МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

В.о. завідувача кафедри інтелектуальних інформаційних систем, канд. техн. наук, доцент

_____ Є. В. Сіденко

«____»_____2023 р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

ІНТЕЛЕКТУАЛЬНІ КОМПОНЕНТИ СИСТЕМИ-ЕМУЛЯТОРА ДЛЯ КОМП'ЮТЕРНОЇ ГРИ В ГОЛЬФ

Спеціальність 124 «Системний аналіз»

124 – МКР – 607.221507001

Виконав студент 6-го курсу, групи 607
_____*Жигалкін І. О.*
«__» _____ 2023 р.
Керівник: *канд. техн. наук, доцент*
_____*І. О. Калініна*
«__» _____ 2023 р.

Миколаїв – 2023

Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Освітньо-кваліфікаційний рівень **магістр**

Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

Спеціальність **124 «Системний аналіз»**
(шифр і назва)

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри інтелектуальних інформаційних систем, канд. техн. наук, доц.

_____ Є. В. Сіденко

«____» _____ 202_ р.

ЗАВДАННЯ

на магістерську кваліфікаційну роботу

Жигалкіну Івану Олеговичу

1. Тема магістерської кваліфікаційної роботи «Інтелектуальні компоненти системи-емулятора для комп'ютерної гри в гольф».
Керівник роботи Калініна Ірина Олександрівна, канд. техн. наук, доцент.
Затв. наказом Ректора ЧНУ ім. Петра Могили від «03» листопада 2022 р. № 197.

2. Строк подання студентом роботи 17 лютого 2023 р.

3. Вхідні (початкові) дані до роботи: параметри впливу на траєкторію польоту та аналіз даних, що впливають на розподіл.

Очікуваний результат: розробка інтелектуальної системи аналізу й рекомендацій тенденцій користувача системи-емулятору гри у гольф.

4. Зміст пояснювальної записки (перелік питань, які потрібно розглянути):

- аналіз сучасного стану використання інтелектуальних інформаційних систем в іграх-емуляторах гольфу;

- розробка компонентів аналізу тенденцій удару гравця;
- розробка рекомендаційної системи для покращення патерну удару гравця;
- розробка рекомендаційної системи для надання рекомендацій щодо використання певної ключки при заданих характеристиках;
- тестування рекомендаційної системи та проведення аналізу результатів удару.

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини: Охорона праці та безпека у надзвичайних ситуаціях на робочому місці.

7. Консультанти:

Розділ	**Прізвище, ініціали та посада консультанта**	**Підпис**
Спеціальна частина з охорони праці	Л. І. Григор'єва	
Методична частина	І. В. Кулаковська	

Керівник роботи __канд. техн. наук, доцент, Калініна І. О._____

_____

(підпис)

Завдання прийнято до виконання ___Жигалкін І. О._____

_____

(підпис)

Дата видачі завдання «__06__» _____листопада_____ 2022 р.

КАЛЕНДАРНИЙ ПЛАН
Виконання магістерської кваліфікаційної роботи

Тема: Інтелектуальні компоненти системи-емулятора для комп'ютерної гри в гольф

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників МКР та визначення керівника	01.09.2022	20.10.2022	Виконано
2	Отримання завдання на виконання МКР	21.10.2022	10.11.2022	Виконано
3	Складання календарного плану роботи на весь період виконання МКР	11.11.2022	15.11.2022	Виконано
4	Отримання завдання на переддипломну практику	05.12.2022	06.12.2022	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до МКР	06.12.2022	20.12.2022	Виконано
6	Розробка звіту з переддипломної практики	20.12.2022	25.12.2022	Виконано
7	Виконання МКР: Інтелектуальні компоненти системи-емулятора для комп'ютерної гри в гольф	13.01.2023	25.01.2023	Виконано
8	Розробка спеціальної частини з охорони праці та методичної частини	26.01.2023	02.02.2023	Виконано
9	Попередній захист МКР на засіданні комісії кафедри	03.02.2023	03.02.2023	Виконано
10	Доробка та остаточне оформлення МКР	04.02.2023	06.02.2023	Виконано
11	Подання МКР рецензенту	09.02.2023	10.02.2023	Виконано
12	Рецензування МКР	11.02.2023	12.02.2023	Виконано
13	Подання МКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	16.02.2023	17.02.2023	Виконано
14	Захист МКР перед екзаменаційною комісією (ЕК)	23.02.2023	23.02.2023	Виконано

Розробив студент Жигалкін І.О. _____
(прізвище та ініціали) (*підпис*)

Керівник роботи канд. техн. наук, доцент, Калініна І. О. _____
(*наук. ступінь, вчене звання, прізвище та ініціали)* *(підпис)*

«____» _____ 202_р.

АНОТАЦІЯ

до магістерської кваліфікаційної роботи

студента групи 607 ЧНУ ім. Петра Могили

Жигалкіна Івана Олеговича

на тему: **«ІНТЕЛЕКТУАЛЬНІ КОМПОНЕНТИ СИСТЕМИ-ЕМУЛЯТОРА ДЛЯ КОМП'ЮТЕРНОЇ ГРИ В ГОЛЬФ»**

Актуальність роботи обумовлена тим що на сьогоднішній день розвиток наукових та технічних досягнень дозволяє розширити задачі емуляторів спорту та використовувати інтелектуальні інформаційні системи для зчитування та аналізу даних користувача й створювати інструменти для аналізу показників гравця, надання рекомендацій щодо його гри тощо.

Об'єктом дослідження є процес збору й обробки даних за допомогою інтелектуальних компонентів що застосовуються в іграх-емуляторах гольфу.

Предметом дослідження є методи збору та аналізу інформації компонентами системи гри-емулятора у гольф та їх тестування за допомогою методів математичної статистики та експертних оцінок.

В результаті виконання роботи було створено компоненти системи емулятору гри у гольф та проаналізовано вили взаємодії основних

Метою даної роботи є підвищення якості комп'ютерної гри в гольф за рахунок системного використання інтелектуальних компонентів. Метою роботи є дослідження можливості та ефективність використання інтелектуальних інформаційних систем – компонентів у комп'ютерній грі-емуляторі гольфу, розробка архітектури проєкту для створення уніфікованих бібліотек API яке може використовуватись при розробці емуляторів спорту. В роботі ставиться за мету розробити інтелектуальні системи-інструменти для аналізу даних гравця, надання рекомендацій й інтегрування таких систем з загальною архітектурою проєкту, розробка клієнт-серверної архітектури проєкту для збору й аналізу даних, що надходять від користувача та визначення можливості використання такої інформації у якості навчального матеріалу для рекомендаційної системи.

Було розроблено систему зчитування даних удару гравця за допомогою різних типів контролерів.

Робота викладена на 139 листах, містить 5 розділів. Робота має наступні розділи:

- вивчення сучасного стану застосування інтелектуальних інформаційних систем у додатках емуляторах гольфу.В першому розділі проведено аналіз сучасних ігор емуляторів гольфу та компонентів систем, які в них використовуються;
- впровадження систем інтелектуального аналізу удару гравця у грі емуляторі гольфу. В другому розділі були проаналізовані й розроблені компоненти систем – інструменти, для проведення аналізу дій гравця та висунення базових рекомендацій щодо техніки гри користувача;
- розроблення архітектури проєкту системи-емулятора для комп'ютерної гри у гольф. Третій розділ надає аналіз щодо архітектури проєкту та використання залежностей та зв'язаності проєкту, проаналізовано можливість створення гнучкої структури компонентів гри емулятора у гольф;
- у четвертому розділі визначена розробка лабораторної роботи з дисципліни «Модулювання ризиків у соціально-економічних системах». Лабораторна робота надає теоретичні й практичні навички використання моделі Леонтьєва у багатогалузевій економіці. Було сформоване завдання для студентів з варіантами його виконання;
- охорона праці та безпеки у надзвичайних ситуаціях П'ятий розділ характеризує правові засади щодо охорони праці та убезпечення працівників компанії. Було розглянуто приклад взаємодії кадрового персоналу з працівниками при роботі в офісі та онлайн.

В роботі містяться 54 рисунки, 17 таблиць, 6 додатків, 43 джерела.

Ключові слова: інтелектуальні компоненти, структура системи-емулятора, система рекомендацій, архітектура проєкту.

ABSTRACT

to the master's qualification work by the student of the group 607
of Petro Mohyla Black Sea National University

Zhyhalkin Ivan

"INTELLIGENT COMPONENTS OF THE SYSTEM-EMULATOR FOR THE COMPUTER GOLF GAME"

The **relevance** of the work is due to the fact that today's development of scientific and technical achievements allows expanding the tasks of sports emulators and using intelligent information systems for reading and analyzing user data and creating tools for analyzing the player's performance, providing recommendations for his game, etc.

The **object** of research is the process of data collection and processing with the help of intelligent components used in golf simulator games.

The **subject** of the research is the methods of collecting and analyzing information by the components of the golf emulator game system and their testing using the methods of mathematical statistics and expert evaluations.

As a result of the work, the components of the golf game emulator system were created and the interactions of the main ones were analyzed.

The **purpose** of this work is to improve the quality of the computer golf game due to the systematic use of intelligent components. The purpose of the work is to investigate the possibility and effectiveness of using intelligent information systems - components in a computer game-golf emulator, to develop project architecture for the creation of unified API libraries that can be used in the development of sports emulators. The thesis aims to develop intelligent systems-tools for analyzing player data, providing recommendations and integrating such systems with the overall project architecture, developing a client-server architecture of the project for collecting and analyzing data received from the user and determining the possibility of using such information in the quality of educational material for the recommender system.

As a result of the work, the components of the golf game emulator system were created and the interaction of the main logic components with the main game module was analyzed.

A system was developed to read the player's shot data using different types of controllers. The thesis is laid out on 139 sheets, contains 5 sections. The work has the following sections:

- study of the current state of application of intelligent information systems in golf emulator applications. In the first section, an analysis of modern golf emulator games and system components used in them is carried out;
- implementation of systems of intellectual analysis of the player's shot in the golf emulator game. In the second section, system components were analyzed and developed - tools for analyzing the player's actions and making basic recommendations regarding the user's game technique;
- development of the project architecture of the emulator system for the computer game of golf. The third section provides an analysis of the project architecture and the use of project dependencies and connectivity, the possibility of creating a flexible structure of golf emulator game components is analyzed;
- the fourth section is development of laboratory work on the discipline "Modulation of risks in socio-economic systems". Laboratory work provides theoretical and practical skills in using the Leontiev model in a multi-sector economy. A task was created for students with options for its implementation;
- occupational health and safety in emergency situations The fifth section describes the legal principles regarding occupational health and safety of the company's employees. An example of the interaction of HR personnel with employees during work in the office and online was considered.

The work contains 45 images, 17 tables, 7 appendices, and 43 sources.

Keywords: intelligent components, emulator system structure, recommendation system, project architecture.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – бази даних

ІІС – інформаційна інтелектуальна система

ІТ – інформаційні технології

МКР – магістерська кваліфікаційна робота

ПЗ – програмне забезпечення

СУБД – система управління базами даних

API – (Application Programming Interface) інтерфейс для взаємодії з частиною програмного забезпечення

AR – (Augmented reality) доповнена реальність

DI – Dependency Injection

HR – human resources specialist

IDE – (Independent development environment) середовище розробки для написання коду

IoC Inversion of Control

MVC – Model-View-Controller

OSHA – Occupational Safety and Health Administration

SNES – Super Nintendo Entertainment System

SO – (Scriptable object)

SOLID – single responsibility, open–closed, Liskov substitution, interface segregation и dependency inversion

UI/UX – (User interface/user experience) інтерфейс та досвід користувача

VR – (Virtual reality) віртуальна реальність

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему:

«ІНТЕЛЕКТУАЛЬНІ КОМПОНЕНТИ СИСТЕМИ-ЕМУЛЯТОРА ДЛЯ КОМП'ЮТЕРНОЇ ГРИ В ГОЛЬФ»

Спеціальність 124 «Системний аналіз»

124 – МКР – 607.221507001

Виконав студент 6-го курсу, групи 607
_____*Жигалкін І. О.*
«__» _____ 2023 р.

Керівник: *канд. техн. наук, доцент*
_____*І. О. Калініна*
«__» _____ 2023 р.

Миколаїв – 2023

ВСТУП

На сьогоднішній день розваги спорт й наука дуже переплетені. В кожній з цих сфер людського життя можна застосувати методи й підходи іншої сфери життя. Так й в ігрових додатках, за допомогою сучасних технологій рівень ігрових симуляторів можна перенести на новий рівень, наприклад за допомогою технологій AR (Augmented reality – доповнена реальність), або VR (Virtual reality – віртуальна реальність) можна створити новий рівень занурення у спортивні дисципліни, тренування, тощо, наприклад як для ігор емуляторів гольфу. Тобто, було припущено, що застосовуючи новітні технології можна створити ігрові додатки які будуть використовуватись не лише як гра, але як засіб тренування та відточування майстерності в реальному світі, а отже коли особа має пряму взаємодію з навколишнім середовищем, то з'являється змога виміряти базові показники дій гравця, покращивши які особа може значно посилити свої навички та спортивні досягнення, відредагувати напрямок удару й підвисити дальність удару у іграх емуляторах, тощо. Така взаємодія буде розглянута на прикладі гри в гольф, у такому випадку гравець може покращити дальність удару завдяки покращеною кута удару, направлення та швидкості маху ключки та точки дотику м'яча до голівки ключки. Все це є вимірювальними параметрами.

Об'єктом дослідження є процес збору й обробки даних за допомогою інтелектуальних компонентів що застосовуються в іграх-емуляторах гольфу. **Предметом** дослідження є методи збору та аналізу інформації компонентами системи гри-емулятора у гольф та їх тестування за допомогою методів математичної статистики та експертних оцінок.

Дана робота є актуальною оскільки новітні технології вже дозволяють використовувати ІІС – інформаційна інтелектуальні системи для зчитування та вимірювання спортивних показників та використання її у тренуваннях, але ці засоби досі не використовуються на постійній основі, та існує невелика кількість

додатків, які використовують ІІС для тренування спортсменів. При цьому існує невелика кількість ігор-програм Launch monitor типу які можуть бути реалізовані як на стаціонарних комп’ютерах так і на мобільних пристроях.

Метою даної роботи є підвищення якості комп’ютерної гри в гольф за рахунок системного використання інтелектуальних компонентів. Мета даної МКР – дослідити можливості використання інтелектуальних компонентів при створенні гри-емулятора гольф, провести дослідження які типи таких компонентів можуть бути використані та безпосередньо як вони можуть застосовуватись для тренування гравців. Поставлено ціль проаналізувати можливість створення бібліотеки яка може застосовуватись як API – Application Programming Interface, тобто інтерфейси для взаємодії однієї частини програмного забезпечення з іншою, для швидкого інтегрування між різними додатками – симуляторами спортивних чи інших ігор. Предметом дослідження в даній роботі будуть компоненти інтелектуальної системи для аналізу ударів гравця ключкою для подальшого збору, систематизації, прогнозування та покращення в подальшому, за допомогою висунення рекомендацій щодо застосування спеціального інвентаря. Будуть створені інструменти для аналізу ударів, такі як теплові карти, проаналізована дисперсія ударів та здійснені додаткові розрахунки, для аналізу ударів.

Для виконання даної роботи будуть застосовані такі наукові **методи** як розрахунки – тобто знаходження необхідних даних виходячи із заданих показників, вимірювання – а саме зчитування базових даних про удари гравця за допомогою додаткового обладнання, буде проведений аналіз отриманих даних на підставі якого буде зроблений висновок про доцільність заміни обладнання. За допомогою синтезу отриманих даних буде зроблено висновок як гравець може покращити поточну гру та вдосконалити свої навички володіння ключкою.

При виконанні даної роботи були використані такі технології:

– мова програмування C# для програмування частини логіки проєкту та створення бібліотек ІІС, які будуть вбудовані в проєкт. Вибір такої мови

програмування обумовлений можливостями мови бути побудованим для любої платформи;

- Microsoft Visual Studio – IDE (Independent development environment) – середовище розробки для написання коду;
- ігровий рушій Unity3d – для створення візуальної частини проєкту, для промальовки ігрового поля та візуалізації інтерфейсів, для базового розрахунку фізичних явищ. Даний ігровий рушій має можливості побудови проєкту для систем під керуванням Windows, Mac, Android, iOS;
- бібліотека FlightEquationsLogic для розрахунку траєкторій польоту м’яча;
- обладнання для зчитування даних про удар гравця – Launch monitor tool;
- Launch monitor simulator – програмний пакет для емуляції удару за допомогою обладнання для налагодження проєкту та тестування;
- NUnit – програмний пакет для впровадження тестування проєкту в режимі реального часу та в режимі середовища;
- PDF Renderer – для стовщення звітів ігрових сесій та надання гравця даних ігрових сесій;
- GPU Instancer – засіб для оптимізації рендерінгу;
- Profiler – програмний компонент для оптимізації виконання проєкту та збору додаткових даних під час виконання проєкту;
- Git – система контролю версій;
- SourceTree – візуальний інтерфейс для системи контролю версій проєкту Git.

Отже в результаті виконання МКР буде визначено як можна використовувати інтелектуальні інформаційні системи в іграх гольф для отримання додаткових даних про навички гравця та інформацію щодо покращення результатів гравця за допомогою різних засобів, які будуть інтегровані в проєкт.

1 ВИВЧЕННЯ СУЧАСНОГО СТАНУ ЗАСТОСУВАННЯ ІНТЕЛЕКТУАЛЬНИХ ІНФОРМАЦІЙНИХ СИСТЕМ У ДОДАТКАХ ЕМУЛЯТОРАХ ГОЛЬФУ

1.1 Наявність ігрових додатків емуляторів гольфу на ринку електронних магазинів

1.1.1 Аналіз наявності інтелектуальних систем в додатках емуляторах комп’ютерної гри у гольф

Наразі існує велика кількість ігрових додатків у всіх доступних магазинах додатків, таких як AppStore, Google Play, Steam. Було проаналізовано пошукову видачу за ключовим словом «симулятор гольфу». Для аналізу необхідно детально розібрати деякі представлені додатки, визначити їх основний функціонал та проаналізувати які інтелектуальні системи в них використовуються та які системи можуть бути вбудовані в подальшому для роботи таких додатків.

Для прикладу було взято пошукову видачу магазину додатків Google play та отримано наступні дані:

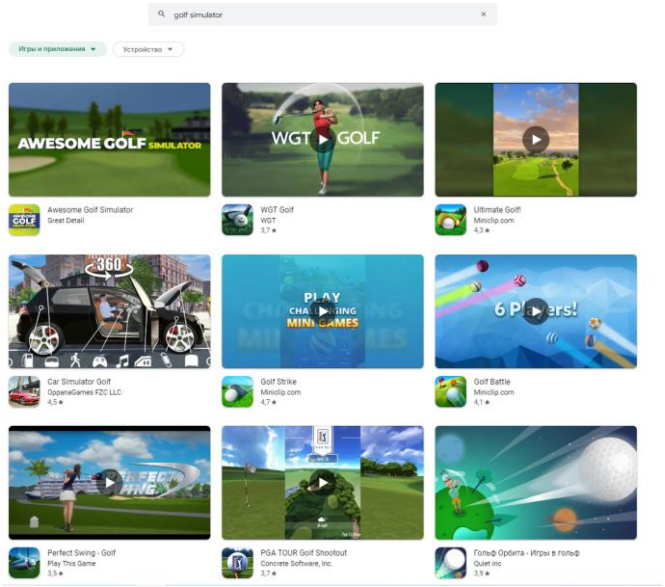

Рисунок 1.1 – Пошукова видача Google Play

Було зроблено пошукову вибірку для магазину додатків Steam. В ньому представлені більші й складніші проєкти з використанням новітніх технологій, таких як VR та AR.

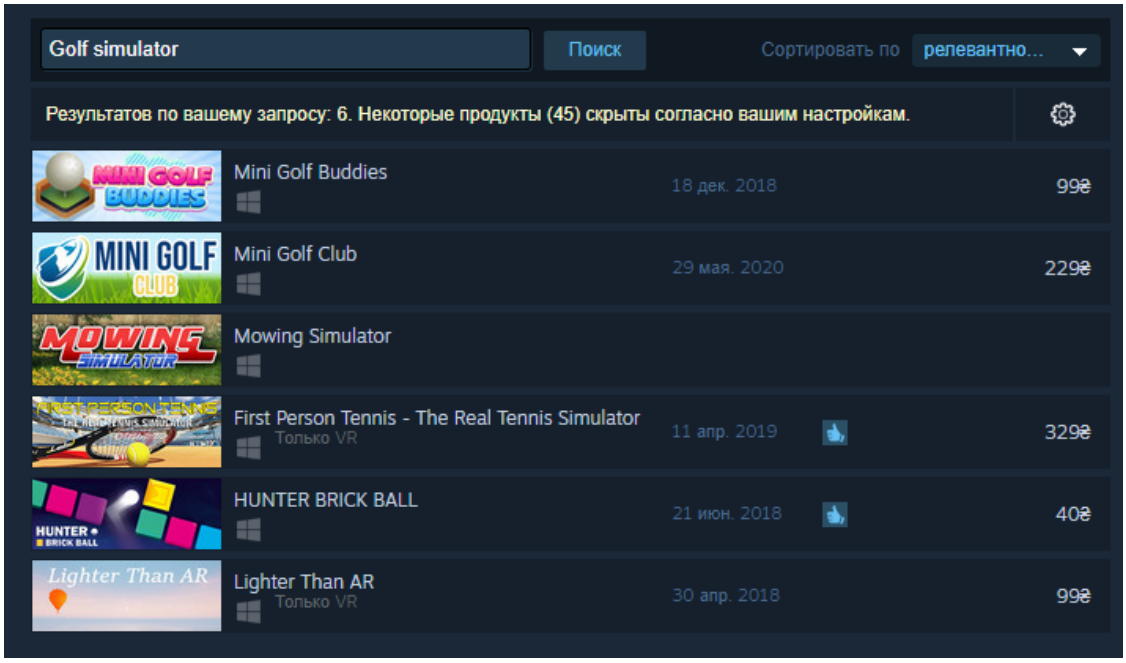

Рисунок 1.2 – Пошукова видача Steam

Проаналізувавши деякі з додатків з пошукової видачі було зроблено висновок, що основний функціонал таких додатків пов’язаний лише з побудовою логіки ігри та побудови траєкторій ударів гравця. Деякі додатки використовують додаткове обладнання для віртуальної реальності. Так були представлені та визначені наступні шари функціоналу у симуляторах гольфу:

– побудова моделі даних для використання її для зберігання, розрахунку та передачі даних;
– модуль фізики, який відповідає за траєкторію польоту м’яча та відскоку, що дає змогу визначити загальну дальність польоту та значення дальності у повітрі (carry – дальність польоту м’яча до першого торкання землі), азимут та деякі інші параметри;

– модуль взаємодій з інтерфейсом. Такий модуль надає змогу гравцю візуально визначити азимут та кут удару;

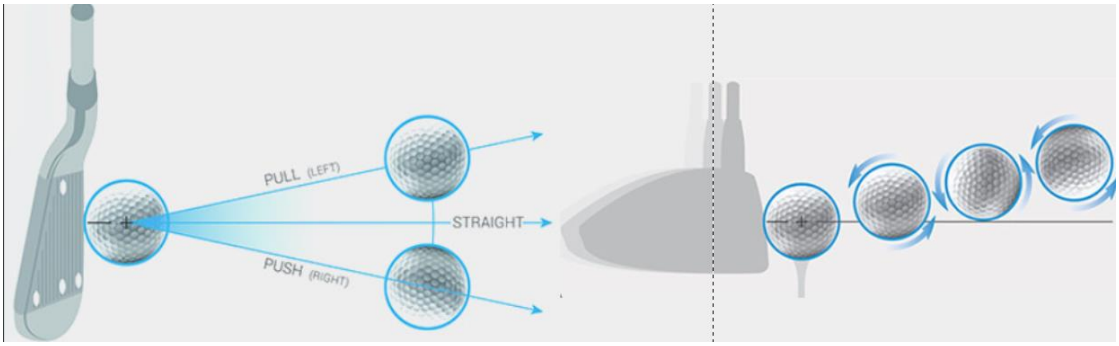

Рисунок 1.3 – Азимут (Azimuth) та кут удару (Angle of attack)

– в деяких додатках також присутня взаємодія з мережею для створення мережевої гри та зберігання даних на сервері для ігор емуляторів гольфу;
– модулі статистики які систематизують інформацію про ігрові сесії гравця, іноді доступний аналіз ударів та надається можливість подивитись траєкторію вже зроблених ударів;
– для створення опонентів використовується мережа через мачмейкінг, або в окремих випадках використовуються боти які не включають в себе системи навчання чи аналізу ударів гравця. Зазвичай вони керуються примітивною логікою з можливість ускладнення за допомогою ступенів влучності та дальності ударів.

Отже, в основній масі ігри – емулятори гольфу не включають в себе дуже багато складної логіки й націлені лише на розважальний сегмент. У всіх проаналізованих іграх інтелектуальні інформаційні системи націлені лише на збір даних щодо окремої сесії гравця та для розрахунку фізичної моделі взаємодії м’яча та ключки. Фізичні моделі ключок задані лише як набір даних з різними параметрами, та не як не відповідає реальним показникам ключок для гольфу.

1.1.2 Визначення сфери застосуванні інтелектуальних систем в подібних емуляторах гри у гольф

Зазначені вище додатки не використовують можливості сучасних технологій повною мірою, оскільки вони не проводять важких розрахунків та не вбудовують додаткові системи аналізу та діагностики гри гравця. Такий підхід переводить спорт у площину вдалих кліків та випадковості, що не відповідає дійсності. Також при виконані ударів гравець не отримує інформацію від додатку про прорахунок параметрів польоту м’яча, таких як загальне, заднє й бокове обертання, що прямо впливає на якісні характеристики побудови траєкторії удару.

У подібних ігрових додатках можна виділити групу інтелектуальних систем які можуть бути застосовані для створення більш якісної основи для впровадження додаткових ігрових механік, які можуть покращити розуміння гравця про гру гольф та відчути більше розуміння контролю над грою.

Згідно до стандарту ISO 2382-1, що затверджене Наказом від 19.12.2017 № 429 Про прийняття національних нормативних документів, гармонізованих з європейськими та міжнародними нормативними документами, та скасування чинності національних нормативних документів [1], інформаційна система – це така система обробки інформації, яка працює у зв’язку з організаційними ресурсами, такими як робітники, техніка або інші ресурси, які забезпечують та розподіляють інформацію, тобто – це сукупність апаратно-програмних та організаційних засобів для обробки інформації, яка може бути збережена для забезпечення інформаційних чи функціональних потреб кінцевого користувача, або поставленої задачі [2].

Такі інтелектуальні інформаційні системи можуть бути поділені на декілька типів для гри у гольф. Так за сферою задач які виконує така система вони можуть бути поділені на системи які впливають на геймплей проєкту, тобто на сам

ігровий процес, системи збору та систематизації даних, системи рекомендацій та навчальні системи.

Проаналізувавши представлені проєкти також інтелектуальні інформаційні системи можуть бути розподілені на локальні, логіка яких виконується у самому додатку, та на клієнт серверні, тобто такі системи де інформація зберігається в базах даних та обробляється на стороні сервера [3].

Для подібних задач краще всього підійде автоматичні системи, які не потребують додаткового втручання персоналу, адже за логікою самих додатків це майже не можливо, лише можливо у тих випадках коли додаток застосовує клієнт серверну архітектуру, але ручне втручання у результати даних буде викликати велику затримку, що негативно вплине на конверсію додатків [4].

Значна частина можливих впроваджених інформаційні системи буде використовуватись для обробки даних. Застосування пошукових систем у даному випадку є недоцільним.

Визначено охоплення дії таких систем, було припущено, що значна кількість додатків застосовується в персональних цілях, при цьому хороша сфера застосування їх буде навчання, тобто особа має змогу отримати для себе якусь корисну інформацію для покращення своїх навичок гри у гольф.

Виходячи з опису того які системи можуть бути застосовані за різними типами застосування та дії було класифіковано наступні задачі, які можуть виконувати ІС в ігрових додатках емуляторах гольфу.

Основна задача, яка може виконуватись – це навчальна, тобто це можна розуміти як використання даного програмного забезпечення для поглиблення своїх знать та розуміння гри гольф. Системи навчання діагностують помилки при вивченні якої-небудь діяльності, наприклад здійснення ударів та вибору кутів удару у іграх емуляторах гольфу, та за допомогою алгоритмів, які виконуються на пристрої підказують правильні дані [5], використовуючи які гравець може покращити свої результати. Вони акумулюють знання про гіпотетичного «учня» і

його характерні помилки, потім у роботі вони здатні діагностувати слабкості в знаннях учнів і знаходити відповідні засоби для їхньої ліквідації. Окрім цього, такими додатками можуть бути розроблені плани навчання за допомогою подібних дій.

Задача інтерпретації даних потрібна для подальшої розробки експертних систем, або створення БД – бази даних, для визначення змісту даних, результати якого мають бути погодженими і коректними. Зазвичай передбачається багатоваріантний аналіз даних [6].

Діагностичні задачі несуть під собою співвідношення об’єкта з деяким класом об’єктів і виявлення несправності в деякій системі. Під об’єктом розуміються удари гравця, які можна порівнювати з базою даних про удари інших гравців у різних умовах. Звідси випливає поняття несправедливості даних, тобто відхилення даних від нормальних показників. Подібне визначення дозволяє використовувати такі системи для різних сфер життя, як здоров’я, так і до аналізу обладнання, або техніки гри у гольф, або справності спортивного обладнання, яке може давати додатковий люфт при ударах. Але при цьому необхідно розуміти специфіку самої системи, та розуміти як коректно перевіряти валідність даних та показників.

Завдання моніторингу полягає у тому щоб на постійній основі інтерпретувати інформацію про ігрові сесії та удари гравця, які були виконані під час гри. Такий моніторинг може виконуватись як після сесії так і у реальному часі, кожен раз під час надходження нової порції даних, при цьому гравець може одразу отримати сповіщення про відхилення якогось показника від норми [7]. Наприклад одним з таких показників може бути кутове, або заднє обертання, що може призвести до відхилення азимуту, або викривленню траєкторії, що призведе до погіршення дальності й точності удару. Але при виконанні такого завдання бувають проблеми пов’язані з неможливістю системи врахувати усі дані необхідні для вирішення проблеми, наприклад бокове обертання може бути корисним коли

гравцю необхідно вдарити м’яч у ціль яка знаходиться у стороні, при цьому для нормального удару граничне значення обертання буде перевищене, а для удару суть якого здвинути тратторію у сторону ця величина буде допустимою. Для вирішення цієї задачі необхідно буде мати більшу кількість інформацію про різні види ударів для різних задач, а також предметну інформацію про тренувальний полігон, на якому проходить гра, з урахуванням показників про вітер та вологість повітря, адже такі дані впливають на траєкторію удару, якщо не беруться до уваги ідеальні умови.

Одним з важливіших завдань – є завдання прогнозування, навіть для додатків, ціллю яких є тільки розважальний контент у сфері гольфу, адже такі дії допоможуть зробити ігровий процес більш цікавий для користувача, так як додаток зможе пристосуватись до гри гравця. Прогнозування може передбачати наслідки деяких подій або явищ на підставі детального аналізу вхідних даних [8] про дії гравця. Прогнозуючі системи логічно виводять ймовірні наслідки із заданих ситуацій. У прогнозуючій системі зазвичай використовується параметрична динамічна модель, в якій значення параметрів «підганяються» під задану ситуацію. Висновки, що виводяться з цієї моделі, складають основу для прогнозів з ймовірними оцінками. Також ці дані можливо використовувати у побудові логіки ботів, які будуть підлаштовуватись під гру гравця й робити процес навчання та гри більш ефективним.

ІІС що ставлять під собою задачу планування мають знаходити план дій, які необхідно зробити гравцю, для поліпшення своєї техніки чи навчання гольфу для більш якісної й правильної гри. Тобто при виконанні таких завдань планування готується певний план дій, наприклад які техніки гри треба вивчити або застосовувати з поточним рівнем вмінь. В подібних ЕС – експертних системах мають використовуватись моделі поведінки реальних об’єктів з тим, аби логічно вивести наслідки планованої діяльності, яка може бути застосована гравцем.

Результати планування дій можуть носити рекомендаційний характер та корегуватись під час отримання системою додаткових даних про гру користувача.

При виконанні завдання підтримки прийняття рішень може бути застосована сукупність процедур для забезпечення прийняття рішень, на приклад про вибір ключки або м’яча, або може бути висунута інформація, що може спростити процес прийняття рішення, або полегшити його. Так може бути наданий ряд альтернатив, що були сформовані згідно з вхідними даними для більш легкого ухвалення рішень.

Отже у підсумку з вищевикладеного випливає що при застосуванні різних типів ІІС можна виконати завдання різних типів, у відповідності до призначення додатків, їх цільової аудиторії, задач та мети використання [9].

Було отримано наступну схему завдань які можуть бути виконані при застосуванні ІІС у іграх емуляторах гольфу:

Рисунок 1.4 – Схема завдань ІІС у емуляторі гри гольф

Також завдання на пряму можуть залежати від наявних контролерів управління та додаткового обладнання. Таке додаткове обладнання з’явилось ще у 1991 р. та випускалось для SNES – Super Nintendo Entertainment System, для відповідних ігрових консолей. Але у той час для зчитування даних гравця

використовувались прості акселерометри та механічні засоби для зчитування швидкості.

Рисунок 1.5 – Приклад контролеру для SNES

Наразі також використовуються більш сучасні контролери для ігрових консолей Nintendo switch, PlayStation VR, Xbox VR та монітори запуску які надають розробнику значно більше даних для обробки та впровадження в ігри емулятори гольфу. Вони використовують високоточні датчики й методи зняття й вимірювання дій гравця.

1.2 Огляд приладів та систем для збору інформації про удари гравця у іграх-емуляторах гольфу.

1.2.1 Аналіз існуючих моніторів запуску для емуляторів гольфу.

Однією з найбільш гнучких та професійних систем контролю дій гравця є монітори запуску. Отже монітор запуску – це є спеціальне електронне обладнання – це пристрій, ціллю якого є який відстеження характеристики польоту м’яча для гольфу, руху, позиції, траєкторії ключки, позиції гравця за допомогою різних датчиків та вимірювальних пристроїв. Пускові монітори здатні надавати

розрахункову відстань, кут старту та удару м’яча, місцезнаходження на стартовій площадці та багато інших даних про траєкторію та політ м’яча. Монітори запуску отримують дані після ударів та співвідносять дані різних датчиків та розраховують додаткові змінні щодо траєкторії удару. Ці розрахунки можуть займати від декількох секунд до декількох хвилин залежно від методів обробки даних.

Функціональна та програмна частина моніторів запуску може відрізнятись від цілей використання приладу та комплектності обладнання. Так існують найпростіші прилади Swing Caddie SC200 PLUS Launch Monitor – це портативний монітор який має можливість зчитувати дані замаху та швидкість ключки, при цьому надаючи базові дані про дальність польоту м’яча, швидкість замаху ключки та швидкість польоту м’яча, а також кут ключки – лофт.

Такі монітори запуску є портативними пристроями, тому вони не вміщають в своїх схемах багато датчиків та не можуть зчитувати багато параметрів. Також за рахунок розміру деякі датчики йдуть у спрощеному вигляді, що призводить до більшої похибки у вимірах таких датчиків. Зазначено, що такі монітори запуску частіше використовуються для особистого використання аніж для професійних тренувань спортсменів з гольфу із-за своєї функціональності та наявних похибок.

Рисунок 1.6 – Портативний монітор запуску

Лофт – це кут між голівкою ключки та землею. Кут позначається градусами і відрізняється для кожного типу ключки. Як правило лофт необхідно враховувати для завдання правильного кута удару (Angle of attack)

Рисунок 1.7 – Вплив лофту на кут удару

«Лофт» або «кут голівки ключки» — це вимірювання в градусах кута, під яким лицьова частина палиці лежить відносно абсолютно вертикальної грані ударної поверхні, представленої древком. Вимірювання такої величини у різних типах ключок різне, у відповідності до їх форм фактору.

Одним з най поширеніших моніторів запуску є Uneekor EYE XO Launch Monitor. Унікальна відмінність такого монітору, що він розміщується на стелі і за рахунок положення має можливість збирати більше даних. Також, на відміну від портативних приладів він має більшу кількість датчиків, що можуть зчитувати більшу кількість інформації про удар користувача, та корегувати все отриманні дані при співставленні інформації різних датчиків. Датчик – технічно відокремлений пристрій, що містить один або декілька первинних вимірювальних перетворювачів. Датчик призначений для вироблення сигналу вимірювальної інформації у формі, зручній для передачі, подальшого перетворення, обробки та зберігання інформації.

На додаток до основних даних, даний пристрій також надає інформацію про шлях ключки, кут голівки ключки при торканні з поверхнею м’яча, інформацію про обертання бічне та обернене, кут запуску, азимут та інше дані для розуміння принципу побудови траєкторії м’яча.

Рисунок 1.8 – Приклад підвісного монітору запуску

Один з найпопулярніших та найбільш просунутих пристроїв моніторів запуску є Foresight Sports GC3 Launch Monitor, який має найбільшу кількість вимірювальних пристроїв, та вважається одним з точніших приладів.

Такі монітори запуску зазвичай використовуються для емуляторів гольфу спортивних школах, тренерами та професійними гравцями. Також дані монітори та програмна частина до них може моделювати удари на чемпіонатах та проводити порівняльний аналіз груп ударів гравців.

Монітор запуску GC3 вимірює низку параметрів даних про ключки та м’ячі, використовуючи комбінацію інфрачервоного відстеження об’єктів і

високошвидкісних камер. Три камери використовуються як частина трископічної фотометричної системи камер. Можна виміряти загалом десять параметрів даних ключки та м’яча, включаючи швидкість м’яча, швидкість голови ключки, кут запуску, бічне обертання, бічний кут, коефіцієнт удару та кут атаки. GC3 працює від літій-іонної батареї та містить акселерометр для автоматичного вирівнювання, а також барометричний датчик, який коригує показання даних залежно від атмосферних умов, а саме вологості й тиску, які можуть значно впливати на побудову траєкторії удару, що змінить як загальну дистанцію, так і дистанцію прольоту м’яча.

Рисунок 1.9 – Монітор запуску GC3

Тобто як такий пристрій є значним інструментом, який доцільно використовуватись для побудови гри симулятора в гольф, за допомогою якого можна побудувати велику ступінь занурення користувача в ігровий та навчальний процес. Отриманні навички та знання можна буде використовувати на справжньому ігровому полігоні.

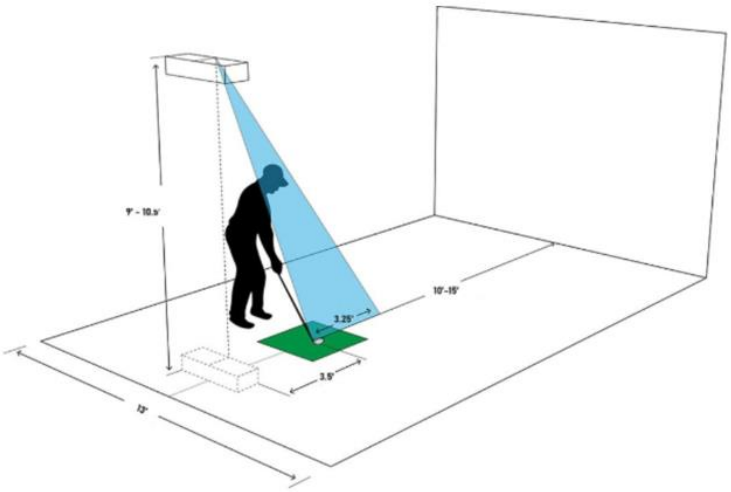

Рисунок 1.10 – Загальна схема використання моніторів запуску

Для кращого представлення інформації щодо функцій моніторів запуску було запропоновано наступну порівняльну таблицю:

Таблиця 1.1 – Порівняння характеристик моніторів запуску

Монітори запуску	Параметри				
	Дальність польоту	Швидкість м’ячу	Швидкість ключки	Кут запуску	Платформа
Caddie SC200	Доступна	Доступна	Відсутня	Відсутня	Андройд
Uneekor EYE	Доступна	Доступна	Доступна	Відсутня	ПК
Foresight Hawk	Доступна	Доступна	Доступна	Доступна	Усі
Foresight GC3	Доступна	Доступна	Доступна	Доступна	Усі

Застосування новітнього обладнання та більш сучасних контролерів збільшує можливості інтелектуальних інформаційних систем щодо зчитування та аналізу даних отриманих гравцем під час ігрових сесій.

Отриманні системою дані можуть застосовуватись майже у будь якому виді спорту, або галузі, адже доповнена реальність, датчики, сенсори кінект роблять

можливим зчитати та проаналізувати будь яку інформацію про положення, швидкість, дій гравця у реальному світі.

1.2.2 Збір даних необхідних для обробки інтелектуальними системами в емуляторах гольфу.

Для забезпечення гравця всією необхідною інформацію для гри в гольф, користувач має отримувати інформацію за допомогою додатка яка надасть йому додаткові знання про удар у емуляторі гольфу, про переміщення та якісні характеристики м’яча. Частина інформації може бути надана монітором запуску, інша частина має бути розрахована програмним забезпеченням за допомогою базових параметрів самого технічного пристрою..

Базові дані, необхідні гравцю щодо польоту м’яча це:

- швидкість м’яча – базовий параметр за допомогою якого розраховується траєкторія польоту;
- кут запуску, також використовується для калькуляції траєкторії та висоти удару;
- напрям запуску – це азимут, що показує сторону в яку був здійсненний удар;
- бокове обертання, що впливає на поперечну позицію траєкторії під час калькуляції удару;
- заднє обертання – характеристика, що впливає на дальність удару користувача;
- швидкість обертання – характеристика м’яча, що буде впливати на модель відскоку;
- нахил осі обертання.

Можна виділити наступні дані що необхідно отримати для отримання характеристик ключки:

- швидкість ключки – це базовий параметр необхідний для розрахунку інших показників;
- ефективність використання ключки;
- кут удару в залежності від типу ключки має різний вплив на обертання та куп запуску м’яча;
- шлях удару ключкою – величина замаху;
- кут обличчя ключки до кута шляху траєкторії;
- поперечний нахил ключки – впливає на азимут удару;
- динамічний лофт – це величина висоти на передній частині клубу під час удару, яка вимірюється відносно горизонту;
- швидкість закриття — це міра обертання обличчя ключки, коли вона наближається до точки удару. Це вимірювання обчислюється відносно шляху клубу. Зокрема, вимірюється кут між лицьовою стороною ключки та траєкторією ключки;
- горизонтальна відстань місця удару відносно центру обличчя;
- відстань місця удару по вертикалі відносно центру обличчя;
- лице до траєкторії є ключовим фактором у визначенні очікуваної кривизни (осі обертання) удару в гольф. Якщо припустити центрований контакт, м’яч повинен вигнутися в напрямку обличчя та від траєкторії ключки.

Для розуміння логіки побудови траєкторії нам необхідно знати параметри польоту м’яча для коректних розрахунків. До таких параметрів належать:

- дальність польоту м’яча – ця характеристика визначає дистанцію в якій м’яч знаходиться в повітрі до першого торкання землі;
- загальна дистанція – визначає відстань польоту м’яча та відстань кочення м’яча після торкання землі до повної зупинки;
- пікова висота – найвища точка траєкторії;
- зсув траєкторії відносно центральної лінії удару (offline);

- кривизна траєкторії;
- кут спуску траєкторії;
- час зависання – час, який м'яч проводить у повітрі після удару, перш ніж впасти на землю.

Можна виділити сукупність характеристик, що будуть задіяні в аналізі групи ударів. До них можна віднести:

- модель дисперсія – це те, наскільки далеко зліва направо охоплює кожен удар і наскільки далеко від ззаду наперед розподілені удари;
- параметр розкиду;
- середній діапазон групи ударів;
- точність у гольфі означає здатність влучити м’ячем близько до обраних цілей;
- точка впливу ключки – середнє значення для групи ударів де м’яч торкає голівки ключки;
- середня точка всіх ударів – це є центр цілі для групи ударів.

Отже отриманні дані були систематизовані за об’єктом до якого вони відносяться та розбиті на різні рівні розрахунків [10]. За об’єктом дані поділяються на розрахунки що відносяться до м’яча, ключки, польоту та групи ударів. Також була виділена групу яка відноситься до сесії гравця. Вона витікає із розрахунків щодо порівняння груп ударів. До таких розрахунків можна віднести точність сесії, загальну дисперсію, середнє квадратичне відхилення ударів у сесії та розрахувати типову тенденцію поточного тренування.

Для реалізації проєкту та покращення моделювання є сенс у включені додаткових характеристик для проведення більш точного розрахунку та аналізу даних. До таких даних можна віднести зовнішні фактори, що впливають на траєкторію, такі як тиск та вологість повітря, а також інформація щодо матеріалів м’яча та ігрового полігону для гольфу.

Була сформована наступна зведена таблиця параметрів, що включає система удару:

Таблиця 1.2 – Таблиця типів параметрів удару користувача

Параметри				
М’яча	Ключки	Траєкторії	Групи	Сесії
Швидкість	Швидкість	Дальність польоту	Дисперсія	Середня швидкість м’яча
Кут запуску	Ефективність	Загальна дальність	Середня дальність	Дальність до цілі
Напрям запуску	Кут удару	Пікова висота	Розкид	Дальність до зеленого поля
Бокове обертання	Шлях ключки	Відхилення	Точність	Удари по краю зеленого поля
Заднє обертання	Кут відносно горизонту	Кривизна траєкторії	Горизонтальна дальність	Загальна точність
Швидкість обертання	Кут древка	Кут зниження	Вертикальна дальність	
Нахил осі обертання	Кут динамічного лофту	Час у польоті	Точка центру	

Також можна розрахувати додаткові параметри які допоможуть розрахувати значення необхідні для гри емулятора гольфу.

Висновки до розділу 1

Отже, було визначено, що наразі існує досить багато ігрових додатків на тему емулятору гольфу, але меншість з них використовують новітні контролери та технології, що можуть покращити ефективність та розуміння гри у гольф користувача. Зараз не так багато додатків які використовують контролери або монітори запуску для зчитування показників удару в іграх емуляторах гольфу. Основним пристроєм, який з легкістю може зчитувати дані про удар стали консолі й додаткові контролери до VR пристроїв, таких як Playstation VR, Xbox VR та інші.

Справжнім засобом для зчитування майже всіх необхідних метрик для спорту стали монітори запуску, які можуть вимірювати все від швидкості м’яча до позиції спортивного знаряддя у просторі, кутів обертання, траєкторії на інших параметрів удару, які необхідні для спортивних ігор та гри у гольф.

У розділі було визначено, що для вдалого існування додатків з контролерами необхідно також впровадження додаткових інтелектуальних інформаційних систем в архітектуру додатків. До таких ІІС можуть бути віднесені різні типи ІІС з різними завданнями. Так було виділено декілька типів за призначенням, аудиторію та цілю гравця.

Було визначено що у відповідності до кожного типу можуть бути застосовані різні ІІС які будуть виконувати різну роботу та надавати необхідну інформацію гравцю, у відповідності до своїх задач.

Кожна із інформаційних систем яка може бути застосована для виконання однієї із своїх задач має свій набір параметрів та розрахунків. Окреслені наступні типи задач, такі як навчальна, діагностична, інтерпретаційна, моніторингова, задача прогнозування, планування, підтримка прийняття рішень, тощо.

Для роботи ІІС окреслена низка параметрів необхідних для роботи інтелектуальних інформаційних систем в емуляторі гри гольф. Ці параметри та

отримані дані допомагають у функціонуванні ІІС та можуть надати користувачу очікувану інформацію.

2 ВПРОВАДЖЕННЯ СИСТЕМ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ УДАРУ ГРАВЦЯ У ГРІ-ЕМУЛЯТОРІ ГОЛЬФУ

2.1 Системи обробки даних у гольф-емуляторах

2.1.1 Системи прорахунку даних удару у іграх гольф

Для створення системи прорахунку ударів гравця було створено систему для зчитування базових даних швидкості ключки, м’яча та інших параметрів удару. Ці дані можуть бути отриманні з монітору запуску або іншого контролеру. Для більш швидкого і зручного тестування та взаємодії з додатком було створено симулятор удару. Такий інструмент зроблено вбудованим до системи гри, основна частина для розрахунків винесена у окрему бібліотеку.

Для побудови траєкторії необхідні наступні параметри як: швидкість м’яча, швидкість ключки, кут запуску, азимут запуску, обернене обертання, бокове обертання, шлях ключки по вертикалі та горизонталі, кут дотику до голівки ключки, кут древка ключки, лофт ключки, та інші параметри.

За допомогою цих базових параметрів була отримана можливість прорахувати траєкторії удару, але перед цим було складено інтерфейс користувача для зчитування базових даних у симуляторі удару. Для цього розроблено вікно з відповідними параметрами та двома полями вводу для кожного параметра, для вибору довільного значення між двох значень. Кожен з параметрів має свої одиниці виміру та для цього була додана функція для переводу значень у різні системи так як у різних країнах використовуються різні системи виміру, так у Європі використовується метрична система виміру, в Великій Британії та Сполучених Штатах Америки використовується імперська система мір. Так наприклад довжина полігону може бути виражена як в ярдах, так і в метрах, при цьому загальна довжина полігону для гольфу має бути однакова. Ця система конвертації систем виміру скрита в самому додатку й недоступна для

безпосередньої взаємодії користувача з конвертером. Користувачу має бути надана можливість зміни системи обчислення через налаштування, для чого була вбудована система перемикачів для переключення виміру дистанції з метрів на ярди, для виміру швидкості з кілометрів на годину на милі на годину, для переключення обертання з роздільного (оберненого та бокового), на загальне обертання, що задається вектором. Також користувачу надана можливість переключитись з відображення азимуту з знаків + чи – на напрям право чи ліво.

Отже було отримано наступний інтерфейс симулятору удару:

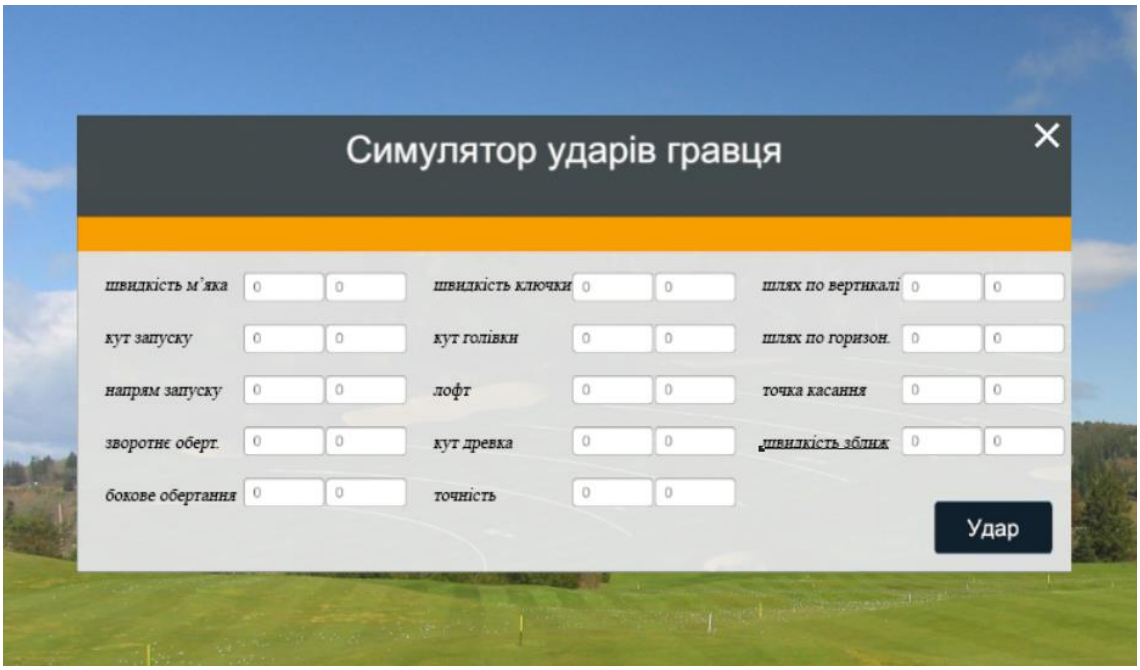

Рисунок 2.1 – Прототип вікна симулятора ударів

Дані що задаються у вікні симулятора ударів надходять у систему додатку у тих одиницях виміру які гравець обрав в налаштуваннях до свого додатку. Після проведення удару дані щодо удару можуть бути скоректовані у будь яку систему виміру, що підтримується додатком. Для зручного користування системою конвертації доцільно використовувати патерн проєктування фасад, у який система

користувача надасть тип параметру, а повернеться сконвертоване значення та одиниці виміру для відображення цих даних на інтерфейсі екрану користувача.

Було розроблено прототип інтерфейсу користувача – панелі налаштування для вибору одиниць виміру для використання їх у проєкті.

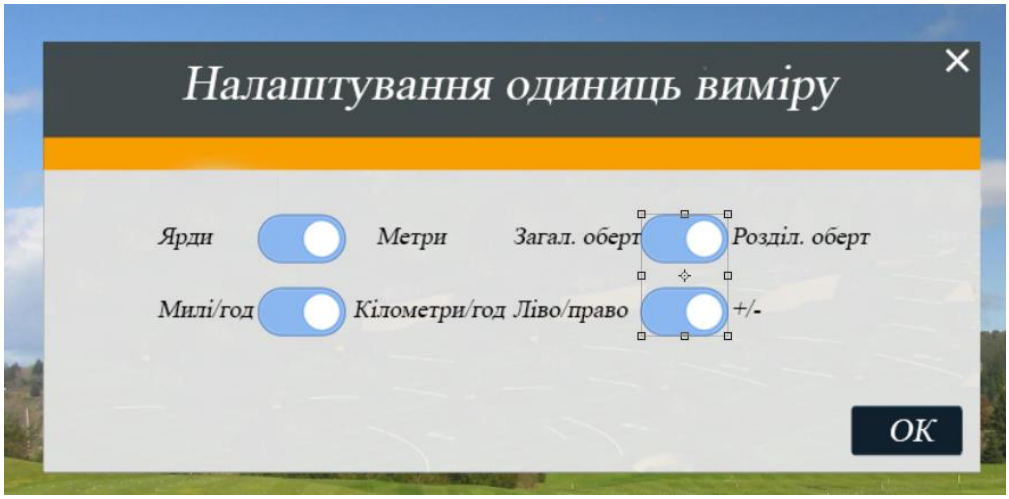

Рисунок 2.2 – Прототип вікна налаштувань одиниць виміру

Оскільки додаток має можливість отримувати дані та вносити їх у систему за допомогою симулятора ударів або за допомогою монітору запуску в проєкт стало необхідно розробити систему за допомогою якої буде розраховуватись траєкторія польоту м’яча. Для цього існує декілька способів розрахунку.

Швидкість м’яча розраховується як швидкість ключки помножена на константу 1,48 – це значення відповідає нормі стрибучості м’яча.

$$x = y * 1.48 \qquad (2.1)$$

де x – це швидкість м’яча;

y – це швидкість ключки.

Кут удару було отримано симулятором або монітором запуску.

Коли м’яч б’ють під кутом до горизонту, рух такого м’яча спочатку рівно уповільнено підіймається, а потім рівно прискорено падає на ігровий полігон гольфу. Було розраховано рух відносно двох координат. Координата напряму може бути додана пізніше для корегування напряму удару. Рух м’яча проходить на малій дистанції від землі, отож прискорення тіла дорівнює вільному падінню, тобто:

$$a = g \tag{2.2}$$

де a – прискорення;

g – прискорення свобідного падіння.

У відповідності до основного закону механіки є другий закон Ньютона, що він пов'язує рівнодіючу зовнішніх сил, які діють на тіло, його масу та прискорення, одержуване внаслідок дії сил [11]. Тобто виникає сума сил.

$$\overrightarrow{Fp} = m\vec{a} \tag{2.3}$$

$$\overrightarrow{Fp} = \overrightarrow{F\text{т}} + \overrightarrow{F\text{с}} + \overrightarrow{F\text{м}} \tag{2.4}$$

де $\overrightarrow{Fp}$ – рівнодіюча сила;

m – маса м’яча;

a – прискорення;

Fт – сила тяжіння;

Fс – сила супротиву;

Fм – сила Магнуса.

У даних розрахунках не були використані сили супротиву та додаткові сили, що впливають на м’яч. Було побудовано графік тіла – м’яча для гольфа, що було ударено під кутом до горизонту.

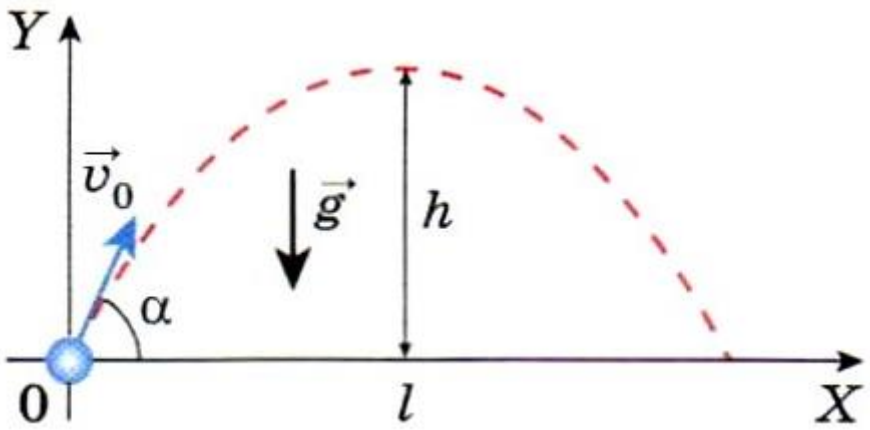

Рисунок 2.3 – Графік тіла кинутого під кутом

Виходячи з графіку видно що вектор швидкості тіла кинутого буд кутом до горизонту направлений по дотичній до траєкторії його руху. Оскільки початкова швидкість не буде направлена уздовж горизонту, то вона буде

$$v_{ox} = v_o cos\alpha \quad (2.5)$$

$$v_{oy} = v_o sin\alpha \quad (2.6)$$

де v_{ox} – проекція удару на вісь x;

v_{oy} – проекція удару на вісь y.

Отже у відповідності до теореми Піфагора може бути розрахований модуль миттєвої швидкості у моменті часу.

$$v = \sqrt{v_{x^2} + v_{y^2}} \quad (2.7)$$

де v – швидкість;

v_{x^2} – швидкість по вісі x;

v_{y^2} – швидкість по вісі y.

Було розраховано мінімальну швидкість м’яча, яка досягається в найвищій точці траєкторії.

$$v_{min} = v_0 \cos\alpha = v_h \quad (2.8)$$

де v_{min} – це мінімальна швидкість м’яча;

v_h – швидкість м’яча у найвищій точці;

$\cos\alpha$ – кут запуску м’яча.

Розраховано час підйому м’яча до найвищої точки траєкторії. Оскільки час підйому дорівнює часу падіння отримаємо наступне значення.

$$t_{\text{підйому}} = \frac{v_0 \sin\alpha}{g} \quad (2.9)$$

де $t_{\text{підйому}}$ – час підйому м’яча на максимальну висоту;

v_0 – початкова швидкість;

$sin\,\alpha$ – кут запуску м’яча;

g – прискорення вільного падіння.

Тому було зроблено висновок, що повний час польоту м’яча з гольфу буде мати наступний вигляд:

$$t_{\text{повне}} = 2 * t_{\text{підйому}} \quad (2.10)$$

Маючи швидкість, висоту та час польоту було визначено загальну дальність удару. Для розрахунку було використано наступну формулу:

$$l = v_0 \cos\alpha \frac{2\, v_0 \sin\alpha}{g} = \frac{2\, v_0^2 \cos\alpha \sin\alpha}{g} = \frac{v_0^2 \sin 2\alpha}{g} \quad (2.11)$$

де $t_{\text{повне}}$ – час підйому м’яча на максимальну висоту;

v_0 – початкова швидкість;

α – кут запуску м’яча.

Для побудови дуги коректної форми необхідно знати висоту м’яча у кожній момент часу польоту. Для цього використовується формула:

$$y = v_0 \sin\alpha \, t \frac{gt^2}{2} \tag{2.12}$$

де y – позиція по вісі ординат у заданий момент часу;

t – момент часу підйому;

v_0 – початкова швидкість;

α – кут запуску м’яча;

g – прискорення вільного падіння.

Також є необхідність дізнатись позицію м’яча по осі абсциси, по вісі X, тобто горизонтальне зміщення тіла у будь-який відрізок часу польоту м’яча для гольфу.

$$x = v_0 \cos\alpha \, t \tag{2.13}$$

де x – позиція по вісі абсцис у заданий момент часу;

t – момент часу підйому;

v_0 – початкова швидкість;

α – кут запуску м’яча.

Було розраховано найвищу точку підйому, тобто найвищу позицію м’яча при ударі:

$$h = \frac{v_0^2 \sin^2\alpha}{2g} \tag{2.14}$$

де h – максимальна висота польоту;

v_0 – початкова швидкість;

α – кут запуску м’яча;

g – прискорення вільного падіння.

Отримавши всі ці дані вони можуть бути реалізовані у коді мови програмування C# для побудови траєкторії польоту м’яча для гольфу. Також у формули слід додати напрям удару, та модифікувати траєкторію польоту за допомогою обертання.

Підставивши базові значення удару у симуляторі гри у гольф можна отримати наступний набір траєкторій.

Рисунок 2.4 – Ізометрична проекція ударів

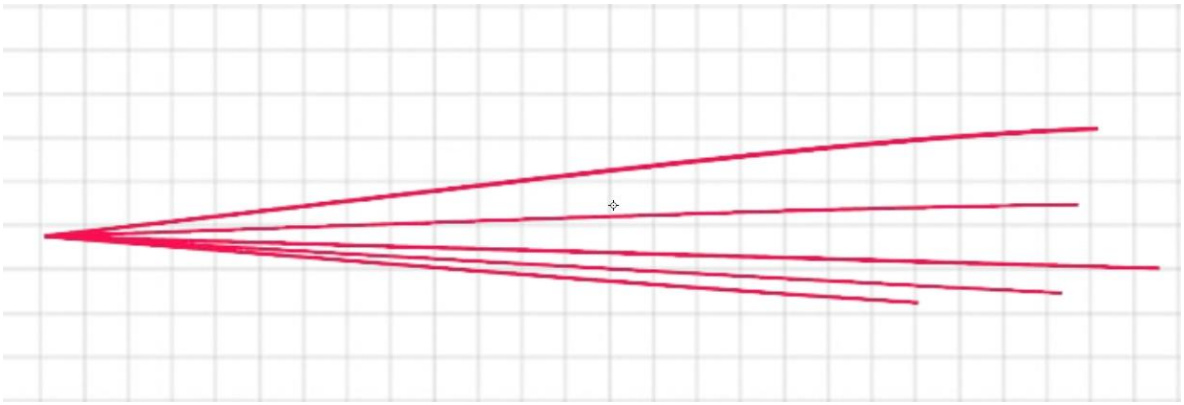

Рисунок 2.5 – Проекція ударів зверху

Рисунок 2.6 – Проекція ударів збоку

Отже маючи можливість робити удари у грі симуляторі гольфу можна проаналізувати на визначити тенденції гравця у поточних умовах у емуляторі гри у гольф.

2.1.2 Система збору інформації про сесії гравця

Для аналізу наявних даних був організований їх збір у локальну й видалену базу даних. Так на стороні клієнта дані збираються у локальну базу даних SQL Lite для швидкого доступу до запису та читання даних. SQL Lite – це легковагова реляційна база даних. Вона представлена у вигляді відокремленої бібліотеки, що дозволяє з легкістю побудувати будь-яку структуру таблиць з даними для будь-якого додатку. Вона підтримується також для збірок під Android та iOS, що дозволяє портувати її на будь який пристрій на якому буде працювати симулятор гри гольф. Така база даних повинна бути локальним сховищем для аналізу даних, та використання їх для навчання ІІС [12].

Також був розроблений веб-сервіс для обробки даних онлайн. Такий сервіс збудований за допомогою технології ASP .Core із застосуванням Entity Framework. ASP.NET Core – це кросплатформерна технологія, для створення за стосунків на базі .NET Core. ASP.NET Core може працювати поверх крос-платформного середовища .NET Core, яке може бути розгорнуто на основних

популярних операційних системах: Windows, Mac OS, Linux. І таким чином, за допомогою ASP.NET Core можна створювати крос-платформні програми. І хоча Windows як середовище для розробки та розгортання програми досі переважає, але тепер система не обмежені тільки цією операційною системою. Тобто з'являється можливість запускати веб-програми не тільки на ОС Windows, а й на Linux і Mac OS, у якості сервера. А для розгортання веб-програми можна використовувати традиційний IIS, або крос-платформний веб-сервер Kestrel [13].

Entity Framework – це інструмент для взаємодії з базою даних через код програми. Включає в себе набір інструментів за допомогою яких можна викликати команди для запису, читання, оновлення чи видалення даних з БД. На стороні клієнту доцільно використовувати MySQL.

MySQL – це система управління базами даних (СУБД), яка розповсюджується як вільне програмне та має відкритий похідний код. Така система надає можливість запису, читання, оновлення та видалення даних [14].

Для гри емулятору гольфу було створено таблицю, що містить в собі інформацію про ігрові сесії гравця. Така таблиця містить в собі ідентифікатор гравця, ідентифікатор сесії, дату сесії, інформацію про кількість груп ударів, загальну кількість ударів, нотатки до сесії, тип полігону для гри у гольф.

Отже загальний вид таблиці сесій бази даних буде мати наступний вигляд:

Таблиця 2.1 – Приклад таблиці сесії бази даних

UserId	SessionId	GoupsCount	HitsCount	Date	SessionDescription	RangeType

У таблицю сесій можуть бути включені й інші дані, що характеризують ігрову сесію гравця, гри емулятору у гольф. Одна сесія може включати в себе багато груп ударів, тобто буде реалізовано зв'язок багатьох до одного. Такий зв'язок буде утворений завдяки реалізації зовнішнього ключа.

Таблиця бази даних що характеризує групу ударів м’ячем для гольфу має ідентифікатор групи ударів, тип ключки, що використовується у даній групі, тип м’яча, нотатки, посилання на зовнішній ключ сесії, дистанція до лунки.

Таблиця 2.2 – Приклад таблиці бази даних групи ударів

GroupId	SessionId	ClubId	BallId	HoleDistance	Notes

Таблиця бази даних що вміщає в себе дані удару матиме все інформацію яка була зчитана з засобу удару та розрахована під час роботи інтелектуальної системи. До таких даних відносяться ідентифікатор зовнішнього ключа сесії, ідентифікатор зовнішнього ключа групи ударів, позиція ігрової лунки, стартова позиція м’яча на полігоні, ідентифікатор удару, інформація про зовнішній тиск та вологість повітря, кут ключки до траєкторії, вісь обертання м’яча, час створення удару, інформація про траєкторію м’яча та ключки [15].

До даних про траєкторію м’яча належать: азимут, бокове відхилення від лінії удару, швидкість м’яча, кут підйому, швидкість обертання бокового та оберненого, загальне обертання, дистанція польоту м’яча, загальна дистанція після падіння та кочення м’яча, максимальна висота, кут падіння м’яча для гольфа.

Інформація про ключку включає в себе: швидкість ключки, кут удару, шлях удару, ефективність удару, кут голівки до напряму удару, кут древка ключки до удару.

Дані удару доцільно зберігати у локальній базі даних та на сервері для навчання системи на прикладі ударів гравців, що використовують додаток, та завантажують дані на сервер. Для оптимізації системи в подальшому ці дані можуть бути використані для навчання штучного інтелекту.

Отже було сформовано таблицю з даними:

Таблиця 2.3 – Приклад таблиці даних удару користувача

SessionId	GroupId	ShotId	HoleDistance	IsRightHanded
Temperature	Barometer	BallPos	ClubType	BallType
Notes	Pressure	Azimuth	Offline	BallSpeed
SpinAxis	SideSpin	BackSpin	Carry	Distance
DescentAngle	ClubSpeed	AttackAngle	Efficiency	ClubPath
Lie	Loft	FaceImpactH	FaceImpactV	ClosureRate
Elevation	Altitude	Date	Height	FaceTarget

Робота з базою даних проводиться через API SQL Lite. Взаємодія з API виконується за допомогою спеціального класу фасаду, для надання доступу для взаємодії однієї частини програмного забезпечення з іншою.

Патерн фасаду – це шаблон проєктування програмного забезпечення, який зазвичай використовується в об’єктно-орієнтованому програмуванні [16]. Логіка фасаду відноситься так само як до фасаду в архітектурі, фасад – це об’єкт, який служить переднім інтерфейсом, що маскує більш складний основний або структурний код [17]. Фасад може: покращити читабельність і зручність використання бібліотеки програмного забезпечення шляхом маскування взаємодії з більш складними компонентами за єдиним API надати контекстно-залежний інтерфейс для більш загальної функціональності, служити відправною точкою для ширшого рефакторінгу монолітних або тісно пов’язаних систем на користь більш слабко пов’язаного коду [18]. Тобто фасад представляє собою програмний інтерфейс, що дає змогу викликати команди більш складного API, у даному випадку інкапсулюючи створення запитів та команд до бази даних.

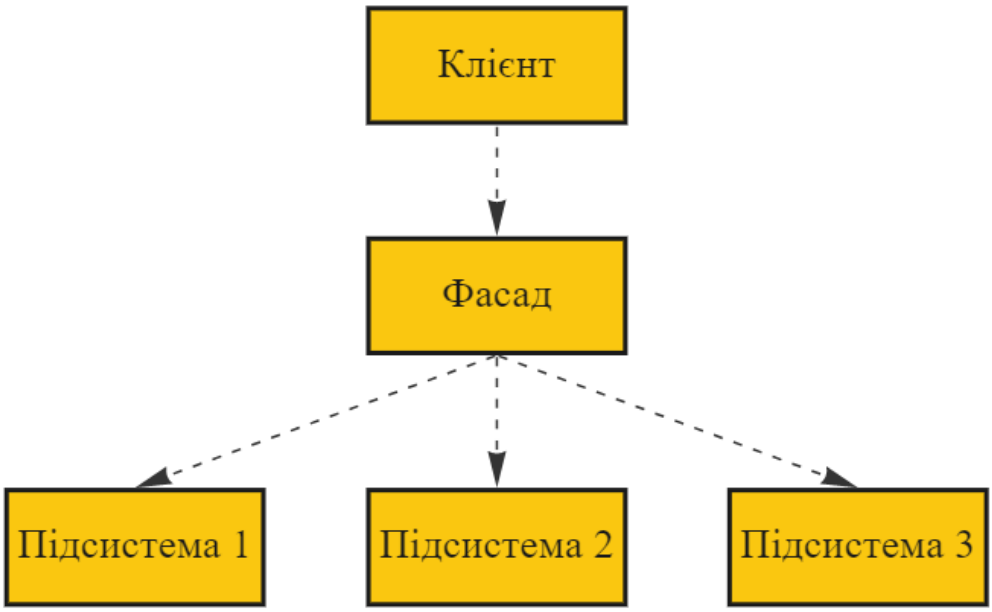

Рисунок 2.7 – Схема структури патерну фасад

Використання такого патерну обумовлене необхідністю інкапсулювати шар логіки з можливістю формування запитів до бази даних, таким чином надано лише можливість створювати, редагувати і видаляти сесії, групи удари у контрольованих умовах у грі емуляторі гольфу. Також використання фасаду дозволяє легко перемкнути сховище даних з локального до видаленого на сервері.

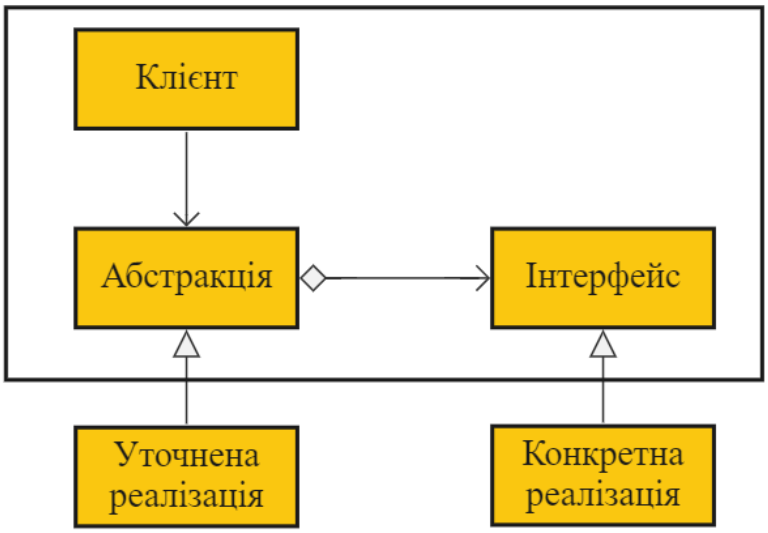

Рисунок 2.8 – Схема структури патерну міст

Таким чином фасад став розширений мостом для переключення API з SQLLite на сервіси взаємодії з мережею. Також це дозволить змінювати частину логіки додатку, що відповідає за зберігання даних, змінювати тип бази даних, тип логування та інше без втручання у фронтенд частину додатку. Таким чином дані стають інкапсульовані від інших шарів логіки проєкту.

2.2 Системи визначення тенденцій ударів гравця у емуляторі гри у гольф

2.2.1 Середньоквадратичне відхилення ударів гравця

Для аналізу даних ударів гравець, тренер та будь який користувач у грі емуляторі гольфу користувач має мати певний набір інструментів для аналізу. Одним з найважливіших параметрів є показники точності гравця, тобто визначення наскільки точно може бути створений наступний удар при аналогічних параметрах, що були задані раніше.

Існує декілька основних метрик за допомогою яких розраховується патерт удару. Це середньоквадратичне відхилення та дисперсія ударів. Окрім цих параметрів слід також враховувати за візерунок, що утворюється в результаті розльоту м’ячів та загальні ігрові тенденції користувача, що випливають з інших ігрових сесій.

Середньоквадратичне відхилення – це статистична величина, показник того, наскільки розкидані числа координат ударів [19]. Такий показник визначається для групи ударів не менше ніж 3 удари. Чим більше число, тим ширший розкид результатів.

Середньоквадратичне відхилення характеризує наскільки гравець може бити удари у межах поставленої цілі та який патерн відхилення вираховується, чи здатен гравець додержуватись патерну групи ударів, які були представлені ним у попередній ігровій сесії та допомагає визначити як змінюється патерн при зміні зовнішніх ігрових параметрів, таких як зміна ігрового полігону, типу ключки чи зміна параметрів удару. Зазвичай гравець мусить притримуватись того самого еліпсу, що утворюється під час ударів. Це може свідчити про його тенденцію гри. Утворення різних еліпсів стандартної дивіації [20] та дисперсії свідчить про те що гравець не контролює свою ігрову тенденцію у емуляторі гри у гольф.

Гравці у гольф розрізняють наступні значення стандартного середньоквадратичного відхилення ударів гравця:

Таблиця 2.4 – Міри середньоквадратичного відхилення ударів користувача

0	ідеальна консистенція ударів гравця (однаковий результат кожного разу)
0<x<10	послідовна тенденція ударів
10<x<20	досить стабільно тенденція ударів
20<x<30	не стабільна консистенція ударів гравця
30<x<40	суперечливо стабільність ударів, великий розкид
40<x<50	досить непослідовна тенденція ударів
50<x<60	дуже суперечливо тенденція ударів, кінці траєкторій знаходяться на великій відстані одне від одного
x<60	удари у довільні позиції, тенденція не спостерігається

Для розрахунку середньоквадратичного відхилення ударів гравця у грі гольф необхідно знайти середню позицію серед всіх ударів, тобто як сума координат ординат і абсцис розділена на кількість ударів [21]. Лістинг коду наданий у Додатку Б:

$$\bar{p}_{avg} = (\sum_i^n \frac{x_i}{n}, \sum_i^n \frac{y_i}{n}) \quad (2.15)$$

де $\bar{p}_{avg}$ – вектор середньої позиції серед всіх ударів;

x_i – координата ординат і-того удару;

y_i – координата абсциси і-того удару;

n – кількість ударів гравця;

i – порядковий номер удару.

$$\bar{s} = (\sum_{i=0}^{n}(p_{xi} - \bar{p}_{xavg})^2, \sum_{i=0}^{n}(p_{yi} - \bar{p}_{yavg})^2) \quad (2.16)$$

де $\bar{s}$ – сума квадратів відхилення;

p_{xi} – координата ординат і-того удару;

$\bar{p}_{xavg}$ – значення x середньої позиції;

p_{yi} – координата абсциси і-того удару;

$\bar{p}_{yavg}$ – значення y середньої позиції;

n – кількість ударів гравця;

i – порядковий номер удару.

Розраховуючи центр середньоквадратичного відхилення необхідно суму квадратів відхилення розділити на кількість ударів та возвести у квадрат.

$$\bar{S}_d = (\frac{\bar{s}_x}{n}^2, \frac{\bar{s}_y}{n}^2) \quad (2.17)$$

де $\bar{S}_d$ – середньоквадратичне відхилення;

$\bar{S}_x$ – сума квадратів відхилення за координатою x;

$\bar{S}_y$ – сума квадратів відхилення за координатою y

n – кількість ударів гравця;

Точка стандартної дивіації є центром еліпсу середньоквадратичного відхилення. Для розрахунку форми та позиції еліпсу використовувалась формула для периметру заснована на другому наближенні Рамануджана Срініваса [22]. Лістинг коду наданий у Додатку В.

Для побудови еліпсу середньоквадратичного відхилення зазвичай береться точність 25%, 50% та 75%. [23] В результаті розрахунку еліпсу для групи ударів отримано наступний патерн:

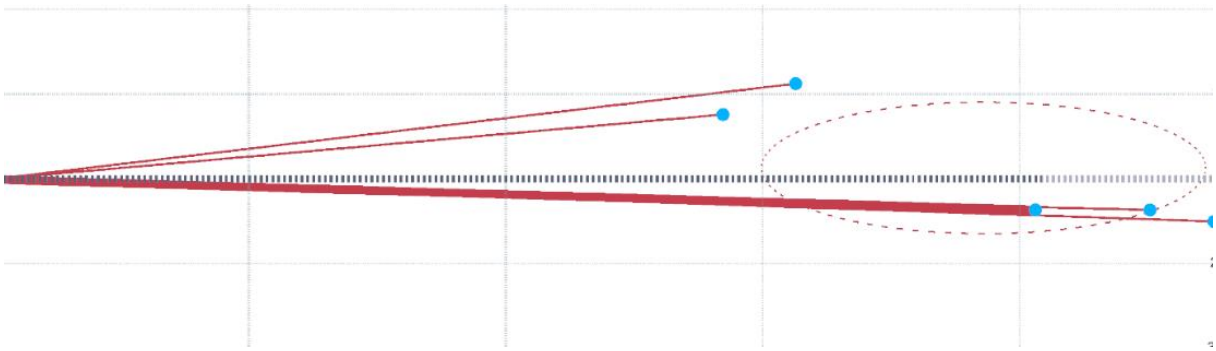

Рисунок 2.9 – Еліпс середньоквадратичного відхилення

Отже виходячи з цих даних можна зробити висновок про точність гравця та про розкид м’ячів.

2.2.2 Дисперсія ударів гравця

Дисперсія у гольфі показує як далеко знаходиться най лівіший удар від най правішого та найкоротший удар від найдовшого. У відповідності до цього будується еліпс дисперсії, що має охоплювати всі удари групи.

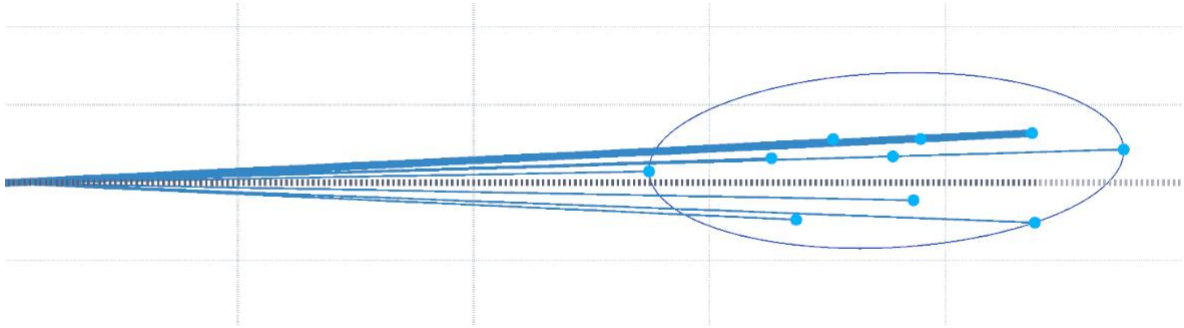

Рисунок 2.10 – Еліпс дисперсії ударів гравця

Дисперсія допомагає визначити тенденцію ударів гравця певною ключкою при грі у симулятор гольфу. Також дисперсія використовується для надання рекомендацій щодо корегування точності та дальності ударів, шляхом винесення

рекомендацій щодо зміни ключки, або щодо зміни певних параметрів удару [24]. При слідуванні таким рекомендаціям розмір дисперсії та середньоквадратичного відхилення зменшує, що підвищує точність гравця [25].

Для знаходження значення дисперсії необхідно виміряти площу еліпсу дисперсії, це робиться за формулою:

$$S = \pi * R * r \tag{2.18}$$

де S – площа еліпсу, тобто значення дисперсії;

π – константа, відношення довжини кола до його діаметру;

R – великий радіус;

r – малий радіус.

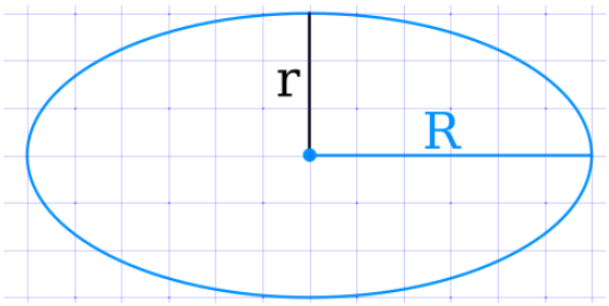

Рисунок 2.11 – Еліпс дисперсії ударів гравця

Чим більше значення дисперсії, тим більший розкид м’ячів для гольфу один від одного, що свідчить про малий контроль горизонтального та повздовжнього положення м’яча після удару.

Для розрахунку еліпсу дисперсії було взято 2 найвіддаленіших вектора один від одного, за алгоритмом пошуку найменшого вектору серед колекції, та взято їх магнітуду від початку до кінцевої точки. Таким чином різниця векторів надає інформацію щодо дистанції між ними [26]. Отже було зроблено перебір дистанцій між векторами за наступною формулою.

$$AB = \sqrt{(B_x - A_x)^2 + (B_y - A_y)^2} \quad (2.19)$$

де AB – відстань між двома векторами;

A_x – координата x першого вектору;

A_y – координата y першого вектору;

B_x – координата x другого вектору;

B_y – координата y другого вектору.

Для побудови форми еліпсу необхідно визначити центр еліпсу. Для цього беруться середнє значення між найближчим значенням вектору та найдовшим. Для того щоб координати відповідали дійсності на ігровому полігоні необхідно зсунути координати центру на дистанцію найкоротшого удару.

$$\bar{E} = (A_x + \frac{(B_x - A_x)}{2}, A_y + \frac{(B_y - A_y)}{2},) \quad (2.20)$$

де $\bar{E}$ – позиція центру еліпсу;

A_x – координата x першого вектору;

A_y – координата y першого вектору;

B_x – координата x другого вектору;

B_y – координата y другого вектору.

Великим радіусом еліпсу буде відстань від центру до найвіддаленішої точки, або половина відстані від найвіддаленіших точок між собою.

Для визначення малого радіусу необхідно перебрати точки, за виключеннями тих, що були використані для визначення великого радіусу за наступною формулою:

$$r = \sqrt{\frac{A_x^2}{(1-\frac{A_y^2}{R^2})}} \tag{2.21}$$

де r – малий радіус еліпсу;

A_x – координата x і-того удару;

A_y – координата y і-того удару;

R – великий радіус еліпсу.

Після побудови еліпсу необхідно правильно розвернути його відносно ігрового поля та ударів. Для цього необхідно зробити наступні обчислення.

$$\theta = \text{arctg}\left(\frac{B_x - A_x}{B_y - A_y}\right) * \omega \tag{2.22}$$

де θ – кут розвороту еліпсу;

A_x – координата x першого удару;

A_y – координата y першого удару;

B_x – координата x другого удару;

B_y – координата y другого удару;

ω – кутова частота, константа, радіани у градуси.

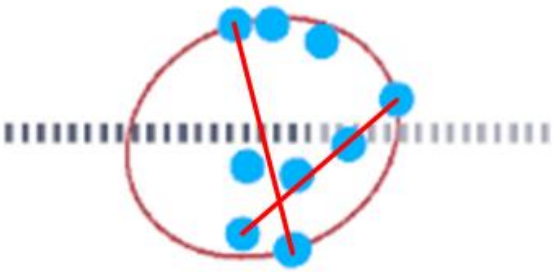

Рисунок 2.12 – Приклад побудова еліпсу дисперсії ударів гравця

В результаті було отримано еліпс розвернутий у відповідності до ударів, що мають найбільшу відстань одне від одного.

2.2.3 Розробка теплової карти ударів ігровому полігоні

Для гри емулятора у гольф важливою метрикою є точність та дальність ударів гравця. Для аналізу ударів впродовж ігрової сесії гри емулятора гольф доцільно розробити теплову карту для відображення позицій де м’яч гравця потрапляє частіше.

Теплова карта ігрового полігону – це спеціальне представлення даних за допомогою графічних засобів, а саме кольорового забарвлення різного кольору, при якому колір зіставляється з матрицею частот попадання даних у цей проміжок. Матриця частот включає в себе відповідність кольору до кількості попадань у певну зону [27]. Так необхідно побудувати теплову шкалу яка буде надавати інформацію користувачу про ступінь попадань у певний квадрант, де червоний колір свідчить про велику кількість попадань гравцем під час гри у гольф у заданий проміжок. Жовте позначення вказує про середню щільність ударів. При цьому зелений колір свідчить про малу влучність у даному районі.

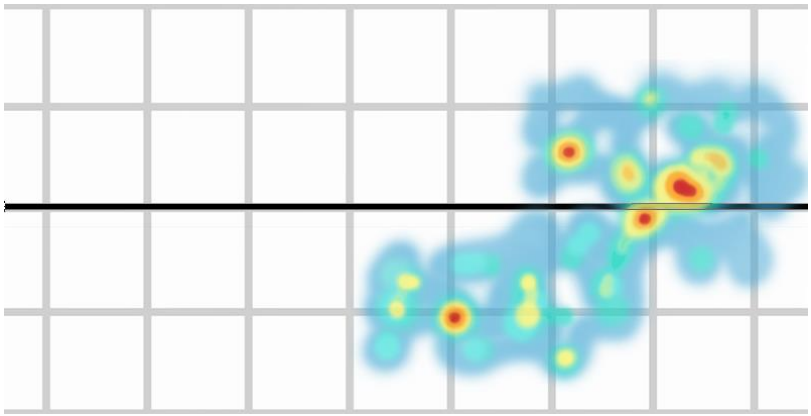

Рисунок 2.13 – Теплова карта ударів ігрової сесії

Для розробки теплової карти полігону необхідно знати його розміри та позиції попадань гравця м’ячем. Для цього доцільно використовувати вектори. Вектор вміщає в себе точку розміщення м’яча.

Дані сесії та ударів усіх груп ударів вміщаються та зберігаються в базі даних у таблиці ударів гравця. Треба припустити що кожен удар при потраплянні у кінцеву точку буде підсвічувати ділянку певного розміру [28]. При потраплянні м’ячів поряд вони будуть посилювати вплив на певну ділянку та колір цієї ділянки зміщується у сторону червоного кольору по шкалі кольорів. Шкала кольорів виконана у якості палітри й реалізована через SO – Scriptable object.

Scriptable Object — це спеціальний серіалізований клас Unity, що дає змогу зберігати велику кількість даних що пересікаються між різними об’єктами, що не залежать від певних екземплярів. Scriptable Object дає змогу гнучко налаштовувати.

Використання Scriptable Objects полегшує керування змінами та налагодження об’єкту, дає можливість створити рівень гнучкого зв’язку між різними системами у грі емуляторі гольфу, для більш легшого застосування та адаптування скрипта протягом виробничого процесу розробки гри. Надає змогу повторно використовувати об’єкт як шаблон для конфігурування.

Оскільки сама теплова карта є частиною візуалізації проєкту, то доцільно розробити її у якості шейдеру, для гнучкого налаштування та рендеренгу та перекладення навантаження з малювання теплової карти на процесор відео карти [29].

Шейдер – це спеціальна програма, що взаємодіє з одним із шарів графічної черги рендерінгу, при реалізації графічних дій [30] у тривимірних й двовимірних іграх. Шейдер задає правила як повинен бути намальований об’єкт, у якому порядку, з якими ефектами та як на нього повинні впливати інші шари графіки, такі як світло, карта нормалей та інше. Таким чином програми шейдери

допомагають реалізувати складну графіку для складних поверхонь із затратою невеликої кількості ресурсів.

Шейдери – це інструкції для наших відеокарт, які говорять, як правильно малювати та трансформувати об’єкти у грі. Для інтеграції роботи рейдеру необхідно взаємодіяти з пайплайном графічного процесу, та розуміти логіку малювання об’єктів [31].

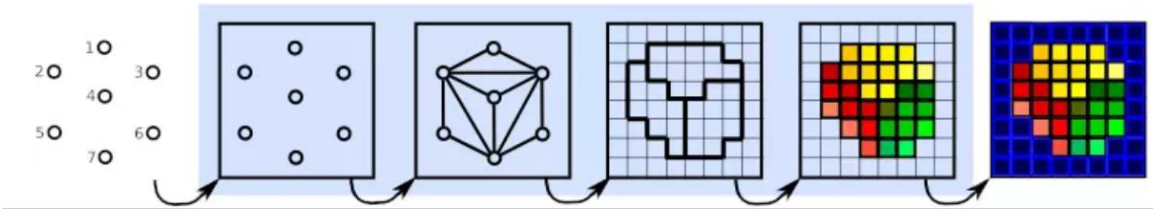

Рисунок 2.14 – Пайплайн роботи шейдеру

Було розроблено шейдер який приймає масив векторів для визначення місця ударів. Зображення ігрового поля повинно бути відображеним на окремому шарі, шар з тепловою картою використовує клас RawImage, й мусить бути вище у ієрархії об’єктів сцени. До RawImage додається RenderTexture зі створеним рейдером матеріалом з зображенням теплової карти, як результат обчислення.

Розроблений код шейдеру наданий у Додатку Д. Для розрахунку кольору шейдеру була застосована система вагів. Текстура була розбита на пікселі та кожному з пікселей була надана вага. Дані про вагу пікселей зберігаються в масиві. Розмір текстури теплової карти повинен бути однаковий незалежно від розміру поля. Оскільки при здійсненні розрахунків відомий розмір ігрового полігону, то застосовується фактор масштабування для коректування розміру текстури.

Шейдер містить в собі додаткові змінні які відповідають за діаметр області на текстурі, в якому задається вага для конкретного пікселя та значення

інтенсивності. В загальному вигляді був отриманий двовимірний масив, з позначкою ваги кожної комірки масиву.

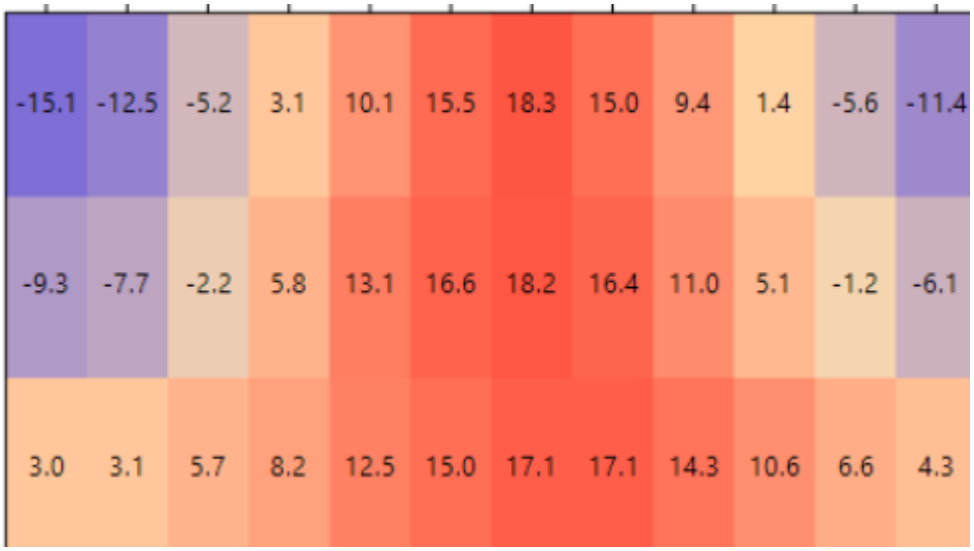

Рисунок 2.15 – Візуальне представлення таблиці ваги комірок

Для розрахунку ваги комірки необхідно сумувати діаметри точок попадання по комірках, при цьому кожна комірка буде накопичувати вагу від кожного попадання.

Таким чином для кожної комірки було прораховано поточну вагу в залежності від дальності попадання.

$$d = w - r \tag{2.23}$$

де d – вага поточної комірки в залежності від дальності удару;

w – максимальна вага;

r – вага в залежності від дальності від центру удару.

Далі розраховується загальна вага комірки цей процес проходить ітеративно для кожного пострілу у залежності від діаметру.

$$T = W + 0.5 * w * i * s * (1 + \sin(t)) \quad (2.24)$$

де T – загальна вага комірки;

W – поточна вага комірки;

w – модифікатор ваги до комірки;

i – значення інтенсивності;

S – сила дії;

t – модифікатор часу.

У залежності від отриманої ваги кожної комірки обирається колір теплової карти у відповідності до шкали. При цьому найінтенсивніше поле буде мати найвище значення – червоне, а найменші значення будуть переходити в синій колір та розсіюватись. Ступінь розсіювання залежить від дальності знаходження від епіцентру попадання удару [32].

2.2.4 Розробка теплової карти ударів на голівці ключки

Подібну ІІС з використанням теплової карт можна використовувати не тільки для аналізу тенденції польоту м’яча в грі емуляторі гольфу, а й для аналізу самої техніки удару, адже від неї залежить напрям, кут й ефективність удару. Під ефективністю удару розуміють співвідношення результату удару до використаних можливостей.

При проведені аналізу удару із застосуванням теплової карти ключки має змогу оцінити яка траєкторія буде у м’яча і як він буде себе вести в процесі польоту. Так за допомогою аналізу удару ключки по м’ячу є змога отримати інформацію кут удару, напрям удару, а також ступінь обертання м’яча в повітрі. При покращені якості удару по м’ячу для гольфа ключкою гравець має змогу покращити дальність траєкторії.

Для розуміння взаємодії точки торкання м’яча для гольфу та траєкторії польоту треба прийняти до уваги кут голівки ключки до лінії удару (кут лофту). Порівняльний аналіз кутів лофту у різних ключках наведений у Додатку Е.

Отже, слід розуміти, що чим більший кут лофту до ігрового полігона, то нижча повинна бути точка торкання до м’ячу, так для прикладу ключка Драйвер з кутом 9 градусів повинна зіштовхнутись з м’ячем серединою голівки, а ключка Пітчінг Ведж, з найбільшим кутом повинна торкнутись м’яча нижньою частиною, оскільки її основна задача вибити м’яч з бункеру набравши найбільшу миттєву висоту.

Відмінність теплової карти ключки від теплової карти ігрової зони відрізняється тим, що кожен тип ключки має різну форму ударної частини. Таким чином необхідно змінити стартові точки початку розрахунків. Доцільно встановити їх у центрі геометрії та починати відлік з цих точок, також немає необхідності розвертати теплову карту.

Для коректного відображення зображення теплової карти, оскільки сама текстура є прямокутною доцільно встановити компонент маску, що буде скривати всі пік селі що знаходяться поза геометрією зображення ключки. Це допоможе зберігати обчислювальні можливості додатку та не виконувати непотрібні калькуляції у сліпій зоні. При цьому слід враховувати що форма голівки ключки однакова навіть у різних виробників такого спортивного знаряддя.

Рисунок 2.16 – Реалізація теплової карти на ключках різного типу

Отримавши результати ударів можна провести та розрахувати додаткову інформацію щодо удару, таку як нахил ключки, кут удару, кут відскоку, ступінь обертання та вісі відскоку м’яча від поверхні ключки та інші дані.

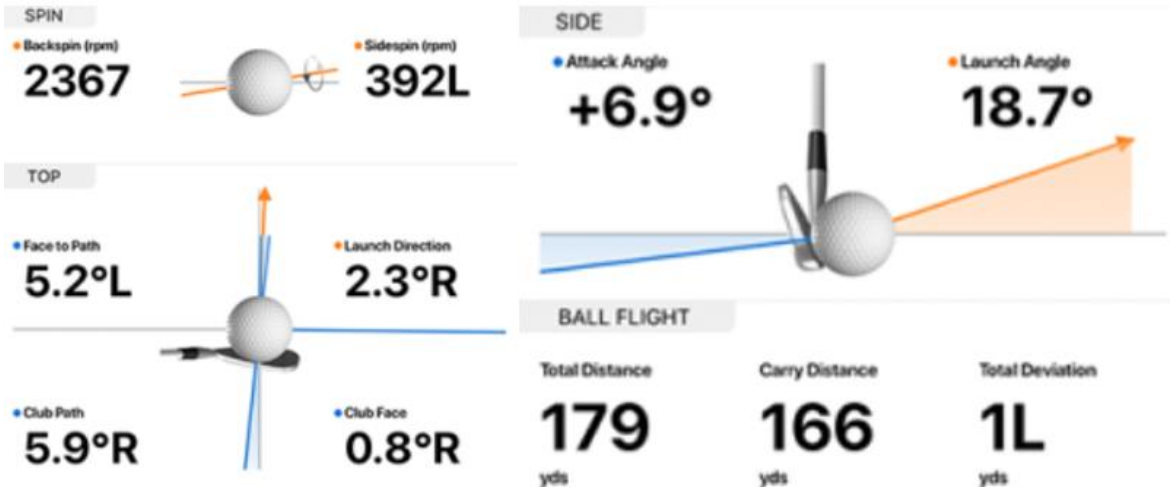

Рисунок 2.17 – Збір та аналіз даних ключки

Розрахунок теплових карт повинен робитись окремо для кожної групи ударів однією з ключок. Аналіз ударів між типами ключок буде неможливий тому що кожна з ключок має різний форм фактор та базові характеристики.

У відповідності до значення теплової карти, та керуючись даними з бази даних можна надати користувачу рекомендації за допомогою експертної системи, який параметр удару необхідно покращити для конкретної ключки, щоб отримати найкраще дальність удару.

Після завершення ігрової сесії у гру емулятор гольфу гравець має змогу проаналізувати як часто він попадав у потрібну точку ключки для удару та проаналізувати залежність траєкторії польоту від точки дотику на обличчі ключки.

2.3 Система рекомендацій ключки

2.3.1 Збір інформації про тенденції траєкторій користувача

Для винесення рекомендацій гравцю стосовно того яку саме ключку доцільно використовувати необхідно отримати від гравця базову інформацію щодо цілі використання ключки гравцем у грі симуляторі гольфу. Так зазвичай можуть бути наступні параметри для винесення рекомендацій стосовно обладнання для гольфу:

- покращення дальності удару;
- оптимізація кута удару у заданих умовах;
- оптимізація точності удару гравця по цілі.

Для покращення дальності удару при зчитуванні даних про удар гравця необхідно звернути увару на такі параметри як кут удару, сила удару, або швидкість ключки, або швидкість м’яча, та обертання м’яча. Такі дані допоможуть розрахувати тип значень які необхідні для досягнення ліпшого результату при рекомендації ключки.

Для прорахування ідеального кута удару необхідно брати до уваги місцевість, тип покриття, та дані щодо ходу ключки, та точки торкання м’яча до голівки ключки, та дальність удару. Саме ці значення дають змогу визначити яку ключку слід обрати в такій ситуації.

Точність удару це комплексний параметр що має враховувати у собі два попередні пункти та загальну тенденцію ударів гравця. У відповідності до тенденції гравця вона може бути відкоригована, при виборі коректної ключки, наприклад з лівої, переведена більш в праву сторону, що вирівняє удар гравця відносно центральної лінії ігрового полігону.

Для отримання базової інформації щодо гри гравця в гру емулятор гольфу є можливість отримати її з бази даних минулих ігор, якщо такі сесії були у попередньому. Якщо гравець не має минулих ігор у базі даних йому має бути

запропоновано провести декілька ігрових сесій з поточною ключкою. Кількість сесій у грі повинна бути не менше двох з мінімальною кількістю ударів по десять ударів на сесію. Альтернативним виходом з цієї ситуації може бути пропозиція гравцю перед утворенням рекомендації провести серію ударів поточною ключкою. Така серія ударів для аналізу не повинна бути менше двадцяти, це дозволить виявити поточну тенденцію гравця з використанням поточного обладнання та перевірити самі характеристики удару.

Оскільки додаток не має можливості зчитати тип обладнання гравця має бути надана можливість внести дані про свою ключку та м’яч. Наразі немає можливості визначити дані лише по бренду та моделі ключки, адже існує багато брендів не стандартизованих всесвітньою асоціацією з гольфу, тобто гравець має змогу обрати ключку із стандартних наданих, або ввести свої параметри самостійно.

Для вибору стандартної ключки для гри симулятора у гольф гравець має ввести бренд, модель, лофт ключки – тобто нахил голівки ключки до лінії удару, шафт ключки – тобто нахил ручки - держака ключки відносно поверхні землі.

У разі якщо ключка має не стандартні характеристики, або бренд відсутній у базі даних, то гравець має ввести дані вручну, а саме лофт від мінімального, що доступний у базі даних до максимального лофту та шафт із запропонованих варіантів.

Надані гравцем дані можуть бути розширені для уточнення роботи алгоритму та надання біль детальних рекомендацій гравцям у майбутньому. Для визначення логіки роботи алгоритму важливо щоб гравець надав інформацію стосовно того яка рука у нього ведуча при грі у емулятор гольфу, яка у нього поточна тенденція гри, і яка бажана, яка в нього траєкторія польоту м’яча и яка бажана. Тенденція – це множина можливостей, що можуть розвиватись у певному напрямку при притримані певних умов. Траєкторію можна поділити на високу, середню та низьку, тенденцію удару по ступеню відхилення від центральної лінії,

тобто пряму ліву чи праву. Траєкторія – це уявна лінія, що проведена від точки удару, що рухається у просторі та закінчується на точці торкання м’яча до ігрового поля. Крім цих даних слід також отримати від гравця дані щодо основної цілі гравця, тобто необхідно отримати інформацію про намір покращити дальність, кут, точність чи обертання м’яча. Слід також отримати інформацію щодо типу м’яча адже вони мають різну вару, та матеріал, що може змінити модель відскоку м’яча від землі та модель кочення м’яча після падіння. Таке опитування організоване у якості меню налаштування з списками, де гравець в межах одного екрану може обрати всі необхідні налаштування для гри емулятора гольфу та алгоритму рекомендацій.

При цьому слід також враховувати, що гравцем можуть бути надані не точні або не коректні дані, тому більше розрахунків слід робити саме на дані з ігрових сесій гравця. При цьому чим більше ігрових сесій буде проведено гравцем, тим більш точними будуть дані щодо тенденції гравця.

Дані гравця можуть бути завантажені на сервер та бути використані як шаблони ударів для інших калькуляцій інформаційних інтелектуальних систем, які можуть бути впроваджені на стороні серверу. До таких систем можуть відноситись як системи створення ботів для гри так і системи щодо рекомендацій, тобто знаючи вхідні параметри й тип обладнання додаток може вирахувати як буде рухатись м’яч та якою буде траєкторія за аналогією з існуючими ударами.

Після зберігання даних вони можуть зберігатись як на стороні серверу так і на стороні клієнта у грі емуляторі гольфу. Гравцю надається можливість завантажити будь яку зіграну сесію для аналізу даних або проведення порівняльного аналізу з поточною сесією.

2.3.2 Створення алгоритму рекомендацій ключок для гри емулятору гольфу

Створення алгоритму рекомендації ключок основане на зібраній інформації щодо ігрових сесій гравця, та вхідних даних стосовно типу рекомендації, яка необхідна.

Алгоритм рекомендації – це складна система математичних алгоритмів, що надає користувачу інформацію-рекомендацію, щодо запиту, у відповідності із задачами які виконуються такою інтелектуальною інформаційною системою [33].

Алгоритм рекомендацій може складатись з менших алгоритмів, у яких кожен шар може виконувати свою логіку, або навпаки кожен шар може конкретизувати рекомендацію від загального до часткового, у рамках дедуктивного методу. Дедукція – це логічний висновок, при якому інформація за загальними даними конкретизується до більш часткових методів [34].

При створені рекомендаційної системи для створення рекомендацій щодо ключок може бути застосовано декілька типів таких інтелектуальних інформаційних систем.

Однією з таких систем може бути експертна рекомендаційна система. Експертна система – це інтелектуальна комп’ютерна система, яка має на меті частково замінити спеціаліста або особу що є експертом у певній галузі знань для вирішення поставленої задачі [35]. Сучасні експертні системи з’явились з 1970 років. Перші інтелектуальні машини – експертні системи, які використовувались у практиці були створені Корсаковим С.Н. у 1983 році. Такі машини надавали змогу знайти рішення по деякому спектру питань [36]. Для прикладу може бути наведена експертна система яка може надати рекомендацію щодо необхідних препаратів – ліків, по заданому списку симптомів пацієнта. Для інформаційних технологій експертні системи можуть бути представлені як бази знань, як модель поведінки експертів в певних знаннях, з використанням логічних висновків та

умов та правил прийняття того чи іншого рішення. При цьому база знань представляє собою збірку фактів і правил, за допомогою яких приймається логічний висновок в заданій площині знань.

Так для гри емулятора гольфу може бути використана база знань та написані правила щодо вибору ключки у різних ситуаціях. Так при необхідності обрати ключку яка необхідна для дальнього удару будуть рекомендуватись більш важкі ключки типу драйвер для нанесення максимально сильного удару, а три необхідності збільшити чи зменшити кут можуть рекомендуватись ключки зі збільшеним чи зменшеним кутом лофту, що призведе до зміни траєкторії польоту м’яча.

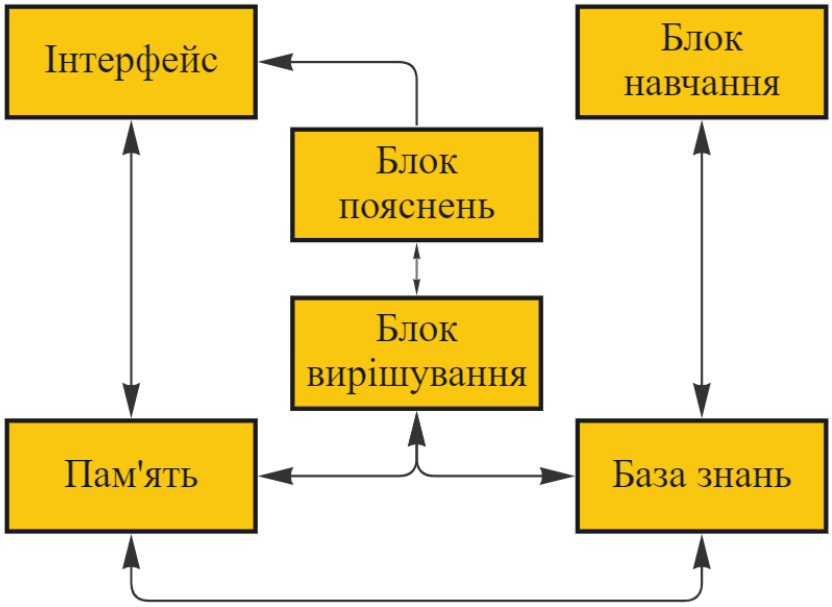

Рисунок 2.18 – Загальна схема рекомендаційної експертної системи

Рекомендаційна експертна система надає інформацію у якості рекомендації, вона не несе обов’язковий характер. Такі рекомендації не завжди можуть бути точними, адже система повинна знати не тільки типи ключок, а й точні характеристики кожної з них у відповідності до різних брендів, адже відсутні

чіткі стандарти щодо характеристик ключок до певного типу. Виходячи з цього є необхідність створити базу даних яка вміщує в себе характеристик ключок, такі як нахил голівки відносно поля, нахил держака відносно поля, загальна вага ключки, вага голівки, довжина та інше.

Ще одним з методів створення рекомендацій для комп’ютерної гри емулятора гольфу є створення математичної системи, що може базуватись лише на характеристиках удару гравця та виходячи з наданих системі даних рекомендувати ту чи іншу ключку в залежності від результатів обчислення. Така система є більш гнучкою та вона має можливість швидко розвиватись за рахунок отримання нових даних про удари у свою базу. Така інтелектуальна інформаційна система має містити в собі 2 бази даних, а саме базу ключок та базу ударів кожною з ключок.

База даних ударів ключок містить в собі удари кожною ключкою із зміною певних параметрів у вигляді інкременту кожного з параметрів окремо. Наразі часто використовуються роботи для створення бази даних таких ударів, тобто робот програмується, наприклад зробити удари із заданою характеристикою, але при кожному ударі кут нахилу ключки буде збільшено на заданий крок. Так отримано базу даних ударів з певними параметрами, при цьому кожен з параметрів може змінювати своє значення.

Рисунок 2.19 – Робот для створення ударів у гольфі

Такий робот розроблений компанією Golf Laboratories, та створює суцільну базу даних різних ударів з різними вхідними даними.

Використання такої бази даних надає змогу робити вибірки будь яких параметричних значень удару при наявності частини відомих даних, що дозволяє також знайти вибірки даних для будь яких невідомих параметрів. Така система може не тільки вказати на рекомендовані параметри, а й надати інформацію про не досконалі параметри, які гравцю слід покращити у емуляторі гри у гольф.

Доцільно використовувати систему математичного розрахунку та експертну систему рекомендацій у поєднанні для надання гравцю найбільш точних даних про удар гравця.

Для раціонального використання ресурсів клієнтського додатку краще зберігати базу даних ударів на стороні сервера та у якості вхідних параметрів отримувати інформацію про гравця, при цьому логіку обчислень слід перенести на серверну частину, за таких умов виробничі ресурси пристрою користувача будуть звільнені від додаткового навантаження та будуть задіяні під час рендерінгу сцен гри емулятора у гольф. При цьому у розробника залишається можливість корегувати та поповнювати базу даних новими даними – навчати систему без додаткової необхідності оновлювати додаток користувача. Також за розробником залишається можливість розширювати й покращувати алгоритм рекомендацій на стороні серверу без потреби випуску патчей для клієнта, а дані про удари клієнта можуть бути використані для поповнення бази даних ударів та корегування рекомендаційного алгоритму. Такий розподіл задач називається клієнт серверною архітектурою. Під клієнт-серверною архітектурою можна розуміти таку побудову додатку, при якому процеси розподілені між різними пристроями і кожен з пристроїв виконує свої задачі у мережі й не залежить від інших пристроїв. Взаємодія в між елементами клієнт-серверної архітектури відбувається за допомогою відправки запитів і отримання відповідей між

елементами мережі. Найпростіша структура клієнт-серверного додатку включає в себе сервер додатків, базу даних та пристрої користувачів.

Для розрахунку базової тенденції удару гравця необхідно взяти ігрові сесії, що аналізуються, взяти масив ударів та взяти показники удару які робив гравець у грі емуляторі гольфу. Так для розрахунку необхідно взяти швидкість удару, кут підйому, напрям удару – азимут, обертання та інші параметри удару. Для знаходження середнього значення по цим даним їх потрібно просумувати та розділити на кількість ударів, що аналізуються.

$$d_c = \frac{d_1 - d_2 \ldots - d_n}{n} \tag{2.25}$$

де d_c – середнє значення показника удару;

d_n – дані n-ного удару;

n – кількість ударів, що аналізуються.

Після розрахунку даних, що відповідають середньому значенню вибірки було побудовано траєкторію польоту за такими даними за допомогою симулятору, що в результаті покаже середню тенденцію гравця у даній виборці.

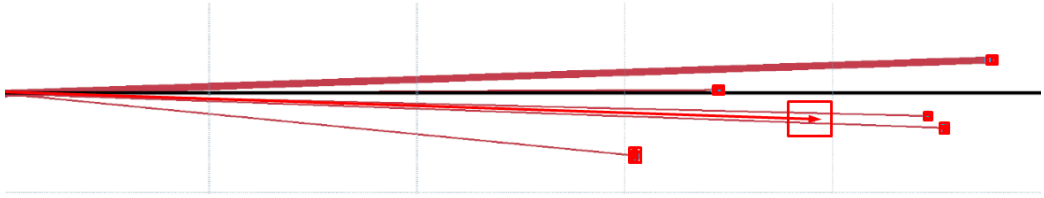

Рисунок 2.20 – Розрахунок середньої тенденції гравця

Для більш точного розрахунку краще взяти декілька вибірок ударів, та визначити середнє значення для кожної з них, в результаті отримані дані будуть представляти собою матрицю даних про, що свідчить про тенденцію ударів гравця.

Всі удари, що аналізуються, мають бути виконані однією ключкою в однакових умовах, без зміни базових параметрів дальності чи оточення.

Прораховані дані є вхідними параметрами для алгоритму надання рекомендацій гравцю. Вхідні параметри – це зовнішні параметри, що надаються у функцію для передання інформації про зовнішній стан об’єкту та впливають на стан такого об’єкту. Параметр – це узагальнена назва певної фізичної, геометричної чи іншої властивості пристрою (процесу). Нормоване значення параметра – це теоретичне значення, яке характеризує ознаки моделі. Дійсне значення параметра – значення параметра, знайдене експериментальним шляхом і настільки наближене до істинного, що його можна використати замість істинного значення для поставленої задачі. У випадку гри емулятору гольфу параметрами є дані удару, а саме швидкість удару, кут підйому, напрям удару, обертання м’яча та характеристики ігрового полігону.

Для отримання оптимального значення удару необхідно зчитати вибір гравця з панелі вводу, так можуть бути обрані будь які параметри для покращення при зміні ключки.

Для рекомендації конкретної ключки на стороні алгоритму необхідно зробити вибірку ударів, що мають такі самі параметри, що були задані гравцем та зробити вибірку всіх ключок з такими параметрами де найбільше значення є значення яке ми шукаємо. Наприклад при аналізі дальності удару необхідно зробити вибірку даних з заданими параметрами де дальність найбільша. Тобто при нанесені аналогічного удару відібраною ключкою дальність буде покращена.

При цьому слід враховувати що кожен параметр при пошуку має свої властивості, так для обрання найкращого кута удару слід зробити вибірку за заданими параметрами та визначити ті кути при яких дальність удару була найбільшою. В такому випадку ключки, що будуть відібрані для оптимізації кута удару не будуть такими самими де параметром для оптимізації виступає лише дальність удару.

При створенні вибірок для обертання м’яча будуть надані ключки, що більш підходять за форм фактором, при цьому буде використано для розрахунку кут удару та вид траєкторії польоту м’яча.

Висновки до розділу 2

Гра емулятор гольфу є складною системою з великою кількістю даних які отримуються від користувача за допомогою засобів вводу, або контролерів. Отримані дані можна аналізувати та збирати для подальшого використання у інтегрованих інтелектуальних системах. Було визначено, що такі дані можна використовувати для багатьох інтелектуальних інформаційних систем, таких як теплові карти на ігровому полігоні та на голівці ключки, а також шляхом аналізу даних можна провести аналіз тенденцій гравця та надати йому рекомендацію стосовно того яке обладнання краще використовувати для поліпшення ігрових параметрів.

Теплова карта ігрової зони допомагає визначити патерн гри гравця для подальшого корегування. Такий інструмент дає змогу надати візуальну картину типової тенденції гравця у рамках ігрової сесії, де відображаються місця на полігоні по яким гравець попадає частіше за все. При цьому може бути доданий параметр часу, який може нашаровувати теплові карти у часовому простору, що може показати динаміку певної тенденції ударів.

Теплова карта також надає змогу визначити як був здійснений удар з боку взаємодії м’яча і ключки, що може надати гравцю знання щодо якості удару, та правильності використання ключки. Такі теплові карти доцільно використовувати для груп ударів. Цей інструмент надає змогу контролювати кут удару гравця та обертання, а азимут у грі емуляторі гольфу та надає можливість бачити тенденцію ударів певною ключкою та виправити їх.

Важливою інформаційною системою, що інтегрована до гри емулятору гольфу є компонент рекомендацій, що надає рекомендації гравцю, стосовно того яке обладнання для гольфу краще використовувати при певних тенденціях гри. Такий компонент працює за допомогою математичних обчислень та експертної системи, які навчаються в процесі надання рекомендацій гравцю. Для висунення рекомендацій гравцю необхідно мати декілька ігрових сесій для аналізу з яких буде розрахована поточна тенденція.

В гру емулятор гольфу можливо додати інші ІІС які допоможуть гравцю оптимізувати показники гри та оптимізувати тенденцію гри.

3 РОЗРОБЛЕННЯ АРХІТЕКТУРИ ПРОЄКТУ СИСТЕМИ-ЕМУЛЯТОРА ДЛЯ КОМП'ЮТЕРНОЇ ГРИ ГОЛЬФ

3.1 Впровадження основних логічних модулів гри-емулятора гольфу та побудова загальної архітектури проєкту

Для функціонування та розширення додатку гри емулятору гольфу та розширення потреб у залежності від потреб користувачів додаток повинен мати гнучку архітектуру та систему взаємодії компонентів додатку та його ключових складових.

Під архітектурою додатку розуміють – комплекс взаємодії між компонентами системи, базами даних, проміжними та допоміжними системами, інтерфейсами користувача та серверними частинами, тобто це логічно визначені зв'язки між компонентами системи, його атомарними частинами, та частинами інфраструктури комп'ютерної системи.

Архітектура додатків – це структурний принцип, за яким створено додаток. Кожному архітектурному виду відповідають свої характеристики, властивості та відносини між компонентами. Компонент – це проста невелика логічна та незалежна частина додатків архітектурної системи. Такі компоненти можуть бути малими та великими логічними формуваннями.

Добре розроблена архітектура додатків може справлятися з різними навантаженнями й адаптуватись до зміни потреб користувача та розробника, забезпечуючи швидку взаємодію з клієнтами. Це підвищує її продуктивність і гнучкість.

Архітектура додатку повинна відповідати наступним критеріям:

– ефективність додатку: програма повинна виконувати поставлені завдання та функції у будь-яких умовах, при цьому система повинна бути продуктивна, надійна та справляється з усіма навантаженнями;

- гнучкість системи має надавати змогу легко змінювати рішення без появи додаткових помилок на різних рівнях абстракції, при цьому зміна чи заміна одного елементу не повинна шкодити іншим логічним елементам системи;
- критерій розширюваність додатка має надавати змогу здійснювати додавання логічних частин як модулів або компонентів;
- масштабованість допомагає зменшити час на розробку та доповнення різних компонентів, що не залежать один від одного, а також дозволяє вести розробку компонентів незалежно один від одного;
- критерій тестування додатку дає змогу додати юніт тести до окремих контролерів, без утворення додаткових зв’язків у додатку, зменшується кількість помилок і збільшується його надійність;
- елементи додатку можна використовувати в різних його частинах, для цього треба використовувати чисті функції та залишати логіку методів на атомарному рівні. Чиста функція - це детермінована функція, яка не справляє побічних ефектів;
- зрозумілість коду, тобто код має бути чистим й читабельним, що дозволяє легко зрозуміти його логіку. У коді мають бути присутні коментарі та пояснення.

Проєктування додатків може проходити трьома способами, залежно від завдань проєкту – це монолітний, модульний, сервіс-орієнтований підхід.

Монолітний підхід до проєктування додатку є одним з найдавніших підходів, в якому немає складних систем. На сервері зберігається необхідна логіка, а в базі – вся потрібна інформація для сервера. Такі додатки дуже прості і вимагають порівняно мало часу на розробку. Але є істотні мінуси, із-за яких цей підхід майже не застосовується. У довгостроковій перспективі програми обов’язково змінюються, оскільки вони повинні відповідати новим платформам, технологіям і операційним системам. У ході життєвого циклу будь-які програми

змінюються. Невеликий функціонал доповнюється новими елементами логіки. Тому монолітна система важка в розробці і не відповідає критеріям гнучкості для забезпечення інтегрування нових інтелектуальних інформаційних систем.

Модульна архітектура являє собою логічну структуру коли додаток розділяється на модулі, кожен з яких відповідає за одну функцію, або логічну систему. Модулі є незалежними елементами системи. При цьому якщо один з модулів дає збій інші продовжують працювати в відокремленому режимі. Це спрощує відстеження помилок у проєкті. Прикладом подібної архітектури є PHP-фреймворки, платформи, на основі яких розробляються веб-додатки. Модулі дають можливість створювати досить складні додатки, які мають багатошарову логіку.

Сервіс-орієнтований підхід є продовженням модульного підходу. З ускладненням додатків деякі модулі виносяться на окремі апаратні частини та сервіси. Модулі тут іноді тримають власні бази даних і розміщуються на окремих пристроях. В цьому є свої плюси і мінуси. Сервіси можна писати на різних мовах програмування та з застосуванням різних технологій, і їх взаємодія налаштовується через інтерфейс між елементами архітектури. Зразок подібних модулів – сервіси електронної пошти вшиті в мобільні додатки або веб-додатки.

Також існують й інші види архітектурних підходів залежно від розподілення модулів по різним обчислювальний середовищам, таким як хмарні чи клієнт-серверні додатки.

Хмарна архітектура – це спосіб поєднання технологічних компонентів для створення хмари, в якому ресурси об’єднуються за допомогою технології віртуалізації та спільно використовуються у мережі. Хмарна архітектура складається з компонентів таких як зовнішня платформа, тобто клієнт або пристрій, що використовується для доступу до хмари, внутрішня платформа, що представлена сервером або сховищем і хмарна модель надання даних додатків з мережевою взаємодією.

Клієнт-серверна архітектура, або клієнт-сервер, або модель Клієнт-Сервер передбачає поділ процесів надання послуг та надсилання запитів на них на різних комп’ютерах у мережі, при цьому кожен з яких виконує свої завдання незалежно від інших. В клієнт серверній архітектурі кілька комп’ютерів-клієнтів, що є віддаленими системами надсилають запити та отримують послуги від централізованої службової машини, тобто серверу, яка також може називатися хост-системою.

Для гри емулятору гольфу можна було обрати монолітну архітектуру у разі залишення її лише з розважальним контентом, без інтегрування інтелектуальних інформаційних систем. При цьому внутрішню структуру додатку можна було розбити на декілька шарів логіки, що б відповідали за дані, за візуалізацію ігрового процесу та за прошарок обчислення з контролерами [37].

Для розробки додатку гри емулятору гольфу, який включає в себе інтелектуальні інформаційні системи, було впроваджено клієнт-серверну архітектуру де шар розрахунків інтелектуальних систем та база даних винесені на сторону серверу. На клієнті доцільно залишити лише локальні обчислення та зберігання локальних даних.

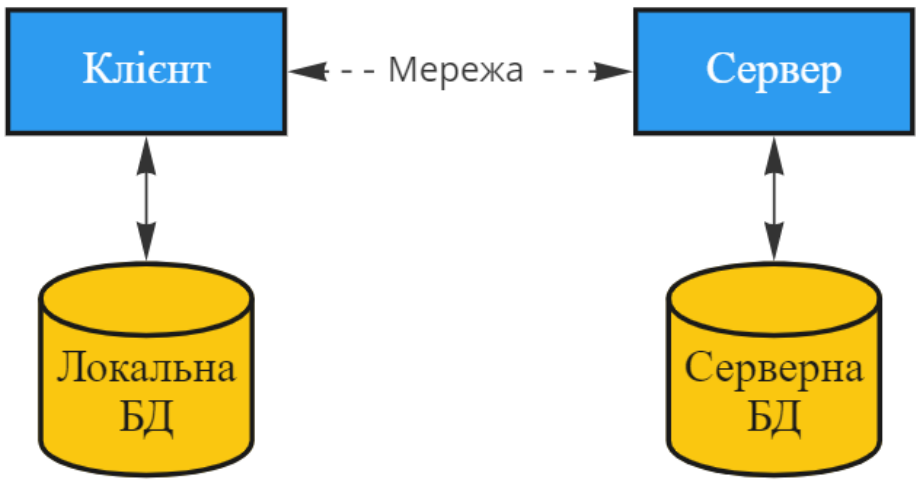

Рисунок 3.1 – Схема клієнт-серверної архітектури додатку

Доцільно існування бази даних на стороні клієнту та на стороні серверу. Такий розподіл обумовлений тим, що різні інтелектуальні інформаційні системи мусять використовувати різні пакети даних. Так на стороні серверу доцільно зберігати таблиці з ключками та тестовими ударами робота, а також деякі завантажені сесії гравця. На стороні клієнту необхідно зберігати сесії, групи і удари гравця при грі у емулятор гри у гольф. Таких даних достатньо щоб обробляти візуальні дані та надавати їх для роботи локальних інтелектуальних інформаційних систем.

Слід відмітити, що для подальшого розвитку системи гри емулятора у гольф можна застосувати хмарний підхід де кожен елемент інтелектуальної системи буде виступати окремим сервісом, який обробляє лише ті запроси, які потребують конкретної логіки. Це дозволить розвантажити навантаження на серверну частину, та зробити систему більш стійкою до збоїв, оскільки самі сервіси не будуть впливати на роботу один одного и будуть виконувати свої обчислення в ізольованій інфраструктурі.

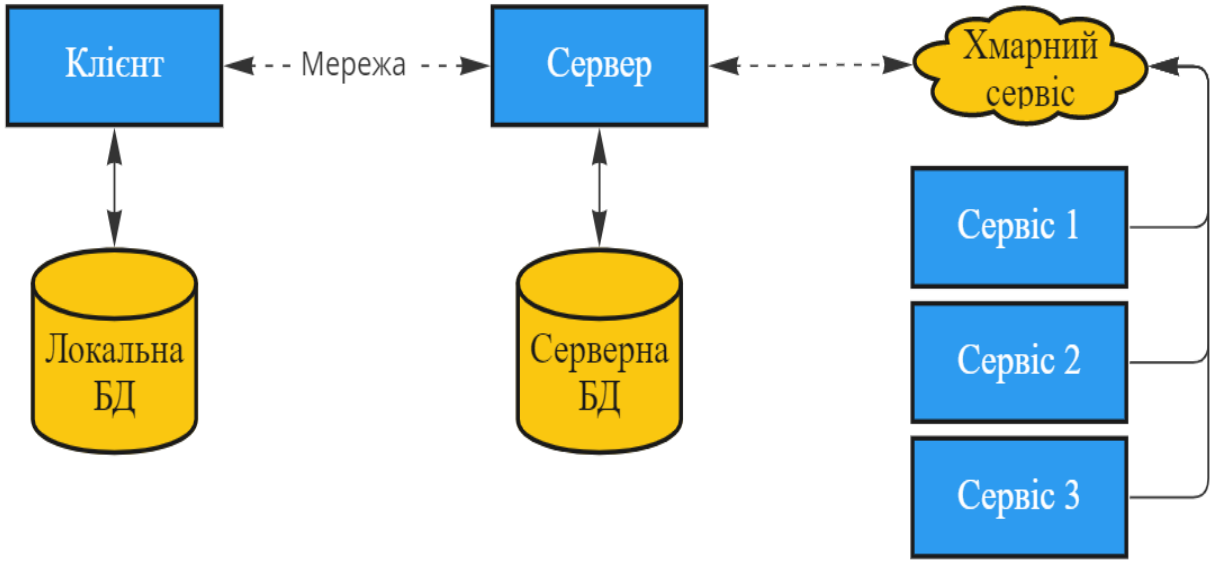

Рисунок 3.2 – Схема хмарної структури додатку

Взаємодія клієнта з сервером провадиться за допомогою веб-запитів. Сервер самостійно взаємодіє з хмарними сервісами й робить до них запит й повертає оброблене значення клієнту без додаткового впливу на самі обчислення. В такому випадку сервер виступає лише як шлюз для запитів та аутентифікації користувача у системі гри емулятора гольфу.

3.2 Інтегрування елементів інтелектуальних систем та сервісно-орієнтовної архітектури у гру-емулятор гольф

На рівні клієнту застосовані архітектурні шаблони програмного забезпечення для створення більшої читабельності та зрозумілості коду, а також для подальшого його розширення. Під архітектурними шаблонами програмного забезпечення розуміється сбір практик, що застосовуються для вирішення архітектурних проблем проєкту, такі шаблони створюють певний фундамент взаємодії елементів системи на різних рівнях. Така схема взаємодії складається із менших підсистем, що також можуть містити в собі інші шаблони проєктування та визначає їх схеми відповідальності та взаємодії.

Архітектурний шаблон – це загальне і повторюване вирішення проблеми архітектури додатків, що часто виникають, в межах заданого контексту. Архітектурні шаблони схожі на шаблони програмного дизайну, однак мають ширше охоплення.

Існують декілька типів пат тернів за логікою їх поведінки й призначенням, а саме, породжуючі патерни, структурні патерни та поведінкові патерни. Породжуючі патерни відповідають за порядок та логіку створення об’єктів. Структурні патерни вибудовують різні способи побудови зв’язків між об’єктами логічної системи. Поведінкові патерни – налагоджують комунікацію між об’єктами та їх основну взаємодію. Для створення гри емулятора гольфу доцільне використання всіх типів задернів для створення гнучкої та ефективної системи.

Для гри емулятору гольфу були використані різні типи архітектурних пат тернів на різних рівнях. На верхньому рівні використовується архітектурний шаблон MVC – Model-View-Controller [38]. Модель-вид-контролер являє собою архітектурний патерн, що поділяє систему на при основні шари, а саме на модель – тобто дані. MVC являє собою об’єктну модель певної предметної області, яка включає дані та методи роботи з цими даними. Модель реагує на запити з контролера, повертаючи дані та/або змінюючи свій стан. При цьому модель не містить у собі інформації про способи візуалізації даних або формати їх подання, а також не взаємодіє з користувачем безпосередньо.

Вид – частина інтерфейсу користувача та всі елементи візуалізації. Вид відповідає за відображення інформації (візуалізації). Одні й самі дані можуть представлятися різними способами й у різних форматах. Прикладом може виступати колекція об'єктів за допомогою різних уявлень можна представити на рівні інтерфейсу користувача як в таблицю, теплову карту, траєкторію польоту м’яча, так і списком; на рівні API можна експортувати дані як у JSON, так і в XML або XSLX.

Контролер – системи, що обробляють дані користувача та взаємодіють з моделлю, що далі передає результат обробки користувачу. Контролер забезпечує зв'язок між користувачем та системою, використовує модель та подання для реалізації необхідної реакції на дії користувача. Як правило, на рівні контролера здійснюється фільтрація отриманих даних та авторизація – перевіряються права користувача на виконання дій або отримання інформації.

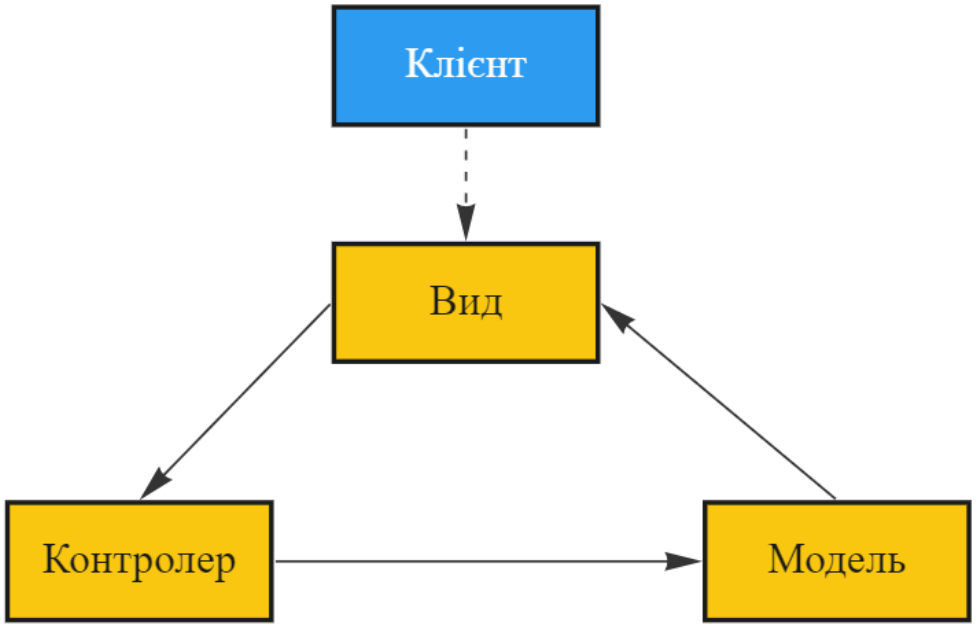

Рисунок 3.3 – Схема патерну модель-вид-контролер

Згідно із схемою користувач взаємодіє з інтерфейсом додатку, а саме здійснює удари, проводить інші ігрові дії та інше, ці дані викликають реакцію контролерів, що здійснюють розрахунки та дії, що необхідні у відповідності до запиту користувача. Такі дії контролеру призводять до зміни у моделі додатку гри емулятора гольфу. При зміні даних моделі викликається подія, що інформує вид про зміну даних після чого відбувається оновлення візуальної частини гравця.

При застосуванні подібного патерну кожен із процесів відповідає своєму шару логіки, що полегшує оновлення та налагодження коду та допомагає у розширені додатку.

Оскільки основною задачею проєкту емулятору гольфу є впровадження додаткових інтелектуальних інформаційних систем для забезпечення гравця інформацією щодо якості його дій у грі та надані аналітичних даних та створення інструментів для аналізу, то стало необхідним створення модульної структури, де існує додаток, як система функціональних систем і модулі, що не мають зв’язків з основним додатком, а лише реагують на нього і викликають додаткові події для

використання основних функцій додатку. Це дозволить підключати додаткові інтелектуальні системи до проєкти без зміни бази коду самого додатку. Для впровадження такої системи необхідно зменшити зв’язаність проєкту.

Для зменшення зв’язаності проєкту доцільне використання DI – Dependency Injection, тобто впровадження залежностей, це такий шаблон проєктування, що передає відповідальність надання залежностей сторонньому компоненту – контейнеру, який використовує IoC – Inversion of Control для того щоб розв’язати залежність об’єкту. Часто такі дії виконуються за допомогою рефлексії. Як правило залежність передається одним із трьох способів, а саме через конструктор, властивість об’єкту чи через метод класу. Інверсія залежностей передбачає собою залежність класів не від безпосередніх реалізацій, а від абстракцій, тобто згідно даного принципу об’єктно-орієнтовного програмування об’єкт класу працює не з конкретною реалізацією, а з абстрактною сутністю, через абстрактний клас або через інтерфейс. Даний принцип є частиною принципів об’єктно-орієнтовного програмування SOLID - single responsibility, open–closed, Liskov substitution, interface segregation и dependency inversion.

Дотримання таких принципів та використання патернів надало змогу впровадити модульну структуру при якій окремий компонент інтелектуальної інформаційної системи є незалежною частиною та може додаватись до проєкту не залежно від основної частини проєкту.

Отже, було виділено такі основні елементи системи – гри емулятора гольфу:

- основний модуль візуалізації. Даний модуль працює у зв’язці з Unity3D та відповідає за рендерінг основного поля, інтерфейсів та додаткових компонентів, інформація про які надходить з додаткових модулів;
- модуль теплових карт, що включає в себе шейдер та контролер що опрацьовує удари гравця, передає їх в шейдер. Результат роботи шейдеру публікується на матеріал, що відображається на інтерфейсі гравця;

- математичний модуль, що розраховує траєкторію польоту м’яча та розрахунок всіх параметрів удару, щодо ключки, м’яча та групи з сесією. Даний модуль отримує сирі дані про удар та повертає колекцію – словник, де вміщується тип параметра та його результат;
- модуль рекомендацій ключок гравця, який включає в себе частину збору даних гравця та частину модуль аналізатор, що аналізує дані гравця та надає рекомендації;
- модуль взаємодії з гравцем та засобами введення, до таких засобів можна віднести контролери, монітори запуску, джойстики та інші пристрої, за допомогою яких гравець має змогу взаємодіяти з додатком;
- модуль взаємодії з мережею оснований на RestApi для відправки сесій на сервер для зберігання даних гравця та подальшого впровадження хмарних розрахунків інтелектуальних інформаційних систем;
- модуль взаємодії з даними відповідає за частину моделі, яка опрацьовується за допомогою модулів взаємодії з локальною базою даних та Json конвертерами, які переводять модель даних у вигляді класу у строку у відповідному форматі. JSON (JavaScript Object Notation) — це легкий формат обміну даними, який легко читається й аналізується, генерується. Цей стандарт базується на підмножині стандарту мови програмування JavaScript ECMA-262 3-ї версії Edition – від грудень 1999 р. JSON – представляє собою текстовий формат, який повністю не залежить від мови, але використовує умовності властиві багатьом мовам програмування;
- модуль геймплею відповідає за порядок ударів гравців та базову калькуляцію рахунку гравця, його рівня та прогресу.

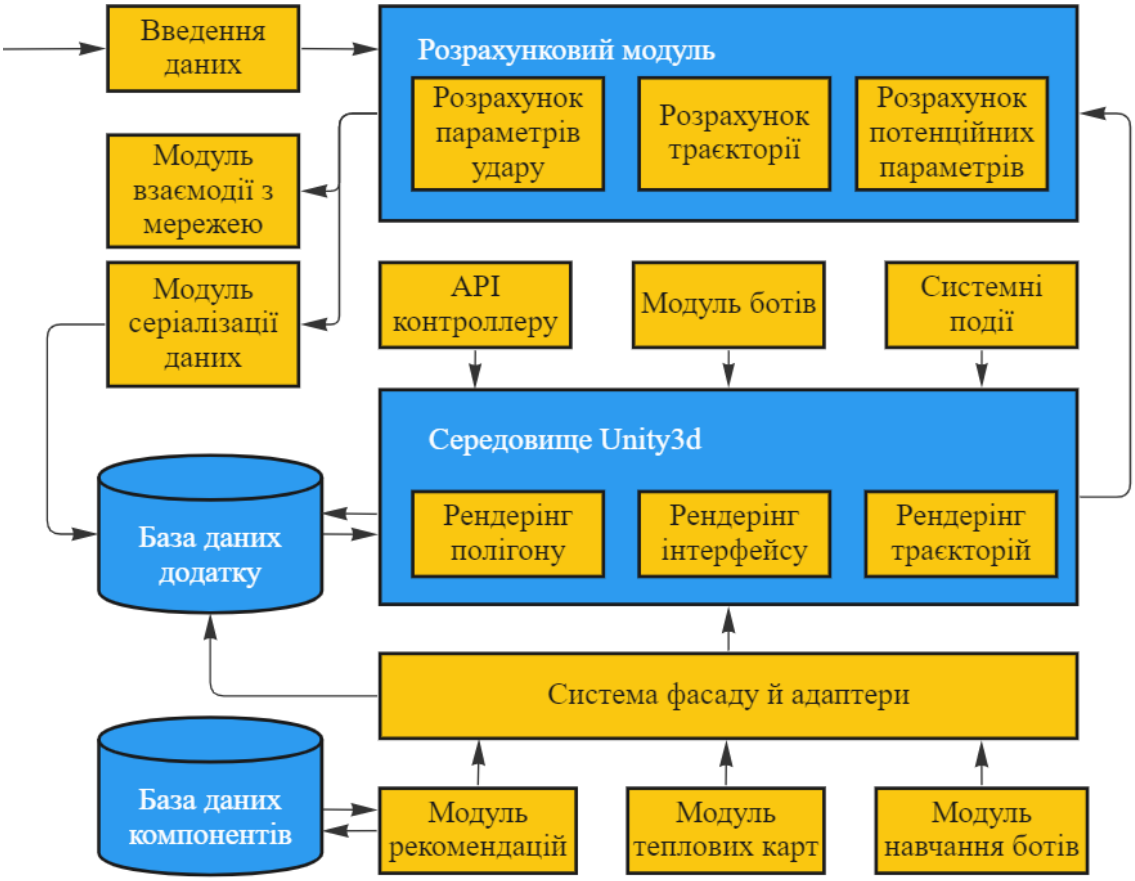

Рисунок 3.4 – Базова схема архітектури додатку

Кожен з модулів має включені в себе більш маленькі елементи, що включаються в нього та мають взаємодіяти лише з даним модулем. Якщо модуль потребує додаткових зв’язків з іншими підсистемами доцільно винести такий елемент у якості окремої логічної структури, що буде додаватись завдяки ін’єкції контейнеру до безпосереднього класу, що потребує такої зв’язаності. Кожен модуль, що не включає бібліотеки Unity3D винесено у окрему dll бібліотеку, та додано Unit-тести для підвищення якості та тестованості коду. Модульне тестування, або юніт-тестування (unit testing) - це процес у програмуванні, що дозволяє перевірити на коректність окремі модулі вихідного коду програми. Для впровадження такого тестування доцільне використання бібліотеки NUnit, яка дозволяє проводити тестування окремих елементів C# коду та тестування

елементів навіть під час виконання додатку для тестування компонентів Unity3D, які працюють та ініціалізуються середовищем Unity3D в основному потоці додатку. Unity3D не має змоги провадити розрахунки в інших потоках окрім основного, тому розбиття на окремі бібліотеки є доцільним, адже не буде обмеження у використанні потоків. Тобто у такому випадку програма містить два чи більше потоків, які можуть виконуватись одночасно і синхронізуватись під час рендерінгу або виконанні спеціальних функцій Unity3D.

3.3 Програмна реалізація та тестування гри-емулятора у гольф

3.3.1 Розробка системи проєкту та інтерфейсу основних систем гри емулятора гольфу

Для забезпечення функціонування проєкту було розроблено систему інтерфейсів для взаємодії користувача з системою та елементами компонентів, що підключені до системи.

Було визначено що проєкт має містити:

- екран основного ігрового процесу;
- панель відображення характеристик удару;
- дерево сесії, тобто дані що включають найменування сесії, групи ударів та самі удари;
- екран аналізу удару ключкою для відображення рекомендацій щодо зміни патерну удару;
- меню налаштувань ігрового полігону;
- екран рекомендацій ключки.

Для взаємодії користувача було створено відповідні панелі для користування додатком та для навігації між його функціональними частинами.

Рисунок 3.5 – Інтерфейс ігрового екрану

У вікні ігрового процесу гравець робить удари та налагоджує систему для відповідного типу гри та аналізу. При імплементації інтерфейсу були застосовані принципи UI/UX дизайну. UI – user interface, це інтерфейс користувача, тобто сама візуальна частина проєкту, яка взаємодіє з системою візуалізації ігрового рушія Unity3D. UX – user experience, це процес налагодження інтерфейсу для його оптимального використання під час роботи з додатком.

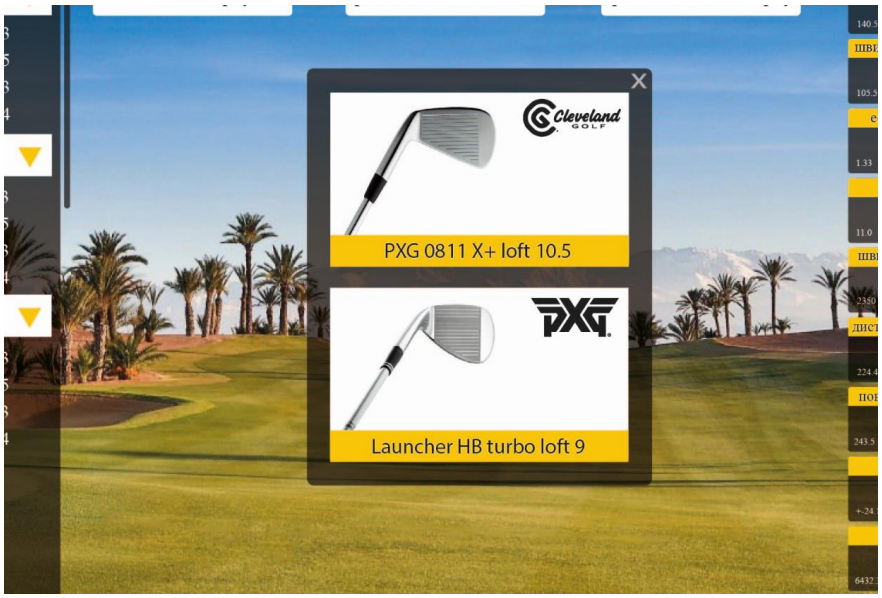

Рисунок 3.6 – Інтерфейс рекомендації ключки

Вікно рекомендації ключки включає в себе список рекомендованих ключок, їх бренд, модель, та кут нахилу голівки ключки. Дана інформація представлена у вигляді скролу у разі надання більшої кількості рекомендацій рекомендаційною системою.

Вікно аналізу ударів включає в себе зображення ключок у чотирьох розворотах для надання базової інформації про удар гравця, а також для представлення інформації щодо покращення тенденції удару. Гравець має змогу змінити кут камери на полігон, та перевести його у двовимірне представлення для спрощення аналізу даних.

3.3.2 Інтеграція інтелектуальних компонентів в проєкт емулятор гольфу

При розробці Інтелектуальних інформаційних систем було розроблено API яке може бути впроваджене до проєкту який має відповідні інтерфейси та готовий опрацьовувати події, згенеровані відповідним компонентом. Для взаємодії з такими компонентами були створені методи для базової взаємодії з будь яким компонентом розумним компонентом, розробленим для даної еко-системи додатків. Тобто такий підхід дозволяє робити будь-яку збірку додатку.

При цьому додаток взаємодіє з системою за допомогою виклику уніфікованих методів, що викликають певну логіку компоненту. До таких методів відносяться методи передачі даних до компоненту, метод передачі налаштувань компонента, метод виклику запуску операцій аналізу.

При цьому компонент надає додатку набір подій, що викликаються при роботі компоненту. Підписка клієнта на події відбувається за допомогою рефлексії. Рефлексія – у філософії це метод самопізнання, при якому пізнання проходить через пізнання самого методу пізнання. У інформаційних технологіях

під рефлексією розуміється процес зчитування типів, їх методів та властивостей під час виконання коду програми.

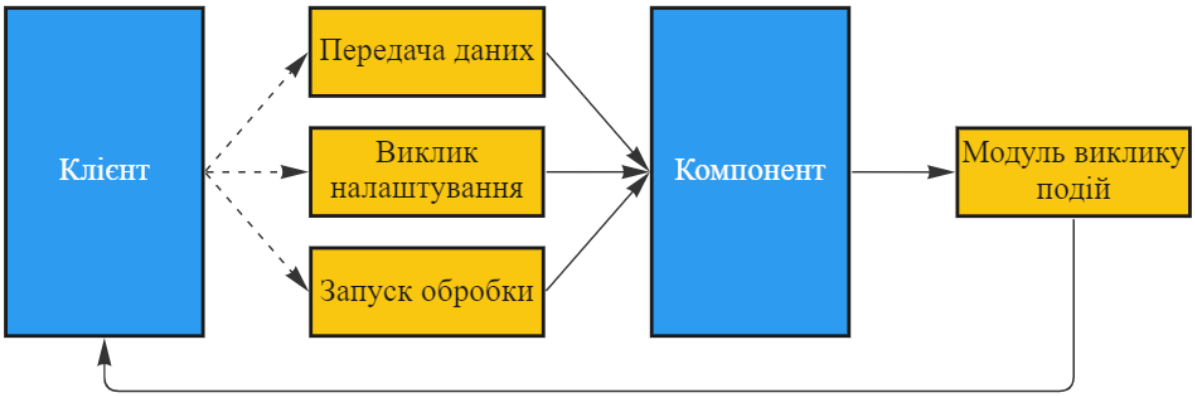

Рисунок 3.7 – Схема інтеграції компоненту у систему

Для ідентифікації завантажених компонентів система зчитує наявність компонентів у відповідній папці та заносить їх у словник з динамічними компонентами де ключем виступає тип компонента, а значенням виступає конкретна реалізація даного типу. При ініціалізації об’єктів проходить підписка до певних подій й генерація необхідних посилань на інтерфейсі для взаємодії з компонентом. Всі візуальні матеріали завантажуються у якості пакета активів при першій ініціалізації компонента у системі. Пакет активів затягується з сервера та утворює файл версії. Додаткові завантаження будуть мати місце лише в тому разі якщо поновлена версія активу.

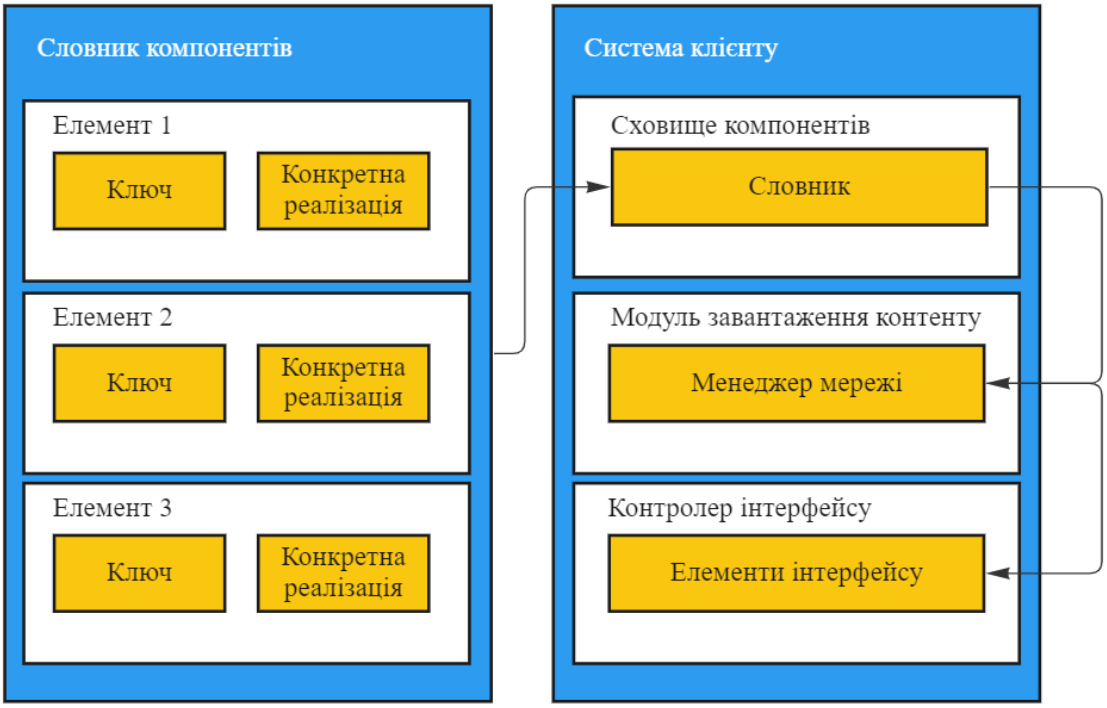

Рисунок 3.8 – Схема ініціалізації компоненту

Інтегрування системи таким чином дозволяє не залежати від базової збірки додатку, У цьому випадку основний додаток має ядро, для візуалізації і калькуляції даних, всі інші елементи є факультативними. За допомогою цього реалізується принцип модульної архітектури проєкти гри емулятору гольфу.

3.3.3 Інтеграція Unit-тестування у проєкт емулятор гольфу

Під час створення основної системи – ядра гри та окремих компонентів проєкту необхідно аналізувати та відстежувати великі пакети даних та розрахунків. У разі не коректної роботи одного з модулів це може призвести до видачі результатів, що не відповідають дійсності.

Для цього було додано систему проведення Unit-тестування для логічних елементів проєкту. Тестові випадки також додаються при інтеграції додаткових компонентів у систему. Для проведення Unit-тестування було використано систему NUnit для ігрового рушія Unity3D.

Валідація даних для тестування була взята з таблиць еталонів з даними, які були розраховані окремо.

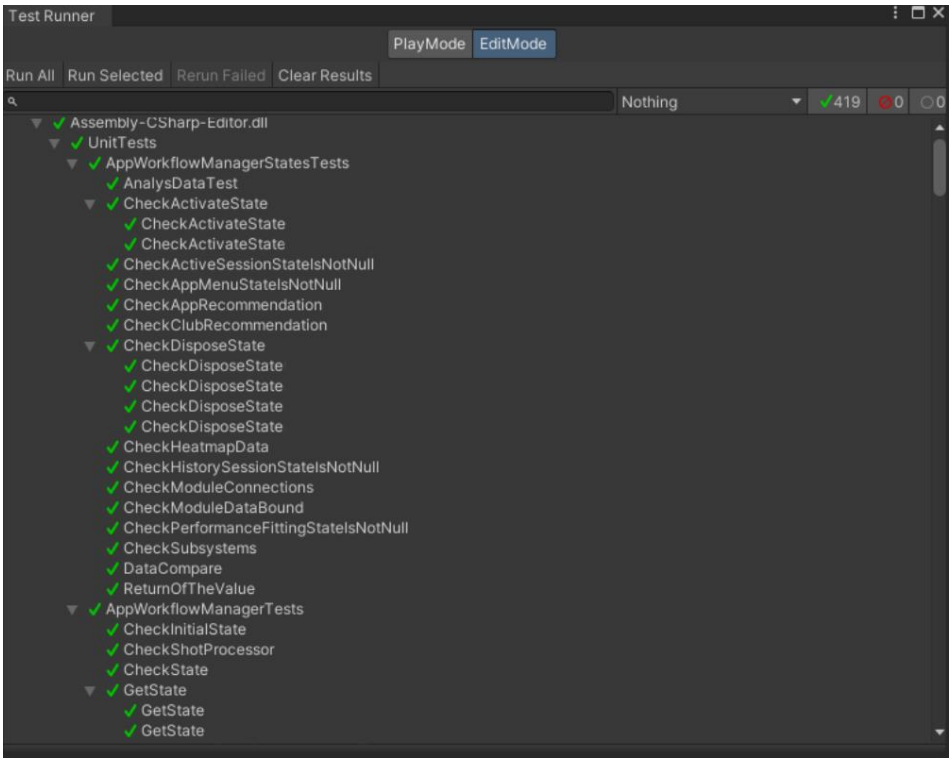

Рисунок 3.9 – Скріншот тест випадків для проєкту

В проєкт було додано 419 тестових випадків для тестування коду й систем компонентів. При цьому додатково були додані тестові випадки для перевірки частин інтерфейсів та Unity3D компонентів на відповідність їх даним, що надаються з підсистем. Запуск Unit-тестів активізується при завантажені проєкту до репозиторію Azure та запуску пайплайну автоматизованої збірки проєкту для систем Windows, Linux, MacOS, iOS, Android. Результат тестування надається у звітах системи Azure.

3.3.4 Аналіз результатів зчитування та розрахунку вхідних даних системи та порівняння з реальними даними

При порівнянні та аналізу ударів гравця в реальному світі та при розрахунку траєкторії у додатку необхідно мінімізувати різницю між даними, що отримує користувач. Для цього було проведено тест при якому робот проводив удари з певним параметром у відкритому просторі та в умовах використання монітору запуску на екрані. Дані зчитувались тим самим пристроєм для уникнення розходжень із-за похибки пристрою. Для тестування були взяті характеристики як кут удару, швидкість польоту м’яча та дані про обертання м’яча.

Таблиця 3.1 – Таблиця ударів у реальному середовищі для швидкості

№	Швидкість м’яча (км/г)	Кут удару (градуси)	Обертання (об/хв)	Дальність (м.)	Дальність з коченням (м.)
1	100	11	2000	113,7	131,5
2	105	11	2000	126	144,8
3	110	11	2000	138,4	156,7
4	115	11	2000	150,8	170,6
5	120	11	2000	163,2	178,5

Таблиця 3.2 – Таблиця ударів та розрахунки у додатку для швидкості

№	Швидкість м’яча (км/г)	Кут удару (градуси)	Обертання (об/хв)	Дальність (м.)	Дальність з коченням (м.)
1	100	11	2000	113,7	131,5
2	105	11	2000	126,3	144,6
3	110	11	2000	139	156,2
4	115	11	2000	151,5	169,5
5	120	11	2000	164,2	177,9

Таблиця 3.3 – Таблиця ударів у реальному середовищі для кута удару

№	Швидкість м’яча (км/г)	Кут удару (градуси)	Обертання (об/хв)	Дальність (м.)	Дальність з коченням (м.)
1	100	12	2000	117,8	135,8
2	100	13	2000	121,5	139,4
3	100	14	2000	124,8	143,5
4	100	15	2000	127,7	141,2
5	100	16	2000	130,3	148,2

Таблиця 3.4 – Таблиця ударів та розрахунки у додатку для кута удару

№	Швидкість м’яча (км/г)	Кут удару (градуси)	Обертання (об/хв)	Дальність (м.)	Дальність з коченням (м.)
1	100	12	2000	116,8	134,9
2	100	13	2000	120,4	138,7
3	100	14	2000	124,1	141,9
4	100	15	2000	128	141,7
5	100	16	2000	128,4	146,3

Таблиця 3.5 – Таблиця ударів у реальному середовищі для обертання м’яча

№	Швидкість м’яча (км/г)	Кут удару (градуси)	Обертання (об/хв)	Дальність (м.)	Дальність з коченням (м.)
1	100	10	2000	109,1	127,3
2	100	10	2500	111,5	126,1
3	100	10	3000	112,5	130,5
4	100	10	3500	113	130,7
5	100	10	4000	113	128,5

Таблиця 3.6 – Таблиця ударів та розрахунки у додатку для обертання м’яча

№	Швидкість м’яча (км/г)	Кут удару (градуси)	Обертання (об/хв)	Дальність (м.)	Дальність з коченням (м.)
1	100	10	2000	112,2	130,3
2	100	10	2500	113,4	129,3
3	100	10	3000	115,5	134,5
4	100	10	3500	115	133,5
5	100	10	4000	116,1	129,3

Як видно із таблиць похибка у даних емулятору є незначною й складає для швидкості приблизно 0,003, для кута удару 0,007, для обертання 0,02. При цьому похибка для польоту м’яча менша ніж похибка загальної дистанції з коченням. Це обумовлено не ідеальною симуляцією фізичної моделі полігону та не відповідності матеріалів полігону та м’яча. Також слід зазначити, що похибка при збільшені обертання росте й збільшує невідповідність у дальності польоту та загальній дальності польоту із коченням. Для мінімізації такої невідповідності слід включити додаткові сили що впливають на м’яч, такі як вологість й тиск.

В результаті тестування отримаємо діаграми похибки для кожного з типів тестів:

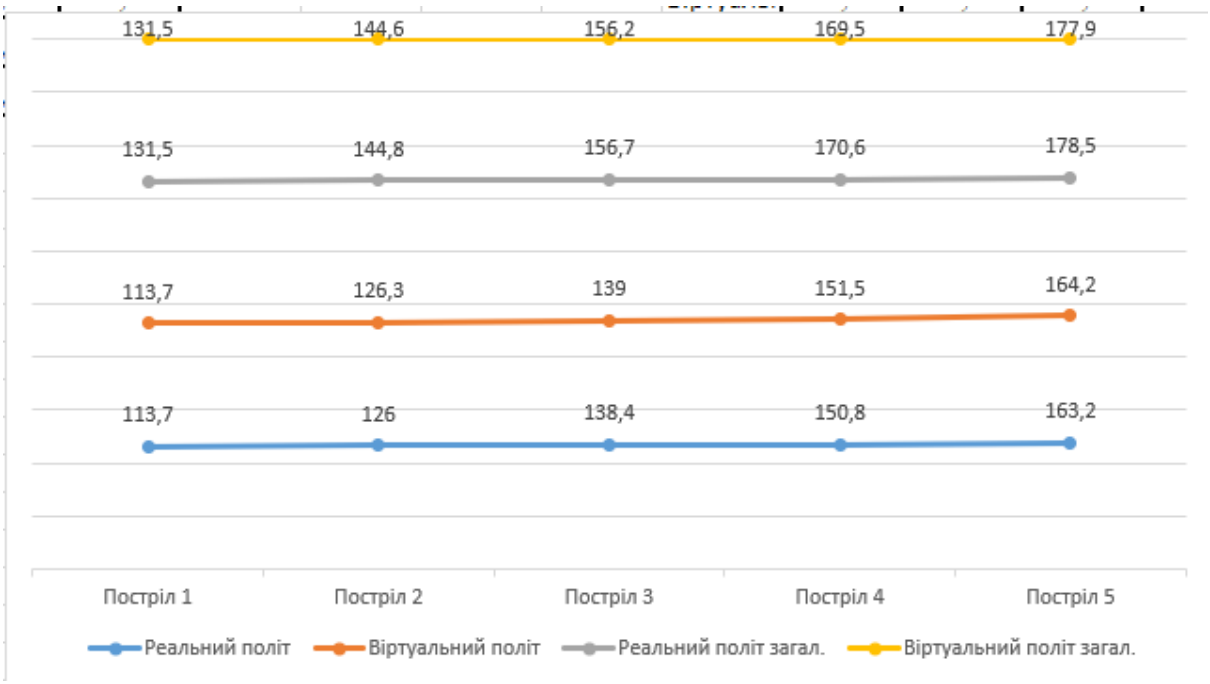

Рисунок 3.10 – Графік розбіжності даних за швидкістю

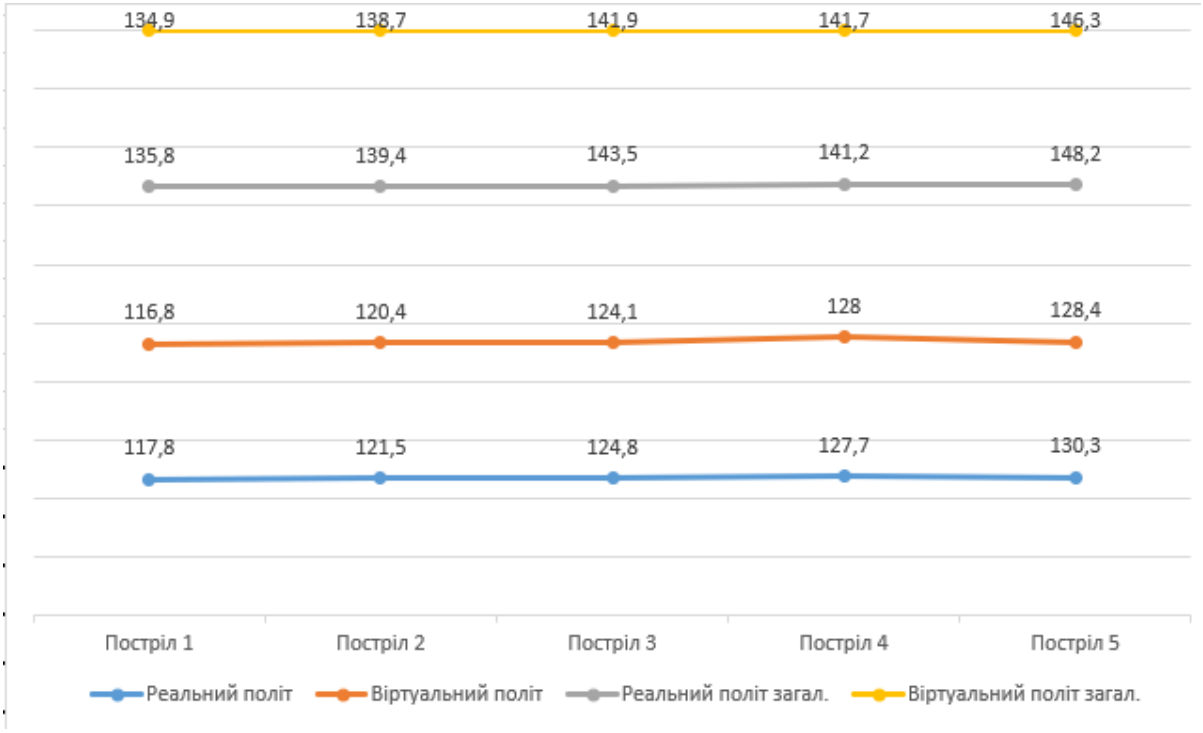

Рисунок 3.11 – Графік розбіжності даних за кутом удару

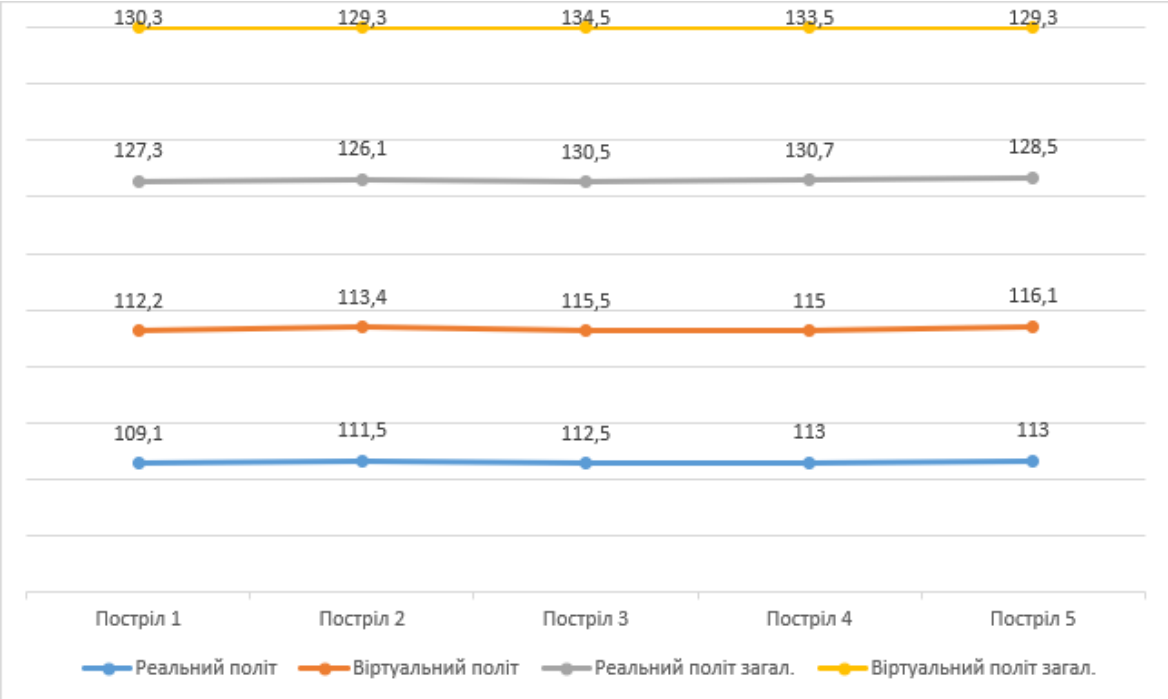

Рисунок 3.12 – Графік розбіжності даних за кутом удару

Отже, можна зробити висновок, що подібна похибка не буде впливати на результати розрахунків та надавати не коректні відомості для інших систем та компонентів.

3.3.5 Аналіз результатів надання рекомендацій щодо зміни патеру удару в грі емуляторі гольфу

Для тестування системи надання рекомендацій щодо удару ключкою було взято групу ударів гравця до рекомендації та після рекомендації. Для отримання більш точних даних необхідно протестувати рекомендації на великій кількості осіб. Очікується що точність користувача повинна підвищитись після винесення рекомендації та дисперсія ударів повинна стати меншою, ніж була до винесення рекомендації.

Таблиця 3.7 – Таблиця ударів до рекомендації

№	Швидкість м’яча (км/г)	Кут удару (градуси)	Обертання (об/хв)	Дальність (м.)	Дальність з коченням (м.)
1	217,5	12,9	2548	202,5	220,6
2	229,1	11	1978	216,5	236,2
3	238,7	12	3002	213	220,8
4	214,2	12,4	3185	191,8	203,9
5	194,9	8,4	2555	156,3	170,7

Гравцю було надано рекомендацію згідно з якою він має змінити кути удару та розвороту ключки, та змінити кут лофту відносно горизонту. Із-за великого обертання м’яча також було запропоновано змінити азимут.

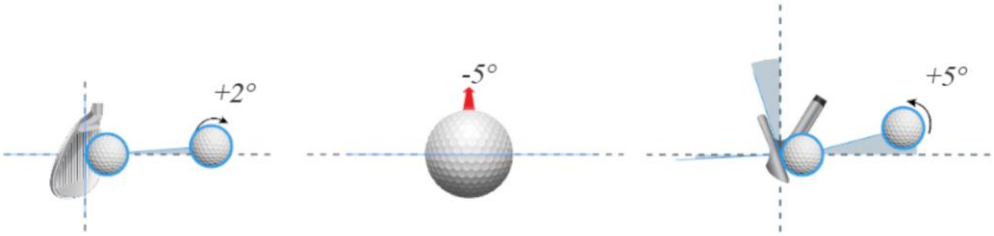

Рисунок 3.13 – Рекомендація щодо покращення патерну удару

Таблиця 3.8 – Таблиця ударів після рекомендації

№	Швидкість м’яча (км/г)	Кут удару (градуси)	Обертання (об/хв)	Дальність (м.)	Дальність з коченням (м.)
1	241,2	14,9	2365	238,3	255
2	223,7	16,8	1846	224,9	244,3
3	232,4	11,9	2650	219,1	240
4	205	12,3	2977	199,5	228,1
5	231,3	14,7	2881	228,4	245

В результаті удару отримуємо 2 еліпси.

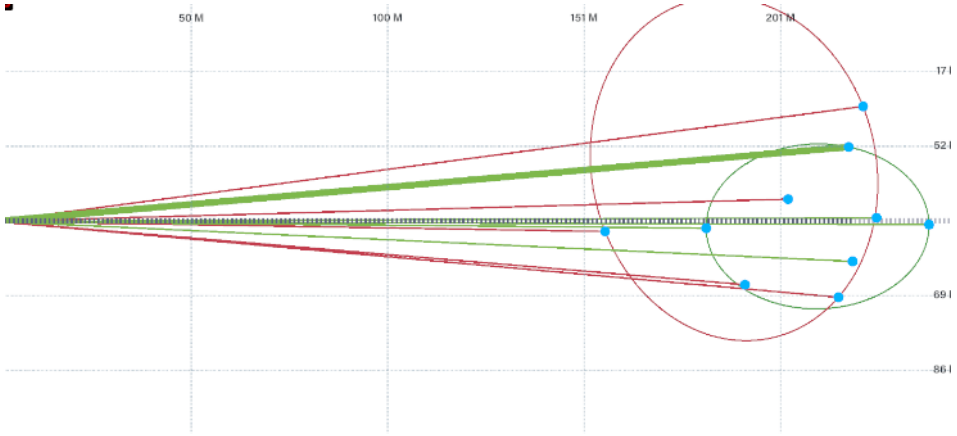

Рисунок 3.14 – Графік розбіжності даних за кутом удару

Виходячи з наведених даних, можемо розрахувати, що точність розрахувавши площу еліпсу. Для першої групи ударів площа складає 1170 метрів квадратних, а для другого 520, що удвічі збільшило показник точності ударів гравця. Така рекомендація не може гарантувати покращення траєкторії, адже додержання рекомендації залежить від користувача. Такі дані доцільно використовувати при навчальному процесі гри у гольф та аналізі тенденцій щодо покращення удару.

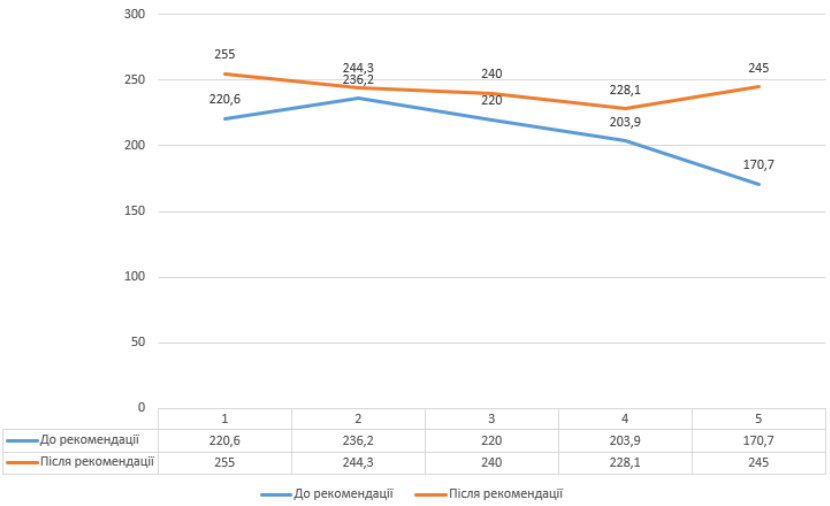

Рисунок 3.15 – Графік розбіжності даних за кутом удару

Також можна стверджувати, що під час тесту збільшилась дальність удару гравця.

3.3.6 Аналіз результатів надання рекомендацій щодо зміни ключки гравця в грі емуляторі гольфу

При надані рекомендації щодо вибору ключки рекомендаційна система враховує параметри, що передаються у групі ударів гравця та правила експертів та характеристики ключок.

Для покращення тенденції удару було зроблено ряд ударів й передано дані про ці удари до компоненту рекомендацій. Була рекомендована одна з ключок яка знаходиться в базі даних для ключок. Після винесення рекомендацій був зроблений такий же ряд ударів, та були отримані результати з покращеними даними.

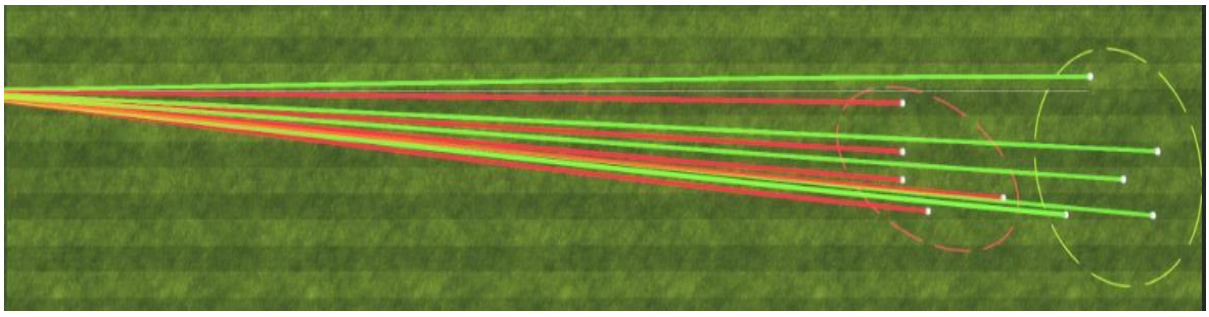

Рисунок 3.16 – Порівняння ударів до та після рекомендації

В результаті отриманих результатів можемо зробити висновок що дальність удару покращилась для кожного із ударів.

Для візуалізації даних можна їх представити у вигляді діаграми з порівнянням ударів до і після застосування рекомендованої ключки.

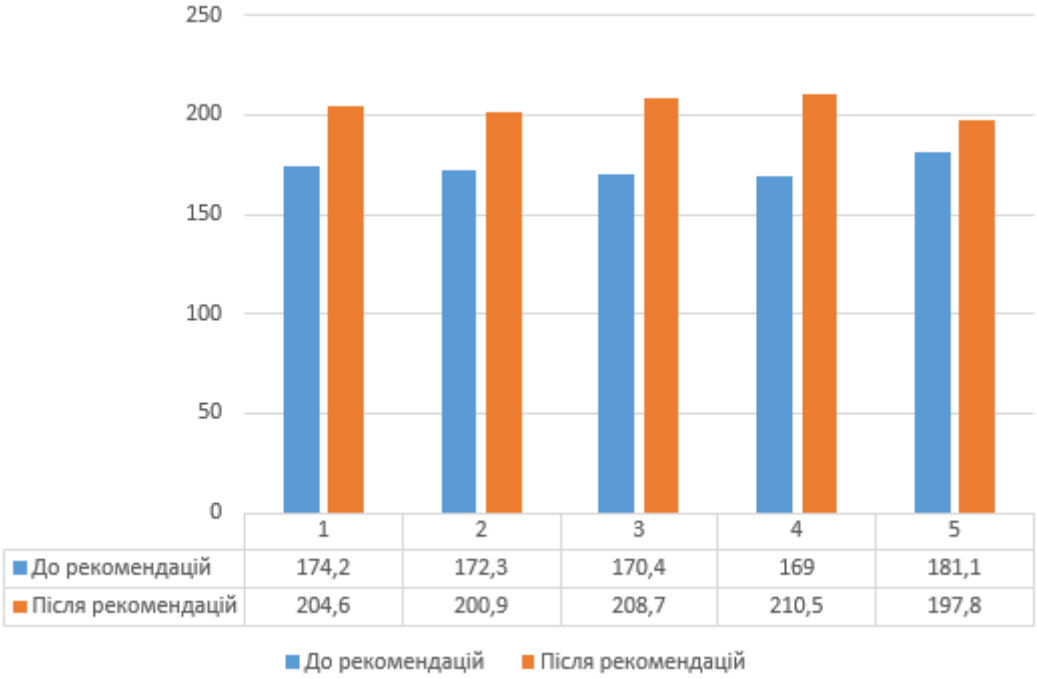

Рисунок 3.17 – Порівняльна діаграма ударів до та після рекомендації

При проведенні тестування системи на великій виборці ударів було визначено, що при застосуванні рекомендованої ключки дальність удару збільшується у 80% випадків, так, серед 50 ударів, 40 з них пішли по поліпшеній траєкторії.

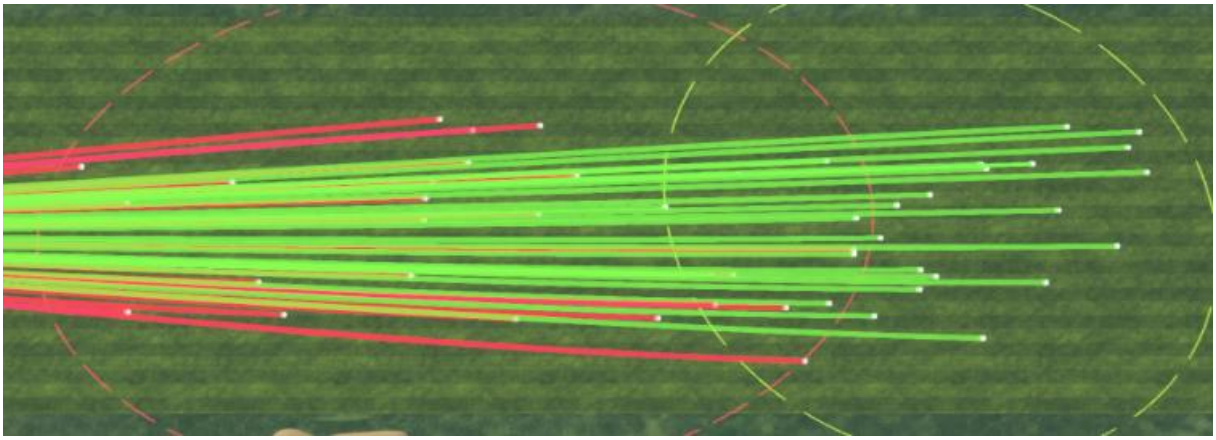

Рисунок 3.18 – Результат тестування великої вибірки ударів

Отже можна зробити висновок, що така рекомендаційна система працює й може застосовуватись для підбору ключки під певні параметри гри користувача.

Для оцінки результатів рекомендації був застосований метод експертних оцінок, для цього були утворені матриці з двома альтернативами та запропоновані питання щодо якості наданої рекомендації. Альтернативами виступає попередня ключка та рекомендована. Для цього використовується схема матриці ієрархій.

$$B_i = \begin{pmatrix} 1 & b_{12} & \dots & b_{1i} & \dots & b_{1m} \\ b_{21} & 1 & \dots & b_{2i} & \dots & b_{2m} \\ \dots & \dots & 1 & \dots & \dots & \dots \\ b_{i1} & b_{i2} & \dots & 1 & \dots & b_{im} \\ \dots & \dots & \dots & \dots & 1 & \dots \\ b_{m1} & b_{m2} & \dots & b_{mi} & \dots & 1 \end{pmatrix} \tag{3.1}$$

де B_i – матриця ієрархій;

b – оцінка критерію.

Для отримання оцінки альтернатив можуть застосовуватись різні способи. Спосіб запропонований Сааті представляє собою використання середнє геометричного усереднення і нормування оптимальних узагальнених оцінок.

$$(B_j) = \begin{pmatrix} \sqrt[m]{b_{11} \cdot b_{12} \cdot \ldots \cdot b_{1m}} = d_1 \\ \sqrt[m]{b_{21} \cdot b_{22} \cdot \ldots \cdot b_{2m}} = d_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \sqrt[m]{b_{m1} \cdot b_{m2} \cdot \ldots \cdot b_{mm}} = d_m \end{pmatrix} \Rightarrow \{D = d_1 + d_2 + \ldots + d_m\} \Rightarrow$$

$$\Rightarrow \left\{ \frac{d_1}{D} = w_1; \frac{d_2}{D} = w_2; \ldots; \frac{d_m}{D} = w_m \right\} \Rightarrow (w_1, w_2, \ldots, w_m) \tag{3.2}$$

де B_j – матриця ієрархій;

d – результат суми оцінки критерію;

m – кількість елементів матриці;

w – вага оцінки критерію;

D – сума критеріїв.

Після чого перевіряється узгодженість елементів за допомогою методів лінійної згортки. В результаті проведення аналізу було підтверджено, що рекомендація є кращою для даних параметрів удару на думку експертів.

Висновки до розділу 3

Додаток емулятор гри в гольф використовує клієнт серверну архітектуру для розвантаження процесів клієнту та перенесення частини інтелектуальних інформаційних систем на сторону серверу, яка надає рекомендації користувачам по зіграним сесіям. Сервісно-орієнтовну архітектура клієнта надає можливість підключення окремих модулів для розвитку додатку та додання нових інтелектуальних інформаційних систем. Сервіс-орієнтований підхід є продовженням модульного підходу. З ускладненням додатків деякі модулі виносяться на окремі апаратні частини та сервіси.

Були виділені основні логічні елементи системи які відповідають за основний потік ігрового процесу та рендерінг, за модулі теплових карт, модулі розрахунку інформації щодо параметрів ударів гравця та інші. В проєкті застосована низка патернів проєктування які уніфікують структуру проєкту для її подальшого розвитку та впровадження додаткових інтелектуальних інформаційних систем, які можуть бути впроваджені як наступний крок проєктування проєкту. Для забезпечення масштабованості гри симулятору у гольф був застосований патерн інверсії управління за допомогою якого була впроваджена ін’єкція залежності, що зменшило зв’язаність проєкту.

Поточна архітектура і структура проєкту дозволяє забезпечити подальший розвиток проєкту та додання нових модулів та навіть зміни візуальної та ігрової

частини гри емулятора в гольф. Так окрім створення гри емулятора в гольф можна додати й інші ігри, такі як міні гольф та інші.

Для створення елементів які потребують зв’язаності використовується патерн фабрика який при створенні об’єктів вирішує залежності та надає безпосередні реалізації класів у якості залежності до конструкторів класів.

Так гра емулятор гольфу утворює гнучку систему модулів які мають змогу для розширення та впровадження нових бібліотек без необхідності основної бази коду.

При реалізації компонентів було проведено тестування компонентів системи, яке показало ефективність застосування рекомендаційних систем та можливості застосування їх для навчання грі у гольф та для коректного підбору ключки під певний патерн удару. Оцінка результатів була проведена за допомогою методів системного аналізу таких як метод математичної статистики та методів експертних оцінок.

ВИСНОВКИ

В даній роботі було розроблено гру емулятор гольфу, що працює з різними пристроями вводу для зчитування ударів гравця, таких як монітори запуску, ігрові джойстики, клавіатура та тачпад. Гра емулятор гольфу виконує основну логіку гри в гольф, тобто гравець має наносити удари по м’ячу та потрапити у ціль, щоб набрати виграшні бали.

Для розширення задач гри були розроблені спеціальні інтелектуальні інформаційні системи, що надають гравцю змогу аналізувати зроблені ним удари та робити висновки про тип траєкторії гравця. Така інформація є корисною для користувача для отримання інформації щодо якості своєї гри й рекомендації того як гравець може її покращити свою гру чи траєкторію удару, спираючись на кут удару та дальність.

Для встановлення якості удару був впроваджений розрахунок траєкторії та всіх необхідних параметрів удару, таких як швидкість м’яча, швидкість ключки, кут запуску, азимут запуску, обернене обертання, бокове обертання, шлях ключки по вертикалі та горизонталі, кут дотику до голівки ключки, кут древка ключки, лофт ключки, та інші параметри.

Для проведення аналізу ударів була створена база даних яка вміщає в себе параметри ударів гравця, де користувач може брати окремі удари ігрової сесії та порівнювати параметри удару й оцінювати якість показників удару. Система підсвічує параметри з кращими результатами при порівняльному аналізі ударів.

Розроблена система надає можливість будувати дисперсію групи ударів та визначати стандартну дивіацію групи ударів, що свідчить точність гравця та можливість дотримання своєї тенденції гри при заданих умовах оточення з поточним обладнанням.

Для проведення аналізу удару система також надає інформацію щодо точок торкання ключки до м’яча за допомогою монітору запуску. Отримання інформації у вигляді теплової карти для групи ударів надає інформацію та довідкові дані

щодо правильності використання ключки та техніки гри з певним обладнанням, а також інформації щодо правильності вибору ключки в поточних умовах.

Оскільки складність у грі емуляторі гольфу корелюється у відповідності до навичок та професіоналізму гравця сбір та аналіз даних сесій гравця надає змогу налаштувати систему ботів таким чином щоб бот завжди грав на рівні гравця, з додатковим коефіцієнтом успішності, для більш точного контролю функції складності й реалізації синусоїдної системи прогресії у грі. Це дозволить гравцю поступово поліпшувати свої навички та постійно суперника, що грає на рівні користувача.

В гру емулятор гольфу було впроваджено систему надання рекомендацій щодо вибору ключки гравцем. Дана інтелектуальна інформаційна система аналізує наявні сесії гравця і за допомогою створеної бази даних та розрахунків тенденції траєкторії визначає ключку яка може виправити траєкторію польоту до більш бажаного. При цьому аналізуються всі параметри удару де цільовими параметрами виступають дальність, кут удару та обертання м’яча.

Наведені вище інформаційні інтелектуальні системи надають можливість перевести додатки з сегменту розваг у навчальний сегмент й набагато розширити завдання таких додатків, до навчальних, задач прогнозу та аналізу. Впровадження інтелектуальні інформаційні системи створює новий шар використання додатків та розширює можливості аналізу поточних й потенційних результатів гравців.

Інтелектуальні інформаційні системи можуть використовуватись в інших спортивних додатках для аналізу даних, адже структура написання коду дозволяє інтегрувати їх у інші системи, що робить подібний інструмент дуже гнучким для використання й надає безмежні можливості для розвитку й розширення систем аналізу гри користувачів у різні спортивні симулятори та інші подібні додатки.

Було встановлено, що у гру емулятор гольфу є можливість включити й інші інтелектуальні інформаційні системи, як із поточним обладнанням, так і з додатковим обладнанням таким як кинект або ж камери, що зчитують рух гравця.

Під час реалізації інтелектуальних інформаційних систем було створено архітектуру додатку й дослідження можливості приєднувати нові модулі ІІС до додатка як окремої бібліотеки, що робить такий інструмент універсальним. Відсутність зовнішніх залежностей дозволяє використовувати розроблені системи й в інших емуляторах спорту, таких як футболу чи баскетболу, при цьому самі модулі не потребують додаткового перепрограмування.

Було визначено що впровадження клієнт-серверної архітектури проєкту дозволяє збирати статистичну інформації щодо ударів і сесій гравця й використовувати її для подальшого навчання системи рекомендацій та інших систем.

Наразі ведеться багато досліджень компаніями – виробниками моніторів запуску щодо створення нових інтелектуальних систем та впровадження їх у додатки для покриття потреб користувачів у даній сфері.

В методичному розділі роботи було розглянуто метод Леонтьєва, що допомагає визначити коефіцієнти повних витрат, валовий випуску цехів, встановлення виробничих програм цехів та визначення коефіцієнтів непрямих витрат для багатогалузевої економіки. Було розраховано приклад лабораторної роботи та надано теоретичні відомості щодо математичної частини метода Леонтьєва.

В спеціальному розділі роботи було розглянуто умови охорони праці та убезпечення робітників підприємства. Було проаналізовано етапи інструктажів та описано нормі праці в рамках підприємства, а також в умовах видаленої роботи працівників. Було розглянуто основні нормативні засади та інструкції що представляються працівниками кадрових відділів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ ISO/IEC 2382:2017 Інформаційні технології. Словник термінів (ISO/IEC 2382:2015, IDT) Київ: Технічний комітет стандартизації «Інформаційні технології» (ТК 20), 2019.

2. Наказ від 19.12.2017 № 429 «Про прийняття національних нормативних документів, гармонізованих з європейськими та міжнародними нормативними документами, та скасування чинності національних нормативних документів» затверджений державне підприємство «український науково-дослідний і навчальний центр проблем стандартизації, сертифікації та якості».

3. Рибіна Г. В. Основи будови інтелектуальних систем / Г. В. Рибіна. – К: Фінанси і статистика, 2010.

4. Ситник В. Ф. Системи підтримки прийняття рішень. – К.: Техніка, 1995.

5. Клир Дж. Системологія. Систематизація рішення системних задач: Пер. с англ. – К.:1990.

6. Дудник І. М. Вступ до загальної теорії систем. – К.: Кондор, 2009.

7. Лесечко М. Д. Основи системного підходу: теорія, методологія, практика: Навч. посіб. – Львів: ЛРІДУ УАДУ, 2002.

8. Орлов П. І., Луганський О.М. Інформаційні системи та технології в управлінні, освіті, бібліотечній справі: Наук.-практ. посіб. – Донецьк: Альфапрес, 2004.

9. Чорней Н. Б. Теорія систем і системний аналіз: Навч. посіб. для студ. вищ. навч. закл. – К.: МАУП, 2005.

10. Шершньова З. Є. Антикризове управління підприємством : Навч. посібник / В.О. Василенко. – К.: ЦУЛ, 2003.

11. Дошина А. Д. Експертні системи. Класифікація. Огляд існуючих експертних систем/ А. Д. Дошина. –Просвіта. – 2016.

12. Антонов А. В. Архітектура додатків. – Х.: Світ, 2004.

13. Волкова В. Н., Денісов А. А. Теорія систем: Навч. посібник для студентів вузів. – К.: ЦУЛ, 2006.

14. Гайдес М. А. Загальна теорія систем (системи и системний аналіз). – Вінниця: Глобус-пресс, 2005.

15. Прокопенко Т. О. Теорія систем і системний аналіз : навч. посіб. Т. О. Прокопенко; М-во освіти і науки України, Черкас. держ. технол. ун-т. – Черкаси : ЧДТУ, 2019.

16. Федорчук Є. Н. Програмування систем штучного інтелекту. Експертні системи / Є.Н.Федорчук, Вид-во Львівської політехніки, 2012.

17. Баклан І. В. Експертні системи. Курс лекцій /Навчальний посібник. – К.: НАУ, 2012.

18. Лунь Ю. О., Козел А. М., Ніколаєв С. М. Експертні системи в міжнародних відносинах – Львів: Видавництво Львіської політихніки, 2011.

19. Катренко А. В. Системний аналіз об’єктів та процесів комп’ютеризації / А. В. Катренко. – Львів : Новий Світ-2000, 2007.

20. Шалютин С. М. Штучний інтелект, – К.: Думка, 2005.

21. Гусаров В. М. Статистика Навч. посібник для вузів /В. М. Гусаров, Є. І. Кузнєцова. - 2-ге вид., перераб. та дод. – К.: Світ, 2008.

22. Ендрю А. Штучний інтелект, – К.: Світ, 2001.

23. Вакалюк Т. А. Хмарні технології в освіті. Навчально-методичний посібник / Т. А. Вакалюк. – Житомир: вид-во ЖДУ, 2016.

24. Н. Вінер “Кібернетика”, К.: Наука, 2003.

25. Шаховська Н. Б. Системи штучного інтелекту: навчальний посібник / Н. Б. Шаховська, Р. М. Камінський, О. Б. Вовк. – Львів : Видавництво Львівської політехніки, 2018.

26. Стефанюк В. Л. Експертні системи і їхнє застосування: Курс лекцій.

27. Олексюк В. Основи хмарних технологій / В. Олексюк. – Тернопіль: Тернопільський обласний інститут післядипломної педагогічної освіти, 2018.

28. Худсон Д. Статистика для фізиків. – Харків.: Світ, 1970.

29. Abelson H., Sussman G. J. and Sussman J. (1996). Structure and Interpretation of Computer Programs. Cambridge MA: MIT Press.

30. Aikins J. S., Kunz J. C., Shortliffe E. H. and Fallat R. J. (1984). PUFF: an expert system for interpretation of pulmonary function data. In Readings in Medical Artificial Intelligence (Clancey W. J. and Shortliffe E. H., eds.). Chapter 19. Reading, MA: Addison-Wesley.

31. Chandrasekaran B. (1984). Expert systems: matching techniques to tasks. In Artificial Intelligence Applications for Business (Reitman W, eds.). Norwood, NJ: Ablex.

32. Cox R. (1946). Probability frequency and reasonable expectation. American Journal of Physics, 14(1).

33. De Kleer J. and Williams B. C. (1987). Diagnosing multiple faults. Artificial Intelligence, 32, p. 97-130.

34. Спицнадель В. Н. Основи системного аналіза: навч. Посібник – К: «Нова-пресса», 2000.

35. Романов В. Н. Системний аналіз для інженерів / В. Н. Романов. – К., 2008.

36. Саати Т. Аналітичне планування. Организація систем / Т. Саати, К. Кернс. – К.:Наука, 1999.

37. Джейсон Прайс. Visual C#. Повне керівництво. / Джейсон Прайс. Майк Гандерлой.: пров. з англ. – К.:НТІ, М: Ентроп, 2004.

38. Довідник з C# – Режим доступу : URL https://msdn.microsoft.com/ru-ru/library/618ayhy6.aspx (відвідано 05.01.2023).

39. Макроекономіка. Математичне обгрунтування: навч. посіб. / О. І. Пономаренко, М. О. Перестюк, В. М. Бурим. – К.: Вища шк., 2004.

40. Базилевич В. Д. Математичні методи в економіці : підручник / За ред. В. Д. Базилевича ; 4-те вид., перероб. і доп. / В. Д. Базилевич, К. С. Базилевич, Л. О. Баластрик. – К. : Знання, 2008.

41. Кочура Є. В. Моделювання макроекономічної динаміки / Є. В. Кочура, В. М. Косарів. – К. : Центр навчальної літератури, 2003.

42. Закон про охорону праці США 1971 року.

43. Загальні стандарти по охороні праці й техніці безпеки.

ДОДАТОК А

Лістинг коду розрахунку траєкторії польоту м’яча

```
private IEnumerator Simulation(Transform t, Rigidbody rb)
{
            while (isInFlight)
            {
                SimulateFlight(t, rigidbody, Time.fixedDeltaTime, true);
                yield return new WaitForEndOfFrame();
            }
            Time.timeScale = 1;
            yield return null;
}

private void SimulateFlight(Transform t, Rigidbody rigidbody, float timeStep, bool
renderLines)
{

    if (!isInFlight)
    {
        GrassImpulse(rigidbody);
        return;
    }
    else
    {
        projTime += timeStep;
        int timeScaler = 100;
        int index = (int)Mathf.Min(projTime * timeScaler, velocities.Count);
        if (index > velocities.Count - 1)
        {
              if (index > velocities.Count - 1)
                  index = velocities.Count - 1;
               LerpProjectileTransform(t, index, projTime);
                SwitchToPhysics(index, rb);
        }
        else if (CheckIntersection(velocities[index], rb))
        {
                 LerpProjectileTransform(t, index, projTime);
                 SwitchToPhysics(index, rb);
        }
        else
        {
                 LerpProjectileTransform(t, index, projTime);
        }
        if (renderLines)
        {
            lineRenderer.positionCount += 1;
            lineRenderer.SetPosition(lineRenderer.positionCount - 1, positions[index]);
            lineRendererProjection.positionCount = lineRenderer.positionCount;
           lineRendererProjection.SetPosition(lineRenderer.positionCount - 1, new
      Vector3(positions[index].x, 0, positions[index].z));
        }
      }
}
```

ДОДАТОК Б

Лістинг коду розрахунку середньоквадратичного відхилення ударів гравця

```
    public Vector3 GetAverageEndpoint(ShotGroup shotGroup, out List<Vector3>
shotEndpoints)
    {
        endpoints = GetShotEndpoints(shotGroup);

        Vector3 sum = new Vector3();
        foreach (Vector3 endpoint in endpoints)
        {
            sum.x += endpoint.x;
            sum.z += endpoint.z;
        }

        return sum / (float) endpoints.Count;
    }

    public Vector2 GetStandardDeviations(ShotGroup group, out Vector3 average)
    {
        List<Vector3> endpoints;
        average = GetAverageEndpoint(group, out endpoints);

        return CalculateStandardDeviationFromEndpoints(endpoints, average);

    }


  private Vector2 CalculateStandardDeviationFromEndpoints(List<Vector3> points, Vector3
average)
    {
        Vector2 sum = new Vector2();
        foreach (Vector3 point in points)
        {
            sum += new Vector2(Mathf.Pow(point.z - average.z, 2f), Mathf.Pow(point.x -
average.x, 2f));
        }

        return new Vector2(Mathf.Pow(sum.x / points.Count, 0.5f), Mathf.Pow(sum.y /
points.Count, 0.5f));
          }
```

ДОДАТОК В

Лістинг коду розрахунку еліпсу середньоквадратичного відхилення

```
   private void SetStandardDeviationEllipseForGroup(Group group, float orthographicSize)
        {
            if (!SavedConfiguration.StandardDeviation)
            {
                return;
            }

            Vector2 standardDeviations =
ellipseCalc.GetStandardDeviationsFromShotGroup(group, out Vector3 averageEndpoint);

            standardDeviations *= SavedConfiguration.StandardDeviationMultiplier;

            var theta = 0f;
            var a = 0f;
            var b = 0f;
            if (standardDeviations.x > standardDeviations.y)
            {
                a = standardDeviations.x;
                b = standardDeviations.y;
                theta = 90f;
            }
            else
            {
                a = standardDeviations.y;
                b = standardDeviations.x;
            }

            Vector3[] ellipsePoints = ellipseCalc.CreateEllipse(a, b, averageEndpoint.x,
averageEndpoint.z, theta, 1000);

            float segmentLength = dottedLineSegmentLengthPercentage * orthographicSize *
orthoMultiplier;
            // formula for perimeter is based on second approximation of Ramanujan
            float h = (Mathf.Pow(a - b, 2f) / Mathf.Pow(a + b, 2f));
            float perimeter = Mathf.PI * (a + b) * (1f + (3 * h) / (10 + Mathf.Sqrt(4 - 3
* h)));
            int idealNumberOfSegments = Mathf.FloorToInt(perimeter / segmentLength);
            int ellipseSegmentLength = Mathf.FloorToInt((float)(ellipsePoints.Length - 1)
/ (float)idealNumberOfSegments);
            int remainderOfTwo = ellipseSegmentLength % 2;

            if (remainderOfTwo != 0)
            {
                ellipseSegmentLength--;
            }

            if (ellipseSegmentLength < 6)
            {
                ellipseSegmentLength = 6;
            }

            bool skipping = true;
```

```
            for (int i = 0; i < ellipsePoints.Length - 1; i += ellipseSegmentLength)
            {
                if (!skipping)
                {
                    GameObject lineObject = Instantiate(linePrefab);
                    lineObject.name = EllipseLineName;
                    lineObject.transform.SetParent(lineParentTopView);
                    lineObject.layer = 15;
                    LineRenderer lineRenderer = lineObject.GetComponent<LineRenderer>();

                    lineRenderer.startWidth = orthographicSize * dottedLineWidthPercentage
* orthoMultiplier;
                    lineRenderer.endWidth = orthographicSize * dottedLineWidthPercentage *
orthoMultiplier;

                    lineRenderer.material.SetColor(ShaderBaseColorName,
((Color)shotGroup.Color).linear);

                    Vector3[] positions = new Vector3[ellipseSegmentLength];
                    int lineLength = positions.Length;
                    for (int k = 0; k < ellipseSegmentLength; k++)
                    {
                        if (i + k < ellipsePoints.Length)
                        {
                            positions[k] = ellipsePoints[i + k];
                        }
                        else
                        {
                            lineLength--;
                        }
                    }
                    if (lineLength != positions.Length)
                    {
                        Array.Resize(ref positions, lineLength);
                    }
                    lineRenderer.positionCount = lineLength;
                    lineRenderer.SetPositions(positions);
                }
                skipping = !skipping;
            }
        }

 public Vector3[] CreateEllipse(float a, float b, float h, float k, float theta, int
resolution)
    {
        Vector3[] positions = new Vector3[resolution + 1];
        Quaternion q = Quaternion.AngleAxis(theta, Vector3.up);
        Vector3 center = new Vector3(h, 0f, k);

        for (int i = 0; i <= resolution; i++)
        {
            float angle = (float)i / (float)resolution * 2.0f * Mathf.PI;
            positions[i] = new Vector3(a * Mathf.Cos(angle), 0f, b * Mathf.Sin(angle));
            positions[i] = q * positions[i] + center;
        }

        return positions;
    }
```

ДОДАТОК Г

Лістинг коду розрахунку дисперсії ударів гравця

```
public float CalculateDispersion(ShotGroup shotGroup)
    {
        float theta;
        Vector3 elipseCenter;
        Vector2 dispersionAxis = GetDispersionAxis(shotGroup, out elipseCenter, out
theta);

        return dispersionAxis.x * dispersionAxis.y * Mathf.PI;
    }

public Vector2 GetDispersionAxis(ShotGroup shotGroup, out Vector3 elipseCenter, out float
theta)
    {
        List<Vector3> shotEndpoints = GetShotEndpoints(shotGroup);
        if (shotEndpoints.Count < 1)
        {
            theta = 0f;
            elipseCenter = Vector3.zero;
            return Vector2.zero;
        }
        int[] furthestPair = GetFurthestPositionsInGroup(shotEndpoints);

        Vector3 shot1Endpoint = shotEndpoints[furthestPair[0]];
        Vector3 shot2Endpoint = shotEndpoints[furthestPair[1]];

        if (shot1Endpoint.x == shot2Endpoint.x || shot1Endpoint.z == shot2Endpoint.z)
        {
            theta = 0f;
            elipseCenter = Vector3.zero;
            return Vector2.zero;
        }

        elipseCenter = new Vector3(shot1Endpoint.x + ((shot2Endpoint.x - shot1Endpoint.x)
/ 2f), 0f, shot1Endpoint.z + ((shot2Endpoint.z - shot1Endpoint.z) / 2f));

        theta = (Mathf.Atan((shot2Endpoint.x - shot1Endpoint.x)/ (shot2Endpoint.z -
shot1Endpoint.z)) * Mathf.Rad2Deg);
        List<Vector3> cleanedShotPositions = new List<Vector3>();
        float maximumB = float.MinValue;

        // rotate all endpoints
        for (int i = 0; i < shotEndpoints.Count; i++)
        {
            Vector3 rotatedShotPosition = RotatePointAroundPivot(shotEndpoints[i],
elipseCenter, Quaternion.Euler(0f, -theta, 0f)) - elipseCenter;
            cleanedShotPositions.Add(rotatedShotPosition);
        }
        // calculate a (radius of major axis)
        float a = Mathf.Abs(cleanedShotPositions[furthestPair[1]].z -
cleanedShotPositions[furthestPair[0]].z) / 2f;
        // calculate minor axis for each point that is not an endpoint of major axis
        for (int j = 0; j < shotEndpoints.Count; j++)
        {
```

```
            if (j != furthestPair[0] && j != furthestPair[1]) // exclude endpoints
            {
                // calculate b(minor axis) for each point that is not an endpoint for
major axis
                float b = Mathf.Sqrt(Mathf.Pow(cleanedShotPositions[j].x, 2f) /
                    (1f - (Mathf.Pow(cleanedShotPositions[j].z, 2f) / Mathf.Pow(a, 2f))));
                if (b > maximumB)
                {
                    maximumB = b;
                }
            }
        }
        return new Vector2(a, maximumB);
           }
```

ДОДАТОК Д
Лістинг коду шейдеру теплової карти

```
Shader "Unlit/Heatmap"
{
	Properties
	{
	  _MainTex("Texture", 2D) = "white" {}
	  _Color0("Color 0",Color) = (0,0,0,1)
		_Color1("Color 1",Color) = (0,.9,.2,1)
		_Color2("Color 2",Color) = (.9,1,.3,1)
		_Color3("Color 3",Color) = (.9,.7,.1,1)
		_Color4("Color 4",Color) = (1,0,0,1)
		_Color5("Color 5",Color) = (0,.9,.2,0)
		_Range0("Range 0",Range(0,1)) = 0.1
		_Range1("Range 1",Range(0,1)) = 0.25
		_Range2("Range 2",Range(0,1)) = 0.5
		_Range3("Range 3",Range(0,1)) = 0.75
		_Range4("Range 4",Range(0,1)) = 1
		_Range5("Range 5",Range(0,1)) = 0

		_Diameter("Diameter",Range(0,1)) = 1.0
		_Strength("Strength",Range(.1,4)) = 1.0
		_PulseSpeed("Pulse Speed",Range(0,5)) = 0
		_ShotIntencivity("Shot Intencivity", float) = 0.1
		_FadeIntencivity("Fade Intencivity", float) = 0.1
	}
		SubShader
	  {
		Tags { "RenderType" = "TransparentCutout" "Queue" = "AlphaTest" }
		LOD 100
		ZWrite Off
		Blend SrcAlpha OneMinusSrcAlpha Cull Off

		Pass
		{
		  CGPROGRAM
		  #pragma vertex vert
		  #pragma fragment frag
		  #pragma multi_compile_fog

		  #include "UnityCG.cginc"

		  struct appdata
		  {
			float4 vertex : POSITION;
			float2 uv : TEXCOORD0;
		  };

		  struct v2f
		  {
			float2 uv : TEXCOORD0;
			UNITY_FOG_COORDS(1)
			float4 vertex : SV_POSITION;
		  };
		  sampler2D _MainTex;
		  float4 _MainTex_ST;
```

```
float4 _Color0;
float4 _Color1;
float4 _Color2;
float4 _Color3;
float4 _Color4;
float4 _Color5;
float _Range0;
float _Range1;
float _Range2;
float _Range3;
float _Range4;
float _Range5;
float _Diameter;
float _Strength;
float _PulseSpeed;
float _ShotIntencivity;
float _FadeIntencivity;
v2f vert(appdata v)
{
    v2f o;
    o.vertex = UnityObjectToClipPos(v.vertex);
    o.uv = TRANSFORM_TEX(v.uv, _MainTex);
    UNITY_TRANSFER_FOG(o,o.vertex);
    return o;
}
float3 colors[6];
float pointranges[6];
float _Hits[3 * 1000];
int _HitCount = 0;

void initalize()
{
    colors[0] = _Color5;
    colors[1] = _Color0;
    colors[2] = _Color1;
    colors[3] = _Color2;
    colors[4] = _Color3;
    colors[5] = _Color4;
    pointranges[0] = _Range5;
    pointranges[1] = _Range0;
    pointranges[2] = _Range1;
    pointranges[3] = _Range2;
    pointranges[4] = _Range3;
    pointranges[5] = _Range4;
}

float3 getHeatForPixel(float weight)
{
    if (weight <= pointranges[0])
    {
      return colors[0];
    }
    if (weight >= pointranges[5])
    {
      return colors[5];
    }
    for (int i = 1; i < 6; i++)
    {
```

```
                    if (weight < pointranges[i])
                    {
                        float dist_from_lower_point = weight - pointranges[i - 1];
                        float size_of_point_range = pointranges[i] - pointranges[i - 1];

                        float ratio_over_lower_point = dist_from_lower_point /
size_of_point_range;

                        float3 color_range = colors[i] - colors[i - 1];

                        float3 color_contribution = color_range *
ratio_over_lower_point;

                        float3 new_color = colors[i - 1] + color_contribution;
                        return new_color;
                    }
                }
                return colors[0];
            }

            float distsq(float2 a, float2 b)
            {
                float area_of_effect_size = _Diameter;
                return  pow(max(0.0, 1.0 - distance(a, b) / area_of_effect_size),
2.0);
            }
            fixed4 frag(v2f i) : SV_Target
            {
                fixed4 col = tex2D(_MainTex, i.uv);

                initalize();
                float2 uv = i.uv;
                uv = uv * 4.0 - float2(2.0,2.0);
                float totalWeight = 0.0;
                for (float i = 0.0; i < _HitCount; i++)
                {
                  float2 work_pt = float2(_Hits[i * 3], _Hits[i * 3 + 1]);
                  float pt_intensity = _Hits[i * 3 + 2];

                  totalWeight += 0.5 * distsq(uv, work_pt) * pt_intensity * _Strength
* (1 + sin(_Time.y * _PulseSpeed));
                }
                if (totalWeight <= _FadeIntencivity)
                {
                    return col + float4(getHeatForPixel(totalWeight),0);
                }
                else
                {
                    return col + float4(getHeatForPixel(totalWeight),
_ShotIntencivity);
                }
            }
            ENDCG
        }
    }
}
```

ДОДАТОК Е
Порівняння кутів голівок ключки для аналізу ударів

Driver	TM Max	TM P790	TM P770	TM 7MB	Cobra	Cobra TEC	Cobra Tour	Callaway B21
3-Wood	9, 12°M				9°, 10.5°			9°, 10.5°, 12.5°
5-Wood	15°, 16.5°				14.5°			15°
7-Wood	18°				18.5°			18°
2-Iron	21°				22.5°			21°
3-Iron								
4-Iron		19.0°	19.5°	21.0°		19.0°	20.0°	
5-Iron	19.0°	21.0°	22.5°	24.0°	19.0°	21.0°	23.0°	19.0°
6-Iron	21.5°	23.5°	25.5°	27.0°	21.0°	23.0°	26.0°	22.0°
7-Iron	25.0°	26.5°	29.0°	31.0°	24.0°	26.0°	29.0°	25.0°
8-Iron	28.5°	30.5°	33.0°	35.0°	27.5°	29.5°	33.0°	29.0°
9-Iron	32.5°	35.0°	37.0°	39.0°	32.0°	34.0°	37.0°	33.0°
PW	38.0°	40.0°	41.5°	43.0°	37.0°	39.0°	41.0°	38.0°
GW/AW	43.5°	45.0°	46.0°	47.0°	42.5°	44.0°	45.0°	43.0°
SW	49.0°	50.0°	51.0°		48.0°	49.0°	50.0°	48.0°
LW	54.0°				54.0°			54.0°
3H	59.0°							60.0°
4H	19.0°				19.0°			19.0°
5H	22.0°				21.0°			21.0°

Callaway Epic	Callaway Apex	Callaway A12	Callaway Pro	Mizuno JPX921	Mizuno Forged	Mizuno Tour	Mizuno MP-20	Mizuno MP-20
9,10.5								
13.5°,15								
18°								
21°								
								16.0°
		19.0°	20.5°				21.0°	19.0°
	20.0°	21.0°	23.0°	19.0°	21.0°	24.0°	24.0°	22.0°
	23.0°	23.5°	26.0°	22.0°	24.0°	27.0°	27.0°	25.0°
	26.0°	26.5°	29.0°	25.0°	27.0°	30.0°	30.0°	28.0°
	30.0°	30.5°	33.0°	29.0°	31.0°	34.0°	34.0°	32.0°
	34.0°	34.5°	37.0°	34.0°	35.0°	38.0°	38.0°	36.0°
	38.0°	38.5°	41.0°	39.0°	40.0°	42.0°	42.0°	41.0°
	43.0°	43.0°	45.0°	44.0°	45.0°	46.0°	46.0°	46.0°
	48.0°	48.0°	50.0°	49.0°	50.0°	51.0°		
				54.0°				
				59.0°				
		19.0°	20.0°					
		21.0°	23.0°					