Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота бакалавра

на тему: «Розроробка мобільної гри "Тисяча" із застосуванням штучного інтелекту»

Виконав: студент групи ІП320-2
Спеціальність 121 «Інженерія програмного забезпечення»
Чуприн Юрій Сергійович
(прізвище та ініціали)
Керівник к.т.н., доцент, Мала Ю.А
(науковий ступінь, вчене звання, прізвище та ініціали)
Рецензент _____
(місце роботи)
_____
(посада)
_____
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2024

АНОТАЦІЯ

Чуприн Ю.С. Розробка мобільної гри "Тисяча" із застосуванням штучного інтелекту.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавр за спеціальністю 121 «Інженерія програмного забезпечення». – Університет митної справи та фінансів, Дніпро, 2024.

Дана кваліфікаційна робота присвячена розробці мобільної карткової гри яка має назву “Тисяча” із застосуванням штучного інтелекту. Розглянуті методи та технічні засоби, необхідні для реалізації мобільної гри "Тисяча". Сформульовані вимоги до програмного забезпечення. Було проведено проектування та розробку гри. Отримані результати мають практичне значення, оскільки сприяють поліпшенню геймплею, забезпечуючи задоволення потреб користувачів та підвищення їх інтересу до гри. Розроблена гра є надійною та ефективною, використовує передові технології, такі як штучний інетелект і експертні системи , та є важливим кроком у покращенні якості мобільних ігор у різних жанрах.

Розроблена гра має серверну та клієнтську частини. Гра зберігає дані про гравців на сервері, та дає змогу продовжити партію в будь-який момент. Навчений ШІ забезпечує високий рівень супротивника, який вимагає стратегічного мислення від гравця. Цей додаток має сучасний та зручний дизайн.

Ключові слова: розробка, штучний інтелект, карткова гра, карткова гра "Тисяча".

ANNOTATION

Chupryn Y.S. Development of the mobile game "Thousand" using artificial intelligence.

Qualification work for obtaining a bachelor's degree in the specialty 121 "Software Engineering." – University of Customs and Finance, Dnipro, 2024.

This qualification work is dedicated to the development of the mobile card game called "Thousand" using artificial intelligence. The methods and technical tools necessary for implementing the mobile game "Thousand" are considered. The software requirements have been formulated. The design and development of the game have been carried out. The results obtained have practical significance, as they contribute to the improvement of gameplay, meeting user needs, and increasing their interest in the game. The developed game is reliable and efficient, utilizing advanced technologies such as artificial intelligence and expert systems, and represents an important step in enhancing the quality of mobile games across various genres.

The developed game has both server-side and client-side components. The game stores player data on the server, allowing the game to be resumed at any moment. The integrated AI provides a high level of opponent difficulty, requiring strategic thinking from the player. This application features a modern and user-friendly design.

Keywords: development, artificial intelligence, card game, Thousand card game.

ЗМІСТ

ВСТУП

Галузь мобільних ігор швидко розвивається, пропонуючи користувачам все більш захопливі та інноваційні рішення. Одним з перспективних напрямків є інтеграція технологій штучного інтелекту (ШІ) в ігровий процес, що дозволяє створювати інтелектуальних віртуальних супротивників, адаптивні ігрові механіки та персоналізований досвід для кожного гравця. Завдяки використанню алгоритмів машинного навчання та глибоких нейронних мереж, ігрові боти можуть аналізувати ходи гравця, вчитися на його стратегіях і тактиках, а також динамічно змінювати складність та стиль гри. Це відкриває нові можливості для створення захопливих ігрових сценаріїв та унікальних ігрових механік, особливо в карткових іграх на взяття, де ШІ може генерувати оптимальні ходи та стратегії в залежності від ситуації на ігровому полі та карт на руках гравців.

Актуальність дослідження зумовлена зростаючим попитом на ігри з елементами ШІ, здатними забезпечити складний і збалансований ігровий процес, який підлаштовується під рівень навичок користувача. Впровадження штучного інтелекту дозволяє уникнути монотонності та передбачуваності, властивих традиційним карткових іграм на взяття з жорстко запрограмованою логікою.

Розумні алгоритми можуть аналізувати стиль гри користувача, передбачати його наміри та розробляти контрстратегії. ШІ може динамічно коригувати складність гри, враховуючи досвід та майстерність гравця, щоб забезпечити оптимальний баланс між викликом і задоволенням від гри. Крім того, ШІ дозволяє створювати різноманітні сценарії розподілу карт та ситуації на ігровому полі, що підвищує варіативність та реіграбельність гри. Таким чином, інтеграція штучного інтелекту в мобільні карткові ігри на взяття має великий потенціал для покращення ігрового досвіду та залучення ширшої аудиторії гравців.

Метою дослідження є розробка мобільної версії популярної карткової гри "Тисяча" з інтегрованим штучним інтелектом, який забезпечить захопливий і збалансований ігровий процес для користувачів різного рівня підготовки.

Крім того, інтеграція штучного інтелекту в мобільні карткові ігри відкриває нові можливості для соціальної взаємодії та змагань між гравцями. ШІ може виступати в ролі віртуального партнера або суперника, надаючи гравцям можливість практикувати свої навички та стратегії в будь-який час, навіть за відсутності реальних опонентів. Це сприяє створенню динамічної та захопливої ігрової спільноти, де гравці можуть кидати виклик один одному та покращувати свої здібності в змаганнях проти ШІ.

Об'єктом дослідження є процес розробки мобільної гри "Тисяча" на платформі Android з використанням технологій штучного інтелекту.

Предметом дослідження є створення інтелектуальної системи, здатної аналізувати ігрову ситуацію, пропонувати оптимальні ходи та адаптувати складність гри відповідно до рівня гравця.

Для досягнення поставленної мети необхідно вирішити такі завдання:

1. Аналіз існуючих мобільних ігор у жанрі "Тисяча" та вивчення методів інтеграції ШІ в ігрові додатки.
2. Вибір оптимальних технологій та інструментів для розробки гри, включаючи мову програмування, середовище розробки та бібліотеки машинного навчання.
3. Проектування архітектури гри та системи штучного інтелекту для аналізу ігрових ситуацій та прийняття рішень.
4. Розробка та налагодження мобільної гри "Тисяча" з використанням обраних технологій.

Практичне значення кваліфікаційної роботи полягає у створенні мобільної гри "Тисяча" з інтегрованим штучним інтелектом, який забезпечить захопливий і збалансований ігровий процес для користувачів різного рівня підготовки. Реалізація цього проекту дозволить створити інноваційну мобільну гру, що поєднує класичний ігровий процес "Тисячі" з сучасними технологіями штучного інтелекту. Це забезпечить гравцям унікальний досвід, в якому вони зможуть змагатися з інтелектуальним супротивником, рівень складності якого буде динамічно підлаштовуватися під їхні навички.

Для забезпечення високої якості та ефективності роботи системи штучного інтелекту в грі "Тисяча" буде проведено ґрунтовне дослідження та аналіз різних алгоритмів і підходів до навчання ШІ. Зокрема, будуть розглянуті методи Resilient Propagation, які дозволяють агенту ШІ навчатися на основі взаємодії з ігровим середовищем та отримання зворотного зв'язку у вигляді винагород або штрафів. Також будуть досліджені алгоритми пошуку оптимальних рішень, такі як мінімаксний алгоритм та альфа-бета відсікання, які допоможуть ШІ приймати найкращі рішення в умовах невизначеності та неповної інформації.

Досягнення поставлених завдань дозволить створити зручну, функціональну та цікаву мобільну гру, яка допоможе користувачам насолоджуватися захопливим ігровим процесом, адаптованим до їхніх навичок та вподобань. Застосування сучасних технологій та передових методик розробки забезпечить високу якість кінцевого продукту, простоту у використанні та інтуїтивний інтерфейс. Ретельне тестування та впровадження заходів безпеки гарантуватимуть захист конфіденційних даних та безперебійну роботу системи. Загалом, ця мобільна гра стане надійним помічником у світі цифрових розваг і допоможе користувачам отримати максимальне задоволення від ігрового процесу.

Робота складається зі вступу, 3-х розділів, висновків, списку використаних джерел з 20 найменувань. Обсяг роботи 59 сторінок, 17 рисунків

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1. Огляд сучасного стану мобільних ігор та штучного інтелекту

Мобільні ігри стали одним із найпопулярніших і найприбутковіших напрямків в ігровій індустрії. Зараз вони не просто розвага, а й важливе джерело заробітку та популяризації контенту через численні платформи, такі як Apple App Store, Google Play та інші. Минуло багато часу, щоб мобільні ігри досягли нинішнього рівня складності та якості, і з кожним роком вони стають дедалі деталізованішими та технологічно досконалішими.

Над створенням провідних мобільних ігор працюють уже не окремі розробники, а цілі ігрові студії та об'єднання компаній, створюючи унікальний, опрацьований і захопливий контент. Такі гіганти, як Supercell ("Clash of Clans", "Clash Royale"), King ("Candy Crush Saga"), Niantic ("Pokémon Go"), Tencent ("Honor of Kings", "PUBG Mobile") та багато інших, задають тренди і встановлюють нові стандарти в мобільній ігровій індустрії. Здебільшого єдиним обмеженням мобільних ігор залишаються самі пристрої, на яких вони запускаються, проте з кожним роком продуктивність смартфонів і планшетів зростає, даючи змогу реалізовувати дедалі амбітніші проєкти.

Паралельно з розвитком мобільних ігор, розвиток штучного інтелекту (ШІ) також вражає і відкриває нові можливості в різних сферах. З кожним роком ШІ стає все розумнішим і досконалішим. Якщо ще три роки тому мало хто міг припустити, що ШІ досягне таких масштабів, що кожна людина зможе створити свій власний ШІ-помічник за допомогою нескладних інструментів і сервісів, то зараз це стало реальністю.

Наразі багато нейронних мереж і систем машинного навчання використовуються для створення картин, побудови складних планів, корекції даних та їх обслуговування. Прикладами видатних досягнень у цій царині є ChatGPT від OpenAI, здатний генерувати різноманітні тексти на будь-яку тему, і нейронна мережа DALL-E від тієї ж компанії, що малює картини, які неможливо

відрізнити від робіт досвідчених художників. Також існують нейронні мережі, здатні писати музику, такі як MuseNet від Google і Jukedeck.

Можемо побачити скільки людей користуються сервисами нейронних мереж кожного дня (див. рис 1.1)

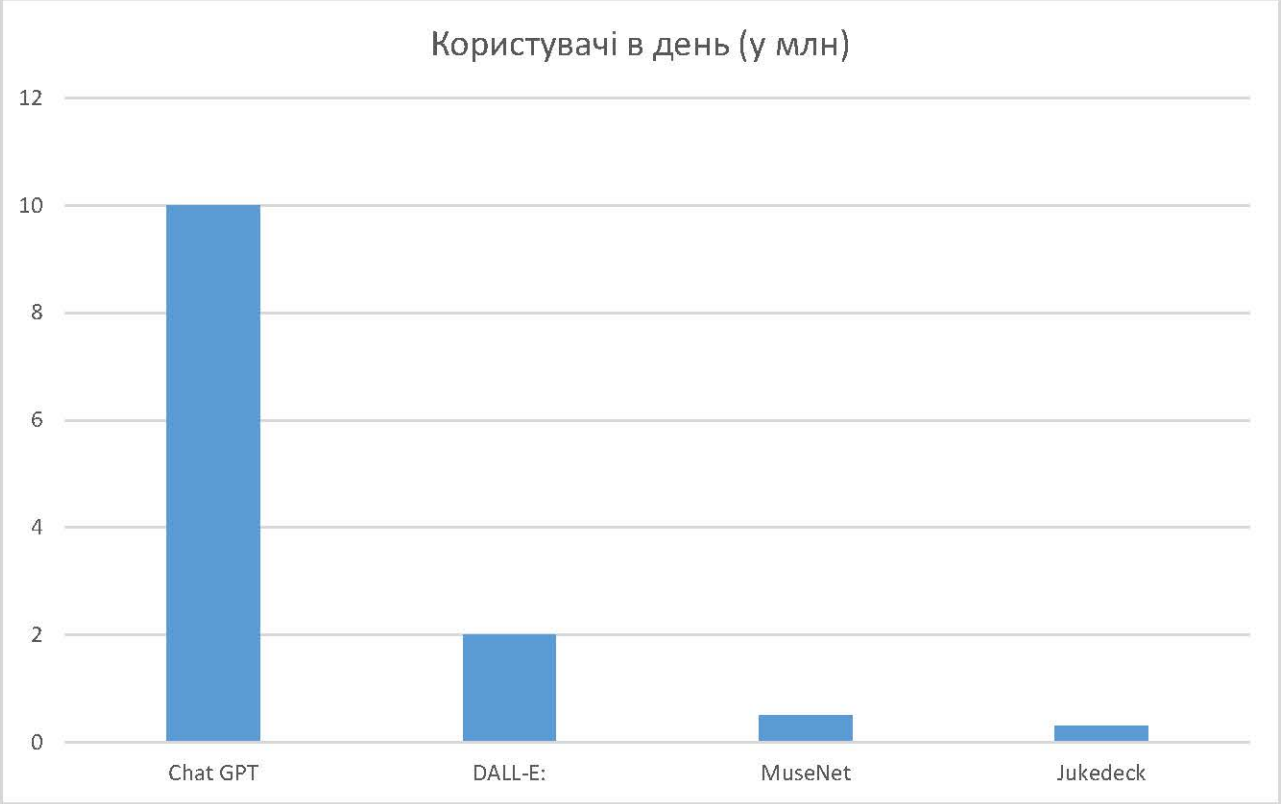

Рисунок 1.1 – Статистика відвідування нейронних мереж

У медичній сфері AI-системи вже активно використовують для аналізу медичних зображень, виявлення патологій і постановки діагнозів. Так, AI-алгоритми від компаній на кшталт Enlitic здатні розпізнавати ознаки раку на рентгенівських знімках із точністю, яку можна порівняти з досвідченими рентгенологами. А стартап Viz.ai розробив систему виявлення інсультів за комп'ютерною томографією, яку успішно застосовують у клініках по всьому світу.

Однак стрімкий розвиток ШІ породжує й етичні питання. Існують побоювання, що просунуті системи ШІ можуть у майбутньому витіснити людей з багатьох професій, призвести до втрати робочих місць. Також є ризик того, що

ШІ використовуватимуть для створення нових видів зброї або для стеження і порушення недоторканності приватного життя.

Ще однією проблемою є можливість ШІ генерувати дезінформацію, "фейкові" новини і навіть цілі фальшиві особистості в соціальних мережах за допомогою технологій синтезу мови, зображень і відео. Такі "глибокі підробки" вже зустрічалися і можуть використовуватися для маніпуляцій громадською думкою [1].

Незважаючи на ці виклики, науковці та розробники ШІ працюють над етичними рамками та системами запобіжників, які дадуть змогу безпечно використовувати потенціал штучного інтелекту, мінімізуючи ризики. Створюються спеціальні комітети з етики ШІ, розробляються принципи "відповідального ШІ". Завдяки проривам у галузі ШІ люди отримують якіснішу медичну допомогу, прискорюються наукові дослідження, спрощується аналіз великих даних. ШІ дає змогу автоматизувати рутинні завдання, вивільняючи людські ресурси для більш творчої та затребуваної діяльності

Незважаючи на те, що ШІ перебуває лише на початковому етапі свого розвитку, він уже може виконувати завдання, наближені до людських можливостей, а в деяких галузях навіть перевершує людей. Технології ШІ впроваджуються в багато сфер, від охорони здоров'я і наукових досліджень до творчості та розваг, відкриваючи нові горизонти і піднімаючи людство на новий рівень розвитку.

У найближчому майбутньому ми можемо очікувати конвергенції мобільних ігор і технологій штучного інтелекту, що призведе до створення більш інтелектуальних, адаптивних і захопливих ігрових світів. ШІ може бути використаний для генерації процедурного контенту, створення більш складніших і реалістичніших ігрових персонажів, адаптації ігрового процесу до вподобань кожного окремого гравця та багато чого іншого.

Уже зараз деякі мобільні ігри, як-от "Pokemon Go" і "Ingress", використовують технології доповненої реальності та геолокації, віщуючи майбутнє злиття цифрового і фізичного світів. Компанії на кшталт DeepMind, що

належить Alphabet, працюють над створенням штучного інтелекту, здатного грати в складні стратегічні ігри на рівні професійних кіберспортсменів [2].

Симбіоз мобільних ігор і ШІ може відкрити нову еру інтерактивних розваг, де ігровий процес буде настільки ж захопливим, наскільки й нескінченно різноманітним та адаптивним. Межі між віртуальною реальністю і фізичним світом стануть дедалі більш розмитими, а ігри перетворяться на повноцінні цифрові всесвіти, що живуть за своїми законами і правилами.

Майбутнє мобільних ігор і штучного інтелекту обіцяє бути воістину захопливим і відкриває безмежні можливості для творчості, інновацій та розваг. Ми перебуваємо лише на початку цього шляху, але вже зараз можна передбачити, що прийдешні зміни докорінно змінять не тільки ігрову індустрію, а й наше сприйняття реальності.

1.2. Актуальність розробки мобільної гри "Тисяча" з використанням штучного інтелекту

Розробка карткової гри "Тисяча" з вбудованим штучним інтелектом є значним досягненням, що демонструє здатність навченої нейронної мережі приймати оптимальні рішення в ігровій ситуації з реальним суперником-людиною. ШІ здатний вдосконалювати свої навички на основі аналізу попередніх ходів і виправлення допущених помилок, відпрацьовуючи мільйони можливих комбінацій.

Ключовою особливістю цього проекту є інтеграція штучного інтелекту в карткову гру "Тисяча", що раніше не було реалізовано. Це робить розробку унікальною та інноваційною у своїй галузі. ШІ здатний здійснювати блискавичні обчислення, прораховуючи потенційні комбінації карт і можливі результати гри на багато ходів вперед.

Нейронна мережа навчена на колосальному обсязі ігрових даних, що дозволяє їй оперативно адаптуватися до різних ігрових ситуацій і стратегій суперника. Система аналізує поточний стан ігрового поля, залишені карти в

колоді та на руках, а також враховує попередні ходи гравця.

У процесі розробки особлива увага приділялася пошуку оптимальних архітектур нейронних мереж і функцій активації для максимально точних обчислень. Були протестовані різні конфігурації, перш ніж вдалося досягти бажаного рівня продуктивності ШІ. Інтегрований у гру алгоритм здатний миттєво підраховувати кількість очок за кожну можливу комбінацію карт і вибирати найбільш вигідний варіант ходу. Він також уміє прораховувати довгострокові наслідки кожного рішення, передбачати пастки та уникати їх [3].

Завдяки здатності до навчання, нейромережа постійно вдосконалює свою гру, витягуючи уроки з кожного поєдинку з людиною. Це забезпечує постійно зростаючий рівень складності для гравця, вимагаючи від нього постійного вдосконалення стратегічного мислення [4].

Інтеграція штучного інтелекту в карткову гру "Тисяча" відкриває нову главу в історії настільних ігор, демонструючи можливості ШІ в прийнятті виважених рішень і адаптації до складних ігрових ситуацій в режимі реального часу. Крім того, це створює унікальні умови для гравців, які можуть навчатися та вдосконалювати свої навички, граючи проти висококваліфікованого суперника. Таким чином, проект "Тисяча" з вбудованим ШІ не тільки розважає, але й навчає, роблячи кожну партію цікавою та корисною для особистісного зростання гравців.

1.3. Постановка задачі дослідження та розробки мобільної гри "Тисяча"

Дослідження ставить за мету детально вивчити методи розробки ігор та впровадження штучного інтелекту (ШІ) у ігровий процес. Основне завдання полягає в аналізі існуючих підходів та виборі найбільш ефективних методик для подальшого застосування у власному проєкті. Підсумком роботи стане розробка технічного завдання (ТЗ), яке стане основою для створення гри з інтегрованим ШІ. Цей ШІ повинен буде втілити в собі всі найкращі практики та позбутися непотрібних або таких, що погіршують продуктивність компонентів.

Оскільки гра планується для платформи Android, вибір середовища розробки є критично важливим етапом. Серед безлічі доступних інструментів необхідно розглянути наступні варіанти :

1. Android Studio - Офіційне середовище розробки для Android від Google, з потужною інтеграцією та інструментами для створення додатків.

2. Unity з плагіном для Android - Крос-платформна ігрова рушій, з можливістю розробки для Android використовуючи спеціальний плагін.

3. Unreal Engine з підтримкою Android - Потужний ігровий рушій з вбудованою підтримкою Android та передовою графікою.

4. Godot Engine - Безкоштовний і відкритий ігровий рушій з підтримкою Android та багатьох мов програмування.

5. Xamarin - Платформа для крос-платформної розробки додатків, що дозволяє використовувати C# для створення мобільних ігор.

6. Corona SDK - Легке та потужне середовище розробки, особливо популярне для створення 2D ігор на Android.

7. GameMaker Studio - Візуальне середовище розробки, що дозволяє легко створювати ігри без глибоких знань програмування.

Кожне з цих середовищ має свої особливості та переваги, які будуть детально розглянуті для вибору найбільш підходящого.

Крім того, важливою частиною дослідження є аналіз фреймворків для інтеграції штучного інтелекту. Для ефективної роботи з ШІ та його інтеграції в ігровий процес слід розглянути кілька фреймворків:

1. Encog Machine Learning Framework - Легкий та потужний фреймворк машинного навчання, що підтримує Java та C#.

2. TensorFlow - Потужний фреймворк для машинного навчання від Google, з підтримкою мобільних платформ.

3. PyTorch - Популярний фреймворк для машинного навчання з Python, що забезпечує високу продуктивність.

4. DL4J (Deeplearning4j) - Бібліотека машинного навчання для JVM, що підтримує кілька мов програмування.

5. Weka - Всеосяжна колекція інструментів машинного навчання з відкритим вихідним кодом.

6. Accord.NET - Бібліотека машинного навчання для .NET Framework, з широким спектром алгоритмів.

7. ML Kit від Google - Бібліотека з різними моделями штучного інтелекту для мобільних пристроїв на Android та iOS.

Кожен з цих фреймворків буде оцінений за різними критеріями, такими як легкість інтеграції, продуктивність, підтримка мобільних платформ та доступність документації. Такий підхід дозволить вибрати оптимальні інструменти для розробки гри з ефективним використанням штучного інтелекту на платформі Android.

Методи оптимізації продуктивності ігор на мобільних пристроях:

1. Стискання текстур та ресурсів - зменшення розміру та покращення завантаження графічних елементів.

2. Оптимізація коду - покращення ефективності за рахунок оптимізації алгоритмів та структур даних.

Використання кешування - зберігання даних у пам'яті для швидшого доступу.

3. Застосування прогресивних рендеринг-технік - покращення візуальної якості без значного впливу на продуктивність.

4. Оптимізація завантаження ресурсів - ефективне керування пам'яттю та завантаженням ресурсів гри.

5. Управління пам'яттю - ефективне використання пам'яті пристрою та звільнення невикористаних ресурсів.

6.Використання спеціалізованих бібліотек для оптимізації - застосування сторонніх бібліотек для поліпшення продуктивності.

Тестування та налагодження мобільних ігор:

1. Інструменти для автоматизованого тестування - засоби для автоматизації тестування функціональності та продуктивності.

2. Симулятори та емулятори Android - програмні та апаратні рішення для емуляції середовища Android.

3. Використання профайлерів продуктивності - інструменти для аналізу та оптимізації продуктивності додатків.

4. Фреймворки для тестування користувацького інтерфейсу - бібліотеки для автоматизації тестування взаємодії з інтерфейсом.

5. Інтеграція з системами контролю

1.4. Висновки до першого розділу

Аналіз предметної області мобільних ігор та штучного інтелекту (ШІ) показав, що обидві сфери перебувають у фазі стрімкого розвитку і взаємного впливу. Мобільні ігри стали однією з найприбутковіших галузей індустрії розваг, завдяки покращенню технологій і зростанню потужності мобільних пристроїв. Водночас, штучний інтелект відкриває нові можливості в різних сферах, включаючи медицину, творчість та ігри. Інтеграція ШІ в мобільні ігри обіцяє створення більш інтелектуальних і адаптивних ігрових світів, що сприятиме підвищенню якості ігрового процесу.

Проект з розробки мобільної гри "Тисяча" з використанням ШІ демонструє потенціал нейронних мереж для прийняття оптимальних рішень у картковій грі з реальними суперниками. Інтеграція ШІ в гру "Тисяча" є унікальним та інноваційним рішенням, що дозволяє ШІ оперативно адаптуватися до різних ігрових ситуацій і стратегій суперника, покращуючи якість гри та забезпечуючи гравцям цікаві та навчальні партії.

Основні завдання дослідження включають вибір оптимальних методик для розробки мобільних ігор та впровадження ШІ, а також аналіз інструментів для розробки ігор на платформі Android. Вибір середовища розробки та фреймворків для інтеграції ШІ є критично важливими етапами для успішного впровадження проекту.

Розробка мобільної гри з використанням ШІ потребує ретельної оптимізації продуктивності, включаючи стискання текстур та ресурсів, оптимізацію коду, використання кешування та спеціалізованих бібліотек.

Тестування та налагодження гри забезпечуються автоматизованими інструментами та симуляторами Android, що дозволяє виявляти та усувати проблеми на ранніх етапах розробки.

На основі цього аналізу можна зробити наступні висновки:

1. Мобільні ігри та штучний інтелект мають великий потенціал для взаємного розвитку, що може призвести до створення нових видів інтерактивних розваг.

2. Інтеграція ШІ в мобільні ігри дозволяє створювати більш реалістичні та адаптивні ігрові світи, покращуючи якість ігрового процесу.

3. Проект з розробки гри "Тисяча" з використанням ШІ є значним кроком вперед у галузі карткових ігор, демонструючи можливості нейронних мереж для прийняття оптимальних рішень у реальному часі.

4. Успішна розробка мобільної гри потребує вибору оптимальних інструментів та методик, а також ретельної оптимізації продуктивності та тестування.

Ці висновки підкреслюють важливість інтеграції новітніх технологій у мобільні ігри та відкривають нові горизонти для розвитку ігрової індустрії.

РОЗДІЛ 2. ОГЛЯД ІСНУЮЧИХ РОЗВ'ЯЗКІВ ТА ВИБІР МЕТОДІВ ВИРІШЕННЯ

2.1. Аналіз існуючих мобільних ігор типу "Тисяча"

"Тисяча" - це популярна карткова гра, особливо в країнах пострадянського простору. Вона доступна у вигляді мобільних додатків, які дають змогу грати як із ботами, так і з іншими гравцями онлайн. Можемо розглянути популярні мобільні версії цієї каркової гри:

- Тисяча (1000) від JagPlay - це один із найпопулярніших додатків для гри в "Тисячу". Він підтримує гру як із ботами, так і з реальними людьми. Додаток має зручний інтерфейс і можливість налаштувати різні правила гри.
- Тисяча Онлайн (1000 Online) - цей додаток пропонує онлайн-режим, що дає змогу грати з друзями або випадковими гравцями. Також є режим гри із ботами. Додаток має хороший рейтинг і позитивні відгуки користувачів.
- Тисяча (1000) від R-Soft LLC - ще один популярний додаток з можливістю гри як в онлайн-режимі, так і проти комп'ютера. Має безліч налаштувань і опцій для зміни правил гри відповідно до вподобань гравців.
- Тисяча Онлайн (Thousand Online) від Appscraft - цей застосунок надає користувачам можливість грати в "Тисячу" онлайн, а також офлайн проти ботів. Додаток має приємний інтерфейс і широкий функціонал.
- Тисяча від Skill Cap - цей застосунок вирізняється гарною графікою і різноманітними налаштуваннями для гри. Підтримує мультиплеєрний режим, а також гру проти комп'ютера

Детально розглянемо і проаналізуємо мобільні ігри "Тисяча (1000)" від JagPlay і R-Soft LLC. Обидві гри пропонують багатий досвід для гравців, але мають свої унікальні особливості та механіки.

Тисяча (1000) від JagPlay

1. Ігровий процес: Підтримка як одиночної гри проти ШІ, так і мультиплеєрних матчів проти реальних гравців. Можливість створювати приватні кімнати для гри з друзями. Різні рівні складності ШІ, що робить гру цікавою як для новачків, так і для досвідчених гравців.
2. Інтерфейс і дизайн: Простий та інтуїтивно зрозумілий інтерфейс, що полегшує навігацію і розуміння гри. Можливість налаштування зовнішнього вигляду карт і столу.
3. Додаткові функції: Система рейтингів і досягнень, що стимулює гравців повертатися до гри і покращувати свої навички. Внутрішньоігрова валюта, яку можна заробити, граючи і перемагаючи в матчах, а потім використовувати для різних поліпшень і налаштувань. Чат для спілкування з іншими гравцями під час гри.
4. Мультиплеєр: Можливість грати онлайн з гравцями з усього світу. Підтримка різних режимів гри, включно з турнірами.

Тисяча (1000) від R-Soft LLC

1. Ігровий процес: Підтримка як одиночної гри проти ботів, так і гри онлайн з іншими гравцями. Регульовані правила гри, що дає змогу адаптувати гру під уподобання конкретних гравців.
2. Інтерфейс і дизайн: Приємна графіка і користувацький інтерфейс, який сприяє комфортній грі. Можливість персоналізації аватарів і профілів.
3. Додаткові функції: Система рейтингів і статистики, що дає змогу відстежувати прогрес і досягнення гравця. Внутрішньоігрові покупки для отримання різних бонусів і поліпшень. Докладні навчальні матеріали та поради для гравців-початківців.
4. Мультиплеєр: Можливість грати онлайн із друзями та випадковими гравцями. Щотижневі турніри та змагання з призами, що додає елемент змагання і зацікавлює гравців.

Порівняння та відмінні риси:

Інтерфейс і користувацький досвід:

- JagPlay: Простий і зручний інтерфейс, який дозволяє легко почати гру навіть новачкам. Основний акцент на легкості використання.
- R-Soft LLC: Більш деталізована графіка і можливість персоналізації, що робить гру більш індивідуальною і візуально привабливою.

Мультиплеєр і соціальні функції:

- JagPlay: Чат і приватні кімнати, що робить гру більш соціальною і дозволяє легко грати з друзями.
- R-Soft LLC: Тури та змагання з призами, що додає змагальний елемент і стимулює гравців брати участь у регулярних заходах.

Додаткові функції:

- JagPlay: Внутрішньоігрова валюта і досягнення, що стимулює прогрес і надає гравцям додаткові цілі.
- R-Soft LLC: Докладні навчальні матеріали та поради, що робить гру доступною і зрозумілою для нових гравців.

Обидві мобільні ігри "Тисяча (1000)" від JagPlay і "Тисяча (1000)" від R-Soft LLC пропонують низку корисних функцій, що роблять ігровий процес зручнішим і захопливішим. Однак, залежно від уподобань гравця, можна виділити ключові переваги кожної з них. JagPlay забезпечує легкий старт і соціальний аспект гри, тоді як R-Soft LLC виділяється великими навчальними матеріалами та змагальними елементами. Зрештою, вибір між цими іграми залежить від того, які функції гравець вважає найбільш важливими і цікавими для себе.

2.2. Вивчення методів та підходів застосування штучного інтелекту в ігровій розробці

Компанія DeepMind, що належить Google, розробила кілька ігор для

демонстрації можливостей своїх нейронних мереж. Однією з таких ігор є AlphaGo, де штучний інтелект може грати в го на професійному рівні, аналізуючи позиції за допомогою глибоких нейронних мереж.

Мобільні ігри з розпізнаванням зображень

Така мобільні гра, як Prisma , використовують згорткові нейронні мережі для аналізу фотографій, зроблених користувачами. Ця нейромережа може розпізнавати об'єкти, стилі та створювати на їхній основі нові художні зображення [5].

Було обрано дві гри які застосовують Штучний Інтелект у процессі користування :

AlphaGo - Ця гра демонструє можливості нейронних мереж під час гри в го - одну з найскладніших настільних ігор.

Інтеграція:

1. Навчання нейромережі на величезних наборах даних зіграних партій професійних гравців для виявлення закономірностей і стратегій.

2. Використання комбінації згорткових нейромереж для аналізу поточної позиції на дошці та оцінювання ймовірностей наступного ходу.

3. Застосування підкріплювального навчання шляхом численних ігор нейромережі самої з собою для підвищення рівня гри.

4. Інтеграція отриманої в результаті тренування нейромережі в ігровий рушій для прийняття рішень і здійснення ходів під час партії з людиною.

Ключові моменти: масивні датасети, глибоке навчання, самонавчання, інтеграція з ядром гри.

Prisma - Мобільний застосунок, що використовує згорткові нейромережі для стилізації зображень у різні художні стилі.

Інтеграція:

1. Попереднє навчання нейромереж на величезних базах художніх зображень різних стилів для виявлення особливостей стилів.

2. Інтеграція навчених нейромереж у мобільний застосунок для опрацювання завантажених користувачем фотографій.

3. Використання GPU для прискорення обчислень нейромереж на мобільних пристроях.

4. Реалізація користувацького інтерфейсу для вибору стилів і параметрів стилізації.

5. Хмарний сервіс для оновлення та донавчання нейромереж на нових даних.

Ключові моменти: переднавчання, інтеграція в мобільні додатки, GPU-прискорення, хмарні сервіси, користувацький інтерфейс.

Після детального аналізу ігор із ШІ було знайдено детальні методики та засоби інтеграцій ШІ до мобільних ігор

1. Попередня підготовка даних:

- Збір і розмітка великих обсягів даних, релевантних для навчання нейромереж (ігрові партії, зображення, звуки тощо)
- Очищення і перетворення даних у формати, придатні для навчання (numpy, TFRecords)
- Розбиття даних на тренувальні, валідаційні та тестові набори

2. Побудова та навчання нейромереж:

- Вибір архітектури нейромережі (згорткові, рекурентні, автокодувальники та ін.) залежно від завдання
- Визначення функцій втрат, оптимізаторів, гіперпараметрів
- Навчання на виділених обчислювальних ресурсах (GPU/TPU кластери) з використанням фреймворків TensorFlow, PyTorch
- Застосування технік регуляризації, перерахунку ваг для боротьби з перенавчанням
- Використання передових методик - переносного навчання, змішаних моделей тощо.

3. Оптимізація для мобільних пристроїв:

- Квантизація ваг і активацій нейромережі для зниження вимог до пам'яті/обчислень

- Видалення надлишкових шарів, застосування технік дистиляції знань
- Використання прискорювачів (GPU, NPU, DSP) і низькорівневих бібліотек (TensorFlow Lite, Core ML)
- Офлайн і онлайн навчені моделі для балансування якості та продуктивності

4. Інтеграція в ігровий рушій:
   - "Обгортання" навченої моделі у відповідний для рушія формат (C++ API, Unity Plugin)
   - Передача відповідних вхідних даних з гри в нейромережу (стан гри, пікселі тощо)
   - Отримання вихідних даних нейромережі (дії, класифікація об'єктів тощо) і їхня інтерпретація рушієм
   - Багатопотокове оброблення для зниження затримок
   - Вбудовування користувацького інтерфейсу для управління ШІ (налаштування складності тощо)
5. Безперервне поліпшення:
   - Збір ігрових даних у режимі онлайн для подальшого донавчання моделей
   - Використання хмарних сервісів і регулярні оновлення для розгортання оновлених нейромереж
   - Ітераційне поліпшення користувацького досвіду на основі аналітики та відгуків
   - Ключовими факторами успішної інтеграції є якісні вихідні дані, потужні обчислювальні ресурси, оптимізація під мобільні пристрої та грамотне вбудовування в ігровий процес.

2.3. Вибір технологій та інструментів розробки для реалізації мобільної гри "Тисяча" з використанням штучного інтелекту

Під час розроблення мобільної гри один із перших важливих кроків - вибір мови програмування (див. рис 2.1), середовища розробки (див. рис 2.2) і, якщо потрібно, фреймворка для підключення нейронних мереж(див. рис 2.3). Правильний вибір цих інструментів має вирішальне значення для успішної реалізації проєкту.

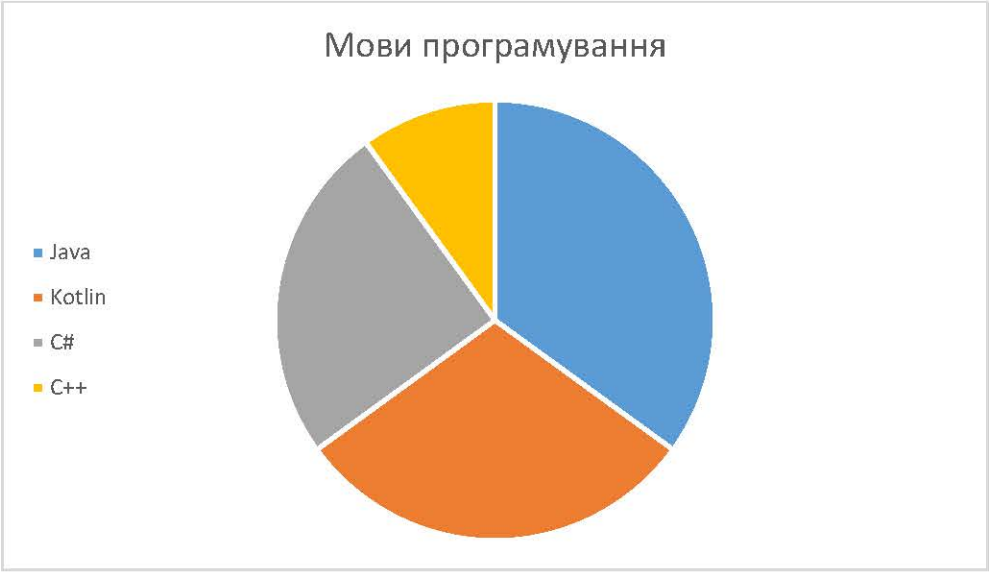

Рисунок 2.1 - Популярність мов програмування для мобільних пристроїв

Як і очікувалося, Java посідає лідируючу позицію з часткою використання 35%, що не дивно, з огляду на її давню історію і широке застосування в Android-розробці.

Однак варто відзначити зростаючу популярність Kotlin, який займає 30% ринку. Kotlin, розроблений компанією JetBrains, поступово завойовує визнання завдяки своїй сучасній конструкції, лаконічному синтаксису і сумісності з Java. Багато розробників віддають перевагу використанню Kotlin для нових проєктів, особливо в галузі Android-розробки [6].

C# і C++ також присутні на діаграмі з частками 25% і 10% відповідно. Ці

мови широко використовуються в рушіях ігор, таких як Unity і Unreal Engine, що робить їх популярним вибором для створення крос-платформних мобільних ігор.

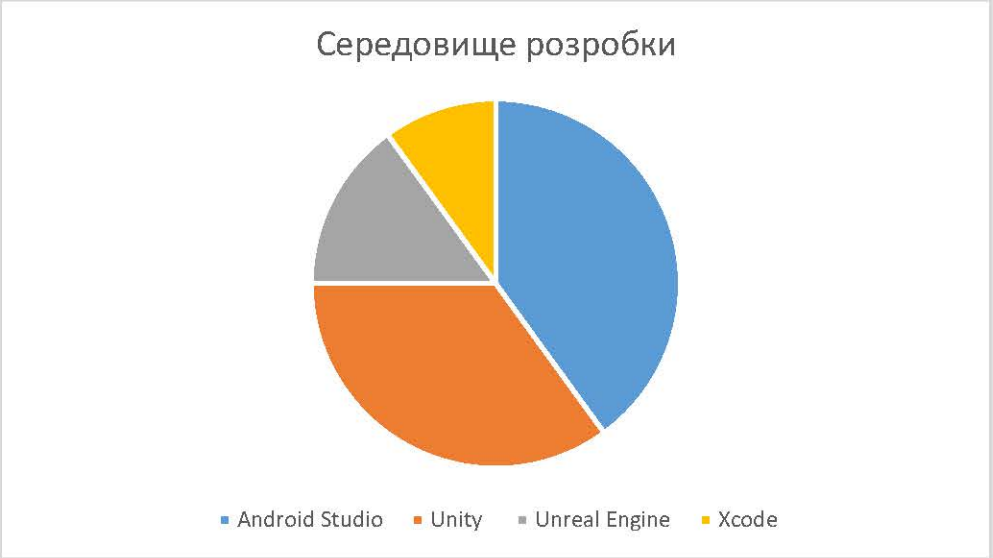

Рисунок 2.2 - Популярність середовищ розробки для мобільних пристроїв

Лідером, буз усяких сумнівів є Android Studio з часткою використання 40%. Це офіційне інтегроване середовище розробки (IDE) для створення додатків на Android, підтримуване самою Google. Її популярність пояснюється тісною інтеграцією з Android SDK, багатим набором інструментів і можливостей для налагодження та профілювання.

На другому місці перебуває Unity з 35% ринку. Цей крос-платформний ігровий рушій високо цінується за свою гнучкість, велику документацію і спільноту розробників. Unity дає змогу створювати ігри для безлічі платформ, включно з мобільними пристроями [7].

Unreal Engine займає 15% ринку. Це потужний рушій, орієнтований на створення високоякісних 3D-ігор і симуляцій. Хоча він часто асоціюється з ПК і консольними іграми, Unreal Engine також підтримує розробку мобільних ігор.

Останнє місце посідає, Xcode від Apple становить 10% ринку. Це офіційне середовище розробки для створення додатків для пристроїв на базі iOS, macOS та інших продуктів Apple. Його використання обов'язкове для розробників, які створюють ігри для платформи iOS.

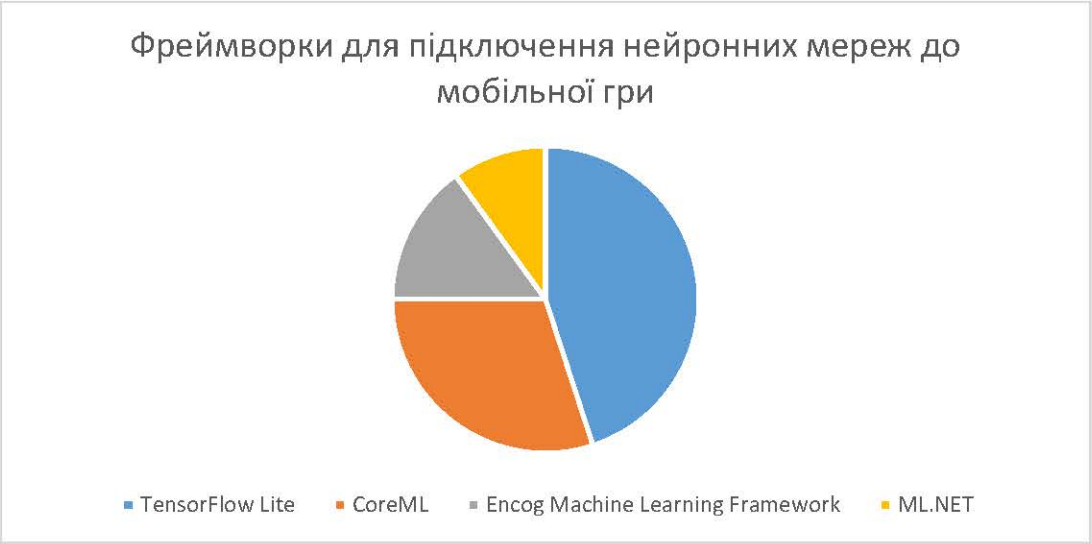

Рисунок 2.3. - Популярність фреймворків для підключення нейронних мереж

Лідером, як видно з діаграми, є TensorFlow Lite з часткою використання 45%. Цей фреймворк, розроблений Google, є полегшеною версією TensorFlow, оптимізованою для роботи на мобільних пристроях і вбудованих системах. Його популярність пояснюється високою продуктивністю, кросплатформеністю і хорошою інтеграцією з Android.

На другому місці знаходиться CoreML від Apple з часткою 30%. Цей фреймворк призначений для вбудовування моделей машинного навчання в застосунки для iOS, iPadOS, macOS і watchOS. CoreML забезпечує високу продуктивність і простоту інтеграції для розробників, які працюють в екосистемі Apple [8].

Encog Machine Learning Framework, згаданий у вашому прикладі, займає 15% ринку. Це кросплатформний фреймворк з відкритим вихідним кодом, що підтримує різні алгоритми машинного навчання, включно з нейронними мережами. Його гнучкість і портативність роблять його привабливим вибором для деяких розробників [9].

Нарешті, ML.NET від Microsoft становить 10% ринку. Цей фреймворк надає інструменти для вбудовування моделей машинного навчання в додатки .NET, включно з іграми для Windows

Після ретельного аналізу різних варіантів і всебічного порівняння

технологій, було прийнято рішення використати мову програмування Kotlin у поєднанні з Encog Machine Learning Framework і середовищем розробки Android Studio для створення мобільної версії популярної карткової гри "Тисяча" з елементами штучного інтелекту.

Цей вибір став результатом глибокого занурення в особливості кожної з технологій, що розглядаються, і ретельної оцінки їхнього потенціалу для реалізації поставлених завдань. Було проведено великі дослідження, вивчили безліч альтернатив і зважили всі "за" і "проти", перш ніж зробити остаточний вибір на користь цього стека.

Комбінація Kotlin, Encog і Android Studio є потужним і гнучким інструментарієм, здатним забезпечити високу продуктивність, безпеку коду і багаті можливості для інтеграції штучного інтелекту в мобільний додаток. Кожен із цих компонентів робить свій унікальний внесок у загальне рішення, доповнюючи і посилюючи сильні сторони один одного.

Мова програмування Kotlin ідеально підходить для розробки гри "Тисяча" з таких причин:

- Безпека і лаконічність: Код гри має бути надійним і легко читабельним. Kotlin з його суворою типізацією і захистом від нульових посилань допомагає уникнути поширених помилок і спрощує підтримку коду.
- Функціональне програмування: Багато аспектів гри, як-от визначення ходів, підрахунок очок і логіка ігрового процесу, можна ефективно реалізувати з використанням функціонального підходу, який добре підтримується Kotlin.
- Сумісність із Java: Завдяки сумісності з Java, у грі можна використовувати наявні бібліотеки та фреймворки для Android, що прискорює розробку та розширює функціональні можливості.

Encog Machine Learning Framework дасть змогу інтегрувати елементи штучного інтелекту в гру "Тисяча", такі як:

- Аналіз ігрової ситуації: Використовуючи алгоритми машинного навчання, як-от нейронні мережі або дерева рішень, можна навчити

систему аналізувати поточний стан гри і пропонувати оптимальні ходи.

- Адаптивний рівень складності: Encog може допомогти у створенні адаптивного штучного інтелекту, який буде підлаштовуватися під рівень гравця, забезпечуючи збалансований і захопливий ігровий процес.
- Розпізнавання образів: Бібліотеку також можна використовувати для розпізнавання ігрових карт або інших елементів інтерфейсу, що дасть змогу реалізувати більш природну взаємодію з грою.

Android Studio, будучи офіційним середовищем розробки для Android, надає такі переваги для створення мобільної версії гри "Тисяча":

- Зручність розробки: Android Studio володіє широким набором інструментів і функцій, які спрощують процес розробки, налагодження та розгортання додатків для Android, що прискорює створення гри.
- Інтеграція з Android SDK: Завдяки тісній інтеграції з Android SDK, розробники можуть використовувати всі можливості платформи, як-от оброблення сенсорних подій, робота з камерою та іншими функціями, що можуть бути корисними в грі.
- Підтримка Kotlin: Повна підтримка Kotlin в Android Studio забезпечує безшовну інтеграцію цієї мови програмування, що дає змогу використовувати її переваги на всіх етапах розробки.

Об'єднуючи ці технології, розробники отримують потужний інструментарій для створення захопливої та інтелектуальної мобільної версії карткової гри "Тисяча". Kotlin забезпечить безпеку та лаконічність коду, Encog Machine Learning Framework додасть елементи штучного інтелекту, а Android Studio надасть зручне середовище розробки, оптимізоване для створення мобільних додатків на Android. Це поєднання дасть змогу створити гру, яка не тільки відповідатиме очікуванням сучасних користувачів, а й пропонуватиме їм унікальний ігровий досвід з адаптивним рівнем складності та інтелектуальним ігровим процесом.

2.4. Висновки до другого розділу

Під час аналізу мобільних ігор типу "Тисяча" було розглянуто кілька популярних додатків, таких як "Тисяча (1000)" від JagPlay та R-Soft LLC, які пропонують багатий ігровий досвід та мають свої унікальні особливості та механіки. Основні відмінності між ними полягають в інтерфейсі, користувацькому досвіді, мультиплеєрних і соціальних функціях, а також додаткових можливостях, таких як внутрішньоігрова валюта, досягнення, навчальні матеріали та змагальні елементи.

Далі був розглянутий досвід застосування штучного інтелекту в ігровій розробці на прикладі ігор AlphaGo та Prisma. В обох випадках було описано методи інтеграції нейронних мереж, включаючи підготовку даних, побудову та навчання нейромереж, оптимізацію для мобільних пристроїв, інтеграцію в ігровий рушій та безперервне поліпшення моделей ШІ.

На основі проведеного аналізу технологій та інструментів розробки було зроблено вибір мови програмування (Kotlin), фреймворка для машинного навчання (Encog Machine Learning Framework) та середовища розробки (Android Studio) для реалізації мобільної версії гри "Тисяча" з використанням штучного інтелекту. Цей вибір обґрунтовано зручністю та потужністю цих інструментів для вирішення поставлених завдань, а також їхньою сумісністю і популярністю серед розробників мобільних ігор.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

3.1. Проектування архітектури мобільної гри "Тисяча"

У сучасному світі мобільна ігрова індустрія швидко розвивається, і вибір відповідних технологій є критично важливим для успішної реалізації ігрового проєкту. При розробці мобільної гри "Тисяча" було необхідно ретельно розглянути різні варіанти та вибрати оптимальний стек технологій, який забезпечить високу продуктивність, простоту розробки, розширюваність та можливість інтеграції штучного інтелекту.

Важливість ретельного планування та проектування архітектури не можна переоцінити, адже це фундамент, на якому будується весь програмний продукт. Архітектура гри повинна забезпечувати ефективну взаємодію між усіма компонентами, бути гнучкою для майбутніх змін і розширень, а також підтримувати високу продуктивність для забезпечення плавного ігрового досвіду [10].

Ще одним ключовим аспектом є реалізація штучного інтелекту, який додасть грі динамічності та викликів для користувача. Використання сучасних інструментів і фреймворків для створення AI допоможе досягти природної та реалістичної поведінки опонентів, що значно підвищить інтерес і задоволення від гри.

Крім того, створення інтуїтивно зрозумілого та привабливого інтерфейсу користувача є важливим елементом успішної гри. Взаємодія користувача з грою має бути максимально простою та зрозумілою, що сприятиме позитивному враженню від гри [11].

Також не менш важливим є забезпечення надійного збереження даних та управління користувачами. Використання сучасних технологій для збереження даних дозволить підтримувати високу надійність і безпеку, а також забезпечить можливість відновлення даних у разі необхідності.

Загалом, успішна реалізація гри "Тисяча" вимагає комплексного підходу до вибору технологій, проектування архітектури, розробки інтерфейсу та

інтеграції різних компонентів. Це дозволить створити якісний продукт, який відповідатиме сучасним вимогам і задовольнить очікування користувачів.

Kotlin був обраний в якості основного мови програмування для цього проєкту з кількох причин. Як сучасна мова, розроблена компанією JetBrains, він пропонує ряд переваг, таких як лаконічний і виразний синтаксис, безпека типів, повна сумісність з Java та екосистемою Android. Крім того, Kotlin активно розвивається і підтримується Google, що робить його перспективним вибором для розробки додатків під Android. Однією з ключових переваг Kotlin є його здатність зменшувати кількість шаблонного коду, що підвищує ефективність розробки та зменшує кількість помилок. Можливість інтеграції з існуючим кодом на Java також спрощує перехід на нову мову та забезпечує високу гнучкість.

Enngo Framework був обраний як рішення для впровадження штучного інтелекту в гру. Цей фреймворк надає широкі можливості для створення ігрових ІІ-агентів, які можуть навчатися і приймати рішення на основі алгоритмів машинного навчання. Інтеграція Enngo дозволить створити більш складних і реалістичних ігрових противників, підвищуючи залученість і інтерес користувачів. Завдяки використанню машинного навчання, ІІ-агенти зможуть адаптуватися до стилю гри користувача, забезпечуючи унікальний досвід кожного разу, коли гравець починає нову гру. Це значно підвищить реіграбельність та додасть глибину геймплею.

Jetpack Compose був обраний для розробки користувацького інтерфейсу. Це сучасний інструмент для створення UI, який спрощує процес розробки завдяки декларативному підходу. Jetpack Compose дозволяє створювати гнучкі та динамічні інтерфейси з меншою кількістю коду, ніж традиційні методи розробки. Завдяки цьому фреймворку можна легко змінювати і тестувати інтерфейс, що прискорює цикл розробки та підвищує якість кінцевого продукту. Jetpack Compose також тісно інтегрується з іншими компонентами Android, такими як ViewModel та LiveData, що спрощує управління станом додатку та реакцію на зміни в даних.

Firebase був обраний для збереження даних і управління користувачами. Це потужна платформа, яка надає безліч сервісів для розробників, включаючи Firebase Realtime Database, Firebase Authentication та Firebase Cloud Messaging. Firebase Realtime Database дозволяє зберігати та синхронізувати дані між користувачами в режимі реального часу, що забезпечує актуальність даних і миттєве оновлення ігрового процесу. Firebase Authentication спрощує процес аутентифікації користувачів, забезпечуючи підтримку різних методів входу, таких як електронна пошта, соціальні мережі та анонімний вхід. Це підвищує безпеку і зручність для користувачів.

Firebase також забезпечує масштабованість і надійність, що є критично важливим для додатків з великою кількістю користувачів. Використання хмарних технологій дозволяє обробляти великі обсяги даних без необхідності в налаштуванні та обслуговуванні власної інфраструктури, що значно зменшує витрати на розробку та підтримку додатку [12].

Таким чином, вибір Kotlin, Enngo Framework, Jetpack Compose та Firebase забезпечує комплексний підхід до розробки мобільної гри "Тисяча", дозволяючи створити високоякісний, продуктивний та масштабований продукт. Завдяки сучасним технологіям та інструментам розробки, ми зможемо забезпечити користувачам захоплюючий ігровий досвід, інтуїтивно зрозумілий інтерфейс та надійне збереження даних. Цей підхід дозволить нам не тільки створити успішну гру, але й забезпечити її подальший розвиток і підтримку, відповідаючи на вимоги та очікування сучасної ігрової індустрії.

У цьому проекті було вирішено використовувати архітектуру MVVM
Існують також інший варінт, архітектура MVC, Розглянемо переваги та недоліки обох архітектур, у таблиці 3.1

Таблиця 3.1- Порівняння MVC і MVVM

Критерій	MVC	MVVM
Розділення обов'язків	Модель представляє дані, View відображає користувацький інтерфейс, а контролер керує логікою та взаємодією між ними.	Модель представляє дані, View відображає користувацький інтерфейс, а ViewModel діє як посередник між ними, реалізуючи логіку та обробляючи команди користувача.
Зв'язок між View та Model	Контролер опосередковує взаємодію між View та Model.	ViewModel діє як посередник між View та Model, забезпечуючи більшу роз'єднаність та полегшуючи тестування.
Простота розробки та підтримки	Простий та зрозумілий патерн для невеликих додатків.	Більш складний патерн, але забезпечує кращу роз'єднаність та тестованість компонентів.
Підтримка зв'язування даних	Зв'язування даних зазвичай реалізується вручну або за допомогою додаткових бібліотек.	Природно підтримує зв'язування даних, що спрощує розробку користувацького інтерфейсу.
Тестованість	Тестування контролерів може бути складним через їх тісний зв'язок з	ViewModel легше тестувати, оскільки він не залежить

	View та Model.	безпосередньо від View чи Model.
Мобільна розробка	MVC широко використовується в мобільній розробці та підтримується багатьма фреймворками.	MVVM набуває популярності в мобільній розробці, особливо у сучасних фреймворках, таких як Xamarin та Flutter.

Один із ключових аспектів MVVM - це здатність створювати високо модульний код. Завдяки чіткому поділу на компоненти Model, View і ViewModel, розробники можуть працювати над кожною частиною додатка незалежно один від одного. Це спрощує супровід і розширення програми, оскільки зміни в одній частині коду рідко зачіпають інші. Наприклад, якщо потрібно змінити логіку роботи гри або додати новий тип об'єктів, це можна зробити, не зачіпаючи призначений для користувача інтерфейс, що істотно скорочує ризик виникнення помилок [13].

Ще одним важливим аспектом MVVM є можливість ефективного тестування. Оскільки бізнес-логіка застосунку міститься у ViewModel, розробники можуть легко писати модульні тести для цієї частини коду. Це дає змогу виявляти та виправляти помилки на ранніх стадіях розроблення, покращуючи якість і надійність застосунку.

Гнучкість MVVM проявляється і в можливості легко адаптувати застосунок під різні платформи та пристрої. Оскільки користувацький інтерфейс (View) є окремим компонентом, його можна легко адаптувати під різні екрани та роздільну здатність, зберігаючи при цьому загальну логіку (див. рис. 3.1). Таким чином, застосунок може бути розгорнуто на різних платформах з мінімальними змінами.

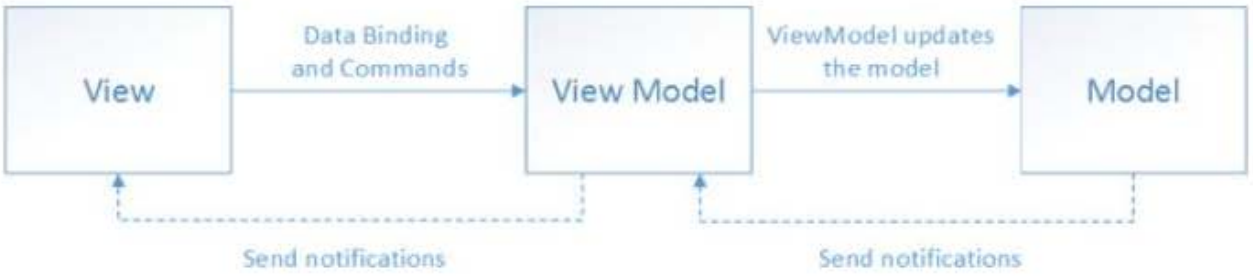

Рисунок. 3.1 – Логіка MVVM

Нарешті, MVVM сприяє поліпшенню читабельності та розуміння коду. Завдяки явному поділу на компоненти Model, View і ViewModel, структура застосунку стає яснішою і зрозумілішою для інших розробників. Це прискорює процес розробки та спрощує співпрацю в команді.

Загалом, використання MVVM у розробці мобільних ігор і застосунків забезпечує безліч переваг, роблячи процес розробки ефективнішим, надійнішим і гнучкішим [14].

Детальніше розглянемо компоненти архітектури мобільної гри "Тисяча" яка використовує шаблон MVVM.

Компоненти архітектури :

1. User Interface (UI)
    - Jetpack Compose: Фреймворк для побудови UI, що дозволяє створювати динамічні і гнучкі інтерфейси.
    - Екрани гри: Включають головний екран, екран налаштувань, екран гри, екран результатів.
2. ViewModel
    - GameViewModel: Клас, який буде містити логіку гри та управління станом UI. Забезпечує комунікацію між View та Model.
3. Model
    - GameLogic: Клас, що містить логіку гри "Тисяча". Включає механізми підрахунку очок, перевірки умов перемоги, управління картами тощо.
    - Player: Клас, що представляє гравця. Містить інформацію про карти гравця, набрані очки тощо.

- AIPlayer: Клас, що представляє штучного інтелектуального гравця, реалізованого за допомогою Encog Machine Learning Framework.

4. Data Management
    - Firebase: Служить для збереження даних користувачів, включаючи налаштування гри та результати. Firebase Authentication буде використовуватися для управління користувачами.

Взаємодія компонентів :

1. Інтерфейс користувача (UI) взаємодіє з ViewModel для відображення даних та отримання введених користувачем команд.
2. ViewModel обробляє ці команди і оновлює Model відповідно до логіки гри.
3. Model оновлює стан гри і повідомляє ViewModel про зміни.
4. ViewModel передає оновлений стан назад до UI для відображення.

Деталізація компонентів :

1. User Interface
    - MainScreen: Відображає головне меню гри, включаючи кнопки для початку нової гри, налаштувань та перегляду результатів.
    - GameScreen: Відображає поточний стан гри, карти гравця, набрані очки та дії гравця.
    - ResultScreen: Відображає результати гри, включаючи підсумковий рахунок та переможця.
    - SettingsScreen: Дозволяє користувачу змінювати налаштування гри.
2. ViewModel
    - GameViewModel: Містить методи для запуску нової гри, обробки дій гравця, оновлення стану гри та взаємодії з Model. Також забезпечує збереження прогресу гри у Firebase.
3. Model
    - GameLogic: Реалізує основну логіку гри, управління картами, перевірку правил та умов гри.

- Player: Містить інформацію про гравця, такі як його карти, набрані очки та інші стани.
- AIPlayer: Реалізує логіку штучного інтелекту, використовуючи Encog для прийняття рішень.

4. Data Management
   - Firebase Realtime Database: Використовується для збереження налаштувань користувачів та результатів гри.
   - Firebase Authentication: Використовується для управління користувачами, забезпечуючи безпеку доступу до даних.

3.2. Реалізація штучного інтелекту для опонентів у грі з використанням Encog Machine Learning Framework

Штучний інтелект може бути використаний для створення інтелектуальних опонентів, які здатні адаптуватися до різних стратегій гравців та забезпечити гідне супротивництво. У даній пояснювальній записці представлено реалізацію системи штучного інтелекту для гри "Тисяча" з ШІ, створеної за допомогою Encog Machine Learning Framework.

Нейронна мережа є потужним інструментом для вирішення складних завдань, таких як оцінювання карт у грі "Тисяча". Вона складається з декількох шарів нейронів, які взаємодіють між собою для вивчення та виявлення закономірностей у даних.

Кожен нейрон у мережі є простою обчислювальною одиницею, яка приймає набір вхідних сигналів, обробляє їх за допомогою активаційної функції та видає вихідний сигнал. Активаційні функції відіграють важливу роль у нейронних мережах, оскільки вони вводять нелінійність, що дозволяє мережі ефективно апроксимувати складні функції [15].

У методі createNeuralNetwork() створюється базова мережа, де вхідний шар має розмірність inputSize, що дорівнює 24, а вихідний шар має розмірність outputSize, що дорівнює 1. Вхідний шар відповідає за прийняття вхідних даних,

в цьому випадку карт у грі "Тисяча". Вихідний шар представляє оцінку, зроблену мережею для поточної руки карт.

Між вхідним та вихідним шарами можуть бути розташовані один або декілька прихованих шарів. Ці шари відповідають за виявлення складних закономірностей та ознак у вхідних даних. Кількість прихованих шарів та нейронів у них впливає на здатність мережі навчатися та узагальнювати інформацію [16].

Вибір функцій активації в нейронних мережах відіграє важливу роль у їх ефективності та здатності вирішувати різноманітні завдання. Розглянемо порівняльну таблицю різних функцій активації та проаналізуємо їх переваги та недоліки.

Таблиця 3.2- Таблиця порінянь функцій активації

Функція активації	Діапазон значень	Переваги	Недоліки
Сигмоїд (Sigmoid)	(0, 1)	- Нелінійна функція	- Проблема зникаючого градієнта
		- Гладка	- Не центрована навколо нуля
		- Обмежений вихідний діапазон	- Насичення при великих позитивних і негативних значеннях
Гіперболічний тангенс (Tanh)	(-1, 1)	- Нелінійна функція	- Проблема зникаючого градієнта
		- Центрована навколо нуля	- Насичення при великих

			позитивних і негативних значеннях
		- Гладка	
ReLU (Rectified Linear Unit)	(0, +∞)	- Нелінійна функція	- Проблема мертвих нейронів
		- Немає проблеми зникаючого градієнта	- Не центрована навколо нуля
		- Швидше збігається	- Проблема вибухаючого градієнта для деяких ситуацій
Лінійна (Linear)	(-∞, +∞)	- Проста	- Лінійна, не вводить нелінійності
		- Немає насичення	
Гаусівська (Gaussian)	(0, +∞)	- Нелінійна функція	- Нецентрована навколо нуля
		- Локалізована активація	- Складніше обчислення

Розглянувши таблицю функцій активації, можна зробити висновок, що функція активації Sigmoid (сигмоїда) є найбільш придатною для використання у нашому випадку реалізації системи штучного інтелекту для гри "Тисяча" з ШІ.

Sigmoid функція має наступні переваги:

- Нелінійність: Сигмоїдна функція є нелінійною, що є важливою вимогою для нейронних мереж, оскільки вона дозволяє мережі вивчати складні нелінійні закономірності в даних. Лінійні функції не можуть ефективно моделювати складні відношення між вхідними та вихідними даними.
- Гладка функція: Сигмоїдна функція є гладкою та диференційованою, що дозволяє використовувати методи оптимізації, засновані на обчисленні градієнтів, такі як метод зворотного поширення помилки. Це забезпечує ефективний процес навчання нейронної мережі.
- Обмежений вихідний діапазон: Сигмоїдна функція має вихідний діапазон від 0 до 1, що робить її зручною для моделювання ймовірностей або пропорцій. У нашому випадку, коли ми оцінюємо руки карт у грі "Тисяча", вихідні значення можуть бути інтерпретовані як ймовірність того, що дана рука є гарною або поганою.

Хоча сигмоїдна функція має деякі недоліки, такі як проблема зникаючого градієнта та насичення при великих позитивних і негативних значеннях, ці недоліки можна пом'якшити за допомогою відповідних методів навчання та архітектури мережі.

Враховуючи переваги сигмоїдної функції та її широке використання у нейронних мережах, її вибір для реалізації системи штучного інтелекту для гри "Тисяча" є обґрунтованим та доцільним.

Після створення структури нейронної мережі, її необхідно навчити на набір даних. Цей процес називається навчанням, і він дозволяє мережі вивчати закономірності та зв'язки у даних, щоб згодом робити точні передбачення.

Для тренування мережі використовується алгоритм зворотнього поширення помилки (Resilient Propagation). Цей алгоритм є одним з методів навчання штучних нейронних мереж. Він використовується для оновлення ваг нейронних мереж під час процесу навчання з метою мінімізації функції втрат.

Основна ідея Rprop полягає в тому, щоб визначати напрямок оновлення ваг на основі знаку похідної функції втрат по вагам. Ваги, для яких похідна має той же знак, збільшуються або зменшуються на певну величину, відому як крок оновлення. Ваги, для яких похідна має протилежний знак, змінюються в інший бік, і крок оновлення для них зменшується. Це дозволяє алгоритму швидко збігатися до мінімуму функції втрат, оминуючи різкі піки та валі, що можуть виникнути в процесі навчання.

Одним із переваг Rprop є те, що він не вимагає налаштування гіперпараметрів, таких як швидкість навчання, як це потрібно в інших методах, наприклад, у методі зворотнього розповсюдження помилки зі сталою швидкістю навчання. Також Rprop може працювати добре навіть у випадках, коли градієнти великі або незначні, що робить його відмінним вибором для навчання нейронних мереж в різних умовах [17].

Дані для навчання, представлені у форматі BasicMLDataSet, використовуються для навчання мережі на основі переданих рук карт та фактичних балів. Кожен приклад даних складається з вхідного вектора, що представляє руку карт, та очікуваного виходу, який є оцінкою цієї руки.

Під час навчання мережа порівнює свій вихід з очікуваним виходом і обчислює помилку. Потім, використовуючи алгоритм зворотнього поширення помилки, ваги між нейронами коригуються таким чином, щоб зменшити цю помилку. Цей процес повторюється багато разів, поки мережа не досягне задовільного рівня точності або не буде виконано максимальну кількість ітерацій.

Було змодельовано дві моделі :

Перша модель - проста модель, що відображає логіку, за якою кожна карта має свою вартість, а оцінка визначається кількістю більш вагомих карт. Ця модель базується на незалежній цінності кожної карти і загальна оцінка розраховується шляхом підсумовування ваг окремих карт. Вхідний шар цієї моделі має розмірність 24 (по одному нейрону для кожної унікальної карти), а вихідний шар - розмірність 1 для визначення імовірності отримання оціночних

очок.

Перша модель містить лише один прихований шар (див. рис. 3.2).

```
private val inputSize = 24
private val outputSize = 1
net.addLayer(BasicLayer(ActivationSigmoid(), true, inputSize))
net.addLayer(BasicLayer(ActivationSigmoid(), true, 50))
net.addLayer(BasicLayer(ActivationSigmoid(), true, outputSize))
```

Рисунок 3.2 – Код першої моделі

Друга модель - складніша, яка відображає логіку наявності зв'язків між картами. У цій моделі необхідно враховувати не лише вагомість окремих карт, але й їхні комбінації. Вона базується на припущенні, що цінність карт не є незалежною, а визначається їх поєднанням і контекстом. Розмірність вхідного та вихідного шарів така ж, як і в першій моделі: 24 і 1 відповідно. Однак ця модель має кілька прихованих шарів для кращого виявлення складних взаємозв'язків між картами.

Використання кількох прихованих шарів у нейронній мережі дозволяє моделі краще розпізнавати та аналізувати ці складні взаємозв'язки. Кожен прихований шар може виявляти певні рівні абстракції в даних, поступово створюючи дедалі більш складні і точні представлення вихідних даних. Це може включати виявлення патернів, які важко розпізнати при простішому підході.

Друга модель вже має 5 скритих шарів (див. рис. 3.3)

```
net.addLayer(BasicLayer(ActivationSigmoid(), true, inputSize))
net.addLayer(BasicLayer(ActivationSigmoid(), true, 50))
net.addLayer(BasicLayer(ActivationSigmoid(), true, 100))
net.addLayer(BasicLayer(ActivationSigmoid(), true, 100))
net.addLayer(BasicLayer(ActivationSigmoid(), true, 50))
net.addLayer(BasicLayer(ActivationSigmoid(), true, inputSize))
net.addLayer(BasicLayer(ActivationSigmoid(), true, outputSize))
```

Рисунок 3.3 – Код другої моделі

Перша модель (див. рис. 3.4)

Рисунок 3.4 – Перша модель

Модель видає одне значення на будь-який набір вхідних даних. Після завершення навчання було згенеровано ще 1000 ігор для подальшого аналізу. Під час цього процесу був визначений середній коефіцієнт впевненості, з яким мережа вважає, що агент візьме свої очки. Також було визначено влучність передбачень мережі.

Для оцінки влучності спершу розраховували оцінку для кожної гри. Якщо ця оцінка перевищувала середній коефіцієнт впевненості, вважалося, що гравець візьме свої очки. Після цього проводилась сама гра, і отриманий результат порівнювався з передбаченням мережі. Таким чином, перевіряпось, чи справді агент взяв свої очки так, як це передбачала модель.

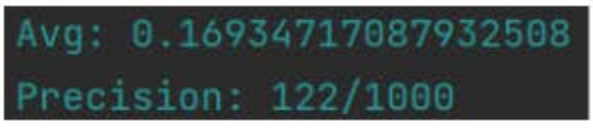

Рисунок 3.5 – Результат навчання першої моделі

Бачимо що результати м’яко кажучи погані, влучність 12.2%. Тобто мережа помилилась у більшості випадків (див. рис. 3.5).

Друга модель (див. рис. 3.6)

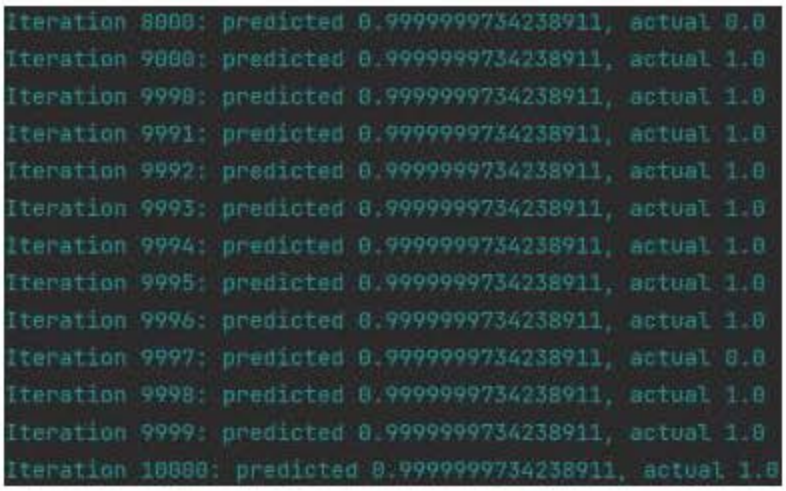

Рисунок 3.6 – Друга модель

Як можна побачити ця модель теж видає одна значення на будь-який набір вхідних даних, до навчання.

Після навчання :

Також порахуємо середній коефіцієнт впевненості та влучність (див. рис. 3.7).

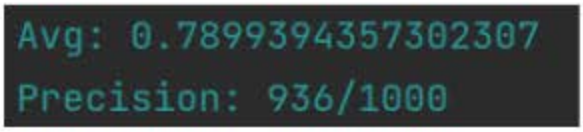

Рисунок 3.7 – Результат навчання другої моделі

Бачимо що модель добре впоралась, влучність 93.6%. Тобто мережа дуже добре передбачує імовірність взяття оціночних очок. Також можна вважати, коли передбачення моделі >0.7 агент гарантовано візьме як мінімум оціночні очки. Далі під час торгів якщо оцінка >0.7, то оціночні очки будуть залишатися як є. А коли оцінка буде <0.7, оціночні очки будуть множитись на цей коефіцієнт впевненості +0.2(при 60% впевненості вартість оцінки буде 80%). Щоб агент одразу не пасував, а спробував хоча б щось набрати.

На основі вищезазначених результатів, друга модель є набагато точнішою

та добре виконує поставлену задачу прогнозування взяття очок для агента

Після завершення процесу навчання, отримана нейронна мережа стає готовою до використання для оцінки поточних рук карт опонента. Метод predict() приймає список карт і повертає оцінку, зроблену мережею щодо цієї руки.

Під час гри система штучного інтелекту може використовувати цю оцінку для прийняття рішень, таких як розіграш або складання карт. Оцінка, зроблена нейронною мережею, базується на закономірностях та зв'язках, які вона вивчила під час навчання на наборі даних. Це дозволяє системі робити інформовані та розумні рішення, забезпечуючи гідний рівень супротивництва для гравців.

Крім того, оскільки нейронна мережа здатна узагальнювати інформацію, вона може ефективно оцінювати також нові, невідомі руки карт, на яких вона не була навчена. Це забезпечує гнучкість та адаптивність системи ШІ до різних ситуацій під час гри.

Крім навчання та використання мережі, клас CardEvaluator містить додаткові методи для підтримки основної функціональності.

Метод endTraining() викликається для завершення навчання та збереження навченої нейронної мережі у файл. Це дозволяє зберігати навчену модель та використовувати її в майбутньому без необхідності повторного навчання.

Метод cardToInputArray() відіграє важливу роль у підготовці вхідних даних для нейронної мережі. Він конвертує карти у вектори входу, які є зрозумілими для мережі. Ця конвертація дозволяє ефективно подавати дані на вхід мережі, забезпечуючи правильне сприйняття та обробку інформації.

Загалом, ця реалізація системи штучного інтелекту для гри "Тисяча" з використанням нейронної мережі демонструє потужність та гнучкість нейронних мереж у вирішенні складних завдань оцінювання та прийняття рішень.

3.3. Розробка користувацького інтерфейсу з використанням Jetpack Compose

Розробка користувацьких інтерфейсів (UI) є ключовим аспектом створення сучасних мобільних додатків. Останніми роками відбулося безліч змін у підходах та інструментах, які використовуються для проектування та реалізації UI. Одним із найбільш значущих кроків уперед стала поява Jetpack Compose - новітнього фреймворка від Google, призначеного для створення UI в додатках під Android.

Jetpack Compose дозволяє розробникам створювати користувацькі інтерфейси більш ефективно, використовуючи декларативний підхід, що значно спрощує процес розробки. Завдяки цьому фреймворку, UI можна створювати за допомогою коду, а не через традиційні XML-файли, що робить процес інтерактивнішим і більш інтегрованим із логікою додатку. Це забезпечує вищу гнучкість і дає змогу швидше вносити зміни та додавати нові функціональні можливості [18].

Він являє собою декларативний підхід до розроблення користувацьких інтерфейсів, що радикально відрізняється від традиційного імперативного програмування, яке використовували в Android протягом багатьох років. Декларативний підхід дає змогу розробникам описувати, який вигляд має мати інтерфейс у поточному стані, замість того щоб детально вказувати, які кроки потрібно зробити для його створення. Це робить код чистішим, легшим для читання та підтримки [19].

Таким чином, Jetpack Compose стає важливим кроком уперед у розробці UI для Android-додатків, пропонуючи сучасний, гнучкий та ефективний інструментарій для створення високоякісних користувацьких інтерфейсів.

Інтерфейс складається з трьох файлів: GamePage.kt, HomePage.kt та NewGamePage.kt, які містять композабельні функції для відображення користувацького інтерфейсу гри "Тисяча" у фреймворку Jetpack Compose.

GamePage.kt (див. рис 3.8, 3.9):

Це основна сторінка гри "Тисяча". Вона відображає список гравців та їхні поточні показники під час гри. Основні функції цього файлу:

1. Відображення TopBar з назвою поточного раунду, кнопкою повернення на головну сторінку та меню швидкого доступу.
2. Виведення карток гравців (PlayerCard) у вигляді сітки з використанням LazyVerticalGrid.
3. Картка гравця (PlayerCard) відображає ім'я гравця, його поточний бал, значення спеціальних показників (розчинення, порожні кидки, діжки), а також круговий індикатор загального балу з можливістю подвійного натискання для встановлення штрафу.
4. Реалізація функції вибору значення балу за раунд за допомогою Picker.
5. Відображення іконки "кидання гральних кісток" біля імені поточного дистриб'ютора.
6. Можливість встановлення декларанта та його заявленого балу за допомогою діалогового вікна.
7. Перегляд історії балів гравця у вигляді діалогового вікна з можливістю встановлення штрафу (-120 балів).
8. Функція розчинення (dissolution) для декларанта, якщо він не зміг виконати своє оголошення.
9. Перехід до наступного раунду з підтвердженням балів усіх гравців за поточний раунд.
10. Виведення діалогових вікон для збереження гри, відображення переможця та підказок.
11. Меню швидкого доступу з кнопками "Підказки", "Зберегти гру" та "Встановити декларанта".

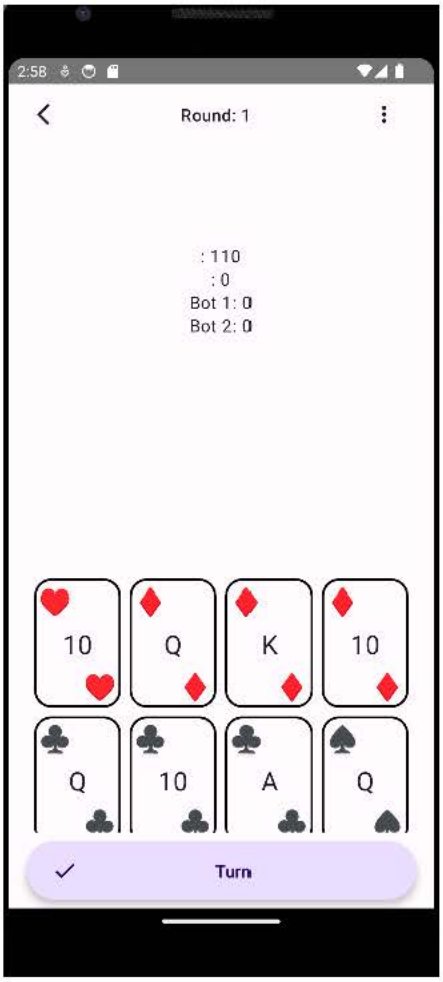

Рисунок 3.8 - Ігрова сторінка

Рисунок 3.9- Вікно із підказками

HomePage.kt (див. рис 3.10):

Ця сторінка є головною точкою входу в додаток. Вона надає можливість розпочати нову гру, завантажити збережену гру або перейти до гри соло. Основні функції:

1. Відображення TopBar з назвою додатку та кнопкою переходу до сторінки завантаження збережених ігор.
2. Кнопка "Продовжити гру" з'являється, якщо є збережена незавершена гра.
3. Кнопка "Гра соло" для переходу до режиму гри соло.
4. Кнопка "Нова гра" для створення нової багатокористувацької гри.
5. Кнопка "Завантажити збережені ігри" для переходу до сторінки завантаження збережених ігрових сесій.

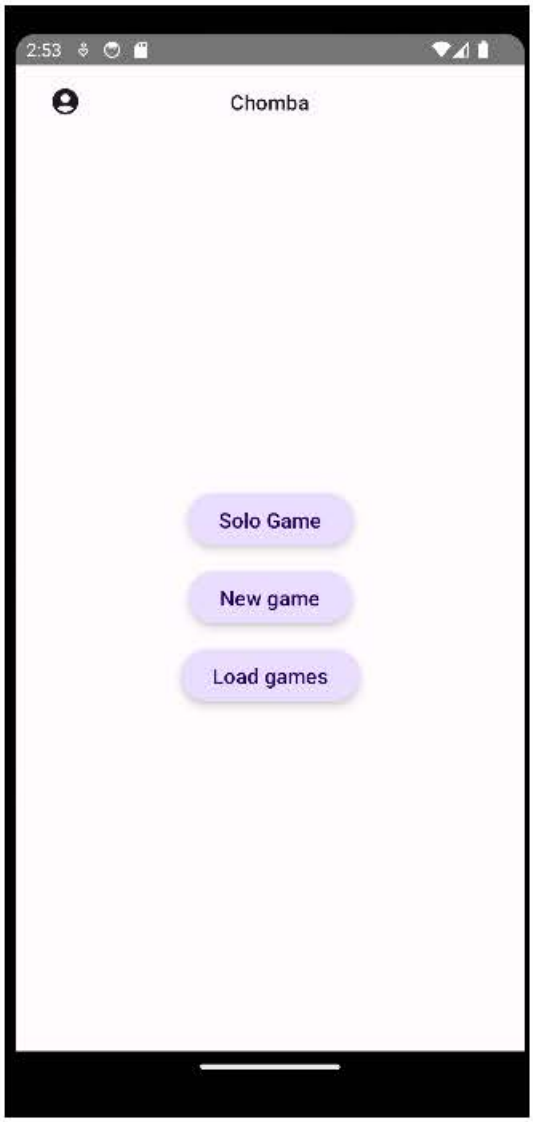

Рисунок 3.10- Головне меню

NewGamePage.kt (див. рис 3.11, 3.12):

1. Ця сторінка призначена для створення нової гри та налаштування гравців. Основні функції:
2. Відображення TopBar з назвою "Нова гра" та кнопкою повернення на головну сторінку.
3. Список гравців (PlayerItem) з можливістю редагування імен, вибору кольору та видалення гравця.
4. Використання OutlinedTextField для введення імені гравця.
5. Інтеграція ColorPicker для вибору кольору гравця за допомогою діалогового вікна.

6. Кнопка додавання нового гравця (з'являється, якщо кількість гравців менше 4).
7. Кнопка "Почати" (з'являється, якщо кількість гравців більше 1) для початку нової гри.

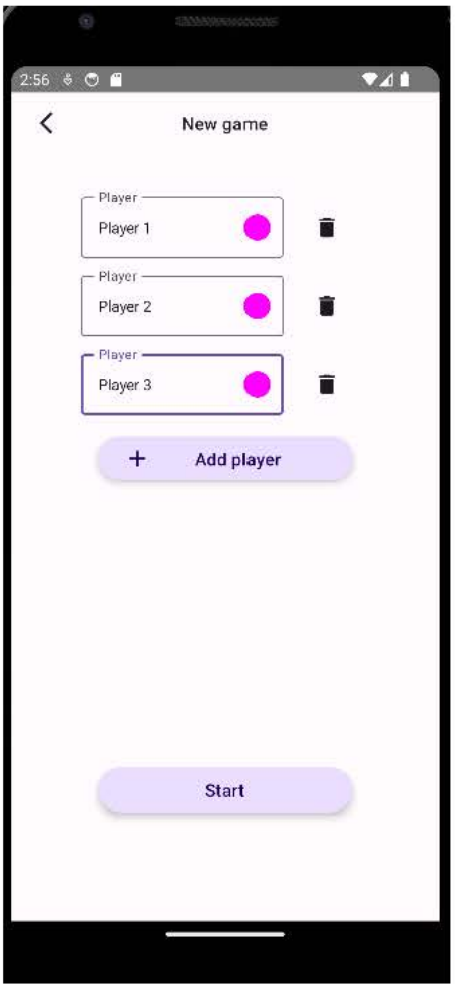

Рисунок 3.11 - Додавання гравців

Рисунок 3.12- Колір гравця

Компоненти, використані в цих файлах:

- TopBar: Верхня панель з назвою, кнопками навігації та меню.
- BasicTextButton, BasicIconButton: Кнопки з текстом або іконками для виконання дій.
- IconButton: Кнопка з іконкою для різних дій (видалення, вибір кольору тощо).
- AlertDialog: Діалогові вікна для відображення інформації, підтвердження дій та отримання введення від користувача.

- OutlinedTextField: Текстове поле для введення імен гравців.
- Picker: Компонент для вибору значень зі списку (наприклад, балів за раунд).
- ColorPicker: Діалогове вікно для вибору кольору гравця за допомогою HsvColorPicker.
- CircularChart: Круговий індикатор для відображення загального балу гравця.
- LazyVerticalGrid: Ефективна сітка для відображення списку елементів (гравців).
- AnimatedVisibility: Анімована видимість елементів інтерфейсу.
- DropdownMenu: Випадаюче меню для швидкого доступу до дій.

3.4. Інтеграція з Firebase для збереження даних та управління користувачами

Firebase - це комплексна платформа для розробки мобільних і веб-додатків, що надається Google. Вона пропонує широкий спектр інструментів і сервісів, які спрощують процес створення, розгортання та масштабування додатків.

Основними компонентами Firebase є:

1. Автентифікація: Firebase Authentication забезпечує просте і безпечне рішення для автентифікації користувачів, включно з підтримкою різних провайдерів, як-от електронна пошта, Google, Facebook, Twitter та інших.
2. Realtime Database: Це хмарна NoSQL база даних, яка синхронізує дані в режимі реального часу між клієнтами. Вона ідеально підходить для додатків, які потребують швидкого обміну даними.
3. Cloud Firestore: Ще одна хмарна NoSQL база даних, орієнтована на документи. Вона забезпечує багатшу модель даних, автоматичне масштабування і потужні запити.

4. Сховище: Firebase Storage надає безпечне сховище для файлів, таких як зображення, аудіо, відео та інші користувацькі дані.
5. Облачні функції: Дають змогу виконувати серверний код у безпечному середовищі без необхідності керування власною інфраструктурою.
6. Хостинг: Firebase Hosting - це простий і швидкий спосіб розгортання веб-додатків на глобальній мережі CDN (Content Delivery Network).
7. Аналітика: Firebase Analytics надає інструменти для збору даних про поведінку користувачів і продуктивність програми
8. Повідомлення: Firebase Cloud Messaging забезпечує надсилання крос-платформних повідомлень на різні пристрої.

Крім цих основних компонентів, Firebase пропонує додаткові інструменти та сервіси, як-от Machine Learning Kit, Remote Config, App Distribution та інші.

Firebase спрощує процес розробки, даючи змогу зосередитися на створенні застосунків, а не на управлінні інфраструктурою. Він інтегрується з багатьма популярними платформами розробки, включно з Android, iOS, веб-додатками та іншими [20].

У грі "Тисяча" Firebase використовується для аутентифікації користувачів, а також для збереження та завантаження даних гри.

Аутентифікація (див. рис 3.13):

- Використовується Firebase Authentication для аутентифікації користувачів.
- Реалізовано аутентифікацію за допомогою електронної пошти та посилання на вхід (sendSignInLinkToEmail, signInWithEmailLink).
- Також реалізовано аутентифікацію через Google (signInWithGoogleToken).
- При успішній аутентифікації зберігається інформація про поточного користувача (isAuthenticated, displayName, userPicture).
- Метод signOut використовується для виходу з облікового запису.

Рисунок 3.13 – Сторінка аутентифікація

Збереження даних гри:

- Використовується Cloud Firestore (база даних NoSQL) для збереження даних гри.
- Кожен користувач має колекцію gameList, де зберігаються його ігри.
- Метод saveGame зберігає поточний стан гри (Game) у колекції gameList користувача.
- При збереженні гри створюється новий документ або оновлюється існуючий, якщо гра вже була раніше збережена.
- Метод loadGames завантажує всі збережені гри користувача із колекції gameList.

Робота з даними:

- Для взаємодії з Firebase Firestore використовується екземпляр FirebaseFirestore.
- Запити до Firestore виконуються асинхронно за допомогою корутин.
- При успішному виконанні запиту дані обробляються та оновлюються у відповідних змінних стану (uiState, profileUi).
- При помилці виконання запиту виводиться відповідне повідомлення.

Структура даних:

- Клас User представляє дані користувача (email).
- Клас Game представляє дані гри (id, дата, список гравців, стан гри).
- Клас Player представляє дані гравця (ім'я, кольор, очки тощо).
- Клас Score представляє очки, отримані гравцем у раунді, та їх тип.

Загалом, Firebase використовується у даному додатку для аутентифікації користувачів, а також для збереження та завантаження даних гри у хмарній базі даних Firestore. Це дозволяє зберігати прогрес гри та продовжувати її пізніше з того самого місця, де вона була припинена.

3.5. Висновки до третього розділу

У цьому розділі було розглянуто основні аспекти проектування та реалізації мобільної гри "Тисяча". В ході розробки було проведено детальний аналіз та вибір відповідних технологій, які забезпечують високу продуктивність, простоту розробки, розширюваність та інтеграцію штучного інтелекту.

Основними технологіями, обраними для проекту, є:

1. Kotlin – основна мова програмування, яка завдяки своїм перевагам, таким як лаконічний синтаксис, безпека типів та повна сумісність з Java, забезпечує ефективність розробки.

2. Enngo Framework – використовується для реалізації штучного інтелекту, дозволяючи створювати реалістичних ігрових противників, що підвищує

інтерес користувачів.

3. Jetpack Compose – інструмент для розробки користувацького інтерфейсу, який спрощує процес створення UI завдяки декларативному підходу.

4. Firebase – платформа для збереження даних та управління користувачами, яка забезпечує масштабованість, надійність і безпеку.

Для архітектури мобільної гри "Тисяча" було обрано шаблон MVVM (Model-View-ViewModel), що забезпечує модульність коду, легкість тестування та можливість адаптації під різні платформи.

Особлива увага була приділена реалізації штучного інтелекту для опонентів у грі з використанням Encog Machine Learning Framework, що дозволяє створювати інтелектуальних опонентів, здатних адаптуватися до стратегій гравця та забезпечувати гідний супротив.

Загалом, успішна реалізація гри "Тисяча" вимагала комплексного підходу до вибору технологій, проектування архітектури, розробки інтерфейсу та інтеграції різних компонентів. Це дозволидл створити високоякісний продукт, який відповідатиме сучасним вимогам і задовольнить очікування користувачів.

ВИСНОВОК

У рамках даного дослідження було розроблено мобільний додаток для гри "Тисяча" з елементами штучного інтелекту, що базується на мові програмування Kotlin та Encog Machine Learning Framework. Використання середовища розробки Android Studio дозволило створити продуктивний, безпечний та інтегрований з AI мобільний додаток.

Застосування Kotlin забезпечило надійність та лаконічність коду, що сприяє його легкості в обслуговуванні та розширенні. Encog Machine Learning Framework був обраний завдяки його потужності та гнучкості, що дозволило ефективно інтегрувати штучний інтелект у гру. Серед інших технологій, які відіграли ключову роль у розробці додатку, варто відзначити Jetpack Compose, що значно спростило створення користувацького інтерфейсу завдяки декларативному підходу, та Firebase, яка забезпечила надійне збереження даних та управління користувачами, що підвищило масштабованість і безпеку додатку.

У процесі дослідження було проаналізовано існуючі мобільні ігри типу "Тисяча" та визначено ключові функціональні можливості, які успішно реалізовані у розробленому додатку. Серед них: гра з ботами та іншими гравцями онлайн, що підвищує інтерактивність та привабливість гри. Особлива увага була приділена реалізації штучного інтелекту для опонентів у грі, що дозволило створювати інтелектуальних опонентів, здатних адаптуватися до стратегій гравця та забезпечувати гідний супротив.

Подальше вдосконалення продукту може включати інтеграцію нових функцій, таких як розширені алгоритми машинного навчання для підвищення рівня гри ботів, а також розробку додаткових режимів гри та турнірів. Впровадження нових моделей нейронних мереж, наприклад, використання глибокого навчання або підкріплення навчання, може значно підвищити адаптивність та інтелектуальні можливості опонентів. Крім того, планується оптимізація продуктивності додатку, зокрема, стискання текстур та ресурсів, покращення ефективності коду, а також застосування прогресивних технік

рендерингу для забезпечення більш плавної та швидкої роботи інтерфейсу.

Важливим напрямком розвитку є також поліпшення системи винагород та досягнень. Введення нових видів бонусів, щоденних завдань та сезонних івентів допоможе зберігати інтерес гравців до гри та мотивувати їх повертатися частіше. Впровадження внутрішньоігрової валюти та магазину, де гравці можуть купувати ексклюзивні карти, косметичні поліпшення та інші предмети, може значно підвищити монетизацію додатку. Особлива увага буде приділена тестуванню та налагодженню, включаючи автоматизоване тестування та використання емуляторів Android, щоб забезпечити максимальну стабільність та високу якість гри на різних пристроях. Окрім цього, планується розширення функціоналу для аналізу та покращення геймплею. Введення детальних статистичних даних про гру кожного гравця, можливість перегляду історії партій та аналізу помилок дозволять користувачам покращувати свою майстерність. Розробка системи рейтингів та ліг додасть додатковий елемент конкуренції, що сприятиме залученню більшої кількості гравців та підвищенню їхнього інтересу до гри.

Було проведено порівняння різних функцій активації для нейронних мереж, зокрема sigmoid, tanh, ReLU та інших. Вибір функції активації Sigmoid обґрунтований її нелінійністю, гладкістю та обмеженим вихідним діапазоном, що робить її зручною для моделювання ймовірностей або пропорцій. Це забезпечило ефективний процес навчання нейронної мережі та оптимальне виконання завдань гри.

Загалом успішна реалізація гри "Тисяча" вимагала комплексного підходу до вибору технологій проектування, архітектури розробки інтерфейсу та інтеграції різних компонентів. Це дозволило створити високоякісний продукт, який відповідає сучасним вимогам і задовольняє очікування користувачів.

Розроблений додаток не лише забезпечує захоплюючий ігровий процес, але й демонструє високий рівень технічної реалізації та інноваційних рішень у сфері мобільних ігор зі штучним інтелектом.

Результати даного дослідження демонструють, що комплексний підхід до

вибору технологій, архітектури та дизайну мобільного додатку, а також ефективна інтеграція штучного інтелекту, дозволяють створювати інноваційні та захоплюючі ігрові продукти. Розроблений додаток для гри "Тисяча" має високий потенціал для подальшого розвитку та вдосконалення, що відкриває широкі перспективи для майбутніх досліджень та впровадження нових функціональних можливостей.

Кваліфікаційна робота виконана у відповідності до стандарту спеціальності 121 «Інженерія програмного забезпечення» і демонструє володіння такими компетентностями як :

- Знання і застосовування відповідних математичних понять, методів доменного, системного і об'єктно-орієнтованого аналізу та математичного моделювання для розробки програмного забезпечення.
- Знання і застосовування на практиці фундаментальних концепцій, парадигм і основних принципів функціонування мовних, інструментальних і обчислювальних засобів інженерії програмного забезпечення.
- Вміння розробляти людино-машинний інтерфейс.
- Знання і застосовування методів розробки алгоритмів, конструювання програмного забезпечення та структур даних і знань.
- Мотивоване обирання мови програмування та технології розробки для розв'язання завдань створення і супроводження програмного забезпечення.
- Знання та вміння застосовувати інформаційні технології обробки, зберігання та передачі даних.
- Знання, аналізування, вибір, кваліфіковане застосовування засобів забезпечення інформаційної безпеки (в тому числі кібербезпеки) і цілісності даних відповідно до розв'язуваних прикладних завдань та створюваних програмних систем.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. The Rise of Deepfakes: How Synthetic Media Will Impact Society. Brookings. URL: https://www.brookings.edu/research/the-rise-of-deepfakes-how-synthetic-media-will-impact-society/

2. Fighting deepfakes when detection fails. Brookings. URL: https://www.brookings.edu/techstream/fighting-deepfakes-when-detection-fails/

3. The TIOBE Index. TIOBE. URL: https://www.tiobe.com/tiobe-index/.

4. Kotlin Programming Language. Kotlin Official Documentation. URL: https://kotlinlang.org/docs/home.html

5. Designing Effective User Interfaces for Games. Gamasutra. URL: https://www.gamasutra.com/view/feature/131412/designing_effective_user_interfaces_.php

6. Object-Oriented Programming Principles. GeeksforGeeks. URL: https://www.geeksforgeeks.org/object-oriented-programming-oop-concept-in-java/

7. Introduction to Android Studio. Android Developers. URL: https://developer.android.com/studio/intro8. Game Programming Pattern. Game Programming Patterns. URL: https://gameprogrammingpatterns.com/contents.html

9. Saving Data in SQL Databases. Android Developers. URL: https://developer.android.com/training/data-storage/sqlite

10. Testing Android Apps. Android Developers. URL: https://developer.android.com/training/testing

11. The Ultimate Guide to Mobile Game Monetization. App Annie. URL: https://www.appannie.com/en/insights/mobile-game-monetization/

12. OWASP Mobile Security Project. OWASP. URL: https://owasp.org/www-project-mobile-security/

13. How to Keep Players Engaged with Regular Updates. Gamasutra. URL: https://www.gamasutra.com/blogs/author/TylerSigman/20180305/315511/How_to_Keep_Players_Engaged_with_Regular_Updates.php

14. Using Game Analytics to Drive Design and Monetization. Game Analytics. URL: https://gameanalytics.com/blog/using-game-analytics-to-drive-design-and-monetization.html

15. The Importance of Player Feedback in Game Development. GDC Vault. URL: https://www.gdcvault.com/play/1025034/The-Importance-of-Player-Feedback

16. Mobile Game Marketing Strategies for Success. Game Developer. URL: https://www.gamedeveloper.com/marketing/mobile-game-marketing-strategies-for-success/

17. 2023 Global Games Market Report. Newzoo. URL: https://newzoo.com/insights/articles/newzoo-global-games-market-report/

18. Ukrainian Game Industry: Current State and Prospects. Ukrainian Game Developers Association. URL: https://uagames.com.ua/en/research

19. Innovations in Game Development. TechCrunch. URL: https://techcrunch.com/tag/game-development/

20. Collaborating with Other Developers: Pros and Cons. Indie Game Developer Network. URL: https://igdn.org/collaborating-with-other-developers/