

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПІДБОРУ
ГРАВЦІВ У КОМП'ЮТЕРНИХ ІГРАХ
SOFTWARE TO SELECT PLAYERS IN COMPUTER GAMES

Виконав(ла): студент(ка) 4 курсу, групи КНТ-229

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

СЕПЕТИЙ О.В.

(ПРИЗВИЩЕ та ініціали)

Керівник ГОФМАН Є.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент СКРУПСЬКИЙ С.Ю.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)

Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2023 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

СЕПЕТОГО Олега Володимировича

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення для підбору гравців у комп'ютерних іграх. Software to Select Players in Computer Games

керівник проєкту (роботи) к.т.н., ГОФМАН Євгеній Олександрович,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 04 ” квітня 2023 року № 87

2. Строк подання студентом проєкту (роботи) 01 червня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ГОФМАН Є.О., доцент		
Нормоконтроль	ЛИПОВЕЦЬ М.В., асистент		

7. Дата видачі завдання “4” квітня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Олег СЕПЕТИЙ
(Імя ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Євгеній ГОФМАН
(Імя ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
92 с., 8 табл., 29 рис., 3 дод., 20 джерел.

VISUAL STUDIO, БАГАТОКОРИСТУВАЦЬКА ГРА, БАЗА ДАНИХ,
ІГРОВИЙ РУШІЙ, ПОШУК МАТЧУ.

Об'єкт дослідження – процес розробки комп'ютерних ігор.

Предмет дослідження – програмні засоби для підбору гравців у комп'ютерних іграх.

Метою роботи є розробка програмного забезпечення для підбору гравців у комп'ютерних іграх.

Матеріали, методи та технічні засоби: мова програмування C++, середовище розробки Visual Studio, мережева бібліотека Boost, система керування базами даних PostgreSQL.

Результати. Розроблено програмне забезпечення для матчмейкінгу за допомогою мови програмування C++, що дозволяє виконувати підбір гравців у комп'ютерних іграх.

Практична цінність роботи полягає у розробці програмного забезпечення для підбору гравців у комп'ютерних іграх.

Висновки. Виконано проєктування програмного забезпечення для підбору гравців у комп'ютерних іграх. Розроблено програмне забезпечення для підбору гравців у комп'ютерних іграх. Здійснено тестування розробленого програмного забезпечення для підбору гравців у комп'ютерних іграх.

Галузь використання – комп'ютерні ігри.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 92 pages, 8 tables, 29 figures, 3 appendixes, 20 sources.

VISUAL STUDIO, MULTIPLAYER GAME, DATABASE, GAME ENGINE, MATCH SEARCH.

The object of research is the process of computer games development.

The subject of the research is software for selecting players in computer games.

The purpose of this work is to develop the software for the selection of players in computer games.

Materials, methods and technical tools: C++ programming language, Visual Studio development environment, Boost network library, PostgreSQL database management system.

Results. The software tools for matchmaking based on the C++ programming language, which allows the selection of players in computer games have been developed.

The practical value of the work lies in the development of software for the selection of players in computer games.

Conclusions. The software for the selection of players in computer games has been designed. The software for the selection of players in computer games has been developed. The developed software for the selection of players in computer games has been tested.

Scope of use – computer games.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області.....	10
1.1 Поняття про комп'ютерні ігри та матчмейкінг	10
1.2 Дослідження програмних засобів для підбору гравців у комп'ютерних іграх.....	23
1.3 Висновки за розділом 1	30
2 Матеріали і методи	32
2.1 Вибір мови програмування та інших технологій розробки	32
2.2 Вибір мережевої бібліотеки.....	34
2.3 Вибір середовища розробки для створення програмного забезпечення підбору гравців у комп'ютерних іграх	37
2.4 Висновки за розділом 2	38
3 Опис програми	40
3.1 Структура процесів програми підбору гравців у комп'ютерних іграх ...	40
3.2 Функціонування програмного забезпечення для підбору гравців у комп'ютерних іграх	41
3.3 Основні рішення щодо реалізації програмного забезпечення для підбору гравців у комп'ютерних іграх	46
3.4 Висновки за розділом 3	51
4 Експлуатація, тестування та експериментальне дослідження програми.....	52
4.1 Призначення й умови застосування програми	52
4.2 Характеристики програми та її встановлення	53
4.3 Інструкція по експлуатації програми.....	53
4.3.1 Звернення до програми.....	53
4.3.2 Вхідні й вихідні дані	55
4.3.3 Повідомлення.....	55
4.4 Виконання програмного забезпечення для збирання даних з вебсайту .	56

4.5 Тестування програмного забезпечення для підбору гравців у комп'ютерних іграх	59
4.6 Висновки за розділом 4	62
Висновки	63
Перелік джерел посилань	65
Додаток А Технічне завдання	67
Додаток Б Фрагмент тексту програми	72
Додаток В Слайди презентації	85

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

IDE – інтегрована середовище розробки;

MMOG – massively multiplayer online game (Масова багатокористувачка онлайн-гра);

SQL – structured query language;

БД – база даних;

ОЗП – оперативний запам'ятовуючий пристрій

ПЗ – програмне забезпечення;

ПК – персональний комп'ютер;

СКБД – систем управління базою даних.

ВСТУП

У наш час широко застосовуються програмні засоби для матчмейкінгу (matchmaking), що дозволяють підтримати процес, за допомогою якого гравці збираються разом у групу для гри в онлайн-ігри зі схожим рівнем навиків та досвіду. Програмні засоби матчмейкінгу зазвичай використовують різні критерії для пошуку подібних гравців, такі як рейтинг, рівень вмінь, географічне положення тощо. Мета матчмейкінгу полягає в забезпеченні балансу та рівності шансів між гравцями, що грають один з одним. Використання ефективних засобів матчмейкінгу у програмному забезпеченні комп'ютерних рівень може підвищити рівень задоволеності від комп'ютерної гри, сприяти більшій активності гравців, що, у свою чергу, може підвищити рівень доходів власників відповідного комп'ютерного бізнесу [1]-[5].

У дипломній кваліфікаційній роботі бакалавра розв'язується актуальне завдання розробки програмного забезпечення для підбору гравців у комп'ютерних іграх.

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для підбору гравців у комп'ютерних іграх;
- здійснити проектування програмного забезпечення для підбору гравців у комп'ютерних іграх;
- створити програмне забезпечення для підбору гравців у комп'ютерних іграх;
- виконати тестування розробленого програмного забезпечення для підбору гравців у комп'ютерних іграх.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття про комп'ютерні ігри та матчмейкінг

Комп'ютерна гра – це гра, яка є керованою комп'ютером (програмним забезпеченням), у якій гравці з метою розваги взаємодіють з об'єктами або іншими гравцями, що відображаються на екрані [1]-[5].

Відеоігри можна використовувати як для веселощів і розваг, так і для змагань і комп'ютерного навчання. Деякі відеоігри призначені для розвитку моторики та координації рук і очей [1]-[5]. Перші відеоігри з'явилися в 1950-1960-х роках, а в 1970-х виникла ціла індустрія, заснована на відеоіграх. Але до 1983 року було зроблено так багато ігор низької якості, що індустрія відчувала труднощі. Ця проблема змусила розробників і видавців відеоігор приділяти більше уваги своїм відеоіграм, і до 2000 року більшість відеоігор виробляли великі компанії. Однак розвиток Інтернету дозволив молодим розробникам почати створювати власні ігри [1]-[5].

Роками відеоігри вважалися розвагою чи розвагою, але за останнє десятиліття кіберспорт почав набирати популярність. Електронні види спорту або Esports – це змагання між професійними гравцями у відеоігри [1]-[5].

Дуже прості ігри, такі як хрестики-нулики, можна було запрограмувати на ранніх комп'ютерах і використовували, щоб показати, наскільки потужними можуть бути комп'ютери. Одним із найбільш недооцінених аспектів відеоігор є те, що багато хто з оригінальних розробників не очікували, що вони будуть популярними, і використовували їх для тестування можливостей комп'ютерів [1]-[5].

Одним із найважливіших аспектів сучасного ринку відеоігор є інновації, тому що люди з більшою ймовірністю купуватимуть нову гру, якщо розробник надасть їм щось нове. Це призвело до швидкої зміни консолей відеоігор, таких як приставки. За більш ніж 50-річну історію відеоігор було дев'ять поколінь обладнання, кожне з яких було кращим за попереднє [1]-[5].

У перші роки розвитку ігор ринок домашніх консолей був дуже різноманітним, на ньому конкурувало багато компаній. Наприкінці 1990-х років багато старих компаній, що займаються відеоіграми, збанкрутували або перейшли на інші ринки, а їх замінили нові компанії [1]-[5].

Відеоігри також мають різні жанри або формати. У деяких іграх від гравців потрібно керувати персонажами та виконувати відповіді чи місії, тоді як в інших від гравців вимагається використовувати стратегію, щоб вести в бій величезні армії. Деякі ігри запрошують гравців керувати такими середовищами, як парки чи міста, тоді як інші дозволяють гравцям грати в імітовані види спорту з друзями онлайн [1]-[5].

У наш час активно розвивається кіберспорт, тобто змагальні відеоігри. Вони часто організовують турніри між професійними гравцями з великими пулами. Жанри відеоігор, які використовуються в кіберспорті, дуже різноманітні, оскільки існують такі кіберспортивні турніри, як шутери від першої особи, файтинги, карткові ігри, стратегії в реальному часі та багатокористувацькі екшн-ігри [1]-[5].

Як і справжній спорт, кіберспорт вимагає багато часу, уваги та тренувань, щоб змагатися на професійному рівні. Багато успішних учасників кіберспорту працюють повний робочий день у сфері освіти та заробляють гроші за допомогою трансляцій і спонсорства [1]-[5].

Існує багато способів використання відеоігор для навчання та викладання. Багато відеоігор включають вирішення проблем або мозковий штурм. Використання пристрою введення, такого як контролер або клавіатура, може допомогти покращити моторику та координацію рук і очей [1]-[5].

Деякі відеоігри розроблені спеціально для освітніх цілей і навчають гравців, як грати в гру. Такі предмети, як історія, географія, наука, музика та мистецтво, можна викладати за допомогою документальних фільмів і новин, так само, як їх викладають за допомогою відеоігор [1]-[5].

У наш час існують багато відеоігор різних жанрів. Жанри відеоігор – це категорії ігор, які мають подібний ігровий процес і ігрову механіку.

Наприклад, рольова гра (інакше відома як RPG) – це класичний жанр відеоігор, у який щодня грають мільйони людей. Спільним у більшості рольових ігор є те, що ви граєте за персонажа, який набуває сили та досвіду, долаючи дедалі складніші виклики та рівні [1]-[5]. Більшість жанрів відеоігор також мають піджанри, які допомагають розділити жанр на вужчі та вузькі категорії. Наприклад, піджанри жанру RPG включають Action RPG, MMORPG, Roguelike RPG, Tactical RPG, Sandbox RPG тощо [1]-[5].

Проаналізуємо жанри відеоігор.

Найпоширенішим жанром є екшн-ігри (Action Games). Екшн-ігри є одним із найпопулярніших жанрів відеоігор у світі, і зазвичай містять фізичні випробування, які долаються стрибками, бійками та стрільбою. Відмінна координація рук і очей і рефлексивні посмикування зазвичай потрібні в екшн-іграх, а дія зазвичай обертається навколо персонажа користувача, який повинен пройти серію випробувань, щоб перейти до наступного рівня. Основними піджанрами є платформерські ігри (або платформери), шутери, файтинги, ігри Beat-em Up Brawler Games, Stealth Action Games, Survival Games, Battle Royale і PlayerUnknown's Battlegrounds (PUBG) [1]-[5].

Пригодницькі ігри містять елементи як бойовиків, так і казуальних ігор. Хоча багато аспектів рольових ігор схожі на екшн-ігри, вони пропонують історії, персонажів, систему оцінки, діалоги та інші функції звичайних ігор. Екшн-ігри часто містять складну комбінацію сюжетних елементів, які представлені гравцям через аудіо та відео. На історію значною мірою впливають дії персонажа гравця, які запускають події історії та, таким чином, впливають на ігровий процес [1]-[5].

Ігри-пригоди – це одиночна гра з вигаданим сюжетом. У цих іграх, більше ніж у будь-якій іншій відеоігрі, значною мірою вплинули сюжет і логіка, щоб створити досвід для одного гравця. Відповідати, писати повідомлення, знаходити та використовувати інструменти, відмикати замкнені двері, досліджувати та відкривати нові місця – усе це частина веселих ігор. Розгадування головоломок допоможе вам дослідити нові сфери гри та

дізнатися більше про сюжет. Столи – альтернативний спосіб грати в багато розважальних ігор. Гравці вважають важливими лише додаткові предмети, оскільки вони можуть отримати певні предмети в грі [1]-[5].

У рольовій грі гравець бере на себе роль персонажа, який отримує силу та досвід під час гри. Гравець повинен подолати складні перешкоди та/або перемогти монстрів, щоб отримати очки досвіду, що представляють прогрес персонажа в обраній професії чи класі, а заробіток певної суми дозволить гравцеві отримати нові здібності [1]-[5].

Імітаційні ігри – це широкий жанр відеоігор, створений для відповідності реальним подіям. Ігри-симулятори намагаються імітувати дії в реальному житті у формі гри з різними цілями, як-от навчання, аналіз, передбачення та розвага. Більшість ігор не мають чіткої мети, і користувач може контролювати лише одного персонажа чи середовище. Ігри-симулятори можуть зменшити нудьгу, пов'язану з традиційними методами в класі, оскільки вони передбачають більше, ніж просто читання та вивчення ідей і концепцій. Учні зможуть «прожити» предмет. Завдяки цьому застосування імітаційних ігор підвищує мотивацію та інтерес учнів до навчання. Ігри-симулятори можуть дати гравцям краще зрозуміти, як люди бачать світ, наприклад їхні інтелектуальні та інтелектуальні якості. Вони також допомагають гравцям зрозуміти емпатію та міжособистісні цінності, дозволяючи їм побачити моральні та етичні наслідки їхніх рішень [1]-[5].

Ритм-гра, також відома як ритм-екшн-гра, – це різновид екшн-ігри, яка перевіряє здатність гравців тримати такт. Прикладами цієї категорії є танцювальні ігри, як-от Dance Dance Revolution, і музичні ігри, як-от Donkey Konga та Guitar Hero. У таких іграх гравець повинен натискати кнопки за певний час: кнопки, які гравець повинен натискати на екрані, гра відображатиме удар і рахунок відповідно до удару. На певному ігровому екрані відображається аватар, який реагує на введення гравцем. Оскільки для глядачів аватар важливіший, ніж актор, відповідь часто малюється на задньому плані. Аватар гравця грає проти керованого комп'ютером

комп'ютера в одиночному режимі, тоді як більшість версій дозволяють двом гравцям, керованим аватаром, змагатися за титул. Із зростанням популярності ритмічних ігор з'явився ринок спеціалізованих пристроїв введення. Серед них також менеджери музичних інструментів, таких як гітари, барабани та маракаси. Для того, щоб взяти участь у танцювальній грі, гравець повинен ступити на килимок на танцполі. З іншого боку, більшість ритмічних ігор дозволяють використовувати кілька пристроїв введення як контролери [1]-[5].

Стратегічні ігри – це різновид відеоігор, у якому наголошується на мисленні та плануванні швидких кроків для перемоги. Гравець повинен виконати серію дій проти одного або кількох супротивників, метою яких є зменшення сили супротивника. Успіх приходить з невеликим гарним плануванням. У багатьох відеоіграх користувач має божественний ігровий світ і безпосередньо керує грою. Як результат, більшість стратегічних ігор включають стратегію, тактику та інші рівні бою. Окрім бою, ці ігри часто перевіряють здатність гравця знаходити економіку та керувати нею [1]-[5].

Артилерійські ігри – це покрокові відеоігри для двох або трьох гравців, у яких ви намагаєтесь знищити танки (або просто бомби). У більшості випадків основний елемент гри полягає в тому, щоб змусити супротивника (траси) у балістичну (іншими словами параболічну криву). Гравець зазвичай має ввести кут і швидкість перед пострілом, і в іграх гравець часто вимагає коригування вітру та інших умов. Гру можна класифікувати як шутер, оскільки гравці стріляють один в одного, але в основному вона виглядає як відеогра [1]-[5].

Автоматичні шахи, які часто називають Auto Chess, – це жанр відеоігор із елементами, схожими на шахи, у яких гравці використовують героїв у сітковій бойовій системі та борються з опонентами, не контролюючи склад команди. Подібно до «Королівської битви», ігри про автомобільні перегони протистоять один одному. Кожен гравець відправляє групу солдатів, відомих як мінйони, щоб створити найсильнішу команду. Після того як кожен гравець обирає свій перший набір, він добровільно приєднується до бою та продовжує

бій. Під час бою одиниці обох гравців розміщуються на дошці та воюють автоматично, зазвичай без втручання гравця. Якщо команда повністю знищена, жоден із гравців не може битися знову, програв карається, і гра продовжується до наступного раунду. Після фази бою гравці купують частини на початку кожної фази, які можна комбінувати, щоб створити більш сильні версії тих самих частин. Підрозділи можна призначити кільком підрозділам, а бойові нагороди присуджуються за відстеження кількох підрозділів одного типу. Якщо гравець програє на все життя, він вибуває з турніру [1]-[5].

Стратегія в реальному часі (RTS) – це піджанр стратегічних відеоігор, у які всі гравці грають одночасно, у «реальному часі». У звичайних іграх RTS можна створювати одиниці та аксесуари для конкурентів, але часто це обмежено. Необхідність використання зібраних ресурсів. Використовується для контролю певних точок на карті та/або для захоплення спеціальних типів підрозділів і споруд, призначених для цієї мети, для отримання ресурсів. Екран типової екшн-гри поділено на дві частини: місцевість, юніти та будівлі, ігровий світ з інструкціями, повномасштабна карта з елементами керування виробництвом, а також «радар» або «мінікарта» огляду карта. Зазвичай ставиться гравець. Якщо на карті є кілька одиниць або будівель, на території можна побудувати додаткові підрозділи та будівлі. Щоб досягти найвищого рівня дерева технологій, гравцеві часто доводиться створювати власного персонажа. Ігри RTS часто, але не завжди, використовують гравця для створення армії, захисту від симуляції людських атак або знищення ворогів, яких вони самі створили. Ігри RTS іноді мають фіксовану кількість одиниць, якими гравець може керувати, і не дозволяють гравцеві створювати більше [1]-[5].

Тактика в реальному часі (RTT) – це піджанр тактичних боїв, який імітує бойові дії та військову тактику в реальному часі. Вона відрізняється від справжніх бойовиків тим, що в ній відсутній традиційний менеджмент ресурсів, створення баз і одиниць, більше уваги приділяється окремим одиницям і відсутня більш складна бойова система. Справжні спортивні ігри

вимагають від гравців зосередитися на спорядженні та виробництві, а не на боротьбі один чи кілька разів, тоді як стратегічні ігри зосереджуються на стратегії та боях, наприклад на створенні підрозділів і використанні місцевості для отримання тактичної переваги. Тактичні ігри в реальному часі характеризуються тим, що гравці, які виконують завдання, використовуючи лише бойові сили, отримують і надають чіткі (або принаймні правдоподібні) докази тактичних і військових операцій. Military Tactics зосереджується на тактичних іграх, що базуються на часі, з відносно коротким часом влучень/винагород, які забезпечують швидший, динамічніший, плавніший тактичний бій і досвід бою, ніж інші типи стратегічних ігор [1]-[5].

Ігри по типу захисту вежі (Tower Defense TD) – це гра, мета якої захищати територію або об'єкти гравця, як правило, будуючи оборонні споруди, щоб стояти на його шляху, зупиняти атаки ворогів або перешкоджати ворогам увійти до їхніх виходів. Зазвичай це передбачає будівництво різних структур, які можуть миттєво блокувати, блокувати, атакувати або знищувати ворогів. Через те, що гра в режимі реального часу походить, захист вежі вважається видом спорту в реальному часі, але багато сучасних ігор захисту вежі мають покрокову стратегію. Найважливішою стратегією цього типу є поєднання та вибір захисних одиниць [1]-[5].

Стратегія трансформації – це піджанр військових симуляторів або стратегічних ігор, які часто маловідомі. Гравці беруть фігури або квадрати і по черзі керують ними на квадратах або прямокутниках. Гравці пересуваються по одній одиниці за хід, атакуючи ворожі одиниці та використовуючи природні ресурси, включаючи пошук додаткових одиниць або створення постійного обладнання. Кожна одиниця може рухатися по лінії та виконувати різні дії. Гра зазвичай закінчується, коли досягається певна мета, наприклад знищення всіх ворожих одиниць (або структур), але також поширені ігри на час [1]-[5].

Покрокова тактика – це тип комп'ютерної та відеоігри, яка симулює психологію та природу активного бою в невеликих конфліктах, а не практичну стратегію. Тактичні рольові ігри характеризуються чіткими (або принаймні

правдоподібними) доказами тактичних і військових операцій, у яких гравці використовують лише наявні бойові сили для виконання завдань. Як впливає з назви, покрокова стратегія характеризується покроковими іграми. Витоки цього жанру – тактика бойових ігор і мініатюри, мініатюри зі складними правилами або відтворення бойових сцен за допомогою простих м'яких обкладинок. Тактичні ігри є більш детальними та складними, ніж інші стратегічні ігри, завдяки тактичним ефектам, таким як висота, щільне укриття та видимість [1]-[5].

Військові ігри – це тип стратегічної відеоігри, яка присвячена війні або стратегії на карті. Військові ігри, як правило, поділяються на чотири основні категорії залежно від того, чи гра заснована на часі чи в реальному часі, і чи вона зосереджена на військовій стратегії чи тактиці. Війна – це гра, яка представляє війну, на відміну від абстрактних ігор, таких як шахи. У військові ігри можна грати для розваги, щоб військові офіцери мислили критично або досліджували природу конфлікту. Більшість війн представляють собою конкретні історичні конфлікти і можуть включати всі війни, а також менші кампанії, битви та бойові дії в цих війнах. Є багато битв, які імітують наземні битви, але є також морські битви, які імітують морські та повітряні бої [1]-[5].

Гра типу великої стратегічної війни – це військова гра, яка зосереджена на великій стратегії: використанні ресурсів країни чи імперії та військовій стратегії на рівні руху. 4X має багато спільного з іграми, але є «симетричним», тобто гравці обмежені більш конкретним графіком і нерівномірно розподілені між вільними одиницями, щоб знайти та завершити відкриту гру. навколишнє середовище. Ігри, що виходять за рамки простих воєн, таких як економіка, географія, історія та політика, особливо важливі в освіті та міжнародних дослідженнях [1]-[5].

Відеоігри, які імітують спортивні дії, називаються спортивними відеоіграми. Багато видів спорту імітують такі ігри, як командні види спорту, легка атлетика, екстремальні види спорту та єдиноборства. У спортивних іграх є фізичні та тактичні випробування, які дають гравцеві хороший і справжній

виклик. Багато спортивних ігор імітують такі спортивні фактори, як швидкість, сила, спритність і точність. У ці ігри грають на стадіонах або аренах, які мають певні межі, як і їхні види спорту. Аудіозаписи часто використовуються для спортивних ігор і кольорових коментарів [1]-[5].

Командні види спорту є одними з найдавніших видів спорту. Вони порівнюють футбол з футболом, а регбі з бейсболом. Командний вид спорту — це змагання між протиборчими командами, у якому гравці змагаються безпосередньо й одночасно за досягнення мети. Команди несуть відповідальність за сприяння переміщенню м'яча або подібним рухам відповідно до правил, щоб отримати очки [1]-[5].

Перегони — це відеоігри, у яких гравець керує транспортним засобом, який змагається з іншими транспортними засобами в серії ігор або рівнів для досягнення найвищого рейтингу. Деякі гоночні відеоігри, такі як Sonic R, не потребують використання автомобілів; Деякі ігри показують лише діючих персонажів. Існує багато видів перегонів, і всі вони різні. Усі вони мають одну мету — дістатися до фінішу раніше за інших, але кожен піджанр має власний ігровий процес і концепцію [1]-[5].

Кіберспортивні змагальні ігри, або кіберспорт, стосується відеоігор, у які змагаються професійні та непрофесійні гравці. Кіберспорт зазвичай поділяється на кілька основних категорій. Бійки, шутери від першої особи, стратегії в реальному часі і багатокористувацькі онлайн-рольові ігри складають більшість кіберспорту. Велика кількість гравців з усього світу змагаються за приз. Серед усіх жанрів відеоігор кіберспорт є жанром, який розвивається найшвидше [1]-[5].

До сольних ігор належать індивідуальні види спорту, такі як катання на ковзанах і сноуборді, і такі відеоігри часто намагаються якомога точніше імітувати реальність [1]-[5].

Ігри-головоломки вимагають від гравців ідентифікувати фігури, розв'язувати ситуації та завершувати слова, викликаючи їхні навички вирішення проблем. Кросворди, кросворди, математичні, кросворди, логічні

головоломки — приклади різних жанрів. Відповіді можуть бути цікаво оформлені, але вони також можуть виникати зі складних математичних або логічних завдань. Логічні питання та ідеї найважливіші в іграх-головоломках. Багатокористувацькі ігри та ігри-головоломки включають головоломки у свій дизайн, тоді як справжні ігри-головоломки віддають перевагу розв'язанню проблем як основному заходу гри [1]-[5].

Ігра-вікторина – це гра, в якій гравцям (індивідуально чи в групах) задають запитання на різні теми, і вони відповідають на якомога більше питань, щоб заробити бали. Конкурси-вікторини часто проводяться як частина вечірок, заходів або розваг. Зазвичай учасники діляться на команди і перемагає команда, яка набрала найбільшу кількість балів [1]-[5]. Такі ігри призначені для того, щоб гравці розважалися під час гри, і їхня головна мета — розвивати та покращувати знання в загальних та окремих сферах, а також стимулювати дебати та хорошу дискусію між учасниками [1]-[5].

Логічні ігри вимагають розв'язувати логічні задачі або орієнтуватися в складних середовищах, як-от лабіринти, щоб здобувати рівні та очки. Головоломки на відповідність є однією з найпопулярніших ігор, оскільки вони підходять для казуальних ігор. Розважальні та навчальні ігри зазвичай належать до цієї категорії [1]-[5].

Неактуальні або бездіяльні ігри (Idle Games) орієнтовані на гравців, із простими завданнями, такими як клацання по печиву. По ходу гри гравець поступово винагороджується певними посиленнями за виконання вищезгаданих місій. Загалом, ці ігри вимагають незначного втручання від гравців і часто в них грають незалежно [1]-[5].

Казуальні ігри – це відеоігри, які не займають багато часу, щоб грати. З іншого боку, «звичайний гравець» відноситься до того, хто любить будь-яку відеоігру, не витрачаючи багато часу, і грає в неї невимушено, випадково або рідко. Казуальні ігри, на відміну від серйозних ігор, часто є вебіграми та орієнтовані на великий масовий ринок. Казуальні ігри можуть бути будь-якого

жанру та включати будь-який стиль гри. Вони, як правило, мають менше правил, коротші курси та менші когнітивні здібності [1]-[5].

Вечірні ігри – це зазвичай багатокористувацькі відеоігри, які дозволяють змагатися великій кількості гравців. Вечірні ігри – це зазвичай набір великомасштабних ігор, які зосереджені на всьому: від збору більшої кількості предметів, ніж інші гравці, до того, як швидко встигнути за все [1]-[5].

Ігри програмування – це відеоігри, які використовуються для навчання навикам програмування. Гра в програмування зазвичай містить комп'ютерну програму, яка дозволяє виконувати команди певною мовою програмування. Зазвичай він представлений візуальною мовою, яка полегшує розуміння концепцій програмування. В іграх з програмуванням гравець не повністю контролює результат гри. Замість цього на спеціальній мові програмування розробляється комп'ютерна програма або сценарій для керування поведінкою гравців (зазвичай це роботи, танки чи бактерії, яких вони хочуть знищити). Гравцям будуть надані інструменти для написання та тестування своїх додатків певною мовою гри, перш ніж надсилати додаток на головний сервер. Потім сервер може запускати програму проти інших гравців, повідомляти про результати гравцеві, а потім змінювати або покращувати програму на основі результатів [1]-[5].

Масова багатокористувацька онлайн-гра (massively multiplayer online game, MMOG) – це жанр відеоігор, який дозволяє великій кількості людей грати одночасно через Інтернет. Ці ігри зазвичай знаходяться в спільному середовищі, до якого гравець може отримати доступ після придбання або встановлення ігрового плану. Багато розробників ігор включили багатокористувацьку мережеву гру в традиційні багатокористувацькі режими завдяки розширенню MMOG. MMOG утворюють цілісну екосистему: від величезних середовищ, які об'єднують велику кількість гравців для виконання місій, до активних економік, які однаково впливають на всіх гравців [1]-[5].

Рекламна гра (також відома як advergame) – це тип відеореklamної продукції, яка створює гру або тісно співпрацює з організацією для

просування певного бренду. Рекламні ігри зазвичай безкоштовні, онлайн і прив'язані до різних пристроїв [1]-[5].

Мистецька комп'ютерна гра – це діяльність, заснована на співпраці з художником, який у комп'ютерній грі надає соціальну чи історичну критику або творчо розповідає історію. Існує особливий спосіб перемогти та досягти успіху в розумовій діяльності, набір етапів (які можуть бути або не бути рівнями), а також основний персонаж або шаблон, який визначає гравця. Аркади можуть поєднувати в собі елементи головоломки та стрілялки з інших жанрів, але залишаються окремим жанром. Вони схильні стирати межу між живою грою та грою, між виставою та традиційними уявленнями про гру. Вони намагаються комбінувати та грати ідеї, щоб направити методи в нові напрямки, і іноді використовують методи, які ще не цікаві [1]-[5].

Навчальні ігри – це ігри, створені з навчальною метою. Навчальні ігри розроблені, щоб допомогти людям вивчити конкретні теми, розширити своє мислення, стимулювати їхній розвиток, зрозуміти історичні та культурні події та отримати навички під час гри. Прикладами ігрових жанрів є настільні ігри, карткові ігри та відеоігри. Цей навчальний інструмент став дуже популярним, оскільки педагоги, уряди та батьки усвідомили необхідність позитивного мислення та переваги гри в навчанні. Ігри та історії навчають цілям, правилам, гармонії, вирішенню проблем і співпраці через гру. Вони задовольняють найважливіші навчальні потреби, створюючи хвилювання, участь, структуру, мотивацію, задоволення, креативність, соціальні зв'язки під час навчання [1]-[5].

Для формування команд у відеоіграх, у т.ч. в кіберспорті, де необхідною є участь декількох гравців в одній команді, використовуються методи матчмейкінгу [1]-[9].

Матчмейкінг – це процес формування команд для гри в комп'ютерні ігри. Його головною метою є створення рівних умов для всіх гравців, де вони матимуть шанс на перемогу незалежно від свого рівня ігрової майстерності. Процес матчмейкінгу базується на алгоритмах, які враховують різні фактори,

такі як ранг, рейтинг, історія гри та інші параметри. Найчастіше матчмейкінг використовується в онлайн-іграх, де гравці можуть взаємодіяти один з одним в режимі реального часу [6]-[9].

Щоб забезпечити правильне формування команд, матчмейкінг використовує різні алгоритми, які можуть бути різні для кожної гри. Наприклад, для одних ігор можуть бути важливі рейтингові очки, тоді як для інших – історія гри. Алгоритми можуть бути також змінені для врахування різних умов, таких як час пошуку гри, кількість гравців, наявність пріоритету для гравців з високим рангом тощо [6]-[9].

У матчмейкінгу важливу роль відіграє баланс між швидкістю формування команд та їх якістю. Надто повільне формування команд може призвести до того, що гравці не зможуть знайти гру через занадто довгий час очікування, але надто швидке формування може привести до того, що гравці будуть постійно перемагати або програвати, що може негативно вплинути на їхню мотивацію грати [6]-[9].

Основні переваги матчмейкінгу – це більш конкурентні матчі, можливість шукати командою та грати з іншою командою, а також різноманітність ігрових конфігурацій. Компроміси включають тривалий час пошуку, щоб знайти вдалий матч, і неможливість вибрати точну карту чи тип гри для гри [6]-[9].

Багато систем підбору гравців використовують оцінки навичок або ранги. Це дозволяє гравцям зі схожими здібностями грати один проти одного, забезпечуючи ближчу та конкурентоспроможну гру для всіх діапазонів навичок [6]-[9].

Списки відтворення також пов'язані зі матчмейкінгом. Вони використовуються в іграх із широкими можливостями налаштування, щоб розділити загальне можливе ігрове поле на кластери подібних типів ігор. Це дозволяє гравцеві мати певний вибір, у який тип гри грати. В онлайн-шутерах списки відтворення містять кілька конфігурацій правил і карт на одну тему [6]-[9].

Визначено, що у наш час широко використовуються комп'ютерні ігри, що являють собою ігри, керовані комп'ютером, де гравці з метою розваги взаємодіють з об'єктами або іншими гравцями, що відображаються на екрані. Відзначено, що існують багато відеоігор різних жанрів тобто категорій ігор, які мають подібний ігровий процес і ігрову механіку. Виконано аналіз жанрів відеоігор. Відзначено, що для формування команд у відеоіграх, у т.ч. у кіберспорті, де необхідною є участь декількох гравців в одній команді, використовуються методи матчмейкінгу, що являє собою процес формування команд для комп'ютерної гри. Його головною метою є створення рівних умов для всіх гравців, де вони матимуть шанс на перемогу незалежно від свого рівня ігрової майстерності. Визначено, що процес матчмейкінгу базується на алгоритмах, які враховують різні фактори, такі як ранг, рейтинг, історія гри та інші параметри.

1.2 Дослідження програмних засобів для підбору гравців у комп'ютерних іграх

Зараз існує значна кількість програм для підбору гравців у комп'ютерних іграх. Проведемо огляд певних програм, що розв'язують завдання матчмейкінгу [10]-[14].

Програмне забезпечення Playfab [10] є бекенд платформою для онлайн-ігор. Надає наступні можливості: керування гравцями, мультиплеер, торгівля, таблиці лідерів, спеціальна логіка, події в грі, панелі інструментів, звіти, потік даних у реальному часі. Матчмейкінг Playfab (рис. 1.1) дозволяє: сортувати, фільтрувати та збирати користувачів для максимальної сумісності та задоволення; встановлювати параметри матчмейкінгу відповідно до стилю та структури вашої гри, отримувати доступ до статистичних даних та постійне налаштування правил матчмейкінгу, щоб оптимізувати досвід гравців[3]. Сучасні мережеві ігри, які використовують Playfab: Rainbow Six Siege, Sea of Thieves, Forza Horizon 4.

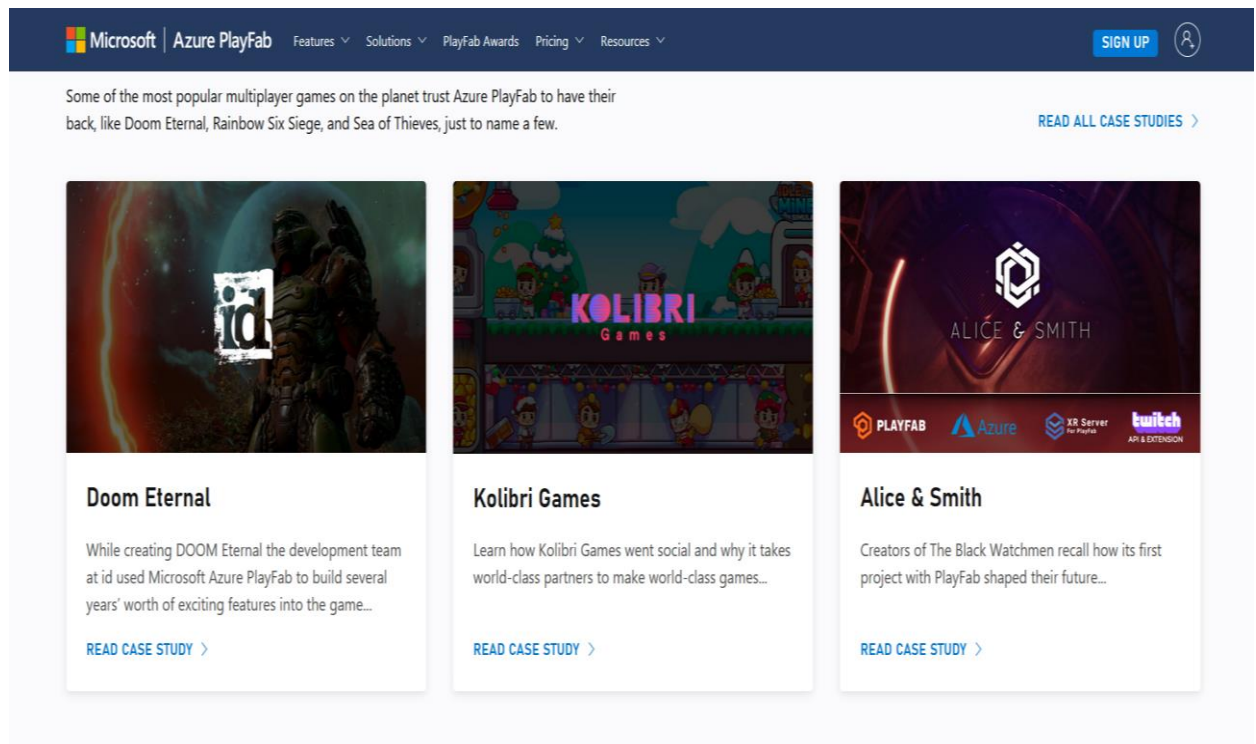


Рисунок 1.1 – Програмне забезпечення Playfab [10]

До недоліків програмного забезпечення Playfab можна віднести вартість оренди серверів матчмейкінгу та серверів статистики, а також специфічну інтеграцію у проєкт.

Програмне забезпечення Amazon GameLift (рис. 1.2) – це повністю керований сервіс для розгортання, керування та масштабування серверів багатокористувацьких ігор на основі сеансів у хмарі. Надає наступні можливості: керування серверами, розташування серверів по всьому світу, саморозширення, захист від DDOS-атак. Можливості FlexMatch Amazon GameLift дозволяють вам узгоджувати гравців на основі визначених правил. Незалежно від того, чи ви вирішите побудувати свою компанію на основі навиків гравця, затримки, або спеціальних критеріїв, проста, але потужна мова правил FlexMatch полегшує будь-кому можливість швидко створювати надійний пошук гравців. Результати FlexMatch автоматично спрямовуються в чергу, яка потім може помістити відповідність у область AWS, яка найкраще балансує затримку гравця та використання сервера [11]. Сучасні мережеві

ігри, які використовують Amazon GameLift FlexMatch: Star Citizen, New World, Crucible [11].

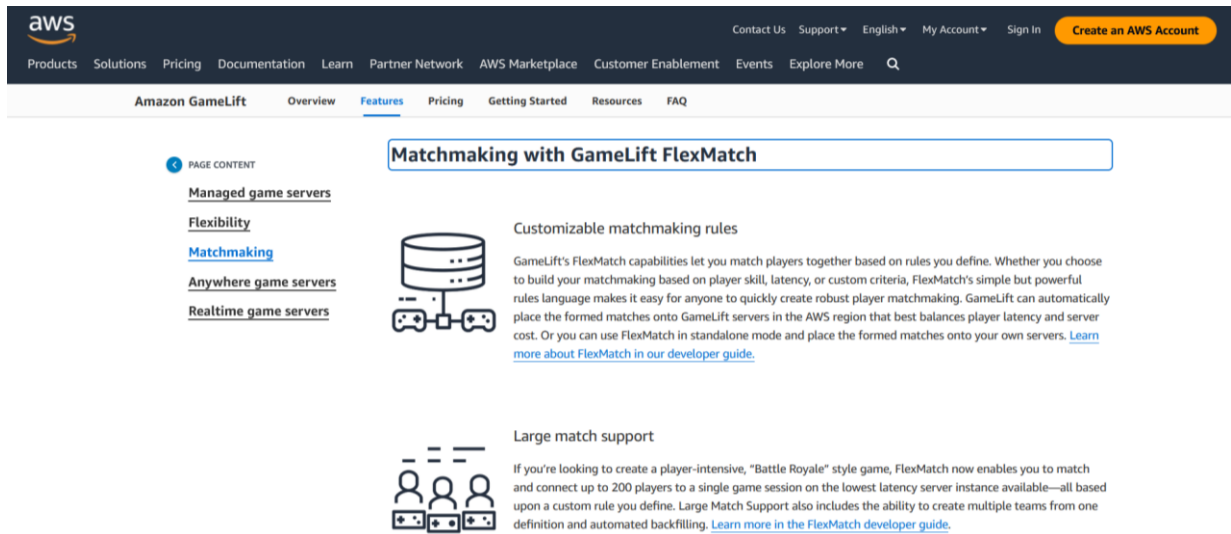


Рисунок 1.2 – Програмне забезпечення Amazon GameLift [11]

До недоліків програмного забезпечення Amazon GameLift можна віднести вартість оренди серверів матчмейкінгу та серверів статистики, а також відсутність підтримки без-екземплярних (Non-instance based) ігрових серверів.

Програмне забезпечення GameSparks (рис. 1.3) – це хмарне рішення для розробників ігор, яке допомагає їм створювати свої серверні компоненти без необхідності налаштовувати і запускати сервер. Надає наступні можливості: налаштування місцевих конкурсів для декількох жорстко підібраних гравців до турнірів, що охоплюють земну кулю, у яких беруть участь велика кількість одночасних гравців на різних рівнях кваліфікації; підтримує head-to-head або групові матчі, щоб заохотити соціальну гру і поглибити участь гравця в грі; дозволяє гравцям засновувати та приєднуватися до матчів зі своїми друзями з реального світу, щоб надати дуже персоналізований та захоплюючий ігровий досвід [12]. Сучасні мережеві ігри, які використовують GameSparks: Hungry Shark Evolution, LaraCroft RelicRun, MMX Racing [12].

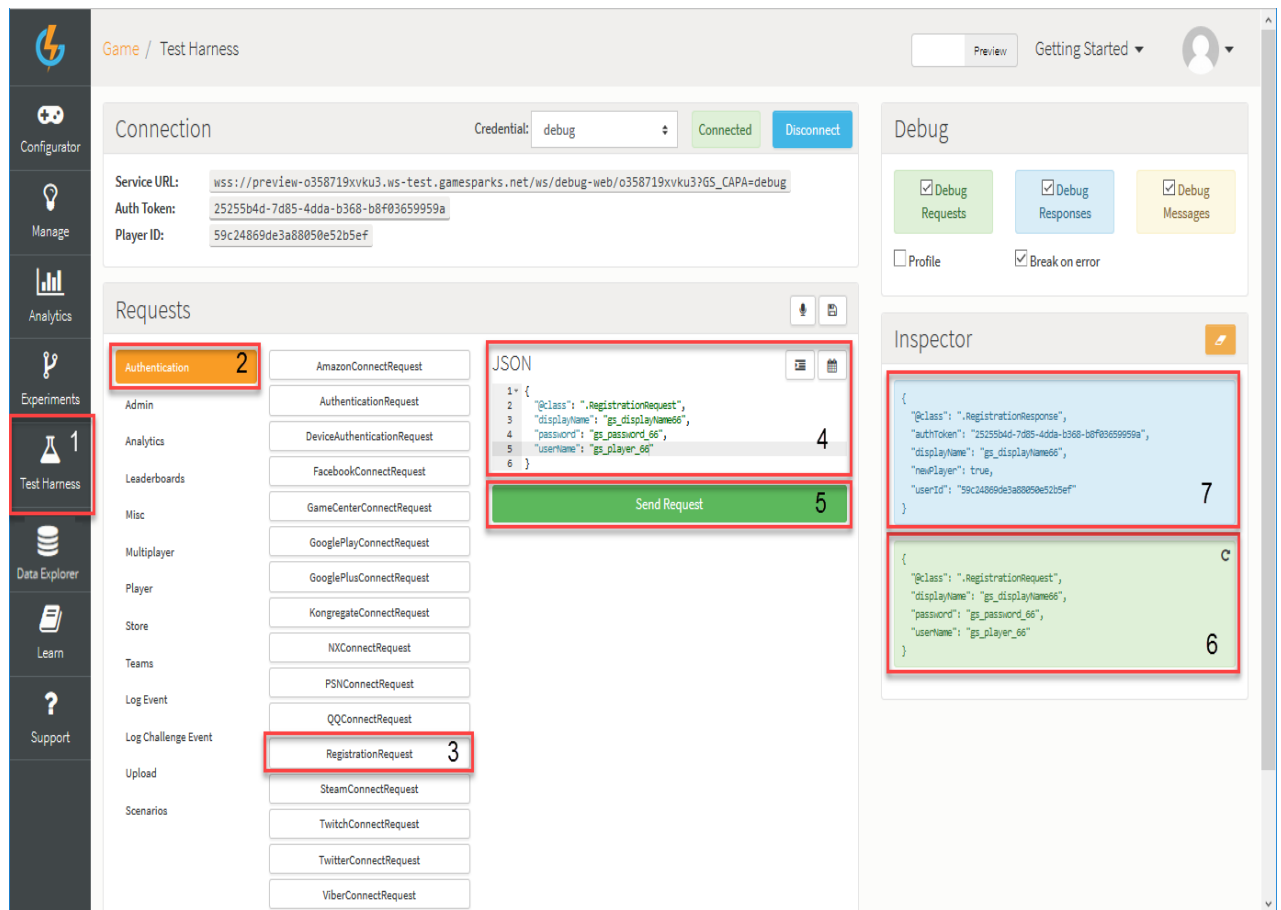


Рисунок 1.3 – Програмне забезпечення GameSparks [12]

До недоліків можна програмного забезпечення GameSparks віднести вартість оренди серверів матчмейкінгу та серверів статистики, а також суттєва недостача документації, що ускладнює інтеграцію.

Програмне забезпечення Photon Engine (рис. 1.4) являє собою механізм розробки багатокористувацьких ігор у реальному часі, який надає різноманітні мережеві рішення для створення масштабованих багатокористувацьких ігор. Він пропонує хмарний сервіс для кількох гравців, який підтримує кросплатформенну розробку та розгортання ігор на кількох платформах, таких як Windows, Mac, iOS, Android тощо. Photon Engine підтримує різні мови програмування, такі як C++, C# і UnityScript, і надає такі функції, як підбір, сервери ретрансляції, синхронізація, чат тощо. За допомогою Photon Engine розробники можуть створювати багатокористувацькі ігри з мінімальною кількістю коду та зменшувати складність мережевої інфраструктури [13].

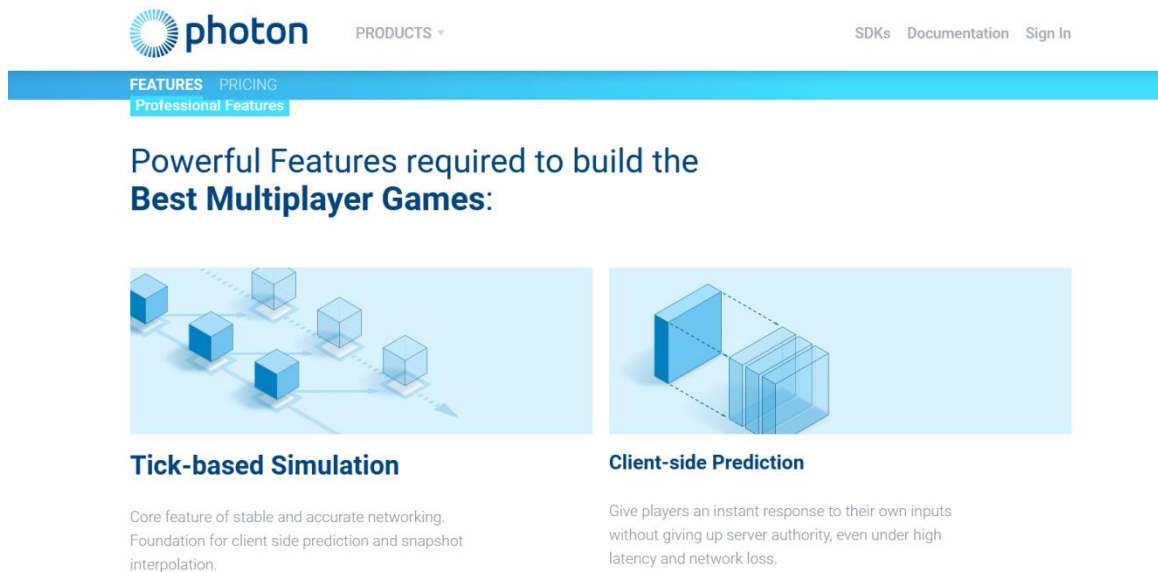


Рисунок 1.4 – Програмне забезпечення Photon Engine [13]

До недоліків можна програмного забезпечення Photon Engine віднести вартість оренди серверів матчмейкінгу та серверів статистики, а також суттєва нестача документації, що ускладнює інтеграцію.

Програмне забезпечення Nakama являє собою розподілений сервер із відкритим кодом для розробки соціальних і багатокористувацьких ігор. Він пропонує широкий спектр функцій, включаючи автентифікацію користувачів, багатокористувацьку гру в режимі реального часу, таблиці лідерів, пошук партнерів, сповіщення тощо. Nakama можна використовувати для розробки ігор на різних платформах, таких як мобільні пристрої, настільні комп'ютери, Інтернет і консолі. Він підтримує кілька мов програмування, включаючи C#, Java, JavaScript, Go та Python, і його можна розгорнути в різних середовищах, включаючи локальні, хмарні та гібридні налаштування. Однією з переваг використання Nakama є його гнучкість і масштабованість, що дозволяє розробникам легко додавати нові функції та розширювати свою гру за потреби [14].

Програмне забезпечення Nakama (рис. 1.5) забезпечує розподілений сервер для соціальних ігор і ігор у реальному часі, програм і сервісів. Він

підтримує протоколи зв'язку HTTP/REST API, WebSocket і RPC (віддалений виклик процедури). HTTP/REST API використовується для зв'язку сервер-клієнт через HTTP і дозволяє клієнтам взаємодіяти з сервером за допомогою стандартних запитів HTTP, таких як GET, POST, PUT і DELETE. Це дозволяє розробникам створювати користувальницькі клієнти будь-якою мовою чи фреймворком, які можуть робити запити HTTP. REST API також підтримує автентифікацію JWT (JSON Web Token), яка забезпечує безпечний контроль доступу до ресурсів сервера [14].

Nakama console

Active Matches
10 running matches found.

Match ID	Presence Count	Authoritative	Handler Name	Tick Rate
0f97a6e1-478c-422a-93be-caad75d4c467.nakama	0	Yes	match	1
Match Label skill=100-150 Current Tick 6 Match State { "debug": true } Match Presences []				
388adf7e-aa3d-4957-a299-2cb7832e4a25.nakama	0	Yes	match	1
Match Label { "skill": 60 } Current Tick 1 Match State { "debug": true } Match Presences []				
> 565e7402-ab8c-49bd-8624-b88c4e09dad2.nakama	1	Yes	match	1
> add86366-628e-4de9-a65a-158010038809.nakama	1	Yes	match	1
> c5ab1abc-d2ae-4da7-b331-7d7ebc33ae01.nakama	0	Yes	match	1
39fc7331-cb0e-46e8-8f43-ffe019f81f53.	1	No		-
70ed3154-62e0-4096-bef2-381bad381a78.	2	No		-
e8b70a47-7f81-43f3-8c8d-35ef35774891.	3	No		-
5ce73d2f-945d-4fae-9c82-ce61c364aa2c.	1	No		-
c0aec9b4-d8b6-4d9e-b5d7-c3efbaaf5df7.	1	No		-

Рисунок 1.5 – Програмне забезпечення Nakama [14]

Адреса сервера Nakama залежить від того, як він був налаштований та розгорнутий. Якщо запускається локальний сервер Nakama, то адресою може бути, наприклад, <http://localhost:7350>. Якщо ви використовується хмарний сервіс Nakama, то адреса може бути різною в залежності від вибраного плану

та регіону. В будь-якому випадку, адресу сервера можна знайти в налаштуваннях проєкту або дізнатися у відповідального за розгортання сервера [14].

До недоліків можна програмного забезпечення Nakama віднести вартість оренди серверів матчмейкінгу та серверів статистики, а також суттєва недостача документації, що ускладнює інтеграцію.

Порівняльну характеристику програмного забезпечення для підбору гравців у комп'ютерних іграх наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмного забезпечення для підбору гравців у комп'ютерних іграх

Критерій порівняння	Playfab [10]	Amazon GameLift [11]	GameSparks [12]	Photon Engine [13]	Nakama [14]
Підтримка декількох режимів матчмейкінгу	—	+	—	+	+
Легкість інтеграції у проєкт	+—	+	+	+	+
Можливість задавання параметрів ігрового серверу при розгортанні	—	+	—	+	+
Низька вартість підтримки	+	+—	+—	+	++
Служба бекенду	+	—	+	—	—
Модель оплати	залежить від використання	залежить від використання	підписка	залежить від використання	вільна

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато програмних систем для підбору гравців у

комп'ютерних іграх. Проте деякі з них є досить дорогими, інші досить складними для використання.

Тому актуальною є розробка програмного забезпечення для підбору гравців у комп'ютерних іграх.

При розробці програми для підбору гравців у комп'ютерних іграх необхідно забезпечити такі функціональні вимоги:

- пошук матчу для одного гравця відповідно до його внутрішньоігрових характеристик;
- пошук матчу для команди гравців відповідно до внутрішньоігрових характеристик команди;
- розробникам функцій задавання критеріїв об'єднання гравців;
- задавання пріоритету критеріїв об'єднання гравців;
- задавання діапазону мережевих портів, які будуть використовуватися для розгортання ігрових серверів;
- надання мережевої адреси розгорнутого ігрового сервера для певної ігрової сесії;
- задавання типу ігрових серверів для розгортання.

1.3 Висновки за розділом 1

Визначено, що у наш час широко використовуються комп'ютерні ігри, що являють собою ігри, керовані комп'ютером, де гравці з метою розваги взаємодіють з об'єктами або іншими гравцями, що відображаються на екрані. Відзначено, що існують багато відеоігор різних жанрів тобто категорій ігор, які мають подібний ігровий процес і ігрову механіку. Виконано аналіз жанрів відеоігор. Відзначено, що для формування команд у відеоіграх, у т.ч. у кіберспорті, де необхідною є участь декількох гравців в одній команді, використовуються методи матчмейкінгу, що являє собою процес формування команд для комп'ютерної ігри. Його головною метою є створення рівних умов для всіх гравців, де вони матимуть шанс на перемогу незалежно від свого

рівня ігрової майстерності. Визначено, що процес матчмейкінгу базується на алгоритмах, які враховують різні фактори, такі як ранг, рейтинг, історія гри та інші параметри.

За результатами попереднього аналізу зроблено висновок про те, що у наш час існує досить багато програмних систем для підбору гравців у комп'ютерних іграх. Проте деякі з них є досить дорогими, інші досить складними для використання. Тому актуальною є розробка програмного забезпечення для підбору гравців у комп'ютерних іграх.

Сформульовано функціональні вимоги до програмного забезпечення для підбору гравців у комп'ютерних іграх.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування та інших технологій розробки

При розробці програмного забезпечення для підбору гравців у комп'ютерних іграх було використано мову програмування C++ [15]-[17].

Мова програмування C++ є мовою програмування середнього рівня, яка працює на різних платформах, таких як Windows, Mac OS та різних версіях UNIX-подібних систем. Мова програмування C++ дуже близькою до апаратного забезпечення, тому є можливість працювати на низькому рівні, що дає великий контроль щодо керування пам'яттю, кращу продуктивність і, нарешті, надійну розробку програмного забезпечення [15]-[17].

Програмування на C++ дає чітке розуміння об'єктно-орієнтованого програмування. Програміст має можливість реалізації низькорівневого поліморфізму при створенні віртуальних таблиць та покажчиків віртуальних таблиць або динамічної ідентифікації типів [15]-[17].

C++ є однією з мов програмування, яку широко використовують у прикладному та системному програмуванні. Таким чином, можна вибрати сферу інтересів розробки програмного забезпечення [15]-[17].

C++ дозволяє ефективно працювати з компілятором, компоувальником і завантажувачем, різними типами даних, класами зберігання, типами змінних, їхніми областями тощо [15]-[17].

Мова програмування C++ є однією з найпоширеніших мов програмування, тому вона присутня майже в усіх сферах розробки програмного забезпечення. Розробка прикладного програмного забезпечення – програмування на C++ використовувалося для розробки майже всіх основних операційних систем, таких як Windows, Mac OSX і Linux. Окрім операційних систем, основна частина багатьох браузерів, таких як Mozilla Firefox і Chrome, написана на C++. C++ також використовувався для розробки найпопулярнішої системи баз даних MySQL [15]-[17].

Мова програмування C++ широко використовується для розробки нових мов програмування, таких як C#, Java, JavaScript, Perl, C Shell UNIX, PHP і Python, Verilog тощо [15]-[17].

Мова програмування C++ широко використовується для розробки медичних та інженерних програм, таких як програмне забезпечення для апаратів МРТ, високоякісних систем систем комп'ютерного проєктування тощо [15]-[17].

Цей тип мови програмування використовує об'єктно-орієнтований підхід та розглядає програму як групу об'єктів, що складаються з даних і програмних елементів, відомих як атрибути та методи. Об'єкти можна повторно використовувати в програмі або в інших програмах. Це робить його популярним типом мови для складних програм, оскільки код легше повторно використовувати та масштабувати [15]-[17]. Мову програмування C++ програмісти використовують для розробки програмного забезпечення широкого призначення, сценаріїв або інших наборів інструкцій для виконання комп'ютерами. Мови сценаріїв, які не потребують компілятора, використовують інтерпретатор для виконання сценарію. Програми, написані на C++, часто компілюються у машинну мову, яку може зрозуміти комп'ютер [15]-[17].

Крім того, мова програмування C++ використовується ученими та дослідниками через високу швидкість і ефективність обчислень [15]-[17].

C++ є надзвичайно швидкою мовою, що дозволяє програмістам виконувати процедурне програмування для функцій, що інтенсивно використовують центральний процесор, і забезпечує більший контроль над апаратним забезпеченням, через що ця мова широко використовується в розробці ігрових двигунів [15]-[17].

Обґрунтування вибору мови програмування для розробки програмного забезпечення підбору гравців у комп'ютерних іграх наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмного забезпечення підбору гравців у комп'ютерних іграх

Критерій порівняння мов програмування	Мова програмування		
	C++	C#	Java
Ефективність та зручність при створенні програмного забезпечення підтримки комп'ютерних ігор	+	—	—
Швидкість виконання коду	+	+—	+—
Багатопарадигмальність	+	+—	+
Кросплатформеність розробки	+	+—	+
Зручність використання	+	+—	+—

Отже, для реалізації програмного забезпечення підбору гравців у комп'ютерних іграх обрано мову програмування C++, яка є статично типізованою, скомпільованою, багатопарадигмальною мовою програмування загального призначення, яка широко використовується у відеоіграх, настільному програмному забезпеченні та розробці вбудованих систем.

2.2 Вибір мережевої бібліотеки

В якості мережевої бібліотеки для забезпечення функцій асинхронного введення-виведення обрано Boost.Asio [18]. Boost.Asio надає інструмент для керування довгими операціями введення-виведення без програмування за допомогою потокового інтерфейсу та явних переривань. Boost.Asio пом'якшує сан призначений для програмістів, які часто використовують C++ для системних програм, які вимагають функцій операційної системи, наприклад мережі.

Boost.Asio є переносимою бібліотекою, яка підтримує набір загальноновживаних функцій і забезпечує стабільну роботу в різних операційних системах [18].

Boost.Asio є широко використовуваною бібліотекою C++, яка забезпечує функції асинхронного введення-виведення. Це одна з найпопулярніших бібліотек для створення мережевих програм, таких як веб-сервери та мережеві ігри [18].

Бібліотека Boost.Asio є масштабованою та підтримує розробку мережевих програм, що працюють з тисячами паралельних з'єднань. Кожна реалізація бібліотеки операційної системи повинна відповідним чином відповідає цьому рівню [18].

Boost.Asio надає інтерфейс API, що широко використовуваним, комплексним і містить розширену документацію. Boost.Asio є простою у використанні бібліотекою. Бібліотека Boost.Asio може бути основою для подальшої абстракції та може дозволити розвиток інших бібліотек, які забезпечують більш високий рівень абстракції [18].

Виконаємо порівняння Boost.Asio з іншими подібними бібліотеками. Бібліотека Libevent також є популярною бібліотекою C для створення мережевих програм. Вона також забезпечує підтримку керованого подіями програмування та асинхронного введення-виведення. Однак Boost.Asio — це бібліотека C++, яка надає більше функцій, наприклад підтримку таймерів, SSL і з'єднання через послідовний порт.

Адаптивне комунікаційне середовище ACE є структурою для створення розподілених програм. Воно надає широкий спектр мережевих протоколів, а також підтримку паралелізму, потоків і програмування, керованого подіями. Boost.Asio є легшим у порівнянні з ACE, і зосередженим саме на асинхронному ввіді-виводі.

Бібліотеки РОСО C++ надають колекцію повторно використовуваних компонентів C++, включаючи мережеві протоколи та серверні

інфраструктури. POCO включає підтримку асинхронного вводу-виводу, але не забезпечує такий самий рівень налаштування, як Boost.Asio.

Бібліотека libuv є бібліотекою на основі мови C для створення мережеских додатків з акцентом на міжплатформенну підтримку. Вона забезпечує кероване подіями програмування та асинхронний ввід-вивід, і це основна бібліотека, яка використовується Node.js. Boost.Asio має більш гнучкий API і краще підходить для створення високопродуктивних мережеских програм.

Обґрунтування вибору мережескої бібліотеки для забезпечення функцій асинхронного введення-виведення наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору мережескої бібліотеки для забезпечення функцій асинхронного введення-виведення

Бібліотека	Підтримувана мова програмування	Практичні застосування	Кросплатформність	Розповсюдженість
Boost.Asio	C++	Мережескі застосунки	+	+
Libevent	C	Мережескі застосунки	+	+
ACE	C++	Розподілені застосунки	+	—
POCO	C++	Мережескі застосунки	—	—
libuv	C	Мережескі застосунки	+	+—

Загалом, Boost.Asio є універсальною та потужною бібліотекою для створення мережеских програм на C++. Вона забезпечує підтримку широкого спектру функцій, включаючи таймери і зв'язок через послідовний порт, а

також має гнучкий API, який дозволяє налаштовувати та забезпечує високу продуктивність. Хоча існують інші бібліотеки, доступні для створення мережових програм, Boost.Asio є популярним вибором завдяки своїй надійності та простоті використання.

Отже, в якості мережової бібліотеки для забезпечення функцій асинхронного введення-виведення обрано Boost.Asio, яка є широко використовуваною бібліотекою C++, що забезпечує створення мережових програм, зокрема і мережових ігор.

2.3 Вибір середовища розробки для створення програмного забезпечення підбору гравців у комп'ютерних іграх

В якості середовища розробки для створення програмного забезпечення підбору гравців у комп'ютерних іграх було обрано Visual Studio [19], [20].

Visual Studio є інтегрованим середовищем розробки (IDE), яке розроблене корпорацією Майкрософт та є одним з найпопулярніших інструментів для розробників для створення, тестування та розгортання програмного забезпечення. Продукт містить усе, що потрібно розробнику, наприклад редактор коду, налагоджувач, компілятори та кілька інших інструментів і служб для оптимізації процесу розробки програмного забезпечення [19], [20].

Крім того, продукт підтримує широкий спектр мов програмування, включаючи C#, C++ і Python, що дозволяє розробникам вибирати мову, яка найкраще підходить для їх проєкту, надаючи необхідні інструменти для роботи з цією мовою. Продукт підтримує широкий спектр шаблонів і фрагментів коду, які можуть прискорити створення нових проєктів і прискорити процес розробки, щоб спростити певні дії для розробників [19], [20].

Обґрунтування вибору середовища розробки для створення програмного забезпечення підбору гравців у комп'ютерних іграх наведено у таблиці 2.3.

Таблиця 2.3 – Обґрунтування вибору середовища розробки для створення програмного забезпечення підбору гравців у комп'ютерних іграх

Критерій порівняння середовищ розробки	Середовище розробки		
	Visual Studio Code	Sublime Text	PyCharm
Безкоштовність	+	+	—
Візуальний редактор коду	+	—	+—
Зручність використання	+	+—	+—
Підтримка рефакторингу	+	—	+
Підтримка хмарних сервісів	+	—	—
Відкритість ПЗ	+	—	—

Таким чином, для створення програмного забезпечення підбору гравців у комп'ютерних іграх обрано середовище розробки Visual Studio.

2.4 Висновки за розділом 2

Для реалізації програмного забезпечення підбору гравців у комп'ютерних іграх обрано мову програмування C++, яка є статично типізованою, скомпільованою, багатопарадигмальною мовою програмування загального призначення, яка широко використовується у відеоіграх, настільному програмному забезпеченні та розробці вбудованих систем.

В якості мережевої бібліотеки для забезпечення функцій асинхронного введення-виведення обрано Boost.Asio, яка є широко використовуваною бібліотекою C++, що забезпечує створення мережевих програм, зокрема і мережевих ігор.

Для створення програмного забезпечення підбору гравців у комп'ютерних іграх обрано середовище розробки Visual Studio.

3 ОПИС ПРОГРАМИ

3.1 Структура процесів програми підбору гравців у комп'ютерних іграх

Структуру програмного забезпечення підбору гравців у комп'ютерних іграх наведено на рисунку 3.1.

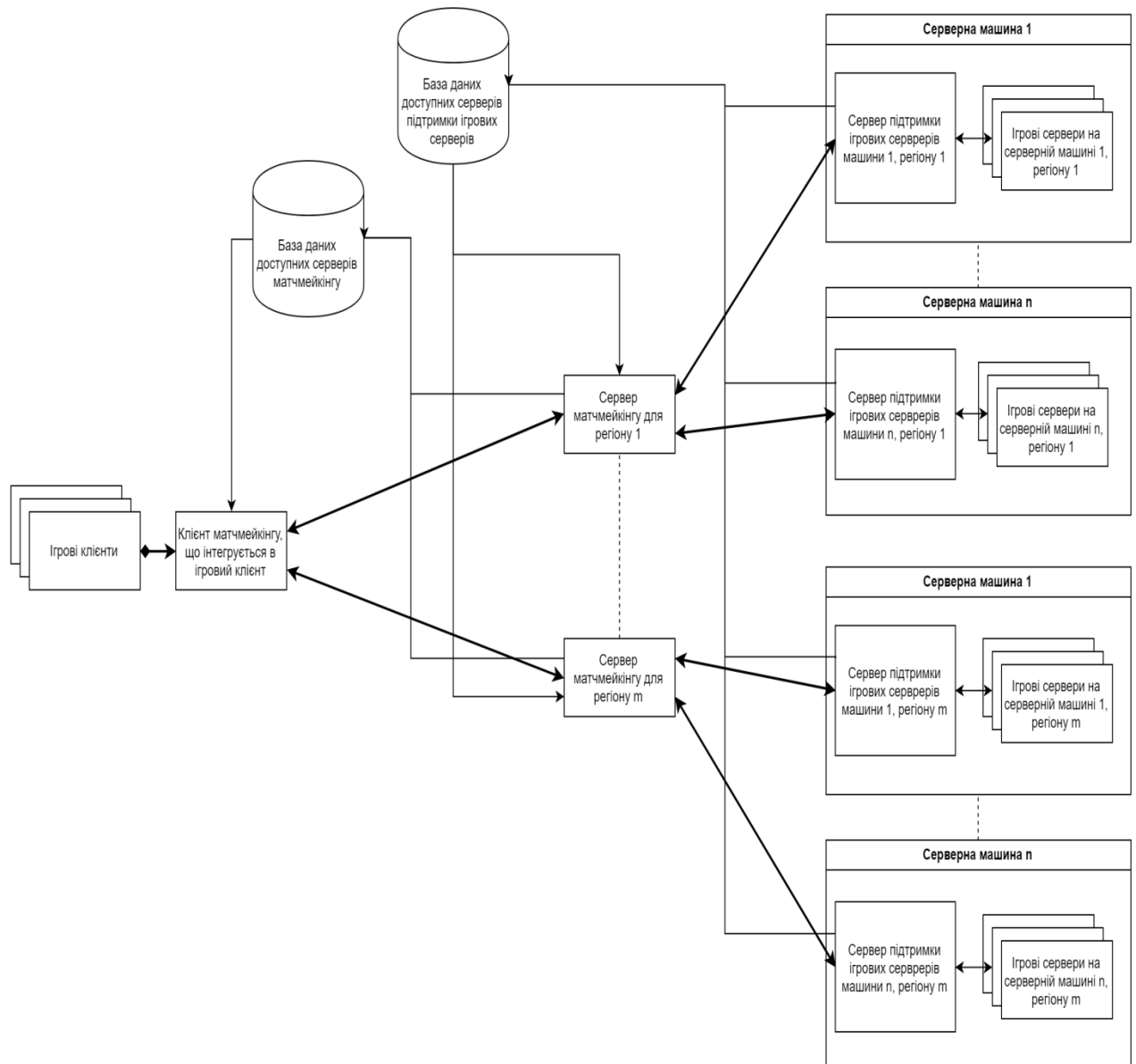


Рисунок 3.1 – Структура програмного забезпечення підбору гравців у комп'ютерних іграх

Як видно з рис. 3.1 програмного забезпечення підбору гравців у комп'ютерних іграх складається з таких модулів:

- клієнт матчмейкінгу, що інтегрується в ігровий клієнт – відповідає за з'єднання з доступним сервером матчмейкінгу та надання інформації ігровому клієнту щодо знайденого ігрового сервера;

- сервер матчмейкінгу – відповідає за отримання ігрової інформації, її сортування за відповідними параметрами, з'єднання з доступним сервером підтримки ігрових серверів для запиту розгортання ігрового сервера, відсилання інформації про ігровий сервер клієнтам матчмейкінгу, для котрих було знайдено сходження, при власному розгортанні – надає інформацію про готовність до бази даних доступних серверів матчмейкінгу;

- сервер підтримки ігрових серверів (сервер розгортання) – відповідає за розгортання та працездатність ігрових серверів, при власному розгортанні – надає інформацію про готовність до бази даних доступних серверів підтримки;

- база даних доступних серверів матчмейкінгу – зберігає інформацію про доступні в поточний час сервери матчмейкінгу;

- база даних доступних серверів підтримки ігрових серверів – зберігає інформацію про доступні в поточний час сервери підтримки ігрових серверів.

3.2 Функціонування програмного забезпечення для підбору гравців у комп'ютерних іграх

Схему взаємодії модулів програмного забезпечення для підбору гравців у комп'ютерних іграх між собою наведено на рисунку 3.2.

Згідно наведеної схеми робота програмного забезпечення для підбору гравців у комп'ютерних іграх складається з таких етапів.

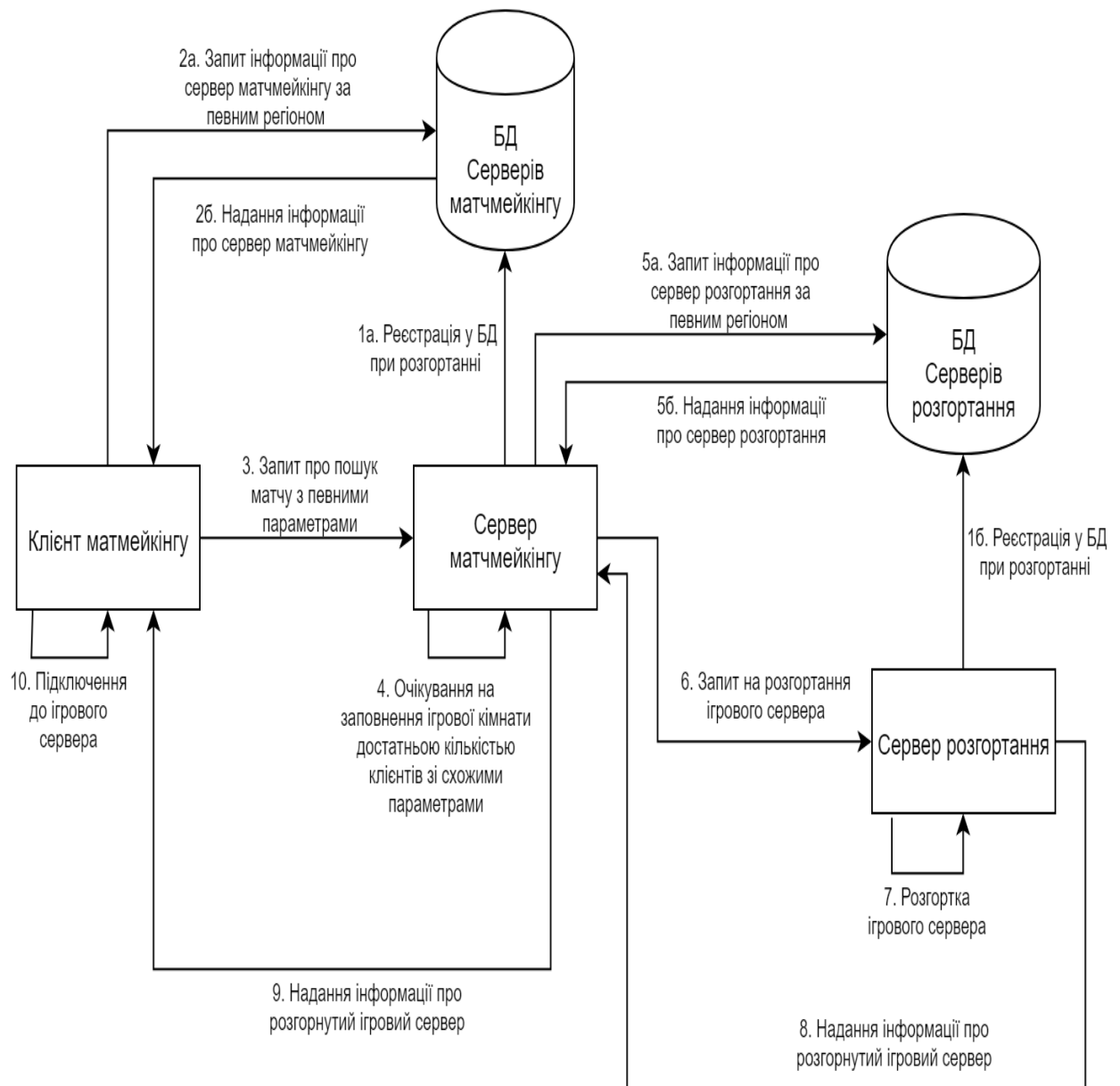


Рисунок 3.2 – Схема взаємодії модулів програмного забезпечення для підбору гравців у комп'ютерних іграх

На першому етапі виконується реєстрація серверів у базі даних:

- реєстрація серверу матчмейкінгу у базі даних (виконується автоматично одразу після запуску);
- реєстрація серверу розгортання у базі даних (виконується автоматично одразу після запуску).

На другому етапі відбувається отримання клієнтом даних для підключення до серверу матчмейкінгу.

Другий етап передбачає виконання таких процесів:

- надсилання запиту до бази даних із зазначеним регіоном;
- отримання інформації про сервер матчмейкінгу у зазначеному регіоні.

На третьому етапі виконується надсилання клієнтом запиту із власними ігровими параметру до серверу матчмейкінгу про пошук матчу.

Далі відбувається створення сервером матчмейкінгу кімнати із певними ігровими параметрами та очікування заповнення достатньою кількістю клієнтів із параметрами, що підходять параметрам кімнати.

На п'ятому етапі виконується отримання сервером матчмейкінгу даних для підключення до серверу розгортання:

- надсилання запиту до бази даних із зазначеним регіоном;
- отримання інформації про доступний сервер розгортання у зазначеному регіоні.

Після цього виконується надсилання сервером матчмейкінгу запиту на розгортання ігрового серверу до серверу розгортання. Далі відбувається розгортання ігрового серверу сервером розгортання із унікальним мережевим портом, надання сервером розгортання інформації про розгорнутий ігровий сервер серверу матчмейкінгу, надання сервером матчмейкінгу інформації про розгорнутий ігровий сервер усім клієнтам із кімнати, закриття кімнати.

На останньому етапі відбувається підключення клієнтом до ігрового серверу за наданою інформацією.

Зобразимо процеси, що відбуваються під час функціонування програмного забезпечення для підбору гравців у комп'ютерних іграх у вигляді діаграми діяльності програмного забезпечення для підбору гравців у комп'ютерних іграх (рис. 3.3) поданої у нотації UML.

Діаграма послідовності програмного забезпечення для підбору гравців у комп'ютерних іграх буде наведено нижче.

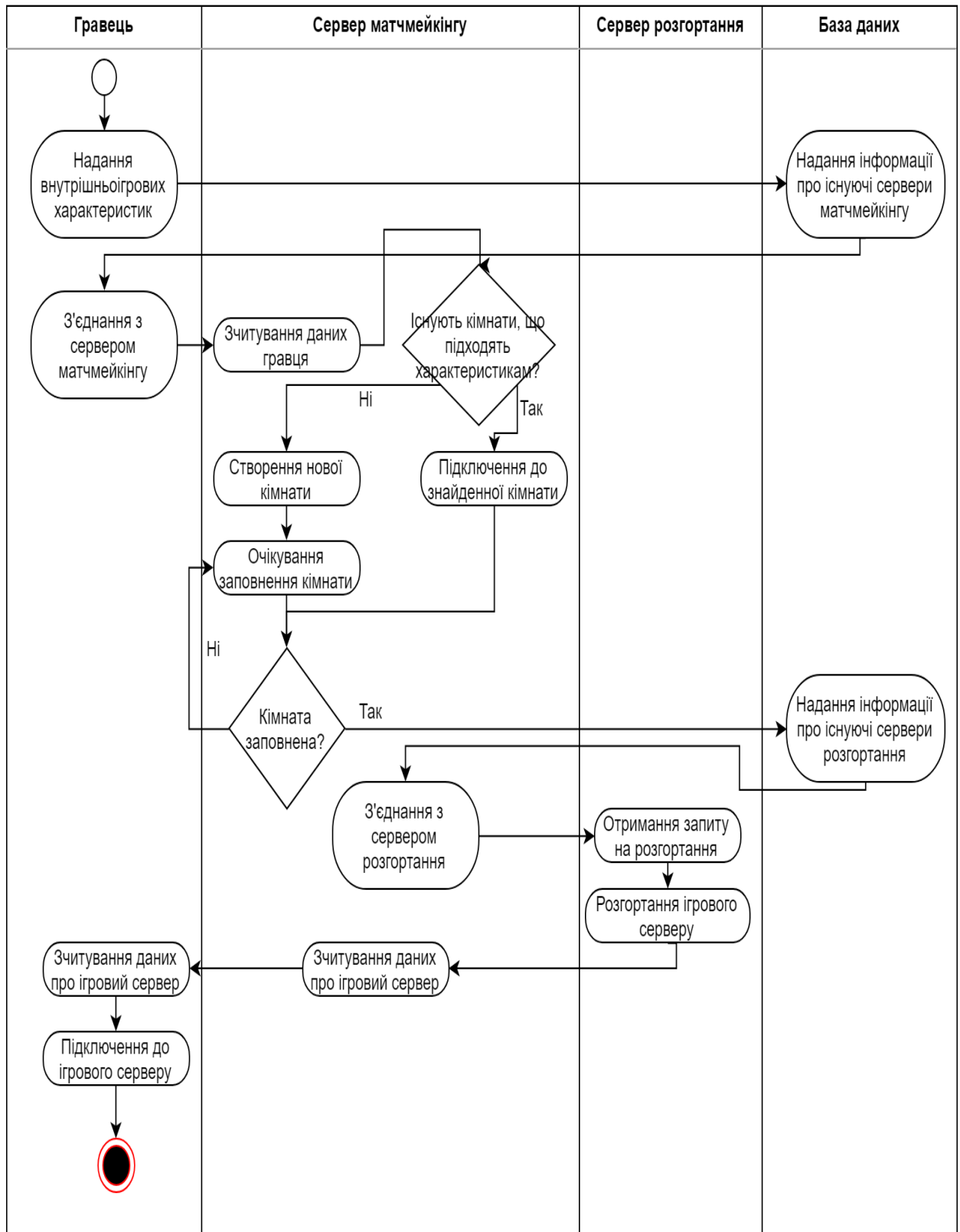


Рисунок 3.3 – Діаграма діяльності програмного забезпечення для підбору гравців у комп'ютерних іграх

Діаграма послідовності програмного забезпечення для підбору гравців у комп'ютерних іграх (рис. 3.4).

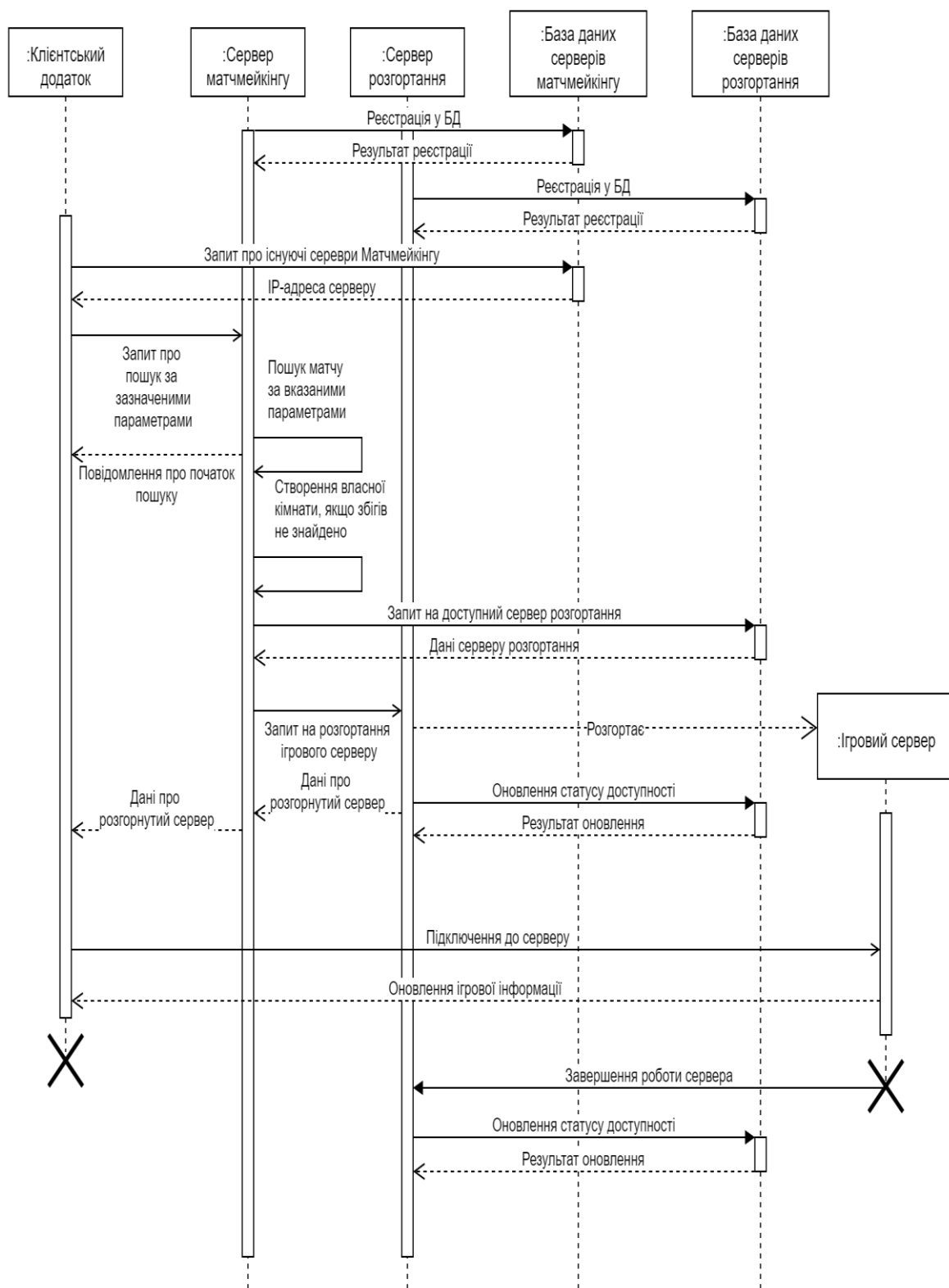


Рисунок 3.4 – Діаграма послідовності програмного забезпечення для підбору гравців у комп'ютерних іграх

3.3 Основні рішення щодо реалізації програмного забезпечення для підбору гравців у комп'ютерних іграх

У таблиці 3.1 наведено основні класи та функції модулів, а також їх призначення.

Таблиця 3.1 – Основні рішення щодо реалізації системи

Назва компоненту	Назва класу або функції	Короткий опис
1	2	3
Matchmaking Server	class DS_DataBase	Реалізує паттерн Singleton, забезпечує з'єднання та надає інструментарій для роботи із базою даних серверів розгортання
	class MS_DataBase	Реалізує паттерн Singleton, забезпечує з'єднання та надає інструментарій для роботи із базою даних серверів матчмейкінгу
	class talk_to_client	Забезпечує мережеву взаємодію із клієнтами
	class talk_to_dsrv	Забезпечує мережеву взаємодію із серверами розгортання
	class GameRoom	Зберігає користувачів зі схожими параметрами до остаточного заповнення і потім повідомляє про необхідність отримати ігровий сервер
Deployment Server	class DataBase	Реалізує паттерн Singleton, забезпечує з'єднання та надає інструментарій для роботи із базою даних серверів розгортання

Продовження таблиці 3.1

1	2	3
Deployment Server	class GameServerInstance	Забезпечує розгортання ігрових серверів, зберігає процес розгорнутого сервера та надає інструментарій роботи з ним
	class talk_to_client	Забезпечує мережеву взаємодію із серверами матчмейкінгу
	std::string GetExternalIP()	Повертає зовнішню IP-адресу комп'ютера
Client	class DataBase	Реалізує паттерн Singleton, забезпечує з'єднання та надає інструментарій для роботи із базою даних серверів матчмейкінгу
	class talk_to_svr	Забезпечує мережеву взаємодію із серверами матчмейкінгу
	class MultithreadedTask	Описує мережеву задачу очікування відповіді, що повинна виконуватись у окремому потоці (для UnrealEngine4)
	bool StartSearching()	Збирає внутрішньоігрову інформацію та розпочинає мережеву взаємодію

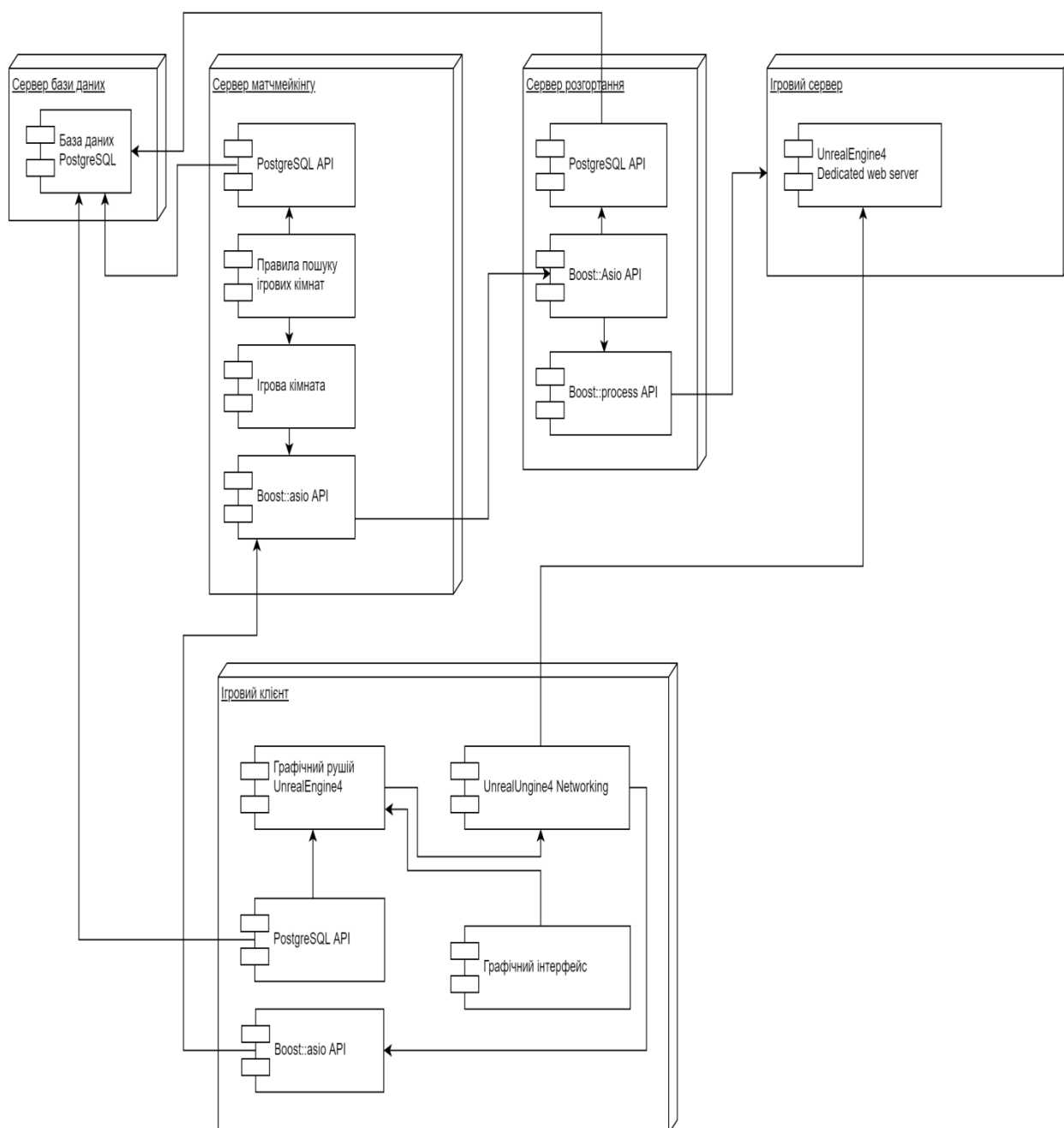


Рисунок 3.6 – Діаграма компонентів програмного забезпечення для підбору гравців у комп'ютерних іграх

Як видно з рис. 3.1 та 3.2, у програмному забезпеченні для підбору гравців у комп'ютерних іграх використовуються база даних серверів матчмейкінгу та бази даних серверів розгортання.

Структура бази даних серверів матчмейкінгу описана у таблиці 3.2.

Таблиця 3.2 – Структура бази даних серверів матчмейкінгу

Назва таблиці	Назва поля	Тип поля	Ключ	Опис
Regionlist	Region_id	Лічильник	Так	Призначене для унікальності записів у полях
	Region_name	Текстовий	—	Призначене для збереження інформації про назву регіону
Serverlist	Server_id	Лічильник	Так	Призначене для унікальності записів у полях
	Server_ipv4	Числовий	—	Призначене для збереження IP-адреси сервера матчмейкінгу
	Server_port	Числовий	—	Призначене для збереження мережевого порту сервера матчмейкінгу
	Server_region	Числовий	—	Призначене для збереження регіону сервера матчмейкінгу

Структура бази даних серверів розгортання описана у таблиці 3.3.

Таблиця 3.3 – Структура бази даних серверів розгортання

Назва таблиці	Назва поля	Тип поля	Ключ	Опис
1	2	3	4	5
DRegionlist	DRegion_id	Лічильник	Так	Призначене для унікальності записів у полях
	DRegion_name	Текстовий	—	Призначене для збереження інформації про назву регіону

Продовження таблиці 3.2

1	2	3	4	5
DServerlist	DServer_id	Лічильник	Так	Призначене для унікальності записів у полях
	DServer_ipv4	Числовий	—	Призначене для збереження IP-адреси сервера розгортання
	DServer_port	Числовий	—	Призначене для збереження мережевого порту сервера розгортання
	DServer_region	Числовий	—	Призначене для збереження регіону сервера розгортання
	DServer_freeslots	Числовий	—	Призначене для збереження кількості доступних ігрових серверів для розгортання

3.4 Висновки за розділом 3

Розроблено програмне забезпечення для підбору гравців у комп'ютерних іграх. Запропоновано структуру підбору гравців у комп'ютерних іграх. Визначено, що програмне забезпечення підбору гравців у комп'ютерних іграх складається з таких модулів: клієнт матчмейкінгу, що інтегрується в ігровий клієнт, сервер матчмейкінгу, сервер підтримки ігрових серверів (сервер розгортання), база даних доступних серверів матчмейкінгу та база даних доступних серверів підтримки ігрових серверів.

Розроблено схему взаємодії модулів програмного забезпечення для підбору гравців у комп'ютерних іграх між собою. Процеси, що відбуваються під час функціонування програмного забезпечення для підбору гравців у комп'ютерних іграх у вигляді, подано у вигляді діаграм діяльності та послідовності у нотації UML. Наведено основні рішення щодо реалізації програмного забезпечення для підбору гравців у комп'ютерних іграх. Подано структуру бази даних серверів матчмейкінгу та бази даних серверів розгортання.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для підбору гравців у комп'ютерних іграх. Програма написана на мові C++, в якості середовища розробки для створення програмного забезпечення підбору гравців у комп'ютерних іграх обрано Visual Studio, що є містить потужні функції для підтримки процесу розробки програм.

Програмне забезпечення для підбору гравців у комп'ютерних іграх забезпечує виконання таких функцій:

- пошук матчу для одного гравця відповідно до його внутрішньоігрових характеристик;
- пошук матчу для команди гравців відповідно до внутрішньоігрових характеристик команди;
- розробникам функцій задавання критеріїв об'єднання гравців;
- задавання пріоритету критеріїв об'єднання гравців;
- задавання діапазону мережевих портів, які будуть використовуватися для розгортання ігрових серверів;
- надання мережевої адреси розгорнутого ігрового сервера для певної ігрової сесії;
- задавання типу ігрових серверів для розгортання.

Для експлуатації програмного забезпечення для підбору гравців у комп'ютерних іграх необхідно таке апаратне та програмне забезпечення:

- 4-ядерний процесор з тактовою частотою 2 ГГц;
- оперативна пам'ять – 8Гб або вище;
- операційна система не старіше за Windows 10 64-bit;
- 50Гб на жорсткому диску;
- монітор, клавіатура та маніпулятор типу «миша»;
- засоби доступу до Інтернету.

Для клієнтської частини програмного забезпечення для підбору гравців у комп'ютерних іграх необхідно таке апаратне та програмне забезпечення: персональний комп'ютер з операційною системою не старіше за Windows 10 64-bit, 4-ядерний процесор з тактовою частотою 3ГГц, 8 Гб ОЗП, 10Гб на жорсткому диску, монітор, клавіатура та маніпулятор типу «миша».

Також для коректної роботи серверів матчмейкінгу та розгортання на машині, на якій вони запускаються, необхідно мати статичну IP-адресу, відкриті порти та правила-виключення у брандмауері для цих портів.

Функціональні характеристики розробленого програмного забезпечення для підбору гравців у комп'ютерних іграх наведено у технічному завданні (додаток А). Фрагмент тексту програмного забезпечення для підбору гравців у комп'ютерних іграх наведено у додатку Б.

4.2 Характеристики програми та її встановлення

Програмне забезпечення для підбору гравців у комп'ютерних іграх працює відповідно до потреб користувача.

Сервери матчмейкінгу та розгортання автоматично реєструються у базі під час запуску, відповідно становляться відразу доступними до мережових з'єднань, при завершенні роботи автоматично видаляються з бази, тому допоміжних корегувань у бази даних не потребують.

Сервера, як і клієнтська частина, працюють асинхронно, тому можуть витримувати велику кількість одночасних підключень, особливого режиму роботи не потребують.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Завантажувальний файл сервера програмного забезпечення для підбору гравців у комп'ютерних іграх має назву MatchmakingServer.exe, для коректної

роботи сервер потрібно запускати через ярлик, який необхідно створити та задати наступні параметри:

- «-port=`номер`» – порт, за яким сервер матчмейкінгу буде прослуховувати вхідні підключення, без вказання параметру використовуватиметься порт `8501`;

- «-msdbip=`IP-адреса`» – IP-адреса бази даних серверів матчмейкінгу, без вказання параметру використовуватиметься IP-адреса `127.0.0.1`;

- «-dsdbip=`IP-адреса`» – IP-адреса бази даних серверів розгортання, без вказання параметру використовуватиметься IP-адреса `127.0.0.1`;

- «-reg=`регіон`» – регіон серверу матчмейкінгу, повинен співпадати із значеннями, вказаними в базі даних серверів матчмейкінгу, без вказання параметру використовуватиметься регіон `EU`.

Завантажувальний файл сервера розгортання має назву DeploymentServer.exe, для коректної роботи сервер потрібно запускати через ярлик, який необхідно створити та задати наступні параметри:

- «-port=`номер`» – порт, за яким сервер матчмейкінгу буде прослуховувати вхідні підключення, без вказання параметру використовуватиметься порт `8927`;

- «-dsdbip=`IP-адреса`» – IP-адреса бази даних серверів розгортання, без вказання параметру використовуватиметься IP-адреса `127.0.0.1`;

- «-reg=`регіон`» – регіон серверу розгортання, повинен співпадати із значеннями, вказаними в базі даних серверів розгортання, без вказання параметру використовуватиметься регіон `EU`;

- «-max=`кількість`» – максимальна кількість ігрових серверів, що можуть бути розгорнуті на цій серверній машині, без вказання параметру використовуватиметься значення `10`.

Ігрові сервера, які розгортаються сервером розгортання на певній машині, використовують мережеві порти починаючи з `7777`, номер порту збільшується з кожним запуском ігровим сервером на `1`, максимальне

можливе значення порту дорівнює сумі початкового порту `7777` і зазначеної максимальної кількості ігрових серверів, зменшеної на `1`.

Шлях до завантажувального файлу ігрового серверу повинен бути зазначений у системній змінній «Path», також файл повинен мати назву «GameServer.exe».

База даних серверів матчмейкінгу називається MM_Servers і зберігається на сервері у різних файлах, база даних серверів розгортання називається MM_DeploymentServers і зберігається на сервері у різних файлах. Доступ до таблиць реалізується за допомогою вебдодатку phpPgAdmin.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення для підбору гравців у комп'ютерних іграх є внутрішньоігрові характеристики користувача, обраний регіон пошуку, кількість людей у команді, які шукають матч, та обраний максимальний розмір команди.

Вихідними даними програмного забезпечення для підбору гравців у комп'ютерних іграх є IP-адреса та мережевий порт ігрового сервера, який використовується клієнтським застосунком для підключення.

4.3.3 Повідомлення

Користувач програмного забезпечення для підбору гравців у комп'ютерних іграх може отримати такі повідомлення: повідомлення про те, що сервер був вдало запущений, та коректно функціонує; повідомлення про невдале з'єднання із базою даних серверів матчмейкінгу, надаючи IP-адресу бази даних, до якої сервер не зміг під'єднатись, потрібно перевірити роботу бази даних та коректність IP-адреси бази даних; повідомлення про невдале з'єднання із базою даних серверів розгортання, надаючи IP-адресу бази даних, до якої сервер не зміг під'єднатись, потрібно перевірити роботу бази даних та

коректність IP-адреси бази даних; повідомлення про те, що клієнт з зазначеним ім'ям створив кімнату із зазначеним ім'ям кімнати; повідомлення про те, що клієнт з зазначеним ім'ям приєднався до кімнати із зазначеним ім'ям кімнати.

4.4 Виконання програмного забезпечення для збирання даних з вебсайту

Оскільки програмне забезпечення для підбору гравців у комп'ютерних іграх являє собою настільну програму, то для її запуску користувачу необхідно запустити виконуваний файл Matchmaking.exe.

Після завантаження програми з'явиться меню пошуку матчу (рис. 4.1). Користувачу необхідно заповнити текстові поля, вказавши ім'я («Username»), коефіцієнт Ело («Elo»), що відображає вміння гравця у числовому вигляді, бажаний максимальний розмір команди у матчі («TeamSize»), розмір поточної групи («PartySize»), та вказати регіон пошуку (праворуч обрати «EU» або «US»).

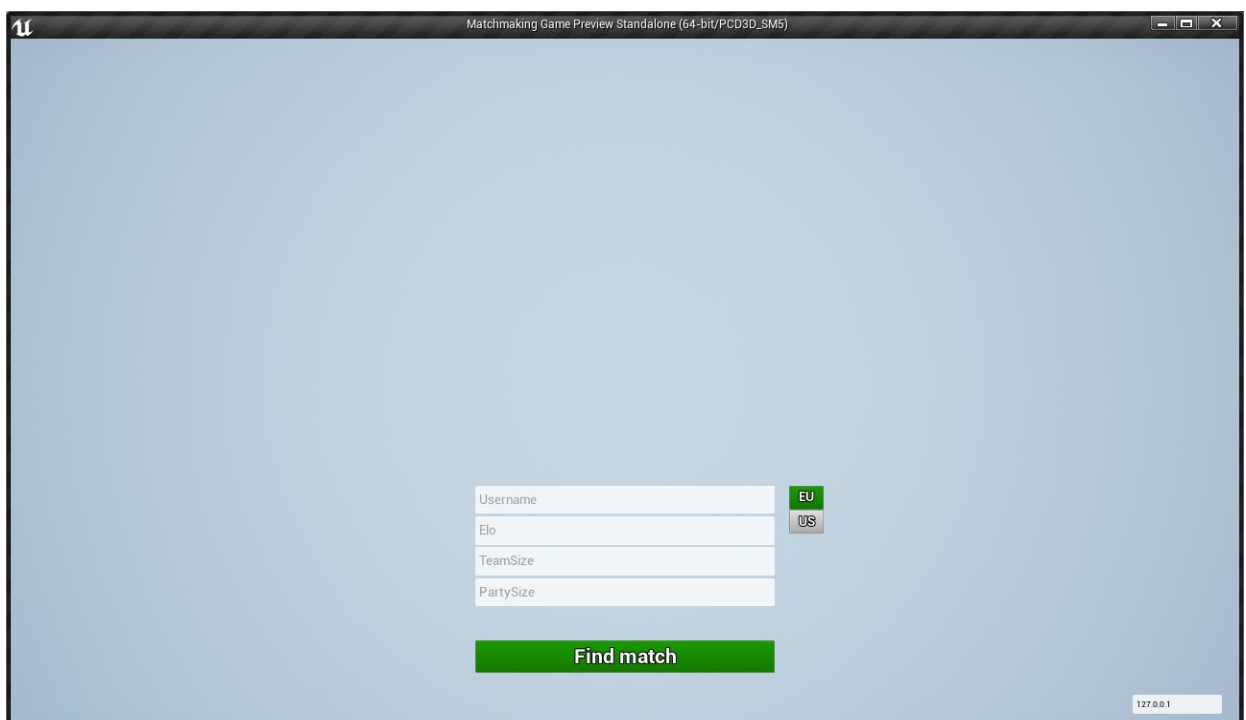


Рисунок 4.1 – Меню пошуку матчу

У правому нижньому куті необхідно задати IP-адресу бази даних серверів матчмейкінгу.

Після заповнення полів потрібно натиснути кнопку «Find match», якщо все правильно заповнено, почнеться пошук, текстові поля заблокуються, а у центрі екрану з'являться зелені квадрати, що обертаються (рис. 4.2). Після того як знайдеться достатня кількість користувачів із схожими параметрами користувача автоматично під'єднає до ігрового серверу.

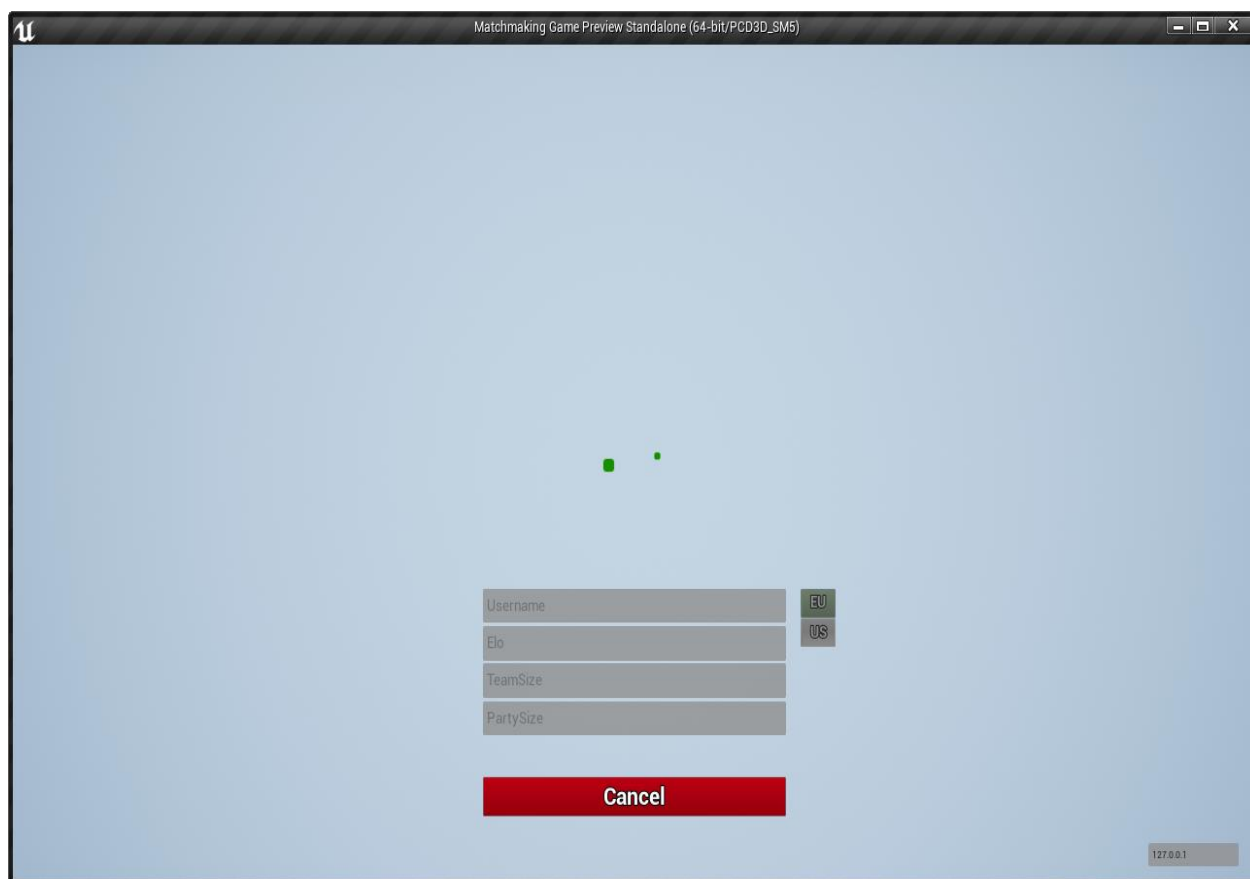


Рисунок 4.2 – Очікування матчу

Після з'єднання з ігровим сервером можна буде побачити характеристики кожного гравця, якого було додано до матчу, приклад зображено на рис. 4.3.



Рисунок 4.3 – Приклад інформаційної дошки біля кожного гравця

При неправильному введенні даних користувачу на екран виведеться повідомлення про те, що він увів неправильні дані (рис. 4.4).

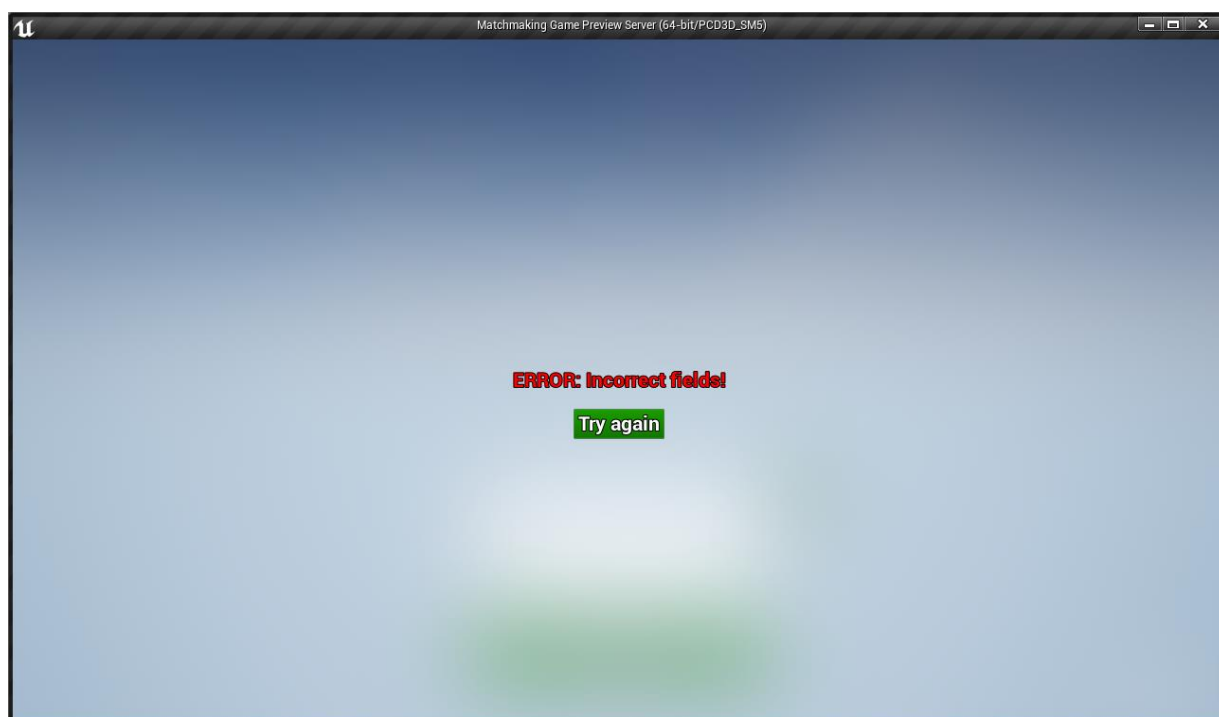


Рисунок 4.4 – Повідомлення про некоректність даних

4.5 Тестування програмного забезпечення для підбору гравців у комп'ютерних іграх

Задача тестування програмного забезпечення для підбору гравців у комп'ютерних іграх полягала у перевірці точності підбору гравців за заданими ігровими характеристиками.

Перелік наданих характеристик, з якими працює програмне забезпечення для підбору гравців у комп'ютерних іграх:

- умовний коефіцієнт Ело – основна ігрова характеристика що відображає ігрову здібність гравця. Числове значення коефіцієнту Ело розраховується незалежно від розроблюваної системи;

- максимальна кількість гравців у кожній команді – обирається гравцем, та впливає на розмір шуканої ігрової кімнати;

- кількість гравців у групі – характеристика, що повинна по можливості автоматично визначатися ігровим клієнтом, позначає кількість гравців у команді, що шукають матч разом.

- регіон пошуку матчу – характеристика, яка визначає регіон, в якому буде проводитись пошук матчу, від обраного регіону пошуку залежить регіон, в якому будуть розгортатися ігрові сервери, для знайденого матчу.

Методика проведення тестування програмного забезпечення для підбору гравців у комп'ютерних іграх:

- задати різні значення для коефіцієнту Ело та кількості гравців у групі різним клієнтам, при однаковій максимальній кількості гравців у кожній команді;

- провести пошук матчу для клієнтів із попереднього пункту декілька разів, змінюючи для всіх максимальну кількість гравців у кожній команді;

- перевірити різницю середніх значення коефіцієнту Ело між командами;

– результати можна вважати успішними, якщо різниця середніх значень коефіцієнту Ело між командами із однаковою максимальною кількістю гравців у кожній команді буде менше за 100.

Результати тестування програмного забезпечення для підбору гравців у комп'ютерних іграх наведено у табл. 4.1.

Таблиця 4.1 – Результати тестування програмного забезпечення для підбору гравців у комп'ютерних іграх

№	Вхідні дані	Звіт серверу матчмейкінгу	Результат
1	2	3	4
1	Username = «Party1», Elo = 1400, TeamSize = 1, PartySize = 1	Додано до команди «team2» в кімнаті «0»	Користувачів «Party1» та «PARTy3» було додано до одного матчу, а користувач «Party2» знаходиться в очікуванні. Team1 Elo = 1301; Team2 Elo = 1400.
	Username = «Party2», Elo = 1300, TeamSize = 1, PartySize = 1	Додано до команди «team2» в кімнаті «1»	
	Username = «PARTy3», Elo = 1301, TeamSize = 1, PartySize = 1	Додано до команди «team1» в кімнаті «0»	
2	Username = «Party1», Elo = 2100, TeamSize = 2, PartySize = 2	Додано до команди «team2» в кімнаті «0»	Користувачів «Party1», «Party2», «Party3» було додано до одного матчу, але в різні команди. Команда «team2»: 2 користувача в групі «Party1», середнє Elo = 2100. Команда «team1»: 1 користувач в групі «Party2», 1 користувач в групі «Party3», середнє Elo = 2090.
	Username = «Party2», Elo = 2050, TeamSize = 2, PartySize = 1	Додано до команди «team1» в кімнаті «0»	
	Username = «Party3», Elo = 2130, TeamSize = 2, PartySize = 1	Додано до команди «team1» в кімнаті «0»	

Продовження таблиці 4.1

1	2	3	4
3	Username = «Party1», Elo = 700, TeamSize = 3, PartySize = 2	Додано до команди «team2» в кімнаті «1»	Користувачів «Party1», «Party2», «Party3», «party555», «Patyу6»
	Username = «Party2», Elo = 780, TeamSize = 3, PartySize = 1	Додано до команди «team1» в кімнаті «1»	було додано до одного матчу, але в різні команди, а користувач
	Username = «Party3», Elo = 820, TeamSize = 3, PartySize = 1	Додано до команди «team2» в кімнаті «1»	«P4rty» знаходиться в очікуванні.
	Username = «P4rty», Elo = 800, TeamSize = 3, PartySize = 1	Додано до команди «team2» в кімнаті «2»	Команда «team2»: 2 користувача в групі «Party1», 1 користувач в групі «Party3» середнє
	Username = «party555», Elo = 760, TeamSize = 3, PartySize = 1	Додано до команди «team1» в кімнаті «1»	Elo = 760. Команда «team1»: 1 користувач в групі
	Username = «Patyу6», Elo = 750, TeamSize = 3, PartySize = 1	Додано до команди «team1» в кімнаті «1»	«Party2», 1 користувач в групі «Party3», 1 користувач в групі «Patyу6», середнє Elo = 763,333.

З таблиці 4.1 можна зробити висновок, що різниця середніх значень Elo команд не перевищує 100 в усіх випробуваннях, а команди, які складаються із двох членів і більше ще й балансуються, тому різниця середніх значень Elo цих команд мінімальна.

Отже результати тестування програмного забезпечення для підбору гравців у комп'ютерних іграх можна вважати успішними.

Таким чином, результати тестування програмного забезпечення для матчмейкінгу показали, що програма відповідає технічному завданню, функціонує коректно та дозволяє виконувати підбір гравців у комп'ютерних іграх.

4.6 Висновки за розділом 4

Описано програмне забезпечення для підбору гравців у комп'ютерних іграх. Виконано тестування розробленого програмного забезпечення для підбору гравців у комп'ютерних іграх. За результатами тестування програмного забезпечення для підбору гравців у комп'ютерних іграх виявлено, що програма відповідає технічному завданню, функціонує коректно та дозволяє виконувати підбір гравців у комп'ютерних іграх.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки комп'ютерних ігор.

Визначено, що у наш час широко використовуються комп'ютерні ігри, що являють собою ігри, керовані комп'ютером, де гравці з метою розваги взаємодіють з об'єктами або іншими гравцями, що відображаються на екрані. Відзначено, що існують багато відеоігор різних жанрів тобто категорій ігор, які мають подібний ігровий процес і ігрову механіку. Виконано аналіз жанрів відеоігор. Відзначено, що для формування команд у відеоіграх, у т.ч. у кіберспорті, де необхідною є участь декількох гравців в одній команді, використовуються методи матчмейкінгу, що являє собою процес формування команд для комп'ютерної гри. Його головною метою є створення рівних умов для всіх гравців, де вони матимуть шанс на перемогу незалежно від свого рівня ігрової майстерності. Визначено, що процес матчмейкінгу базується на алгоритмах, які враховують різні фактори, такі як ранг, рейтинг, історія гри та інші параметри.

За результатами попереднього аналізу зроблено висновок про те, що у наш час існує досить багато програмних систем для підбору гравців у комп'ютерних іграх. Проте деякі з них є досить дорогими, інші досить складними для використання. Тому актуальною є розробка програмного забезпечення для підбору гравців у комп'ютерних іграх.

Сформульовано функціональні вимоги до програмного забезпечення для підбору гравців у комп'ютерних іграх.

Для реалізації програмного забезпечення підбору гравців у комп'ютерних іграх обрано мову програмування C++, яка є статично типізованою, скомпільованою, багатопарадигмальною мовою програмування загального призначення, яка широко використовується у відеоіграх, настільному програмному забезпеченні та розробці вбудованих систем.

В якості мережевої бібліотеки для забезпечення функцій асинхронного введення-виведення обрано Boost.Asio, яка є широко використовуваною бібліотекою C++, що забезпечує створення мережевих програм, зокрема і мережевих ігор.

Для створення програмного забезпечення підбору гравців у комп'ютерних іграх обрано середовище розробки Visual Studio.

Розроблено програмне забезпечення для підбору гравців у комп'ютерних іграх. Запропоновано структуру підбору гравців у комп'ютерних іграх. Визначено, що програмне забезпечення підбору гравців у комп'ютерних іграх складається з таких модулів: клієнт матчмейкінгу, що інтегрується в ігровий клієнт, сервер матчмейкінгу, сервер підтримки ігрових серверів (сервер розгортання), база даних доступних серверів матчмейкінгу та база даних доступних серверів підтримки ігрових серверів.

Розроблено схему взаємодії модулів програмного забезпечення для підбору гравців у комп'ютерних іграх між собою. Процеси, що відбуваються під час функціонування програмного забезпечення для підбору гравців у комп'ютерних іграх у вигляді, подано у вигляді діаграм діяльності та послідовності у нотації UML. Наведено основні рішення щодо реалізації програмного забезпечення для підбору гравців у комп'ютерних іграх. Подано структуру бази даних серверів матчмейкінгу та бази даних серверів розгортання.

Виконано тестування розробленого програмного забезпечення для підбору гравців у комп'ютерних іграх. За результатами тестування програмного забезпечення для підбору гравців у комп'ютерних іграх виявлено, що програма відповідає технічному завданню, функціонує коректно та дозволяє виконувати підбір гравців у комп'ютерних іграх.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Video Games [Electronic resource]. – Access mode: <https://www.twinkl.com/teaching-wiki/video-games>.
2. The Definitive Guide to Video Game Genres and Game Types [Electronic resource]. – Access mode: <https://codakid.com/video-game-genres/>.
3. The Many Different Types of Video Games & Their Subgenres [Electronic resource]. – Access mode: <https://www.idtech.com/blog/different-types-of-video-game-genres>.
4. The Complete Guide to Video Game Genres: From Scrollers, Shooters, to Sports [Electronic resource]. – Access mode: <https://www.gamedesigning.org/gaming/video-game-genres/>.
5. Guide to Video Game Genres: 10 Popular Video Game Types [Electronic resource]. – Access mode: <https://www.masterclass.com/articles/guide-to-video-game-genres>.
6. How to Ensure Your Matchmaking Experience for Multiplayer Games Is Fast and Full [Electronic resource]. – Access mode: <https://subspace.com/resources/multiplayer-matchmaking>.
7. Matchmaking Algorithm: Skill-based Matchmaking [Electronic resource]. – Access mode: <https://www.pubnub.com/blog/skill-based-matchmaking-multiplayer-games-tutorial/>.
8. Suggestions for Matchmaking Algorithm [Electronic resource]. – Access mode: <https://datascience.stackexchange.com/questions/57974/suggestions-for-matchmaking-algorithm>.
9. Matchmaking [Electronic resource]. – Access mode: <https://www.giantbomb.com/matchmaking/3015-3334/>.
10. Playfab [Electronic resource]. – Access mode : <https://playfab.com/>.
11. Amazon GameLift [Electronic resource]. – Access mode: <https://aws.amazon.com/ru/gamelift/>.

12. GameSparks [Electronic resource]. – Access mode: <https://www.gamesparks.com/product/>.
13. Photon Engine [Electronic resource]. – Access mode: <https://www.photonengine.com/>.
14. Nakama [Electronic resource]. – Access mode: <https://heroiclabs.com/nakama/>.
15. C++ Tutorial [Electronic resource]. – Access mode: <https://www.tutorialspoint.com/cplusplus/index.htm>.
16. Learn C++ [Electronic resource]. – Access mode: <https://www.learncpp.com/>.
17. How to Learn the C++ Programming Language [Electronic resource]. – Access mode: <https://www.freecodecamp.org/news/how-to-learn-the-c-programming-language/>.
18. Boost.Asio [Electronic resource]. – Access mode: https://www.boost.org/doc/libs/1_76_0/doc/html/boost_asio.html.
19. Visual Studio [Electronic resource]. – Access mode: <https://www.incredibuild.com/integrations/visual-studio>.
20. Welcome to the Visual Studio IDE [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022>.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення для матчмейкінгу може використовуватися для підбору гравців у комп'ютерних іграх.

А.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення для підбору гравців у комп'ютерних іграх», затверджене наказом Національного університету «Запорізька політехніка» № 87 від 4 квітня 2023 р.

А.2 Призначення розробки

Програмний продукт призначений для підбору гравців у комп'ютерних іграх.

А.3 Основні вимоги до програми, що розробляється

А.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для підбору гравців у комп'ютерних іграх забезпечує виконання таких функцій:

- пошук матчу для одного гравця відповідно до його внутрішньоігрових характеристик;
- пошук матчу для команди гравців відповідно до внутрішньоігрових характеристик команди;
- розробникам функцій задавання критеріїв об'єднання гравців;
- задавання пріоритету критеріїв об'єднання гравців;
- задавання діапазону мережевих портів, які будуть використовуватися для розгортання ігрових серверів;

- надання мережевої адреси розгорнутого ігрового сервера для певної ігрової сесії;
- задавання типу ігрових серверів для розгортання.

A.3.2 Вимоги до інтерфейсу програми

Інтерфейс програми для підбору гравців у комп'ютерних іграх повинен бути зручним для користувачів. Взаємодія між програмою та користувачем повинна бути організована за допомогою стандартних засобів введення-виведення, зручних для використання користувачем.

A.3.3 Вимоги до надійності

Програмне забезпечення для підбору гравців у комп'ютерних іграх повинно забезпечити надійне функціонування та передбачити можливість запобігання некоректних ситуацій.

A.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для підбору гравців у комп'ютерних іграх необхідна наявність персонального комп'ютеру.

A.3.5 Вимоги до складу та параметрів технічний засобів

Для експлуатації програмного забезпечення для підбору гравців у комп'ютерних іграх необхідно таке апаратне та програмне забезпечення:

- пошук матчу для одного гравця відповідно до його внутрішньоігрових характеристик;

- пошук матчу для команди гравців відповідно до внутрішньоігрових характеристик команди;
- розробникам функцій задавання критеріїв об'єднання гравців;
- задавання пріоритету критеріїв об'єднання гравців;
- задавання діапазону мережевих портів, які будуть використовуватися для розгортання ігрових серверів;
- надання мережевої адреси розгорнутого ігрового сервера для певної ігрової сесії;
- задавання типу ігрових серверів для розгортання.

Для експлуатації програмного забезпечення для підбору гравців у комп'ютерних іграх необхідно таке апаратне та програмне забезпечення:

- 4-ядерний процесор з тактовою частотою 2 ГГц;
- оперативна пам'ять – 8Гб або вище;
- операційна система не старіше за Windows 10 64-bit;
- 50Гб на жорсткому диску;
- монітор, клавіатура та маніпулятор типу «миша»;
- засоби доступу до Інтернету.

A.3.6 Вимоги до маркування і пакування

Програма для підбору гравців у комп'ютерних іграх може бути записана на будь-якому носії інформації. На пакуванні повинна бути назва програми – «Програмне забезпечення для підбору гравців у комп'ютерних іграх».

A.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення для підбору гравців у комп'ютерних іграх є внутрішньоігрові характеристики користувача, обраний

регіон пошуку, кількість людей у команді, які шукають матч, та обраний максимальний розмір команди.

Вихідними даними програмного забезпечення для підбору гравців у комп'ютерних іграх є IP-адреса та мережевий порт ігрового сервера, який використовується клієнтським застосунком для підключення.

ДОДАТОК Б
Фрагмент тексту програми

```

class DataBase
{
private:
    std::string pghost;
    std::string pgport;
    PGconn *conn;
    PGresult *res;
    bool connected;
    int DServerID;
    /*DataBase()
    {
        pgport = "5432";
        connected = ConnectToDB();
    }*/
    DataBase(const std::string &pgh)
    {
        pghost = pgh;
        pgport = "5432";
        DServerID = 0;
        connected = ConnectToDB();
    }
    bool ConnectToDB()
    {
        conn = PQsetdbLogin(pghost.c_str(), pgport.c_str(), NULL, NULL,
"MM_DeploymentServers", "DServerUser", "DServerUser1");
        if (PQstatus(conn) == CONNECTION_OK)
        {
            std::cout << "Successfully connected to DB" << std::endl;
            return true;
        }
        else
        {
            std::cout << "Connect to DB failed" << std::endl;
            return false;
        }
    }
    bool UnregisterServerinDB()
    {
        //extern DataBase &db;
        DataBase &db = DataBase::getInstanceByHost();
        if (db.isConnected() && db.getServerID()>0)
        {

```

```

        db.ExecCommand("Delete    from    \"DServerlist\"    Where
\"DServer_id\" = \"\" +std::to_string(db.getServerID()) + \"\");
        if (PQresultStatus(db.getRes()) == PGRES_COMMAND_OK)
        {
            db.ClearRes();
            return true;
        }
        else
        {
            db.ClearRes();
            std::cout << "bad ip unregistration request result" <<
std::endl;
        }
    }
    else
        std::cout << "db is not connected or DServerID<=0" << std::endl;

    return false;
}

public:
    DataBase(const DataBase&) = delete;
    DataBase& operator=(DataBase&) = delete;

    bool isConnected() { return connected; }
    PGresult* getRes() { return res; }
    void ClearRes() { PQclear(res); }
    void setServerID(int id) { DServerID = id; }
    int getServerID() { return DServerID; }

    static DataBase& getInstanceByHost(const std::string &pgh = "127.0.0.1")
    {
        static DataBase pInstance(pgh);
        return pInstance;
    }

    void setHost(const std::string &pgh)
    {
        getInstanceByHost(pgh);
    }

    void ExecCommand(const std::string &comm)
    {
        if (connected)
        {

```

```

        res = PQexec(conn, comm.c_str());
    }

}
~DataBase()
{
    // del ip from db when closes
    UnregisterServerinDB();
    PQfinish(conn);
    //PQclear(res);
    system("pause");
}
};

io_service service;

class talk_to_client;
typedef boost::shared_ptr<talk_to_client> client_ptr;
typedef std::vector<client_ptr> varray;
varray clients;

#define MEM_FN(x)    boost::bind(&self_type::x, shared_from_this())
#define MEM_FN1(x,y) boost::bind(&self_type::x, shared_from_this(),y)
#define MEM_FN2(x,y,z) boost::bind(&self_type::x, shared_from_this(),y,z)

//ProcessFinishHandler

struct ProcessFinishHandler : boost::process::extend::async_handler
{
    typedef boost::shared_ptr<GameServerInstance> gsptr;

    int port;//boost::shared_ptr<GameServerInstance> gserver;
    gsptr pointer;
    ProcessFinishHandler() {}
    ProcessFinishHandler(int p)//boost::shared_ptr<GameServerInstance> g
    {
        port = p;
        std::cout << "port in structure:" << port << " input p = " << p <<
std::endl;
    }
    ProcessFinishHandler(boost::shared_ptr<GameServerInstance>          sho);
//boost::shared_ptr<GameServerInstance> g

```

```

void on_close();

template <typename ExecutorType>
std::function<void(int, std::error_code const&)>
on_exit_handler(ExecutorType&)
{
    return [=](int exit_code, std::error_code const& ec) // before without
'this'
    {
        if (ec)
        {
            std::cout << "process exit error: " << ec << std::endl;
            // handle error
        }
    }

#ifdef BOOST_POSIX_API
    if (WIFEXITED(exit_code))
        exit_code = WEXITSTATUS(exit_code);
#endif

    /*std::vector<int>::iterator it =
std::find(VPORTS_IN_USE.begin(), VPORTS_IN_USE.end(), port);
VPORTS_IN_USE.erase(it);*/
    //UpdateServerFreeslotsInDB();

    on_close();
    std::cout << "exit code: " << exit_code << std::endl; //port
    //on process close

};

}

};

class GameServerInstance : public
boost::enable_shared_from_this<GameServerInstance>, boost::noncopyable //,
boost::process::extend::async_handler
{
    typedef GameServerInstance self_type;
    GameServerInstance(const std::string &exe, std::vector<std::string> prms, int
p)
    {
        exename = exe;

```

```

        params = prms;
        port = p;
        //
    }

    void start()
    {
        handler = { shared_from_this() };
        processInstance = boost::process::child{
boost::process::search_path(exename), params, handler, service };
        if (processInstance.running())
        {
            VPORTS_IN_USE.push_back(port);
            ++NUMBER_OF_GAMESERVERS_RUNNING;
            UpdateServerFreeslotsInDB();
            gameservers.push_back(shared_from_this());
        }
    }

public :
    typedef boost::shared_ptr<GameServerInstance> gsiptr;
    static gsiptr pstart(const std::string &exe, std::vector<std::string> prms, int
port)
    {
        gsiptr new_(new GameServerInstance(exe, prms, port));
        new_->start();
        return new_;
    }
    bool isRunning() { return processInstance.running(); }

    void stop()
    {
        for (std::vector<int>::iterator it1 = VPORTS_IN_USE.begin(), it2 =
VPORTS_IN_USE.end(); it1 != it2; ++it1)
        {
            std::cout << "it1 = " << *it1 << " port = " << port << std::endl;
            if (*it1 == port)
            {
                std::cout << "Erasing port" << std::endl;
                VPORTS_IN_USE.erase(it1);
                break;
            }
        }
        --NUMBER_OF_GAMESERVERS_RUNNING;
        UpdateServerFreeslotsInDB();
    }

```

```

        std::cout << "deleting ptr to game server instance" << std::endl; //port
        gsiptr self = shared_from_this();
        gsInstancesVec::iterator it2 = std::find(gameservers.begin(),
gameservers.end(), self);
        if (it2 != gameservers.end())
        {
            gameservers.erase(it2);
        }
    }
private:
    boost::process::child processInstance;
    std::string exename;
    int port;
    std::vector<std::string> params;
    ProcessFinishHandler handler;
};

```

```

//GameServerInstance::GameServerInstance(const std::string &exe,
std::vector<std::string> prms, int p)

```

```

ProcessFinishHandler::ProcessFinishHandler(boost::shared_ptr<GameServerInstanc
e> sho)
{
    pointer = sho;
    std::cout << "port in structure:" << port << " input p = " << std::endl;
}
void ProcessFinishHandler::on_close()
{
    std::cout << "Game server is closing..." << std::endl;
    pointer->stop();
}

```

```

//void update_clients_changed();

```

```

class talk_to_client : public boost::enable_shared_from_this<talk_to_client>,
boost::noncopyable
{
    typedef talk_to_client self_type;
    talk_to_client() : sock_(service), started_(false), isReading(false) {}
public:
    typedef boost::system::error_code error_code;
    typedef boost::shared_ptr<talk_to_client> ptr;

```

```

void start()
{
    started_ = true;
    clients.push_back(shared_from_this());
    //last_ping = boost::posix_time::microsec_clock::local_time();
    // first, we wait for client to login
    do_read();
}
static ptr new_()
{
    ptr new_(new talk_to_client);
    return new_;
}
void stop()
{
    if (!started_) return;
    started_ = false;
    sock_.close();
    isReading = false;
    ptr self = shared_from_this();
    varray::iterator it = std::find(clients.begin(), clients.end(), self);
    clients.erase(it);
    //update_clients_changed();
}

bool started() const { return started_; }
ip::tcp::socket & sock() { return sock_; }
//std::string username() const { return username_; }
//void set_clients_changed() { clients_changed_ = true; }

private:
    void on_read(const error_code & err, size_t bytes)
    {
        std::cout << "on_read, error:" << err.message() << "bytes: " << bytes
        << std::endl;
        if (err) stop();
        if (!started()) return;
        isReading = false;
        // process the msg
        std::string msg(read_buffer_, bytes);
        memset(read_buffer_, 0, max_msg);

        std::cout << "msg: " << msg << std::endl;

        if (msg.find("startserver") == 0) create_gameserver_instance(msg);
    }

```

```

//else if (msg.find("portok") == 0) stop();
else
    std::cerr << "invalid msg:" << msg << std::endl;

}

void create_gameserver_instance(const std::string & msg)
{
    std::cout << "Creating child process... " << std::endl;
    if (NUMBER_OF_GAMESERVERS_RUNNING <
MAX_NUMBER_OF_GAMESERVERS)
    {
        int tempGamePort = INITIAL_GAME_PORT;
        while (true)
        {
            if (std::find(VPORTS_IN_USE.begin(),
VPORTS_IN_USE.end(), tempGamePort) == VPORTS_IN_USE.end())
            {
                break;
            }
            ++tempGamePort;
            std::cout << "iterated port: " << tempGamePort <<
std::endl;
        }
        std::cout << "unbusy port: " << tempGamePort << std::endl;

        std::string textPort = "-port=" + std::to_string(tempGamePort);
        //-----
        -----Game server parameters
        GameServerInstance::gsiptr inst =
GameServerInstance::pstart("GameServer.exe", std::vector<std::string>{"-log",
textPort}, tempGamePort);
        // boost::process::child child{
boost::process::search_path("DS.exe"), std::vector<std::string>{"-map", "-
port=1231"}, ProcessFinishHandler{tempGamePort}, service };
        //ProcessFinishHandler{tempGamePort},
        if (inst->isRunning())
        {
            std::cout << "Instance of GameServer.exe was successfully
crated with port " << tempGamePort << std::endl;
            do_write("port " + std::to_string(tempGamePort) + "\n");
        }
        else
        {

```

```

        std::cout << "child IS NOT running()" << std::endl;
        do_write("maxlimit\n");
    }
}
else
{
    std::cout << "NUMBER_OF_GAMESERVERS_RUNNING >=
MAX_NUMBER_OF_GAMESERVERS" << std::endl;
    do_write("maxlimit\n");
}
}

void on_last_write(const error_code & err, size_t bytes)
{
    std::cout << "last write operation, error" << err.message() << std::endl;
}

void on_write(const error_code & err, size_t bytes)
{
    memset(write_buffer_, 0, max_msg);
    std::cout << "write complete, error" << err.message() << std::endl;
    if (!isReading)
        do_read();
}

void do_read()
{
    isReading = true;
    std::cout << "reading..." << std::endl;
    async_read(sock_, buffer(read_buffer_),
        MEM_FN2(read_complete, _1, _2), MEM_FN2(on_read, _1,
_2));
    //post_check_ping();
}

void do_write(const std::string & msg)
{
    if (!started()) return;
    std::copy(msg.begin(), msg.end(), write_buffer_);
    std::cout << "writing:" << msg << std::endl;
    if (msg.find("port") == 0)
    {
        std::cout << "shas budu slat port: " << msg << std::endl;
        sock_.async_write_some(buffer(write_buffer_, msg.size()),
MEM_FN2(on_last_write, _1, _2));
    }
}

```

```

        }
        else
        {

                sock_.async_write_some(buffer(write_buffer_, msg.size()),
MEM_FN2(on_write, _1, _2));
        }

}
size_t read_complete(const boost::system::error_code & err, size_t bytes)
{
    if (err)
    {
        std::cout << "read_complete ERROR: " << err.message() <<
std::endl;
        return 0;
    }
    bool found = std::find(read_buffer_, read_buffer_ + bytes, '\n') <
read_buffer_ + bytes;
    std::cout << "read_buffer: " << read_buffer_ << std::endl;
    std::cout << "read_complete, found:" << found << std::endl;
    // we read one-by-one until we get to enter, no buffering
    return found ? 0 : 1;
}
private:
    ip::tcp::socket sock_;
    enum { max_msg = 1024 };
    char read_buffer_[max_msg];
    char write_buffer_[max_msg];
    bool started_;
    bool isReading;

};

ip::tcp::acceptor acceptor(service, ip::tcp::endpoint(ip::tcp::v4(), 8001));

void handle_accept(talk_to_client::ptr client, const boost::system::error_code & err)
{
    client->start();
    talk_to_client::ptr new_client = talk_to_client::new_();
    acceptor.async_accept(new_client->sock(), boost::bind(handle_accept,
new_client, _1));
}

```

```

bool RegisterServerInDB(const std::string &strIp, const std::string &strReg, const
int &newPort)
{
    //extern DataBase &db;
    DataBase &db = DataBase::getInstanceByHost();
    db.ExecCommand("SELECT \"DServer_id\" FROM \"DServerlist\" Where
\"DServer_ipv4\"=\"" + strIp + "\"");
    if (PQresultStatus(db.getRes()) == PGRES_TUPLES_OK)
    {
        if (PQntuples(db.getRes())>0)
        {
            db.ClearRes();
            std::cout << "Server with IP " << strIp << " already running!" <<
std::endl;
            return false;
        }
    }
    db.ClearRes();

    int regID = 0;
    db.ExecCommand("SELECT \"DRegion_id\" FROM \"DRegionlist\" Where
\"DRegion_name\"=\"" + strReg + "\"");
    if (PQresultStatus(db.getRes()) == PGRES_TUPLES_OK)
    {
        std::stringstream ss(PQgetvalue(db.getRes(), 0, 0));
        ss >> regID;
    }
    else
    {
        std::cout << "bad region id request result" << std::endl;
        db.ClearRes();
        return false;
    }
    db.ClearRes();

    db.ExecCommand("INSERT INTO \"DServerlist\" (\"DServer_ipv4\",
\"DServer_region\", \"DServer_freeslots\", \"DServer_port\") VALUES(\"" + strIp +
", \"" + std::to_string(regID) +
std::to_string(MAX_NUMBER_OF_GAMESEVERERS) +
std::to_string(newPort) + "\")");
    if (PQresultStatus(db.getRes()) == PGRES_COMMAND_OK)
    {
        db.ClearRes();
    }
}

```

```

    }
    else
    {
        db.ClearRes();
        std::cout << "bad ip registration request result" << std::endl;
        return false;
    }

    int tempServerID = 0;
    db.ExecCommand("SELECT \"DServer_id\" FROM \"DServerlist\" Where
\"DServer_ipv4\"    =\"\"    +    strIp    +    \"\"    and    \"DServer_port\"    =
\""+std::to_string(newPort)+"\"");
    if (PQresultStatus(db.getRes()) == PGRES_TUPLES_OK)
    {
        std::stringstream ss(PQgetvalue(db.getRes(), 0, 0));
        ss >> tempServerID;
    }
    else
    {
        std::cout << "bad server id request result" << std::endl;
        db.ClearRes();
        return false;
    }
    db.ClearRes();
    if (tempServerID>0)
    {
        db.setServerID(tempServerID);
    }
    else
    {
        std::cout << "tempServerID<=0" << std::endl;
        return false;
    }
    VPORTS_IN_USE.push_back(newPort);
    return true;
}

```

ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»

Програмне забезпечення для підбору гравців у комп'ютерних іграх

Олег СЕПЕТИЙ

ст. групи КНТ-229

керівник: Євгеній ГОФМАН

к.т.н., доцент НУ «Запорізька політехніка»

Рисунок В.1 – Слайд 1

Об'єкт, предмет та мета роботи

Об'єкт дослідження: процес розробки
комп'ютерних ігор.

Предмет дослідження: програмні засоби для
підбору гравців у комп'ютерних іграх.

Мета роботи: розробка програмного
забезпечення для підбору гравців у
комп'ютерних іграх.

Рисунок В.2 – Слайд 2

Задачі роботи

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для підбору гравців у комп'ютерних іграх;
- здійснити проектування програмного забезпечення для підбору гравців у комп'ютерних іграх;
- створити програмне забезпечення для підбору гравців у комп'ютерних іграх;
- виконати тестування розробленого програмного забезпечення для підбору гравців у комп'ютерних іграх.

3

Рисунок В.3 – Слайд 3

Порівняння програмного забезпечення для підбору гравців у комп'ютерних іграх

Критерій порівняння	Playfab	Amazon GameLift	Game Sparks	Photon Engine	Nakama
Підтримка декількох режимів матчмейкінгу	–	+	–	+	+
Легкість інтеграції у проект	+–	+	+	+	+
Можливість задавання параметрів ігрового серверу при розгортанні	–	+	–	+	+
Низька вартість підтримки	+	+–	+–	+	++
Служба бекенду	+	–	+	–	–
Модель оплати	залежить від використання	залежить від використання	підписка	залежить від використання	вільна

4

Рисунок В.4 – Слайд 4

Порівняння мов програмування

Критерій порівняння мов програмування	Мова програмування		
	C++	C#	Java
Ефективність та зручність при створенні програмного забезпечення підтримки комп'ютерних ігор	+	–	–
Швидкість виконання коду	+	+–	+–
Багатопарадигмальність	+	+–	+
Кросплатформеність розробки	+	+–	+
Зручність використання	+	+–	+–

5

Рисунок В.5 – Слайд 5

Обґрунтування вибору мережевої бібліотеки

Бібліотека	Підтримувана мова програмування	Практичні застосування	Кросплатформеність	Розповсюдженість
Boost.Asio	C++	Мережеві застосунки	+	+
Libevent	C	Мережеві застосунки	+	+
ACE	C++	Розподілені застосунки	+	–
POCO	C++	Мережеві застосунки	–	–
libuv	C	Мережеві застосунки	+	+–

6

Рисунок В.6 – Слайд 6

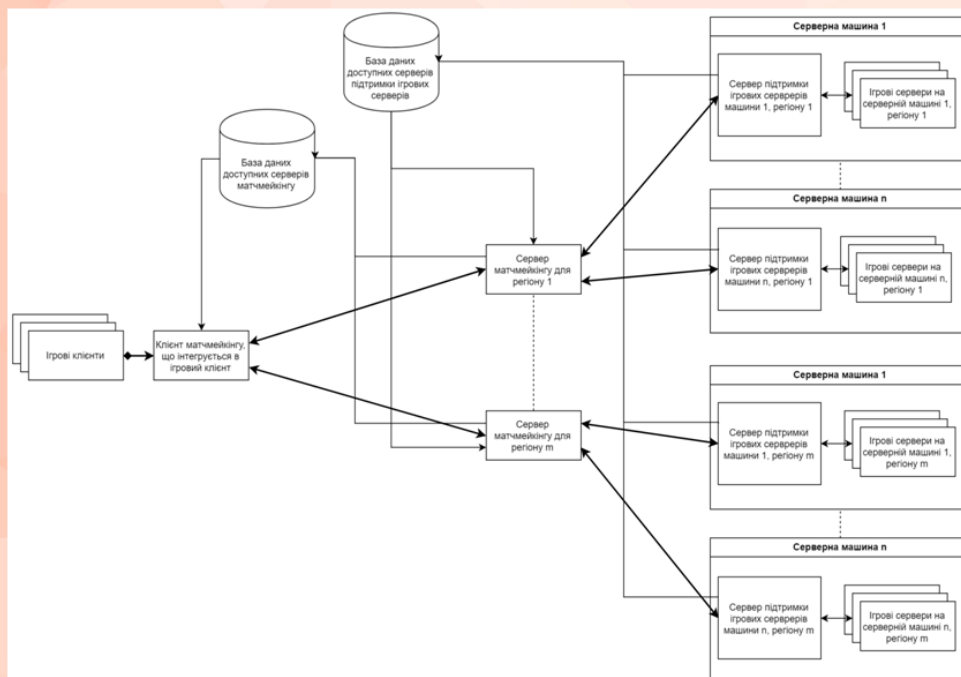
Порівняння середовищ розробки

Критерій порівняння середовищ розробки	Середовище розробки		
	Visual Studio Code	Sublime Text	PyCharm
Безкоштовність	+	+	—
Візуальний редактор коду	+	—	+—
Зручність використання	+	+—	+—
Підтримка рефакторингу	+	—	+
Підтримка хмарних сервісів	+	—	—
Відкритість ПЗ	+	—	—

7

Рисунок В.7 – Слайд 7

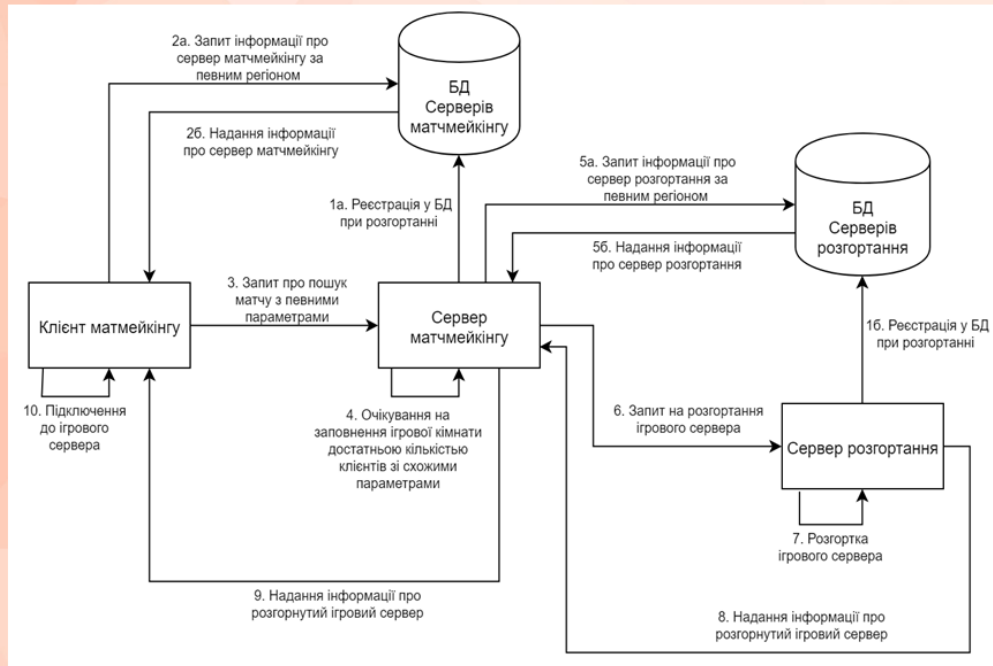
Структура програмного забезпечення підбору гравців у комп'ютерних іграх



8

Рисунок В.8 – Слайд 8

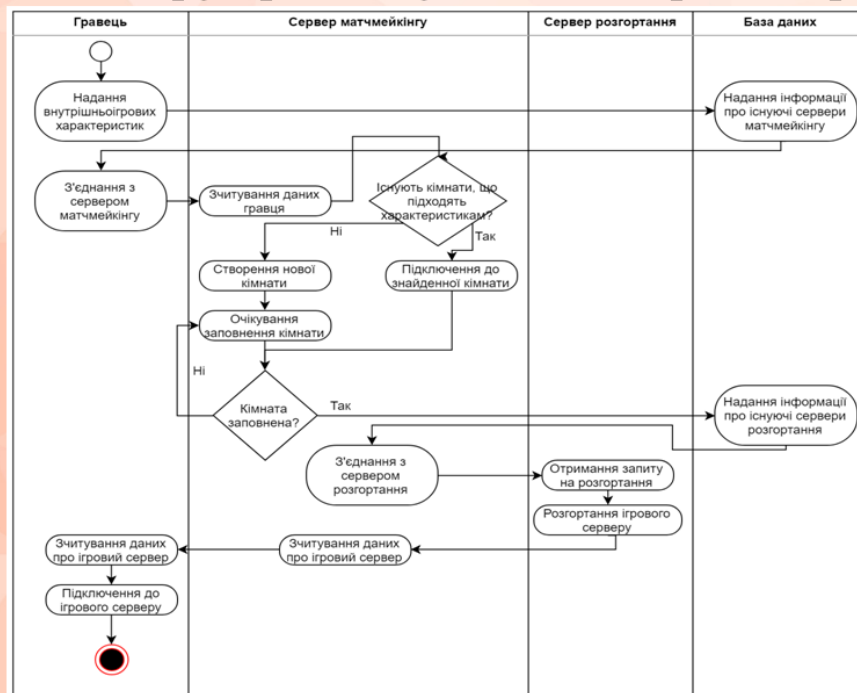
Схема взаємодії модулів програмного забезпечення



9

Рисунок В.9 – Слайд 9

Діаграма діяльності програмного забезпечення для підбору гравців у комп'ютерних іграх



10

Рисунок В.10 – Слайд 10

Робота з програмою

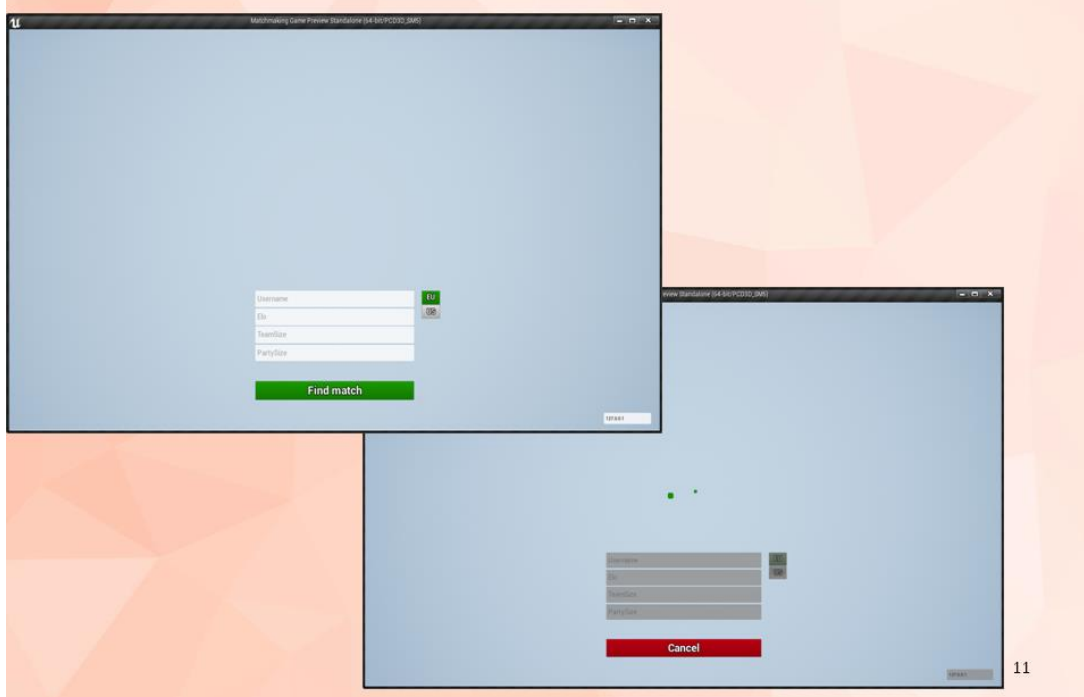


Рисунок В.11 – Слайд 11

Робота з програмою. Приклад інформаційної дошки



12

Рисунок В.12 – Слайд 12

Висновки

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки комп'ютерних ігор.

Визначено, що у наш час широко використовуються комп'ютерні ігри, що являють собою ігри, керовані комп'ютером, де гравці з метою розваги взаємодіють з об'єктами або іншими гравцями, що відображаються на екрані. Відзначено, що для формування команд у відеоіграх, у т.ч. у кіберспорті, де необхідною є участь декількох гравців в одній команді, використовуються методи матчмейкінгу, що являє собою процес формування команд для комп'ютерної гри.

Для реалізації програмного забезпечення підбору гравців у комп'ютерних іграх обрано мову програмування C++, мережеву бібліотеку для забезпечення функцій асинхронного введення-виведення Boost.Asio, середовище розробки Visual Studio.

13

Рисунок В.13 – Слайд 13

Висновки

Розроблено програмне забезпечення для підбору гравців у комп'ютерних іграх. Запропоновано структуру підбору гравців у комп'ютерних іграх. Визначено, що програмне забезпечення підбору гравців у комп'ютерних іграх складається з таких модулів: клієнт матчмейкінгу, що інтегрується в ігровий клієнт, сервер матчмейкінгу, сервер підтримки ігрових серверів (сервер розгортання), база даних доступних серверів матчмейкінгу та база даних доступних серверів підтримки ігрових серверів.

Розроблено схему взаємодії модулів програмного забезпечення для підбору гравців у комп'ютерних іграх між собою. Процеси, що відбуваються під час функціонування програмного забезпечення для підбору гравців у комп'ютерних іграх у вигляді, подано у вигляді діаграм діяльності та послідовності у нотації UML. Наведено основні рішення щодо реалізації програмного забезпечення для підбору гравців у комп'ютерних іграх. Подано структуру бази даних серверів матчмейкінгу та бази даних серверів розгортання.

Виконано тестування розробленого програмного забезпечення для підбору гравців у комп'ютерних іграх. За результатами тестування програмного забезпечення для підбору гравців у комп'ютерних іграх виявлено, що програма відповідає технічному завданню, функціонує коректно та дозволяє виконувати підбір гравців у комп'ютерних іграх.

14

Рисунок В.14 – Слайд 14