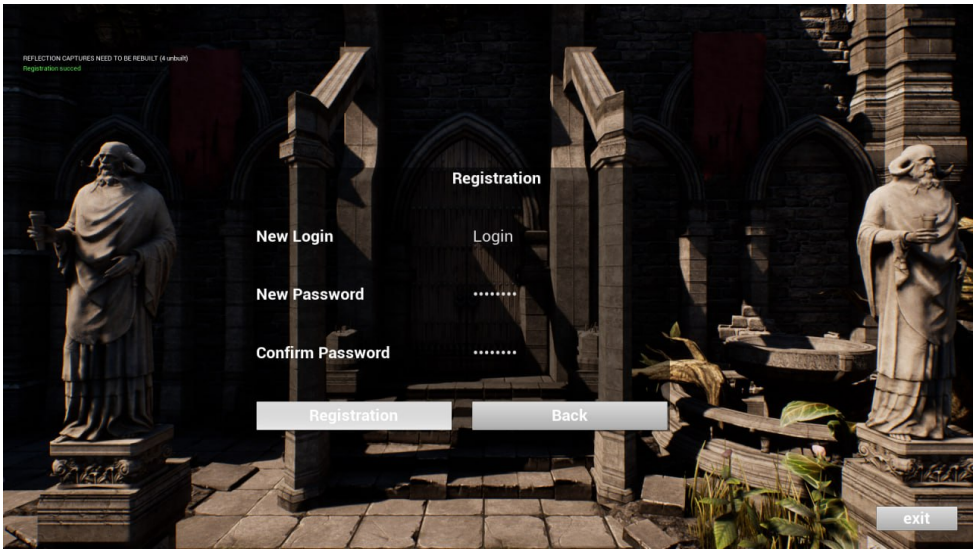


Рис. 4.6. Успішна реєстрація користувача

- *Авторизація*: користувач переходить в меню автентифікації (рис. 4.7), заповнює поля логіну та пароля згідно з одним із існуючим обліковим записом. Очікуваний результат – перехід на рівень вибору героя.

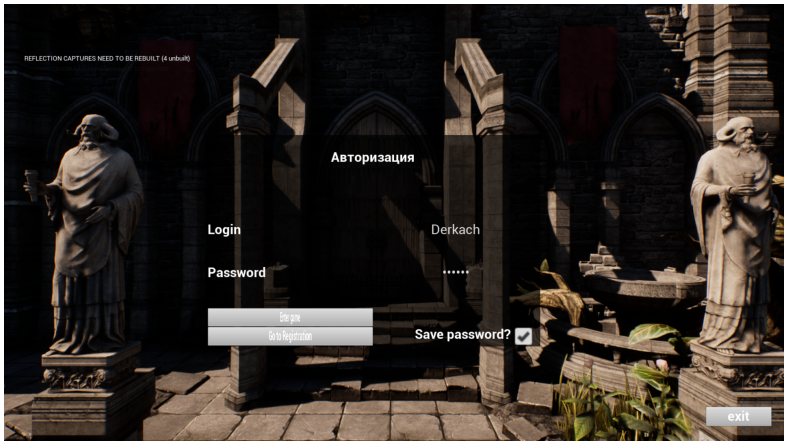


Рис. 4.7. Меню автентифікації

- *Збереження даних аутентифікації*: користувач ставить прапорець у відповідному місці та успішно проходить автентифікацію. Очікуваний результат – в папці гри з'явиться файл із зашифрованими даними автентифікації (рис. 4.8).

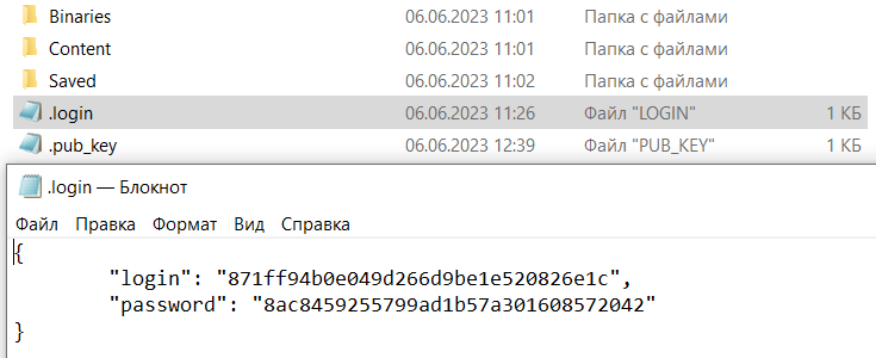


Рис. 4.8. Файл із зашифрованими даними автентифікації

Сценарій 2 – перевірка механік керування героями:

- *Створення героя*: на рівні вибору героя користувач натискає на кнопку "Create character". Після цього користувач обирає клас

героя, вказує ім'я героя та натискає кнопку "Create" (рис. 4.9). Очікуваний результат – в головному меню з'являється створений персонаж із вказаним іменем.

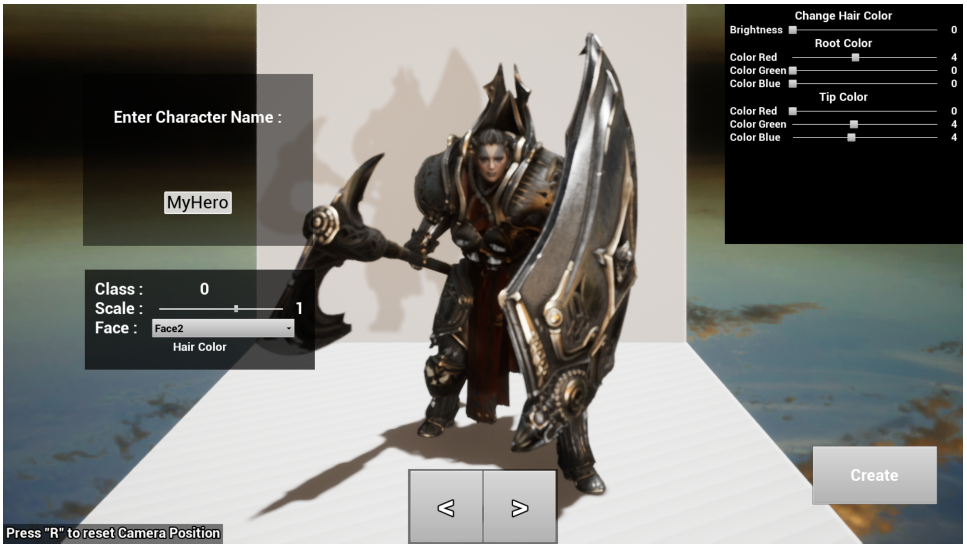


Рис. 4.9. Створення героя

- *Вибір героя*: користувач натискає на ім'я одного із вже створених героїв. Очікуваний результат – в центрі екрану з'являється обраний герой (рис. 4.10).

Рис. 4.10. Зміна героя

- *Приєднання до гри*: користувач обрав героя та натиснув на кнопку "Enter game". Очікуваний результат – обраний герой з'являється на головному рівні (рис. 4.11).

Рис. 4.11. Результат приєднання до гри

Сценарій 3 – перевірка основних механік симуляції:

- *Продаж товарів в магазині*: користувач знаходиться в меню магазина (рис. 4.12) та перетягує товар зі свого інвентаря в магазин. Очікуваний результат – товар зникає з інвентаря, грошовий баланс збільшується.

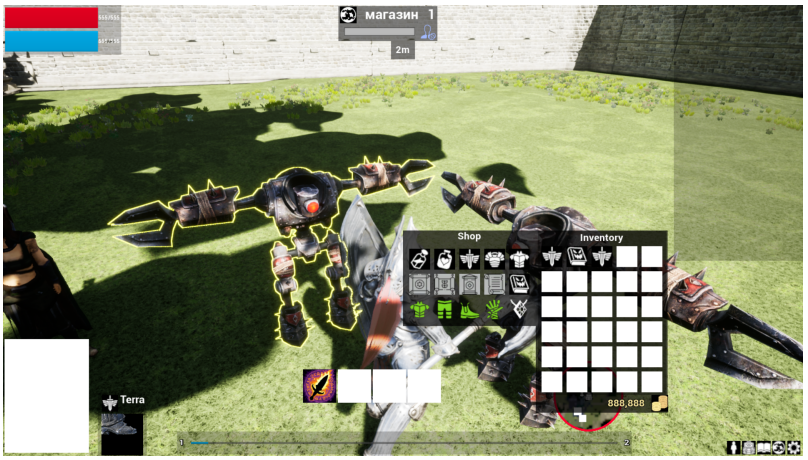


Рис. 4.12. Меню в магазині

- *Купівля товарів в магазині*: користувач знаходиться в меню магазина, клікає на товар в магазині та вказує кількість товару (рис. 4.13), який хоче купити. Очікуваний результат – зменшення грошового балансу, в інвентарі з'являється обраний товар.

Рис. 4.13. Меню купівлі товару в магазині

- *Додавання елемента екіпіровки*: користувач відкрив меню екіпіровки та інвентаря. Користувач перетягує предмет екіпіровки з інвентаря до відповідного слоту екіпіровки (рис. 4.14). Очікуваний результат – предмет зникає з інвентаря і залишається у відповідному слоті екіпіровки.

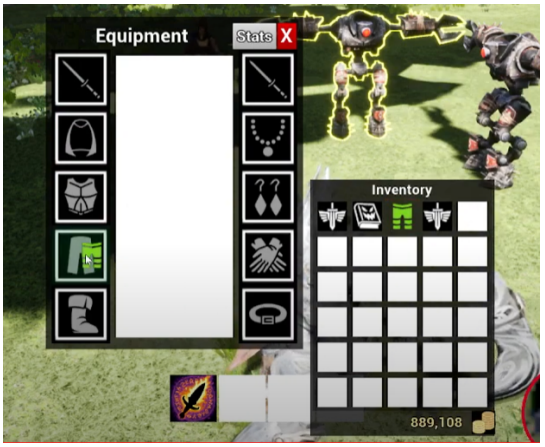


Рис. 4.14. Додавання елемента екіпіровки

- *Фокусування на об'єкті*: користувач дивиться в сторону об'єкту, який може бути знищений, та натискає кнопку tab. Очікуваний результат – об'єкт обводиться червоним контуром, зверху виводиться інформація про обрану ціль (рис. 4.15).

Рис. 4.15. Фокусування на об'єкті

- *Знищення ботів*: користувач сфокусувався на боті та забрав у нього достатню кількість здоров'я (рис. 4.16). Очікуваний результат – бот перестає рухатись, збільшується досвід героя.

Рис. 4.16. Нанесення шкоди боту

Тести майже всіх сценаріїв використання застосунку пройшли успішно, за винятком отримання досвіду героєм після знищення бота. Ця помилка буде усунена найближчим часом.

4.5. Шляхи вдосконалення системи

Покращення проєкту розглядається в таких напрямках як: поліпшення архітектури, оптимізація, виправлення помилок, вдосконалення штучного інтелекту ботів, впровадження квестової системи, створення механіки ближнього бою, додавання нових рівнів та локацій.

Під поліпшенням архітектури мається на увазі знаходження занадто великих модулів, які важко розширювати, та проведення декомпозиції таких модулів. Одним з таких модулів є PC_ArenaBase, який реалізує загальні механіки, які будуть спільними для всіх рівнів самої симуляції. Через недостатність досвіду роботи з системою Unreal Engine на початку роботи, цей модуль має більше сімдесяти методів. Цей модуль має бути декомпозованим з використанням можливостей класу ActorComponent. Такий крок значно спростить пошук помилок, додавання нових механік та оптимізацію вже існуючих.

Під час тестування було виявлено помилку, яка буде виправлена найближчим часом. Для подальшого вдосконалення системи, слід розробити систему повідомлень про помилки від гравців, активно відстежувати такі повідомлення, та оперативно виправляти їх. Це допоможе забезпечити стабільність та надійність гри, а також позитивний досвід для гравців.

Також варто розробити більш складні та реалістичні алгоритми штучного інтелекту для ботів. Це дозволить їм краще адаптуватися до дій гравців, приймати стратегічні рішення та забезпечувати виклик для гравців у битвах або завданнях. На поточний час штучний інтелект в розробленому застосунку являє собою просте пересування між контрольними точками.

Також необхідно додати квестову систему, нові захоплюючі квести, які розширять світ гри та нададуть гравцям цікаві завдання. Квести можуть включати розв'язання головоломок, збір рідкісних предметів або виконання складних місій. Система квестів допоможе утримувати гравців навіть при низькому онлайні.

Наостанок, для збільшення різноманіття симуляції, слід розширити світ гри шляхом створення нових унікальних локацій. Це можуть бути просторі відкриті місцевості, загадкові печери, міста або фантастичні світи. Нові локації додадуть різноманітності та можливості гравцям для дослідження.

ВИСНОВКИ

Розроблене програмне забезпечення являє собою симуляцію, яка пропонує механізми кооперації та взаємодії гравців проти ботів та один одного. Вимоги до проєкту були сформовані на основі, вивчення та аналізу існуючих програмних рішень, таких як World of Warcraft, The Elder Scrolls Online т Guild Wars2.

У ході роботи була розроблена архітектура трьох рівнів, кожен з яких має власне призначення. Перший відповідає за авторизацію та реєстрацію користувачів, другий – за створення та керування героями, і третій – головний світ симуляції, на якому відбувається, власно, взаємодія користувачів. Таке розбиття на рівні спрощує майбутнє масштабування симуляції.

Розроблена система реалізовує основні механіки жанру ММОРПГ, такі як торгівля, керування інвентарем, екіпіровкою, мапою та інші важливі елементи гри. Це створює зручні умови для взаємодії та спільної гри для користувачів.

Крім того, була спроєктована інфраструктура, що забезпечує безпечну передачу чутливих даних між користувачами. Це дозволяє запобігти можливість несанкціонованого доступу до особистої інформації.

У рамках проєкту була реалізована система автоматизованого налаштування серверів, що спрощує процес налаштування та підтримки ігрового середовища.

Отриманий продукт був протестований, і більшість виявлених помилок були виправлені, що гарантує стабільну та надійну роботу програмного забезпечення.

Також були сформовані шляхи майбутнього розвитку проєкту, що містять можливості розширення функціоналу, впровадження нових функцій та покращення взаємодії користувачів.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. The Future of the Video Game Market [Електронний ресурс]. – Режим доступу: https://www.marketsandmarkets.com/Market-Reports/video-games-market-197.html.
2. Статистика світового ринку відеоігор [Електронний ресурс]. – Режим доступу: https://www.statista.com/statistics/292056/video-game-market-value-worldwide/.
3. Статистика жанру рольових ігор (RPG) на SteamSpy [Електронний ресурс]. – Режим доступу: https://steamspy.com/genre/RPG.
4. World of Warcraft [Електронний ресурс]. – Режим доступу: https://worldofwarcraft.com/.
5. The Elder Scrolls Online [Електронний ресурс]. – Режим доступу: https://www.elderscrollsonline.com/.
6. Guild Wars 2 [Електронний ресурс]. – Режим доступу: https://www.guildwars2.com/.
7. Linux Game Development [Електронний ресурс]. – Режим доступу: https://www.gamedeveloper.com/programming/q-a-understanding-the-future-of-linux-game-development.
8. macOS Game Development [Електронний ресурс]. – Режим доступу: https://learn.unity.com/tutorial/mac-build.
9. Windows Game Development [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/windows/apps/games/.
10. Game Development with Unity [Електронний ресурс]. – Режим доступу: https://learn.unity.com/course/unity-advanced-game-development.
11. Unreal Engine Documentation [Електронний ресурс]. – Режим доступу: https://docs.unrealengine.com/

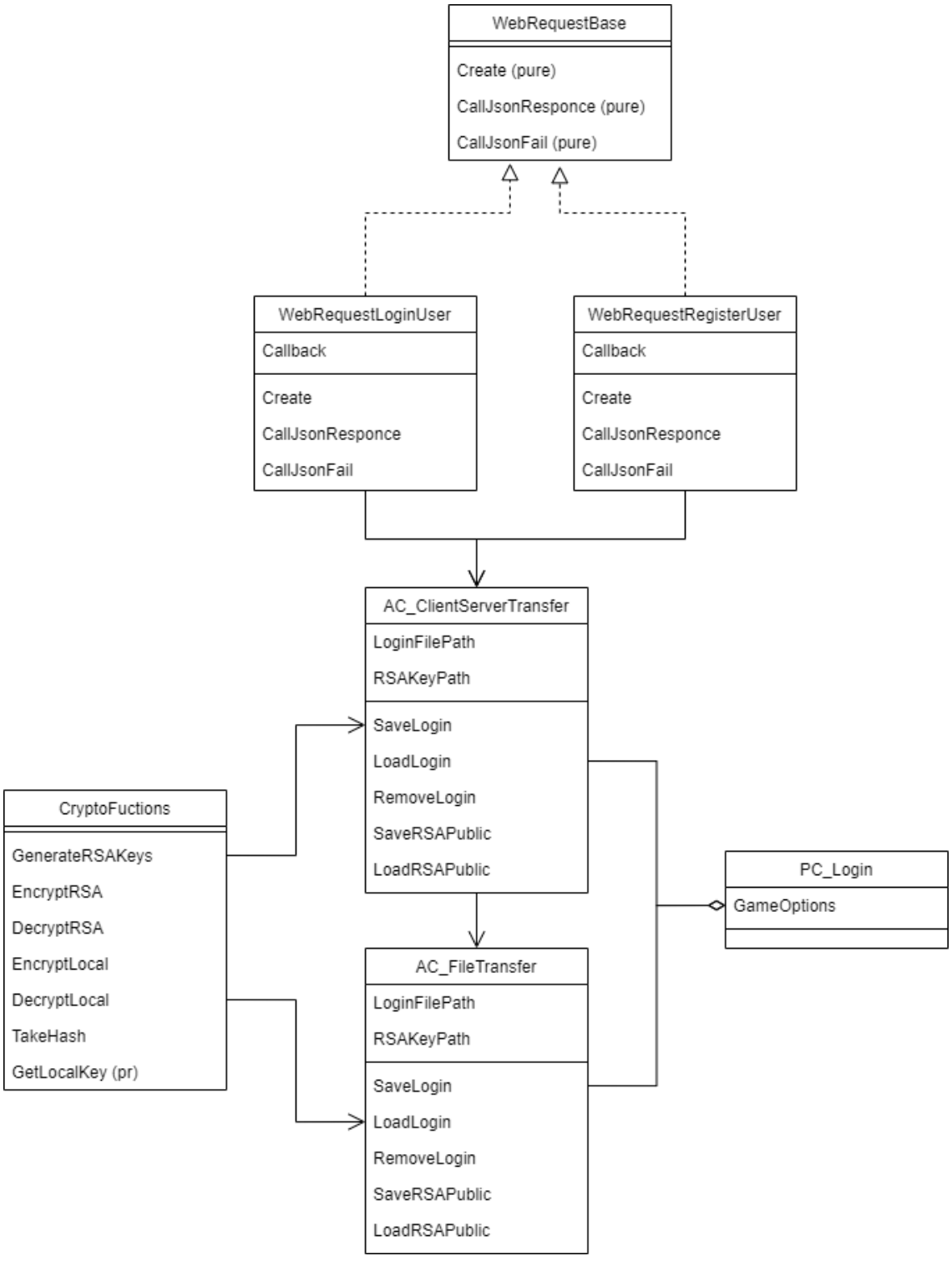
12. Програмування C++ в Unreal Engine [Електронний ресурс]. – Режим доступу: https://docs.unrealengine.com/en-US/Programming/Introduction/index.html.
13. Introduction to Blueprints in Unreal Engine [Електронний ресурс]. – Режим доступу: https://docs.unrealengine.com/en-US/Blueprints.
14. Database connection in Unreal Engine [Електронний ресурс]. – Режим доступу: https://www.unrealengine.com/marketplace/en-US/store/category/extensions/code-plugins/database.
15. MySQL Connector [Електронний ресурс]. – Режим доступу: https://dev.mysql.com/doc/connector-cpp/8.0/en/.
16. Backend Development with C++ [Електронний ресурс]. – Режим доступу: https://www.educative.io/blog/backend-development-with-cpp.
17. The Complete Node.js Developer Course [Електронний ресурс]. – Режим доступу: https://www.udemy.com/course/the-complete-nodejs-developer-course-2/.
18. PHP Object-Oriented Programming (OOP) [Електронний ресурс]. – Режим доступу: https://www.php.net/manual/en/language.oop5.php.
19. PostgreSQL: The World's Most Advanced Open Source Relational Database [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/.
20. MongoDB Documentation [Електронний ресурс]. – Режим доступу: https://docs.mongodb.com/.
21. MySQL Documentation [Електронний ресурс]. – Режим доступу: https://dev.mysql.com/doc/.
22. Neo4j [Електронний ресурс]. – Режим доступу: https://neo4j.com/
23. MD5 Algorithm [Електронний ресурс]. – Режим доступу: https://www.rfc-editor.org/rfc/rfc1321.
24. SHA-256 Algorithm [Електронний ресурс]. – Режим доступу: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf.

25. AES Encryption Algorithm [Електронний ресурс]. – Режим доступу: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf.
26. RSA Encryption Algorithm [Електронний ресурс]. – Режим доступу: https://www.rfc-editor.org/rfc/rfc8017.
27. Elliptic Curve Cryptography (ECC) [Електронний ресурс]. – Режим доступу: https://www.globalsign.com/en/ssl-information-center/what-is-elliptic-curve-cryptography.

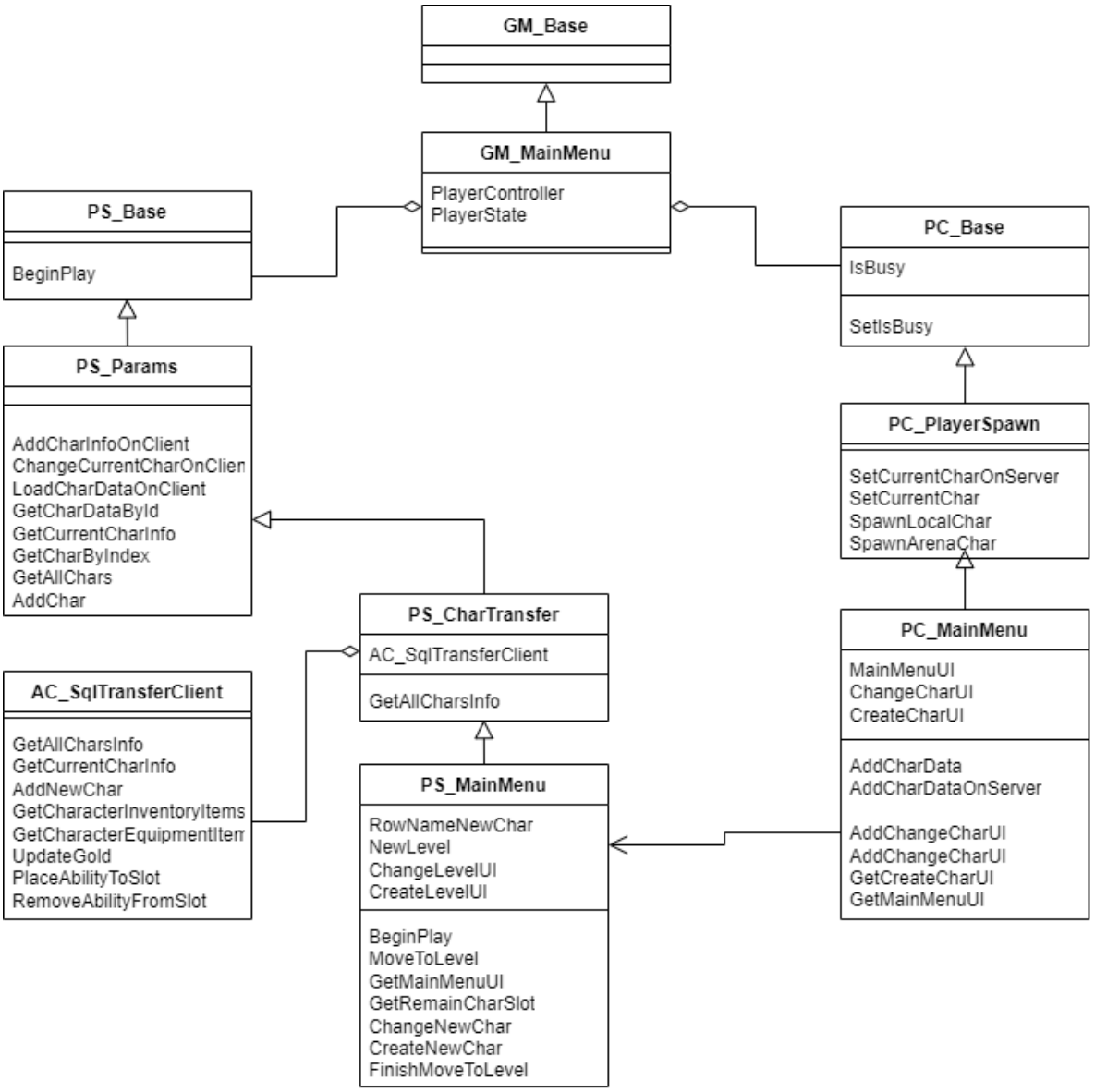
ДОДАТКИ

Додаток 1

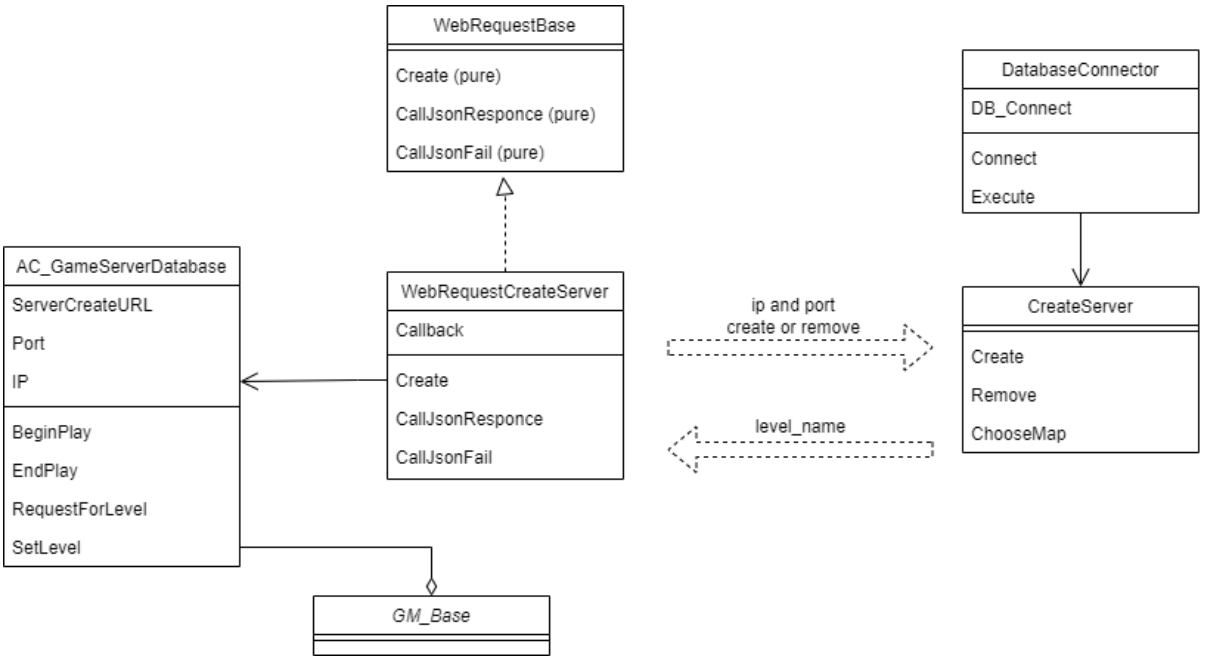
Копії графічних елементів



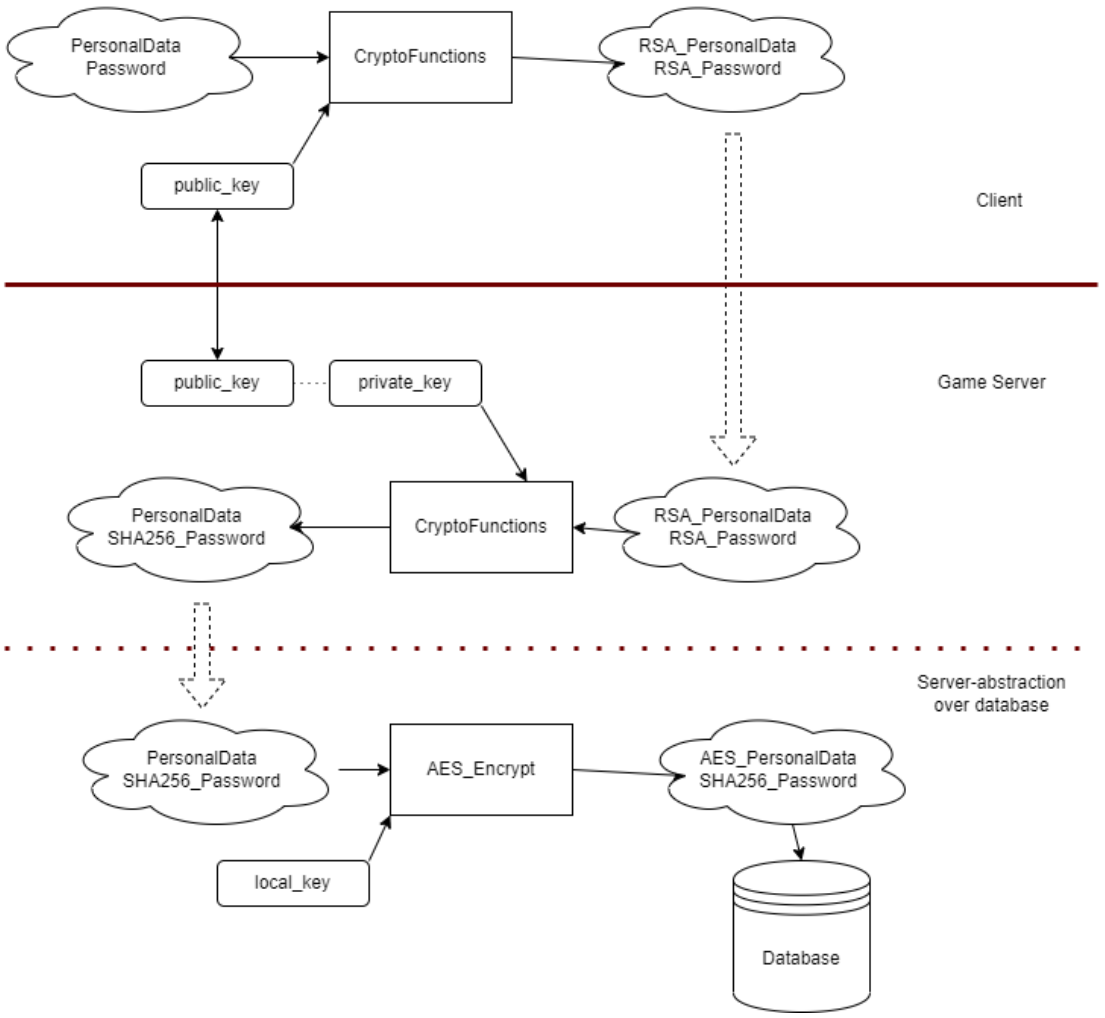
ДП.045480-06-99
Програмне забезпечення для симуляції віртуальної багатокористувацької рольової взаємодії. Рівень авторизації. UML-діаграма класів



ДП.045480-07-99
Програмне забезпечення для симуляції віртуальної багатокористувацької рольової взаємодії. Рівень головного меню. UML-діаграма класів



Система автоматизованого налаштування серверів
Деркач С.Д., група КП-93



Інфраструктура передачі чутливих даних
Деркач С.Д., група КП-93

Додаток 2

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

Програмне забезпечення для симуляції віртуальної багатокористувацької рольової взаємодії

Виконав: студент групи КП-93 Деркач Станіслав Дмитрович

Керівник: доцент кафедри ПЗКС, к. ф.-м. н, доцент Нещадим О.М.

Київ – 2023

ПОСТАНОВКА ЗАДАЧІ

Мета проекту: розробити програмне забезпечення для симуляції віртуальної багатокористувацької рольової взаємодії.

Завдання:

1. Проаналізувати:
   - існуючі рішення
   - існуючі інструменти розробки
2. Розробити:
   - Клієнт-серверну архітектуру
   - Основні механіки
3. Протестувати розроблений застосунок

АКТУАЛЬНІСТЬ

Актуальна проблема:

- Нестача соціальної взаємодії
- Загострення на фоні пандемії та війни

Спосіб вирішення:

Розроблення застосунку, який може задовольнити одну з основних соціальних потреб, таку як відчуття єдності з іншими людьми.

Існуючі рішення

Месенджери:

Подібні системи:

Засоби реалізації

Графічний двигун:

Unreal Engine (UE)

Мова розробки основної частини:

C++ і Blueprints

СУБД:

MySQL

Мова розробки САБ:

PHP

Криптографічні алгоритми:

SHA-256, RSA, AES

UML-діаграма рівня авторизації

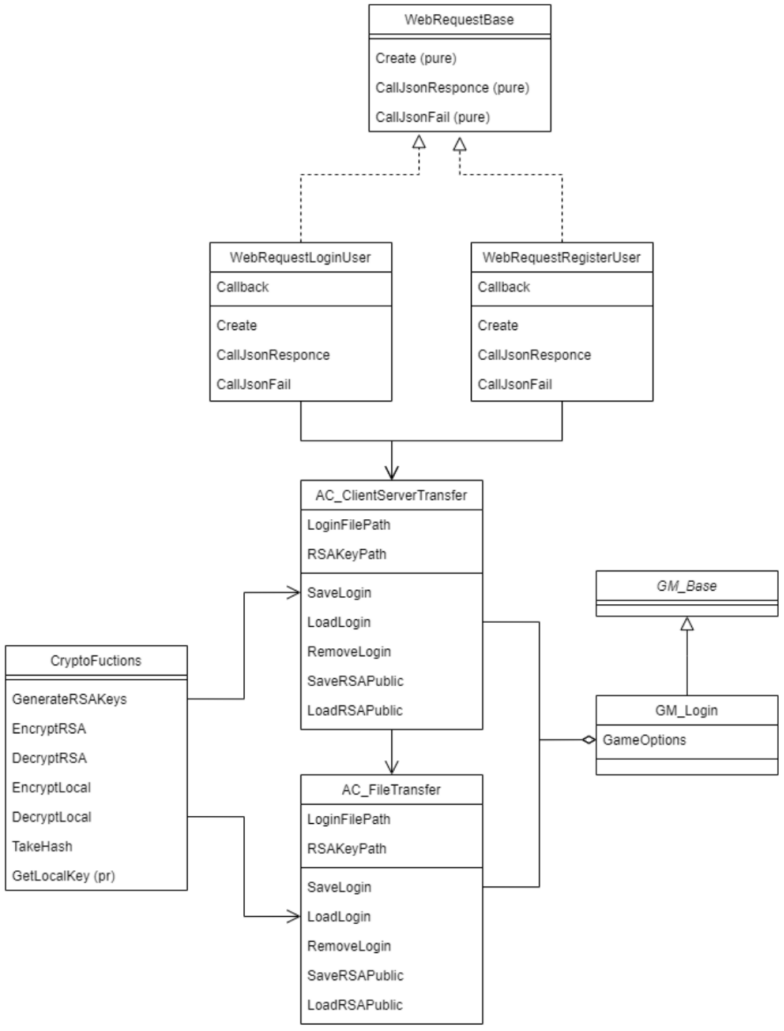
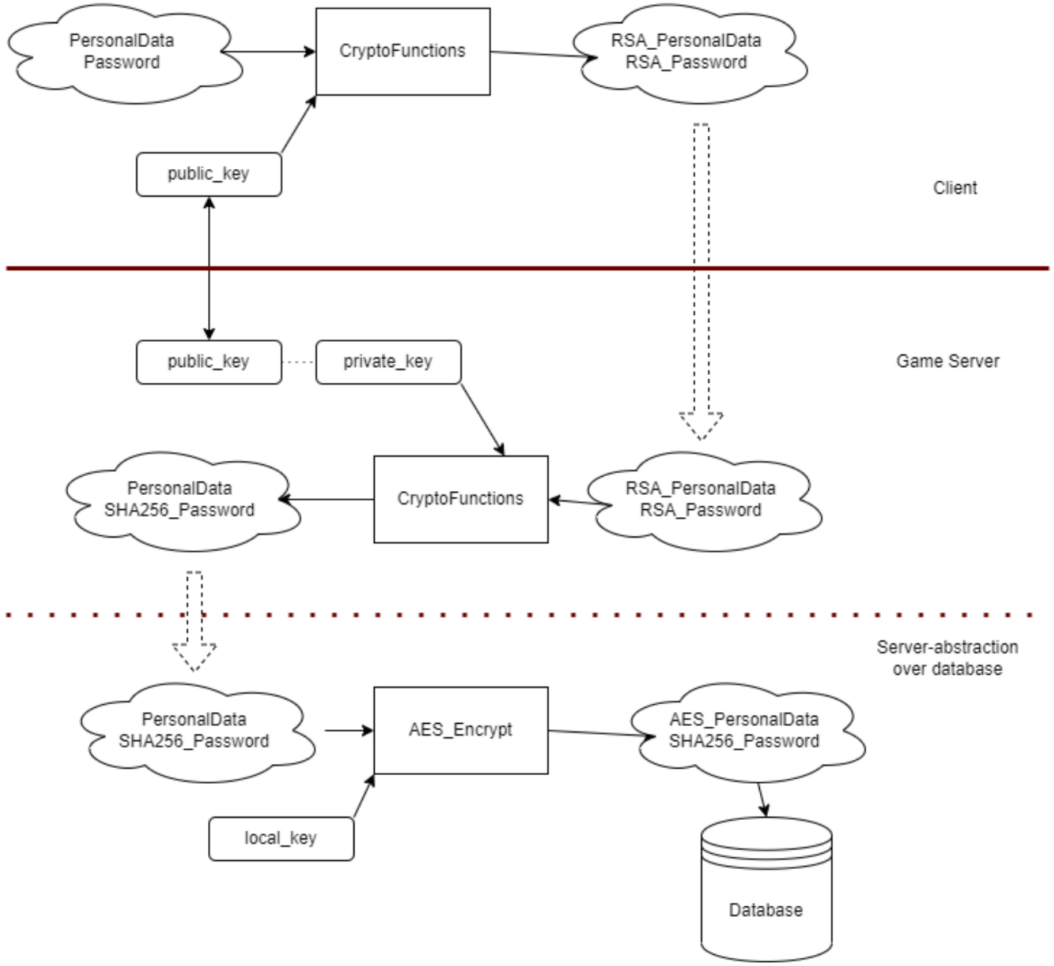
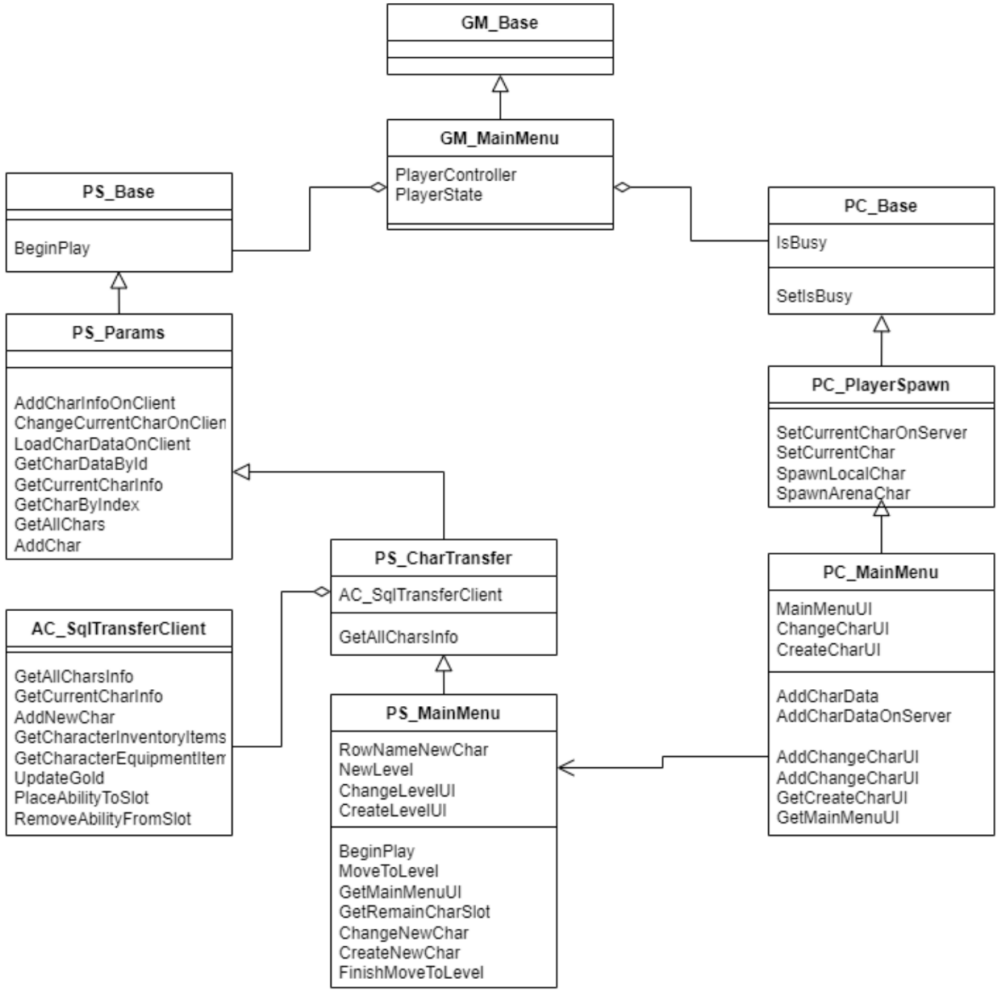


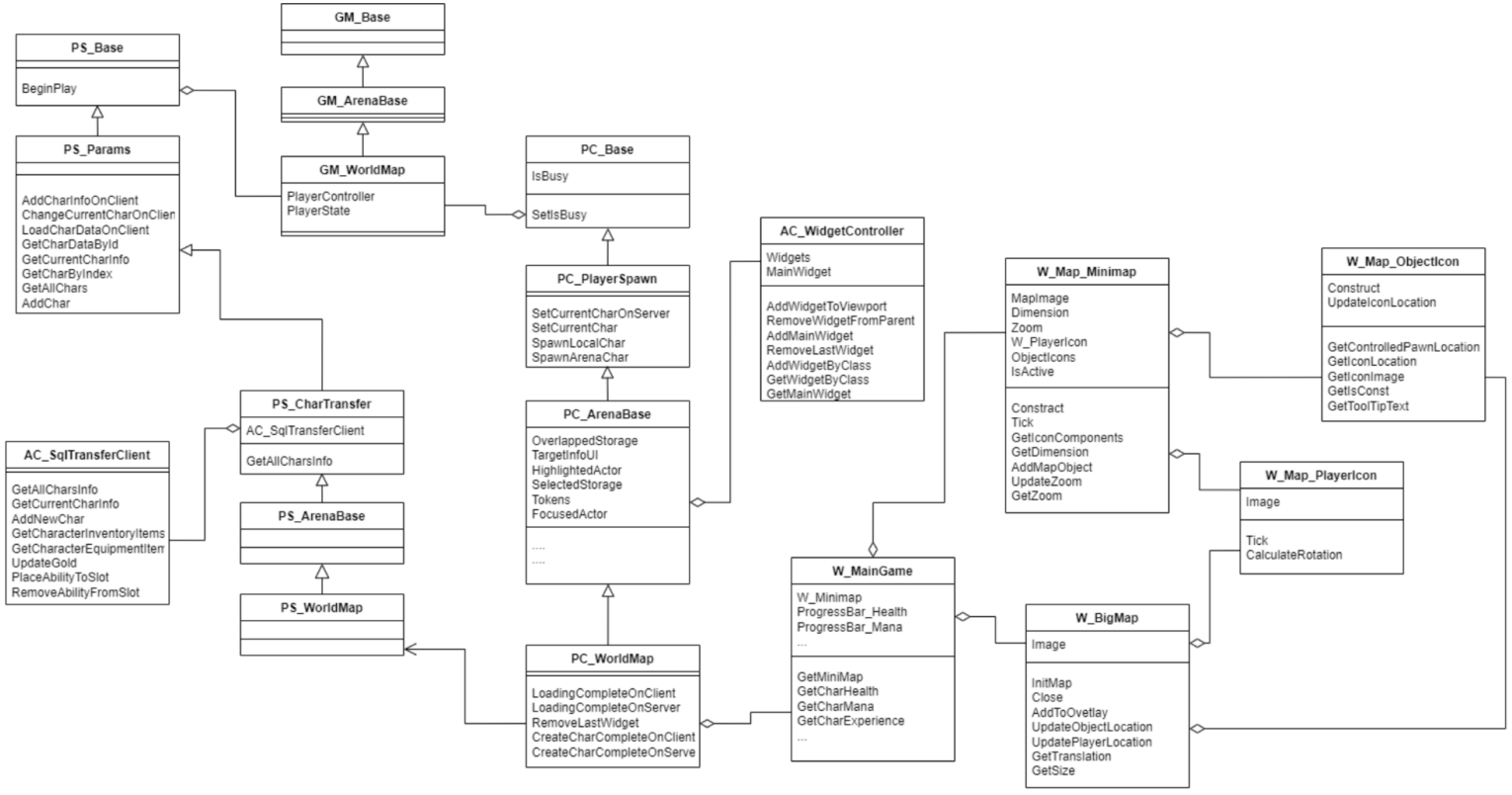
Схема передачі чутливих даних



UML-діаграма рівня вибору героя

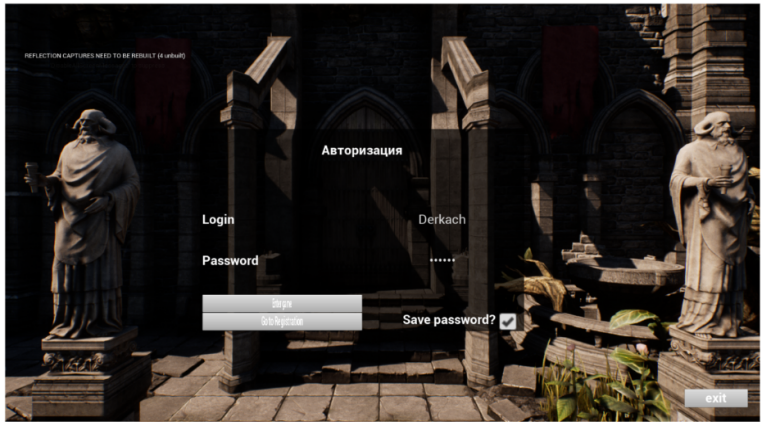


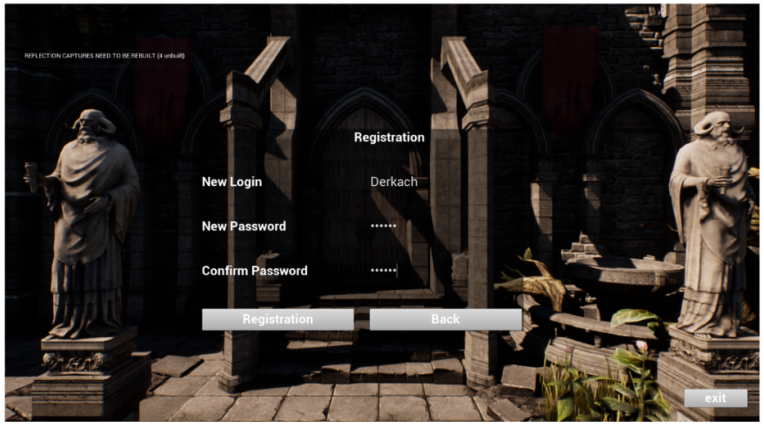
UML-діаграма центральної частини рівня світу



Приклади роботи

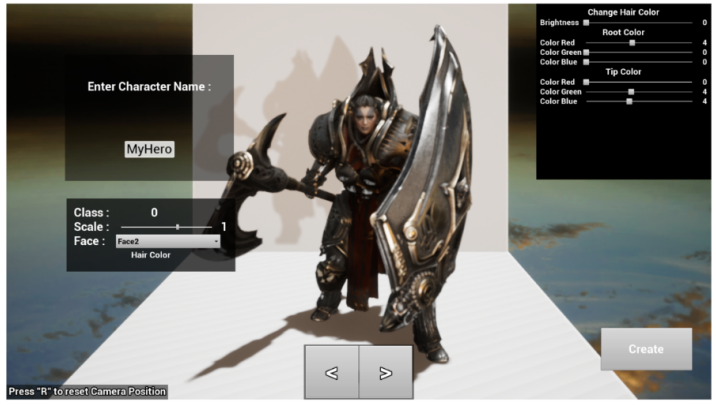
Рівень авторизації:





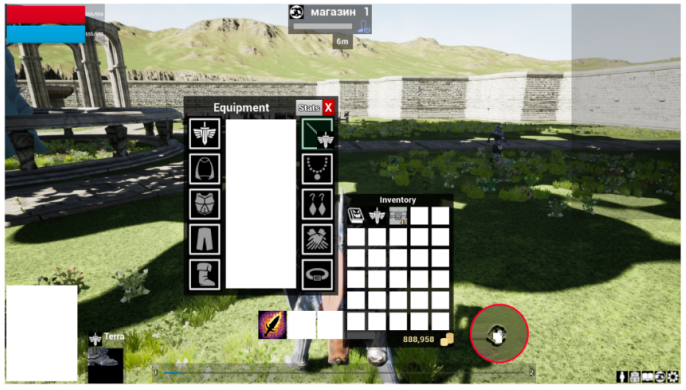
Приклади роботи

Рівень головного меню:



Приклади роботи

Рівень відкритого світу:



Приклади роботи

Повна демонстрація проекту за QR-кодом або посиланням:

https://youtu.be/zaxHsJwEgsU

Результати тестування

Під час тестування методом чорного ящику було виявлено наступні помилки:

- Спотворення даних після дешифрування (+)
- Боти не зникають після знищення (-)
- Неправильна валідація предметів екіпіровки (+)
- Знищення ворогів не збільшує досвід героя (-)

Напрямки розвитку

- Поліпшення архітектури
- Виправлення знайдених помилок
- Вдосконалення штучного інтелекту ботів
- Впровадження квестової частини
- Створення механіки ближнього бою
- Створення нових локацій

ВИСНОВКИ

1. Проаналізовано існуючі програмні рішення.
2. Розроблено архітектуру трьох рівнів, та механізму безпечної передачі даних.
3. Реалізовано систему автоматизованого налаштування серверів.
4. Реалізовано основні механіки симуляції.
5. Отриманий продукт протестовано, більша частина знайдених помилок виправлена
6. Сформовано шляхи майбутнього розвитку проекта

UNICHECK
by Turnitin

Національний технічний університет України "Київський політехнічний інститут"

Ім'я користувача:
Нещадим Олександр Миколайович

ID перевірки:
1015631191

Дата перевірки:
17.06.2023 03:17:35 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
17.06.2023 13:33:27 EEST

ID користувача:
100012517

Назва документа: **Diploma_for_antiplugiat**

Кількість сторінок: **50** Кількість слів: **10508** Кількість символів: **76888** Розмір файлу: **240.15 KB** ID файлу: **1015277672**

1.72%
Схожість

Найбільша схожість: **0.18%** з джерелом з Бібліотеки (ID файлу: **1015242566**)

0.32% Джерела з Інтернету	21	Сторінка 52
1.4% Джерела з Бібліотеки	51	Сторінка 52

0% Цитат

Не знайдено жодних цитат

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Дякую за увагу!

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2022 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ СИМУЛЯЦІЇ ВІРТУАЛЬНОЇ БАГАТОКОРИСТУВАЦЬКОЇ РОЛЬОВОЇ ВЗАЄМОДІЇ

Програма та методика тестування

ДП.045440-04-51

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Олександр НЕЩАДИМ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____Станіслав ДЕРКАЧ

2022

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Об'єктом випробувань є програмне забезпечення для симуляції віртуальної багатокористувацької рольової взаємодії, розроблене на базі рушія Unreal Engine.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1. Цілісність даних від час їх запису в БД.
2. Коректність взаємодії з базою даних.
3. Тестування верстки для різних екранів.
4. Тестування механіки керування героями.
5. Тестування механік головного рівня симуляції.
6. Відповідність вимогам технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Gray Box Testing. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

1. Функціональне тестування.
2. Модульне тестування.
3. Тестування інтерфейсу.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Працездатність додатку перевіряється шляхом:

1. Динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати.

2. Динамічного ручного тестування на відповідність функціональним вимогам.
3. Статичного тестування коду.
4. Тестування при різному навантаженні.
5. Тестування стабільності роботи на різних пристроях.
6. Тестування зручності використання.
7. Тестування інтерфейсу.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2023 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ СИМУЛЯЦІЇ ВІРТУАЛЬНОЇ БАГАТОКОРИСТУВАЦЬКОЇ РОЛЬОВОЇ ВЗАЄМОДІЇ

Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Олександр НЕЩАДИМ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Станіслав ДЕРКАЧ

2023

ЗМІСТ

1. Опис структури програмного забезпечення

Програмне забезпечення для симуляції віртуальної багатокористувацької рольової взаємодії складається з трьох рівнів. Кожен рівень можна умовно поділити на модулі.

Модулі рівня авторизації:

- менеджер серверів;
- модуль криптографічних функцій;
- модуль клієнт-серверної взаємодії;
- модуль взаємодії з САБ.

Модулі рівня керування героями:

- модуль керування героями;
- модуль керування поточної картою;
- модуль клієнт-серверної взаємодії.

Модулі головного рівня симуляції:

- центральний логічний модуль;
- модуль керування поточним стану героя;
- модулі механик: мапа, інвентар, магазин тощо.

2. Процедура авторизації та реєстрації користувача

У разі згоди з політикої конфіденційності, користувач має змогу зареєструватись або авторизуватись. При авторизації користувач має можливість запам'ятати свої дані аутентифікацію (рис. 1). Якщо цей прапорець буде стояти та користувач успішно авторизується, при наступному відкритті застосунку поле логіну та паролю автоматично заповниться даними з попередньої автентифікації.

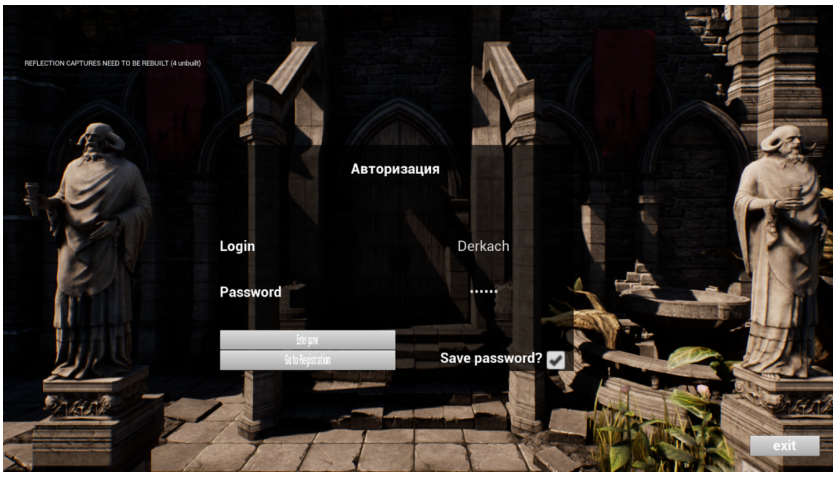


Рис. 1. Сторінка автентифікації

Для реєстрації користувач має відкрити окно реєстрації та заповнити запропоновані поля (рис. 2). Якщо вказаний логін раніше не був зареєстрований, та підтвердження паролю співпадає із паролем, користувач побачить повідомлення про успішну реєстрацію.

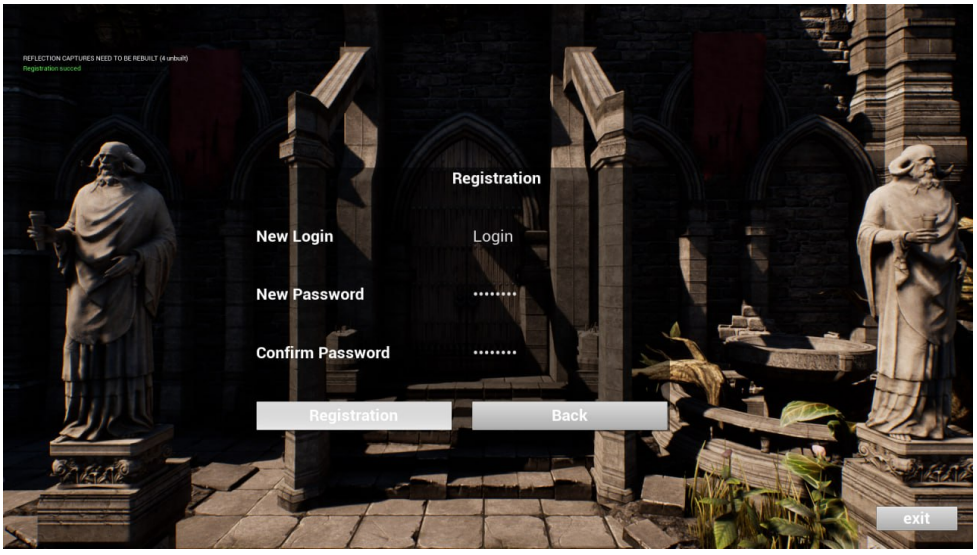


Рис. 2. Успішна реєстрація

3. Вибір та створення героя

У разі успішної авторизації, користувач бачить своїх створених героїв (рис. 3), має можливість створити нового або обрати вже існуючого.

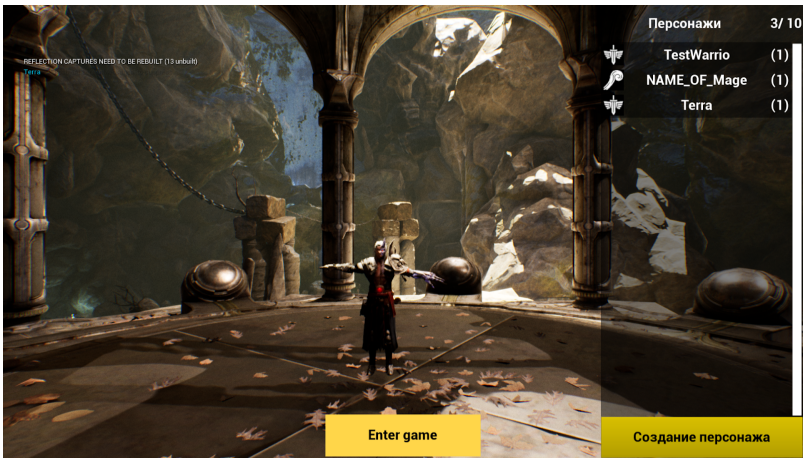


Рис. 3. Рівень керування героями

На поточний час можна створити героя одного з двох існуючих класів: маг (рис. 4) та воїн. Для воїна доступно дві моделі (рис. 5).

Рис. 4. Створення мага.

Рис. 5. Створення воїна

Після вибору класа та моделі героя, користувач має увести ім'я свого героя (рис. 6). Також наявний інтерфейс для редагування зовнішності героя, але на поточний час такий функціонал не реалізовано.

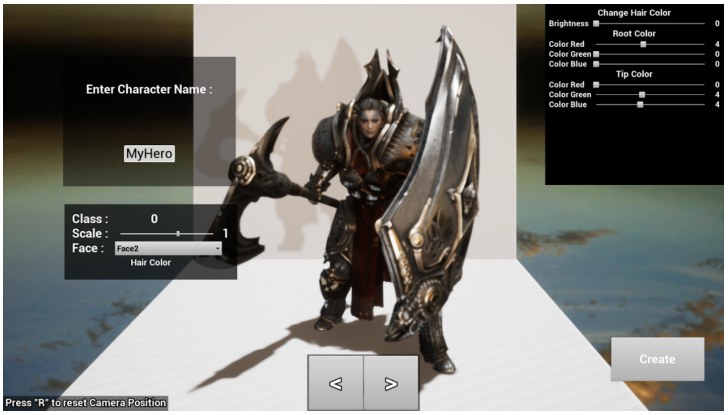


Рис. 6. Налаштування нового героя

Якщо користувач готов долучитись до гри, він натискає кнопку "Enter game", після чого приєднується до самої симуляції.

4. Використання механік самої симуляції

Після приєднання до відкритого світу, користувач з'являється на карті. Щоб керувати камерою, необхідно натиснути та утримувати праву кнопку миші (рис. 7).

Рис. 7. Поворот камери

За допомогою системних кнопок зліва знизу можна подивитись інвентар (рис. 8), поточний баланс, поточну екіпіровку (рис. 9), мапу світу. Для закриття віджетів необхідно натиснути кнопку ctrl.

Рис. 8. Перегляд інвентаря

Рис. 9. Перегляд екіпіровки

Для взаємодії з об'єктами, на них треба підійти до них достатньо близько та клікнути по об'єкту лівою кномишою. Інтерактивні об'єкти підсвічуються при наведенні (рис. 10).

Рис. 10. Підсвітлення інтерактивних об'єктів

Знизу зліва можна побачити міні-мапу (рис. 10). Для зміни масштабу, треба навести на неї машею та скористатисом колесиком (рис. 11).

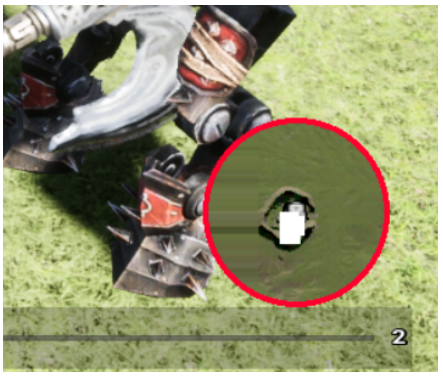


Рис. 10. Відображення міні-мапи

Рис. 11. Міні-мапа після змінення масштабу

Щоб скористатися послугами магазину, треба провзаємодіяти з відповідним персонажем, після чого відкриється меню магазину (рис. 12). Тут ви можете купувати або продавати речі та екіпіровку.

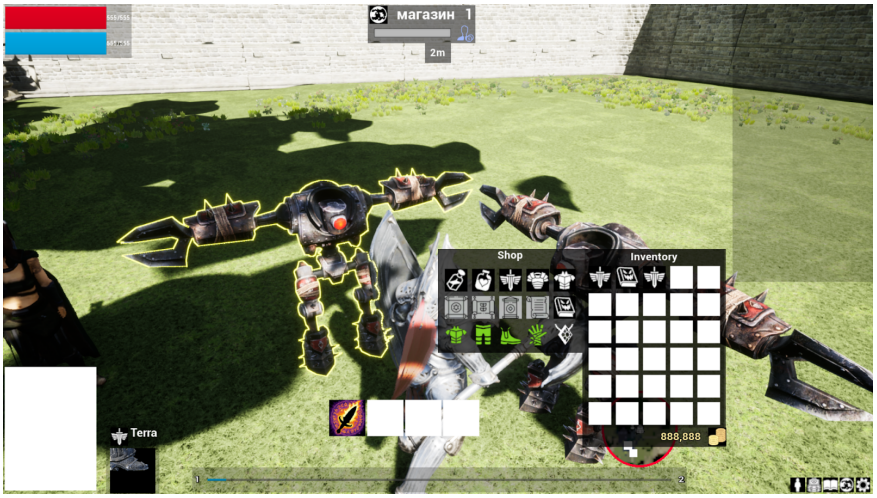


Рис. 12. Меню магазину

Для того щоб сфокусуватись на потенційному ворогу треба натиснути tab, після чого ворог буде обведений червоним, а зверху можна бути бачити інформацію про свою ціль (рис. 13).

Рис. 13. Фокусування на об'єкті

Фокус на об'єкті дозволяє його атакувати. Для атаки можна використовувати заклинання. Для цього має бути виділений ворог, а гравець має натиснути на слот із заклинанням, який хоче активувати. Спливаючі числа означають нанесена шкода (рис. 14).

Рис. 14. Атака об'єкта фокусу

Після знищення ворога, з'являється можливість його обшукати та забрати собі знайдені речі (рис. 15). Кількість та види таких предметів залежить від вбитої цілі, проте фактичний лут завжди залишається випадковим.

Рис. 15. Обшук бота після знищення

Перед тим як забрати речі собі, або перекласти свої речі до іншого, користувач може обрати кількість речей, які він хоче перекласти (рис. 16).

Рис. 16. Вибір кількості речей для отримання

Для того щоб одягнути елемент екіпіровки, треба відкрити інвентар та меню екіпіровки, після чого перетягнути предмет з інвентаря у відповідний слот екіпіровки (рис. 17). Якщо цей предмет підходить до класу гравця, він залишиться у відповідному слоті.

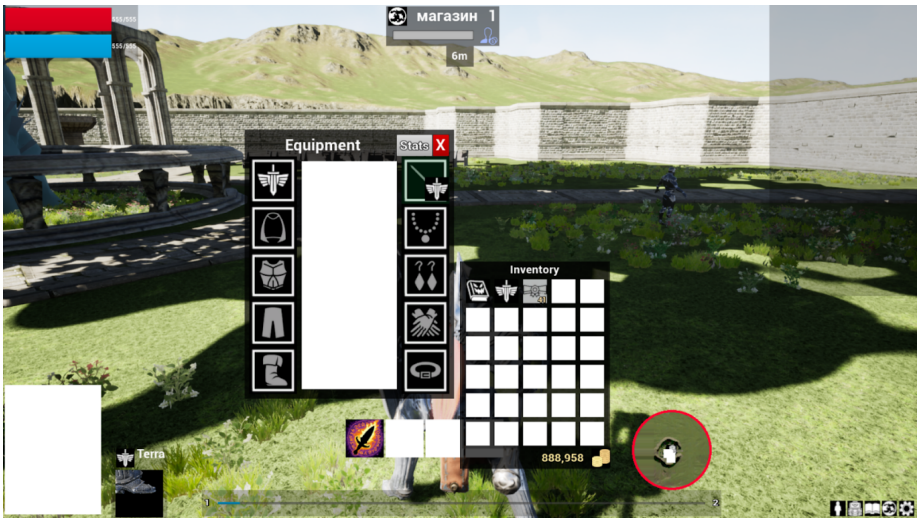


Рис. 17. Одягання елементу екіпіровки