Plag

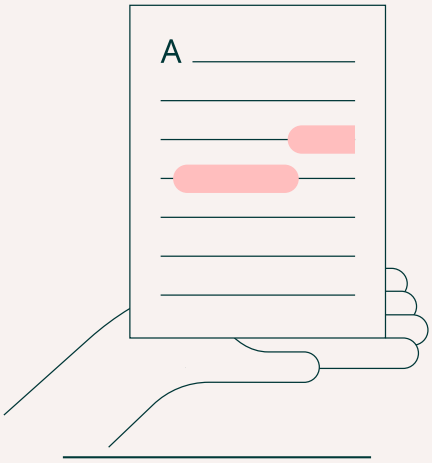
Звіт про оригінальність

● Оцінка схожості %7

● Ризик плагіату НАЙВИЩИЙ

Ігор Кагало 2025-06-05 22:46

Посилання на звіт: ZSn7 / Посилання користувача: qfC8



Ось вона –
Ваша звіт про оригінальність!

Ми раді повідомити, що перевірка вашого документа завершена, і результати вже готові! Наші алгоритми старанно працювали, щоб знайти збіги в наших базах даних.

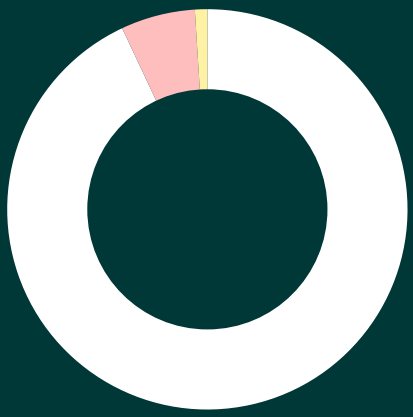
На наступних сторінках ви знайдете результати перевірки:

Бали
Збіги
Посилання

Ваш документ було перевірено за такими джерелами:

- ✓ База даних інтернет-джерел
- ✓ База даних наукових статей
- ○ Глибока перевірка (наш вдосконалений алгоритм)

Бали



Збіги тексту	6%
Перефразування	1%
Цитований текст	0%
Неправильне цитування	0%
Збігів не знайдено	93%

Ризик плагіату НАЙВИЩИЙ

Ризик плагіату вказує, як збіги тексту розподілені по документу. Вищий ризик виникає, коли збіги з'являються близько один до одного, наприклад, у тому самому абзаці або розділі.

Оцінка схожості 7%

Оцінка схожості показує, скільки слів або символів у вашому документі збігаються з текстами інших документів, включаючи перефразовані тексти або неправильні цитати.

Збіги

16 НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

13 ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ

13 «ФАХОВИЙ КОЛЕДЖ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ 16 «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

16 ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту

Фаховий молодший бакалавр

(освітньо-професійний ступінь)

на тему: Розробка програмного забезпечення для пошуку оптимальних стратегій у грі

Виконав студент

IV

курсу, групи

ОК-42

ОПП 16 «Обслуговування 10 комп’ютерних систем та мереж»

10 Спеціальності 16 123 Комп'ютерна інженерія

16 Гладкий 16 Володимир Андрійович

10 (прізвище, ім'я по батькові)

10 Керівник

Любомира Кужій

(підпис) (ім'я прізвище)

Нормоконтролер

Любомира Кужій

(підпис) (ім'я прізвище)

Рецензент

(підпис) (ім'я прізвище)

Голова ЕК

Олег Гіщак

(підпис) (ім'я прізвище)

Члени ЕК

Любомира Кужій

(підпис) (ім'я прізвище)

Андрій Селемонавічус

(підпис) (ім'я прізвище)

Дипломний проєкт захищений в ЕК « __ » _____2025 р.

з оцінкою « _____ »

Львів 2025

16 НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА» 13 ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ

13 «ФАХОВИЙ КОЛЕДЖ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

13 НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ 16 «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

13 Циклова комісія

10 Комп'ютерних систем і мереж

10 Освітньо-професійний 13 ступінь

13 Фаховий молодший бакалавр

13 Освітньо-професійна 13 програма

16 Обслуговування 10 комп’ютерних систем та мереж

10 Спеціальність

10 123 Комп'ютерна інженерія

10 ЗАТВЕРДЖУЮ

10 Завідувач відділення

10 «Комп’ютерних систем і мереж»

10 _____ Володимир СТАХІВ

«___» _____ 2025 року

ЗАВДАННЯ

НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТУ

Гладкому Володимиру Андрійовичу

16 (прізвище, ім'я та 3 по батькові)

3 1. Тема проєкту

3 Розробка програмного забезпечення для пошуку

оптимальних стратегій у грі

керівник проєкту

Кужій Любомира Іванівна

(3 прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

3 затверджені наказом директора від «20» березня 2025 року № 20 - ст

3 2. Строк подання студентом проєкту «10» червня 2025 34 року

3 Вихідні дані до проєкту

3 3.1 Мова програмування Kotlin

3.2 Інтегроване середовище розробки Android studio

3.3 Мова розмітки XML

3.4 Android-додаток «Хрестики-нулики»

3 Зміст розрахунково-пояснювальної записки

3 4.1 АКТУАЛЬНІСТЬ ПРОБЛЕМИ ПРИЙНЯТТЯ РІШЕНЬ В ТЕОРІЇ ІГОР

4.2 МЕТОД МІНІМАКСУ ДЛЯ ПОШУКУ ОПТИМАЛЬНИХ СТРАТЕГІЙ

4.3 РОЗРОБКА ТА ДОСЛІДЖЕННЯ ANDROID-ГРИ «ХРЕСТИКИ-НУЛИКИ»

4.4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ

4.5 23 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА ЖИТТЄДІАЛЬНОСТІ

3 5. Перелік графічного матеріалу

3 5.1.

Лист 1 Основні компоненти пакету ai

5.2.

Лист 2 Дерево рішень для гравця Max в грі

5.3.

Лист 3 Екран налаштування гри

5.4.

Лист 4 Витрати на розробку та впровадження проектного рішення

3 6 Консультанти розділів проєкту

58 Розділ

58 Ім'я, прізвище 3 та посада консультанта

3 Підпис, дата

3 Завдання видав

Завдання отримав

Техніко-економічне обґрунтовування

Тетяна Підкуймуха

23 Охорона праці та безпека життєдіяльності

23 Роман 23 Томків

3 7. Дата видачі завдання « 01»квітня 2025 34 року

34 КАЛЕНДАРНИЙ ПЛАН

34 №

34 з/п

34 Назва етапів дипломного проєкту

34 Термін

34 виконання

34 Примітка

1

Актуальність проблеми прийняття рішень в теорії ігор

20.04.2025

2

Метод мінімаксу для пошуку оптимальних стратегій

24.04.2025

3

Розробка та дослідження android-гри «хрестики-нулики»

1.05.2025

4

Техніко-економічне обгрунтування

7.05.2025

5

23 Охорона праці та безпека життєдіяльності

23 10.05.2025

6

Вступ, реферат, висновки, зміст, додатки

20.05.2025

7

Розробка обов’язкових креслень

25.05.2025

Студент

Володимир Гладкий

(58 підпис)

58 (ім'я, прізвище)

58 Керівник проєкту

Любомира Іванівна

(підпис)

(ім'я, 60 прізвище)

60 РЕФЕРАТ

60 Пояснювальна записка дипломного проєкту: 143 стор., 15 рис., 3 табл., 8 додатки, 17 посилань.

Об’єкт проєктування – Android-додаток «Хрестики-нулики».

Метою дипломного проєкту є створення гри «Хрестики-нулики» для мобільної операційної системи Android з використанням мови програмування Kotlin, XML та алгоритму Minimax.

Галузь використання – мобільні пристрої на базі Android.

У проєкті проаналізовано базові принципи теорії ігор, а також методи побудови оптимальних стратегій. Досліджено особливості розробки Android-додатків. Реалізовано гру «Хрестики-нулики» з графічним інтерфейсом, декількома рівнями складності, підтримкою гри з ботом або іншим гравцем. Для побудови штучного інтелекту реалізовано алгоритм Minimax. Проведено тестування додатку на реальному пристрої.

ANDROID, KOTLIN, XML, MINIMAX, FRAGMENT, UI, ГРА, МОБІЛЬНИЙ ДОДАТОК, ШТУЧНИЙ ІНТЕЛЕКТ, БОТ, ТЕОРІЯ ІГОР, ЧИСТІ СТРАТЕГІЇ.

ЗМІСТ

59 ВСТУП

59 Актуальність обраної теми дипломного проєкту зумовлена зростаючим попитом на розробку мобільних ігор для платформи Android, що поєднують простоту реалізації з використанням сучасних підходів у галузі штучного інтелекту. Гра «Хрестики-нулики» слугує базовим прикладом для реалізації алгоритмів прийняття рішень, зокрема алгоритму Minimax, що використовується для побудови оптимальної стратегії в ігрових ситуаціях.

Метою дослідження є створення Android-додатку «Хрестики-нулики» з реалізацією інтелектуального супротивника на основі алгоритму Minimax.

Завдання дослідження:

проаналізувати принципи роботи алгоритму Minimax та його використання в ігрових стратегіях;

розробити архітектуру Android-додатку з використанням Single Activity та фрагментів;

реалізувати користувацький інтерфейс із налаштуваннями, профілем, меню, грою з ботом і з другом;

протестувати додаток на реальному пристрої.

Об’єкт дослідження – мобільний додаток гри «Хрестики-нулики» на платформі Android.

Предмет дослідження – методи реалізації інтелектуальної гри з використанням алгоритмів пошуку оптимальної стратегії.

Методи дослідження – аналіз, проєктування, програмування, моделювання, тестування.

Структура роботи. Дипломна 45 робота складається з п’яти розділів. У першому розділі розглянуто актуальність проблеми прийняття рішень у теорії ігор, наведено основні

поняття, класифікації та приклади застосування. У другому – детально описано метод Мінімакс для пошуку оптимальних стратегій, включаючи побудову алгоритму та блок-схему. Третій розділ присвячено розробці та дослідженню Android-гри «Хрестики-нулики», у межах якого описано постановку задачі, вибір інструментів, реалізацію графічного інтерфейсу, ігрової логіки, алгоритму та проведено тестування програми. Четвертий розділ присвячено економічному обґрунтуванню розробки, а п'ятий – питанням охорони праці. У кінці подано висновки, перелік використаних джерел та додатки з фрагментами коду.

Фактологічна основа дослідження – офіційна документація Kotlin та Android, ресурси з відкритим доступом (наприклад, Android Developers, Kotlin Documentation), матеріали щодо теорії ігор та алгоритму Minimax, графічні ресурси з сайту IconScout.

1 АКТУАЛЬНІСТЬ ПРОБЛЕМИ ПРИЙНЯТТЯ РІШЕНЬ В ТЕОРІЇ ІГОР

1.1 Основні поняття ти визначення теорії ігор

Теорія ігор, як наукова дисципліна, бере свій початок у першій половині XX століття. Хоча окремі ідеї та концепції, пов'язані зі стратегічною взаємодією, можна знайти в працях більш ранніх мислителів, таких як Ернст Цермело, який досліджував шахові стратегії, справжній прорив відбувся у 1944 році з виходом фундаментальної праці 48 Джона фон Неймана та Оскара Морґенштерна «Теорія ігор та економічна поведінка». Ця книга заклала основи для систематичного вивчення стратегічних взаємодій між раціональними агентами.

У 1950-х роках Джон Неш зробив вагомий внесок у розвиток теорії ігор, запропонувавши концепцію рівноваги, яка згодом отримала його ім'я. Рівновага Неша стала ключовим інструментом для аналізу стратегічної взаємодії в різноманітних галузях, включаючи економіку, політику, соціологію та інформатику.

З плином часу теорія ігор продовжувала розвиватися та розширювати свої сфери застосування. У другій половині XX століття були розроблені нові концепції та методи аналізу, такі як ігри з неповною інформацією, запропоновані Джоном Харсані. Ці розробки дозволили моделювати більш складні та реалістичні сценарії, в яких гравці мають обмежені знання про наміри та можливості своїх опонентів.

У 1980-х і 1990-х роках теорія ігор почала активно проникати в галузь обчислювальної техніки та штучного інтелекту. Алгоритми, засновані на принципах теорії ігор, знайшли застосування в автоматизованих системах прийняття рішень, економічному моделюванні, кібербезпеці та інших сферах.

З початком XXI століття теорія ігор стала невід'ємною частиною досліджень у галузі

машинного навчання, багатокритеріальної оптимізації та розподілених систем. Зокрема, алгоритми, що використовують принципи теорії ігор, застосовуються в системах штучного інтелекту для навчання агентів у конкурентному середовищі, що має ключове значення для розробки стратегічного програмного забезпечення.

Таким чином, теорія ігор пройшла довгий шлях розвитку, перетворившись з абстрактної математичної теорії на потужний 35 інструмент для аналізу та моделювання стратегічних взаємодій у різноманітних галузях.

Теорія ігор 19 – це розділ прикладної 19 математики, який 19 вивчає математичні моделі прийняття оптимальних рішень в умовах конфлікту інтересів. Вона аналізує стратегічні взаємодії між раціональними суб'єктами (гравцями), де результат дій одного гравця залежить від дій інших гравців.

Основна мета теорії ігор – визначити оптимальні стратегії для кожного гравця в умовах, коли їхні інтереси можуть частково або повністю суперечити один одному.

Теорія ігор застосовується в 35 різних галузях, таких як економіка, політика, біологія, інформатика та штучний інтелект. Вона допомагає аналізувати та прогнозувати поведінку гравців у стратегічних ситуаціях.

Ключові аспекти теорії ігор:

Аналіз стратегічних взаємодій.

Визначення оптимальних стратегій.

Прогнозування результатів ігор.

19 Прийняття рішень в умовах невизначеності.

19 Гравці – це суб'єкти, які беруть участь у грі та приймають рішення.

Стратегії – це набір можливих дій, які може вибрати гравець.

Чисті стратегії – це вибір однієї конкретної дії.

Змішані стратегії – це вибір дій з певною ймовірністю.

Домінантні стратегії – це стратегії, які приносять найкращий результат незалежно від дій інших гравців.

Доміновані стратегії – це стратегії, які завжди приносять гірший результат, ніж інші стратегії.

Виграш – це результат гри, який може бути виражений у вигляді чисел, що відображають користь або втрати для кожного гравця.

Рівновага Неша – це стан гри, при якому 57 жоден гравець не може збільшити свій виграш, змінивши свою стратегію, якщо інші гравці не змінюють своїх стратегій. Рівновага Неша є ключовим поняттям у теорії ігор, оскільки вона дозволяє прогнозувати результат гри.

Дилема в'язня – ситуація, де двоє гравців можуть співпрацювати або зраджувати, і хоча зрада дає індивідуальну вигоду, спільна співпраця приносить кращий загальний результат. На рисунку 1.1 зображено матрицю виграшів для двох гравців (А і Б), де кожен може обрати співпрацю або зраду. Числа в клітинках показують виграші (або втрати) для кожного гравця залежно від їхніх рішень.

Рисунок 1.1 – Матриця виграшів у дилемі в'язня

Типи ігор:

Ігри 36 з нульовою сумою – 36 це ігри, в яких 36 сума виграшів усіх гравців дорівнює нулю. Тобто виграш одного гравця означає програш іншого.

Ігри з ненульовою сумою – це ігри, в яких сума виграшів гравців може бути більшою або меншою за нуль.

Некооперативні ігри – це ігри, в яких гравці діють незалежно один від одного.

Ігри з повною інформацією – це ігри, в яких кожен гравець знає всі минулі ходи інших гравців, а також їхні можливі стратегії та виграші.

Ігри з неповною інформацією – це ігри, в яких гравці не мають повної інформації про минулі ходи або про стратегії інших гравців.

Статичні ігри – це ігри, в яких гравці приймають рішення одночасно.

Динамічні ігри – це ігри, в яких гравці приймають рішення послідовно.

1.2 Основні завдання теорії ігор

Теорія ігор допомагає аналізувати ситуації, в яких рішення одного гравця впливають на результати інших гравців. Це дозволяє зрозуміти, як гравці взаємодіють між собою та які фактори впливають на їхні рішення.

Аналіз стратегічних взаємодій дозволяє прогнозувати поведінку гравців та 67 виявляти можливі сценарії розвитку подій.

67 Одним з основних завдань теорії ігор є визначення оптимальних стратегій для кожного гравця. Оптимальна стратегія – це стратегія, яка максимізує виграш гравця, враховуючи можливі дії інших гравців.

Теорія ігор розробляє методи та алгоритми для пошуку оптимальних стратегій у різних типах ігор.

Теорія ігор дозволяє прогнозувати результати стратегічних взаємодій, враховуючи раціональну поведінку гравців.

Прогнозування результатів ігор допомагає приймати обґрунтовані рішення в умовах конфлікту інтересів.

Теорія ігор допомагає приймати рішення в умовах, коли інформація про дії інших гравців є неповною або відсутньою.

Вона розробляє методи для аналізу ризиків та невизначеностей, що дозволяє приймати обґрунтовані рішення в складних ситуаціях.

Теорія ігор використовується для моделювання реальних ситуацій, в яких присутні стратегічні взаємодії, таких як економічні ринки, політичні переговори, військові конфлікти та інші.

Моделювання реальних ситуацій дозволяє аналізувати їхні особливості та розробляти ефективні стратегії.

1.3 Класифікація ігор

Теорія ігор розглядає широкий спектр ігор, які можна класифікувати за різними критеріями. Ось основні класифікації:

За кількістю гравців:

Ігри з двома гравцями – це найпростіший тип ігор, в яких беруть участь два гравці.

Ігри з кількома гравцями – це ігри, в яких беруть участь три або більше гравців.

За типом виграшу:

Ігри з нульовою сумою – це ігри, в яких сума виграшів усіх гравців завжди дорівнює нулю. Виграш одного гравця означає програш іншого.

Ігри з ненульовою сумою – це ігри, в яких сума виграшів гравців може бути більшою або меншою за нуль. Гравці можуть як виграватися, так і програвати разом.

За можливістю співпраці:

Кооперативні 8 ігри – це ігри, в яких гравці можуть укладати угоди та координувати свої 8 дії для досягнення спільної мети.

8 Некооперативні ігри – це ігри, в яких гравці діють незалежно один від одного, не маючи можливості укладати угоди.

За типом інформації:

Ігри з повною інформацією – це ігри, в яких кожен гравець знає всі минулі ходи та стратегії інших гравців.

47 Ігри з неповною інформацією – це ігри, в яких гравці не мають повної інформації про минулі ходи або про стратегії інших гравців.

За часом прийняття рішень:

Статичні 8 ігри – це ігри, в яких гравці приймають рішення одночасно, не знаючи про рішення інших гравців.

Динамічні 8 ігри – це ігри, в яких гравці приймають рішення послідовно, знаючи про попередні ходи.

За типом стратегій:

Ігри з чистими стратегіями – це ігри, в яких кожен гравець вибирає одну конкретну дію.

Ігри зі змішаними стратегіями – це ігри, в яких кожен гравець вибирає дії з певною ймовірністю.

На рисунку 1.2 візуально зображено класифікацію ігор за різними критеріями.

Рисунок 1.2 – Класифікація ігор за різними критеріями

1.4 Критерії прийняття рішень з використанням методів теорії ігор

Теорія ігор надає різні критерії для прийняття рішень у стратегічних ситуаціях. Вибір критерію залежить від типу гри, доступної інформації та цілей гравців. Ось основні критерії:

Мінімаксний принцип. Цей принцип використовується в іграх з нульовою сумою, де метою гравця є мінімізація максимальних втрат. Гравець вибирає стратегію, яка забезпечує йому найкращий результат у найгіршому можливому сценарії. Мінімаксний принцип є консервативним, оскільки він орієнтований на уникнення ризику.

Критерій Байєса. Цей критерій використовується в іграх з неповною інформацією, де гравці мають ймовірнісні оцінки щодо дій інших гравців. Гравець вибирає стратегію, яка максимізує його очікуваний виграш, враховуючи ймовірності дій інших гравців. Критерій Байєса є раціональним, оскільки він орієнтований на максимізацію очікуваного виграшу.

Критерій Севіджа (мінімізації ризику). Цей критерій орієнтований на мінімізацію жалю, який гравець може відчути, якщо його рішення виявиться невдалим. Гравець вибирає стратегію, яка мінімізує максимальний можливий жаль. Цей критерій корисний, коли гравець боїться прийняти рішення, яке 8 може призвести до значних втрат.

Критерій Гурвіца. Цей критерій є компромісом між оптимізмом та песимізмом. Гравець вибирає стратегію, яка максимізує зважену суму найкращого та найгіршого можливих результатів. Вага, що надається найкращому та найгіршому результатам, визначається коефіцієнтом оптимізму.

Рівновага Неша. Це стан гри, в якому 47 жоден гравець не може 63 збільшити свій виграш, змінивши свою стратегію, якщо інші гравці не змінюють своїх стратегій. Рівновага Неша є ключовим поняттям у теорії ігор, оскільки вона дозволяє прогнозувати результат гри.

Алгоритми пошуку рівноваги Неша. Існують різні алгоритми для пошуку рівноваги Неша, такі як алгоритм Лемке-Хаусона. Ці алгоритми дозволяють знаходити оптимальні стратегії в складних іграх.

Методи машинного навчання. Методи машинного навчання, такі як навчання з підкріпленням, можуть бути використані для пошуку оптимальних стратегій у складних іграх. Ці методи дозволяють агентам навчатися оптимальним стратегіям шляхом взаємодії з середовищем.

Генетичні алгоритми. Генетичні алгоритми можуть бути використані для пошуку оптимальних стратегій шляхом еволюційного процесу. Ці алгоритми дозволяють знаходити оптимальні стратегії в складних іграх, де інші методи можуть бути неефективними.

1.5 Застосування теорії ігор у програмуванні

Теорія ігор знаходить широке застосування в програмуванні, особливо в тих областях, де необхідно моделювати та аналізувати стратегічні взаємодії між агентами або системами. Ось основні напрямки застосування:

Штучний інтелект (ШІ):

Теорія ігор використовується для розробки інтелектуальних агентів, які здатні приймати оптимальні рішення в умовах конфлікту або співпраці.

Навчання з підкріпленням. Алгоритми, засновані на теорії ігор, використовуються для навчання агентів оптимальним стратегіям шляхом взаємодії з середовищем.

Багатоагентні системи. Теорія ігор допомагає моделювати та аналізувати поведінку багатоагентних систем, де агенти взаємодіють між собою.

Розробка неігрових персонажів (NPC) в комп'ютерних іграх.

Розробка ігор:

Теорія ігор використовується для створення збалансованих та цікавих ігор, в яких гравці приймають стратегічні рішення.

Розробка алгоритмів для комп'ютерних супротивників. Теорія ігор допомагає розробляти алгоритми, які дозволяють комп'ютерним супротивникам приймати оптимальні рішення.

Аналіз ігрового балансу. Теорія ігор допомагає аналізувати ігровий баланс та виявляти можливі проблеми.

Оптимізація алгоритмів:

Теорія ігор може бути використана для оптимізації алгоритмів, особливо в тих випадках, коли необхідно враховувати стратегічні взаємодії між різними компонентами алгоритму.

Розробка алгоритмів для вирішення завдань маршрутизації, розподілу ресурсів та планування.

Мережеві технології:

Теорія ігор використовується для аналізу та оптимізації роботи комп'ютерних мереж, де різні користувачі або пристрої взаємодіють між собою.

Розробка алгоритмів для управління трафіком, розподілу пропускної здатності та запобігання перевантаженням.

Розробка алгоритмів для персоналізації рекомендацій та реклами.

2 МЕТОД МІНІМАКСУ 40 ДЛЯ ПОШУКУ ОПТИМАЛЬНИХ СТРАТЕГІЙ

2.1 Розробка та опис алгоритму мінімаксу

Розробка алгоритму мінімаксу є ключовим етапом у 40 створенні програмного забезпечення для пошуку оптимальних стратегій у грі. Алгоритм використовується в іграх із повною інформацією, де гравці мають доступ до всіх даних про поточний стан гри та можливі ходи. Він дозволяє прогнозувати майбутні дії супротивника та вибирати найкращий варіант розвитку подій. Основні етапи розробки алгоритму:

Моделювання гри. На цьому етапі необхідно формалізувати правила гри, визначити можливі дії гравців та описати умови завершення гри. Основні аспекти моделювання:

Визначення набору допустимих ходів для кожного гравця.

Визначення станів гри та умов її завершення (наприклад, перемога одного з гравців або нічия).

Представлення ігрового середовища у вигляді структури даних (наприклад, матриця для гри в хрестики-нулики або список можливих ходів для шахів).

Формалізація дерева рішень. Мінімакс-алгоритм будує дерево рішень, де:

Кожен вузол представляє певний стан гри.

Гілки від вузла відповідають можливим ходам гравців.

Листові вузли відповідають кінцевим станам гри (перемога, поразка або нічия).

Структура дерева залежить від правил гри та максимальної глибини розгляду ходів.

Оцінка станів гри. 22 Для прийняття рішень необхідно створити функцію оцінки стану гри. Функція має:

Повернути числове значення, яке відображає вигідність конкретного стану для гравця.

Враховувати такі фактори, як кількість доступних ходів, контроль над важливими зонами ігрового поля та потенційні загрози від супротивника.

У простих іграх (наприклад, хрестики-нулики) оцінка може базуватися на наявності виграшних комбінацій, у складніших (наприклад, шахи) – на сумі вартостей фігур та їх позиційному розташуванні.

Рекурсивний перебір. Мінімакс використовує рекурсивний підхід для пошуку найкращого ходу:

Гравець, що намагається максимізувати свій виграш (Max), вибирає хід із найбільшим значенням функції оцінки.

Гравець, що 22 намагається мінімізувати виграш противника (Min), вибирає хід із найменшим значенням оцінки.

Цей процес повторюється, поки не буде досягнуто кінцевого стану або заданої глибини рекурсії.

Оптимізація алгоритму (α-β відсічення). Оскільки повний перебір варіантів може бути дуже ресурсозатратним, застосовується метод α-β відсічення:

Якщо один із гравців уже має гарантовано кращий хід, інші варіанти не розглядаються.

Це дозволяє значно скоротити кількість обчислень, відкидаючи гілки дерева, що не можуть вплинути на кінцевий вибір.

Оптимізований алгоритм працює швидше, особливо в іграх із великою кількістю можливих ходів (наприклад, шахи).

Алгоритм мінімаксу використовується для знаходження найкращого можливого ходу в іграх із двома гравцями. Він працює шляхом рекурсивного перебору всіх можливих станів гри та оцінки їхньої вигідності для кожного з гравців.

Мінімакс – алгоритм 22 працює за принципом чергування 22 гравців: один 22 гравець намагається максимізувати свій виграш (Max), тоді як інший – мінімізувати виграш супротивника (Min). Алгоритм проходить через всі можливі варіанти розвитку гри та вибирає той хід, який приводить до найкращого можливого результату для активного гравця.

Побудова дерева рішень:

Кореневий вузол дерева рішень відповідає поточному стану гри.

Кожен рівень дерева представляє черговий хід.

Вузли чергуються між гравцями: Max прагне максимізувати виграш, а Min— мінімізувати його.

На рисунку 2.1 показано уривок дерева рішень для гри.

Рисунок 2.1 –Дерево рішень для гравця Max в грі хрестики-нулики

Гравець Max обирає хід, що максимізує його виграш. Кожен вузол дерева представляє ігрову позицію (у цьому випадку – поле для гри в хрестики-нулики), а стрілки показують можливі ходи, які ведуть до нових позицій.

Оцінка стану гри:

Для кожного кінцевого стану гри використовується функція оцінки, яка повертає числове значення (наприклад, +1 для перемоги Max, -1 для перемоги Min, 0 для нічиєї).

Проміжні вузли оцінюються за допомогою мінімаксного підходу: кожен гравець вибирає найкращий для себе варіант із доступних.

Кроки виконання алгоритму:

Генерується список всіх можливих ходів.

Для кожного ходу визначається новий стан гри.

Алгоритм рекурсивно оцінює всі наступні можливі ходи.

Використовується α-β відсічення для скорочення кількості перевірених ходів.

Обирається найкращий хід для поточного гравця.

2.2 Блок-схеми основних функцій алгоритму

На рисунку 2.2 зображено блок-схему функції evaluate, яка є частиною алгоритму мінімаксу і використовується для оцінки стану гри.

Рисунок 2.2 – Блок-схема функції evaluate (оцінка стану гри)

Рекурсивна функція послідовно перебирає всі можливі ходи у грі, оцінюючи їх наслідки для обох гравців. Для ходів комп'ютера функція максимізує потенційну вигоду, а для ходів гравця – мінімізує її. Функція спочатку оцінює поточний стан гри і перевіряє базові випадки: якщо стан є виграшним для комп'ютера, повертається позитивна оцінка з урахуванням глибини (чим швидше перемога, тим краще); якщо стан є виграшним для гравця, повертається негативна оцінка; якщо всі клітинки заповнені або досягнута максимальна глибина пошуку, повертається нейтральна оцінка. Якщо жоден з базових випадків не виконується, функція рекурсивно аналізує наслідки кожного можливого ходу, симулюючи його на дошці, викликаючи себе для протилежного гравця, а потім скасовуючи хід. При згортці дерева рекурсії функція послідовно повертає найкращу оцінку для кожного рівня, що в кінцевому підсумку дозволяє визначити оптимальний хід для поточного стану гри. На рисунку 2.3 зображено блок-схему цієї функції, яка ілюструє процес рекурсивного пошуку та оцінки ходів.

Рисунок 2.3 – Блок-схема функції minimax, що реалізує алгоритм мінімаксу для пошуку оптимальних ходів у грі

Алгоритм вибору найкращого ходу перебирає всі доступні варіанти, оцінює їх за допомогою minimax і повертає хід з найвищою оцінкою. На рисунку 2.4 зображено блок-схему завершального етапу роботи алгоритму – вибору найкращого ходу.

Рисунок 2.4 – Блок-схема вибору найкращого ходу на основі оцінок minimax

3 РОЗРОБКА ТА ДОСЛІДЖЕННЯ ANDROID-ГРИ «ХРЕСТИКИ-НУЛИКИ»

3.1 Постановка задачі

Завданням дипломного проєкту є розробка мобільної гри «Хрестики-нулики» для операційної системи Android, що має забезпечити повноцінний досвід гри як 8 для двох гравців, так і з ботом. Головною метою є створення інтуїтивно зрозумілого та привабливого графічного інтерфейсу, реалізація механізму гри з ботом з використанням алгоритму мінімакс для оптимального вибору ходів, а також забезпечення можливості налаштування параметрів гри (наприклад, вибір складності).

У рамках проєкту планується реалізувати наступні функціональні можливості:

Головне меню, яке забезпечує навігацію в додатку, зокрема запуск ігрового процесу, перехід до розділу налаштувань, вибір рівня складності комп'ютерного супротивника, а також вибір режиму гри – проти комп'ютера або між двома гравцями на одному пристрої.

Профіль користувача, в якому зберігається ім'я гравця та його аватар.

Розділ налаштувань гри, який надає можливість обрати гравця, що розпочинає гру (користувач або комп'ютер), а також визначити ігровий символ (хрестик або нолик).

Режим гри з комп'ютером, де реалізовано супротивника з кількома рівнями складності.

Режим гри 8 для двох гравців, який дозволяє двом користувачам здійснювати ходи по черзі, використовуючи один пристрій.

Однією з основних задач є розробка алгоритму для бота, який здатний адекватно реагувати на різні стратегії користувача та вибирати оптимальні ходи. Для цього буде використано алгоритм мінімакс, який дозволяє оцінити всі можливі варіанти ходів і вибрати найкращий.

Проєкт також передбачає інтеграцію функціоналу, що дозволяє проводити тестування гри на реальних Android-пристроях, перевіряти стабільність роботи програми, коректність алгоритмів та зручність інтерфейсу.

Завдання цього проєкту полягає в тому, щоб забезпечити повний функціональний цикл гри, від вибору режиму гри до визначення переможця, і надати користувачеві комфортний досвід гри з ботом або двома гравцями.

3.2 Вибір мови програмування

Kotlin – це статично типізована 46 мова програмування, розроблена компанією JetBrains, яка працює на платформі Java Virtual Machine (JVM) і 46 може бути використана для розробки програм для різних платформ, зокрема для Android. Kotlin була офіційно представлена у 2011 році, але її популярність значно зросла після 2017 року, коли Google оголосив про підтримку Kotlin як першокласної мови для розробки Android-додатків.

У 2017 році Google оголосив, що Kotlin є офіційною мовою для розробки додатків під Android, що стало важливою віхою для розвитку цієї мови. З того часу Kotlin став стандартом для розробки на платформі Android завдяки своїм численним перевагам перед Java, зокрема у зручності синтаксису та можливості інтеграції новітніх фіч у розробку мобільних додатків.

Переваги Kotlin для Android-розробки:

Kotlin має чистий та виразний синтаксис, що робить код компактним, легким для читання та підтримки. Це дозволяє розробникам писати менш об'ємний код порівняно з Java, зберігаючи при цьому високу продуктивність і безпеку типів.

Однією з найбільших проблем Java є використання null-значень, що часто призводить до помилок у коді (NullPointerException). Kotlin пропонує вбудовану систему обробки null-значень через nullable типи, що значно знижує ймовірність виникнення таких помилок. Це забезпечує більшу стабільність програм.

Kotlin є повністю сумісним з Java, що дозволяє легко використовувати існуючі Java-бібліотеки та інтерфейси в Kotlin-проектах. Це дає змогу поступово перейти з Java на Kotlin, не вимагаючи повної переписування існуючого коду.

Завдяки лаконічному синтаксису Kotlin дозволяє створювати більш короткий і зрозумілий код, що знижує ризик помилок і полегшує його підтримку. Наприклад, в Kotlin є можливість використовувати лямбда-вирази, розширення функцій та інші сучасні конструкції, які роблять код більш гнучким і легким у використанні.

Kotlin підтримує корутини – механізм, що спрощує роботу з асинхронними задачами, такими як завантаження даних з мережі чи робота з базами даних. Завдяки корутинам можна працювати з асинхронними задачами без необхідності використовувати складні зворотні виклики або багатокрокові блокування потоків.

Kotlin надає безліч розширень, таких як розширення функцій для стандартних бібліотек, що дозволяє створювати власні розширення та модифікації класів без необхідності змінювати їх базову реалізацію. Це полегшує організацію коду та покращує його повторне використання.

Kotlin активно підтримує нові фічі, що з'являються в Android SDK і дозволяє інтегрувати сучасні технології, такі як Jetpack, зручні для побудови складних UI, а також забезпечує підтримку новітніх API, наприклад для роботи з потоками даних або інтеграції з іншими системами.

Після офіційної підтримки Kotlin з 2017 року, Google активно рекомендує цю мову для розробки Android-додатків. У своїх документаціях, на офіційному сайті Android та в інших матеріалах Google підкреслює, що Kotlin є сучасним, потужним і зручним інструментом для мобільної розробки. Окрім цього, Kotlin також підтримується в основних Android-фреймворках і бібліотеках, що ще більше підвищує її популярність серед розробників.

3.3 Вибір інструментів та технологій розробки

У розробці Android-додатків доступні два основні підходи до створення графічного інтерфейсу користувача (GUI): традиційна XML-розмітка та Jetpack Compose – сучасний декларативний фреймворк від Google. Обидва підходи мають свої переваги та недоліки, і вибір між ними залежить від потреб проєкту, технічних вимог, а також рівня підготовки розробника.

XML-розмітка є стандартним способом створення інтерфейсу з моменту появи Android. Розмітка представлена у вигляді окремих XML-файлів, у яких описуються елементи інтерфейсу (кнопки, текстові поля, зображення тощо), їхні властивості та структура. Перевагами такого підходу є:

Чітке розділення логіки і дизайну. XML-файли відповідають лише за вигляд, а вся бізнес-логіка реалізується окремо в Kotlin/Java-коді;

Сумісність зі всіма Android SDK і бібліотеками;

Наявність великої кількості прикладів, туторіалів і документації;

Широка підтримка в Android Studio (візуальний редактор, попередній перегляд).

Однак XML має і свої мінуси: велика кількість шаблонного коду, ускладнене створення динамічних UI-компонентів, складніші механізми для побудови реактивного інтерфейсу.

Jetpack Compose – це новий підхід, заснований на декларативній парадигмі. Інтерфейс описується безпосередньо в Kotlin-коді за допомогою функцій. Це дозволяє легко будувати гнучкий та реактивний інтерфейс, спрощуючи взаємодію UI з даними. Compose підтримує всі сучасні фічі Android-UI, такі як анімації, теми, адаптивність до екранів, реактивність через State.

Compose активно розвивається і є рекомендованим способом побудови інтерфейсу в нових проєктах, але має і деякі обмеження:

Вища крива навчання для початківців;

Менше прикладів у відкритому доступі, особливо для складних компонентів;

Не всі бібліотеки ще повністю адаптовані під Compose.

У цьому проєкті було прийнято рішення використовувати XML-розмітку для реалізації графічного інтерфейсу. Основні причини вибору:

Знання XML вже були отримані під час навчання, тому немає потреби опановувати новий фреймворк;

Стабільність і перевіреність підходу – XML використовується роками і не має критичних проблем у сумісності;

Достатній функціонал для створення простого ігрового інтерфейсу (меню, кнопки, сітка гри тощо);

Наявність візуального редактора, що полегшує створення макетів і підвищує швидкість розробки.

Також важливо враховувати, що для гри «Хрестики-нулики» інтерфейс є відносно простим – не потребує складної анімації чи реактивної побудови, тому переваги Compose у цьому випадку не є критичними.

Отже, XML залишається оптимальним вибором у контексті цього навчального проєкту, забезпечуючи ефективну розробку, хорошу підтримку інструментів і високу передбачуваність поведінки UI.

Для створення Android-додатку було обрано Android Studio – офіційне інтегроване середовище розробки (IDE) від Google для Android. Воно забезпечує повний набір інструментів, необхідних для повного циклу розробки мобільного додатку: написання коду, створення інтерфейсу, компіляції, тестування, відлагодження та збірки APK.

Основні переваги Android Studio:

Офіційна підтримка від Google – постійне оновлення, інтеграція з новими Android SDK, бібліотеками Jetpack, Gradle та інструментами Google;

Повна підтримка Kotlin – автодоповнення, підсвітка синтаксису, рефакторинг, аналіз коду, швидкий доступ до документації;

Інструменти для дизайну UI – зручний графічний редактор для роботи з XML-розміткою, попередній перегляд інтерфейсу на різних пристроях;

Вбудовані емулювальники Android-пристроїв – дозволяють тестувати додаток без фізичного телефону;

Інтеграція з системами контролю версій (Git);

Гнучка система збірки Gradle – дає змогу керувати залежностями, конфігураціями, створювати різні білди (debug/release).

Також Android Studio підтримує зручну роботу з логами, налагодженням, Unit-тестами, профілюванням продуктивності, а також має великий вибір плагінів для розширення функціоналу.

З урахуванням усіх цих факторів, Android Studio є найкращим вибором для розробки додатків під Android і особливо підходить для проєктів на Kotlin, що робить його логічним і ефективним рішенням у межах цієї роботи.

3.4 Загальна архітектура графічного інтерфейсу

Основою для створення графічного інтерфейсу користувача (GUI) в операційній системі Android є використання мови розмітки XML (Extensible Markup Language). На відміну від програмного підходу, де кожен елемент інтерфейсу створюється та налаштовується безпосередньо в коді (Java або Kotlin), XML дозволяє застосовувати декларативний підхід. Це означає, що структура та зовнішній вигляд екранів програми описуються в окремих файлах ресурсів (.xml), зазвичай розташованих у директорії res/layout проекту.

Основні концепції Android XML для GUI:

1. Ієрархічна структура: XML-файли компонування (layout files) мають деревоподібну структуру. Кожен файл містить один кореневий елемент, який є контейнером (ViewGroup), а всередині нього розташовуються інші контейнери або окремі елементи керування (View).

2. View та ViewGroup – це два фундаментальні класи, на яких будується весь інтерфейс. View представляє окремий компонент інтерфейсу, з яким користувач може взаємодіяти або який відображає інформацію. Прикладами є TextView (для тексту), Button (кнопка),

EditText (поле для введення тексту), ImageView (зображення), ProgressBar (індикатор прогресу) тощо.

ViewGroup є невидимим контейнером, завдання якого – організовувати дочірні View та/або інші ViewGroup. ViewGroup визначає правила розташування елементів усередині себе. Поширеними прикладами є LinearLayout (розміщує елементи послідовно, горизонтально або вертикально), RelativeLayout (розміщує елементи відносно один одного або батьківського контейнера), ConstraintLayout (дозволяє створювати складні та гнучкі інтерфейси за допомогою зв'язків між елементами, є рекомендованим підходом Google), FrameLayout (зазвичай використовується для відображення одного елемента або елементів, що накладаються один на одного).

3. Атрибути: Кожен елемент (View або ViewGroup) у XML-файлі має набір атрибутів, які визначають його властивості та зовнішній вигляд. Найважливіші атрибути включають:

android:id унікальний ідентифікатор елемента (@+id/my_element), який дозволяє звертатися до нього з програмного коду;

android:layout_width та android:layout_height визначають ширину та висоту елемента. Можуть приймати значення match_parent (зайняти весь доступний простір батьківського контейнера), wrap_content (зайняти мінімально необхідний простір для вмісту) або конкретні розміри (наприклад, 100dp);

android:text встановлює текст для елементів типу TextView, Button, EditText;

android:textColor, android:textSize, android:background визначають колір тексту, розмір шрифту, фон елемента відповідно;

android:padding внутрішні відступи елемента;

android:layout_margin зовнішні відступи елемента від сусідніх елементів або меж контейнера.

4. Ресурси: XML-компонування тісно інтегровані із системою ресурсів Android. Замість жорсткого кодування значень (текст, кольори, розміри) безпосередньо в XML, рекомендовано виносити їх в окремі ресурсні файли (res/values/strings.xml, res/values/colors.xml, res/values/dimens.xml тощо). Посилання на ці ресурси здійснюється через спеціальний синтаксис (@string/app_name, @color/colorPrimary, @dimen/default_margin). Це значно спрощує локалізацію програми, підтримку різних тем оформлення та внесення змін до дизайну.

Переваги XML для GUI в Android-додатках:

Код (логіка) і опис інтерфейсу окремо – простіше підтримувати й розвивати.

Layout Editor показує зміни в реальному часі без перезапуску.

Ресурси та ViewGroup-контейнери гарантують коректне відображення на різних розмірах і щільності пікселів.

Можна правити XML без заглиблення в Kotlin/Java.

У рамках реалізації графічного інтерфейсу застосовано архітектуру з єдиною активністю (MainActivity) та множиною фрагментів, що відповідають за різні екрани гри. Для динамічного відображення фрагментів використовується FragmentContainerView, а навігаційна панель реалізована за допомогою MaterialToolbar. Такий підхід дозволяє ефективно управляти навігацією в межах одного екрану, спрощує обробку життєвого циклу компонентів і забезпечує масштабованість інтерфейсу.

У більшості екранів гри як фонову анімацію використано ефект снігопаду, реалізований за допомогою бібліотеки com.github.jetradarmobile:android-snowfall. Щоб уникнути дублювання коду, розмітку з ефектом снігу було винесено в окремий XML-файл, який підключається до потрібних екранів за необхідністю через тег <include>. Такий підхід спрощує підтримку інтерфейсу та забезпечує єдиний стиль оформлення для всього застосунку.

Фрагмент коду, що відповідає за підключення фону з анімацією снігу:

<include layout="@layout/snow_background" />

3.5 Структура гри та навігація між екранами

Після запуску гри користувач потрапляє на головне меню, реалізоване у вигляді окремого фрагмента MenuFragment. Інтерфейс цього екрана містить заголовок гри у верхній частині та три основні кнопки: для гри з ботом, онлайн-гри та локального мультиплеєра. Для створення ефектного вигляду до кожної кнопки застосовано індивідуальний фон із градієнтом.

У верхньому правому куті відображається іконка профілю гравця, натискання на яку відкриває екран редагування профілю. Ця іконка реалізована через інтерфейс HasCustomAction, що дозволяє гнучко додавати дії до тулбара з різних фрагментів.

Після натискання кнопки «ОДИН ГРАВЕЦЬ» користувач переходить на екран вибору складності гри з ботом. Заголовок тулбара змінюється на «Оберіть складність», а в правому верхньому куті з’являється іконка налаштувань, що відкриває меню, де можна вибрати, хто ходить першим, і змінити ігровий символ (хрестик або нулик).

На самому екрані розміщено три кнопки, кожна з яких відповідає певному рівню складності: легкий, середній і складний бот. Для кожної кнопки передбачено відповідну стилізацію й візуальний супровід у вигляді іконок. Фонову анімацію снігу реалізовано через включення snowfall_background, що забезпечує єдиний візуальний стиль застосунку.

На головному екрані гри також присутня кнопка «МУЛЬТИПЛЕЄР», яка зарезервована для майбутньої реалізації онлайн-режиму гри. Наразі ця функціональність ще не впроваджена, проте її наявність у інтерфейсі дозволяє легко масштабувати застосунок у майбутньому та додати можливість гри з іншими користувачами через інтернет без змін у структурі меню.

Остання кнопка на головному екрані – «Гра удвох». Натиснувши на неї, ви потрапляєте на екран, де двоє гравців можуть зіграти на одному пристрої. Ігрове поле розташоване по центру з рівномірними відступами, а над ним – по обидва боки – картки гравців: ваш профіль та умовного опонента. Хід і символ кожному призначаються випадковим чином. Після кожного ходу черговість змінюється, а коли гра закінчується – відображається екран з результатом: перемога, поразка або нічия.

На першому екрані натиснувши на іконку профіля ви попадаєте на екран редагування профіля.

Екран редагування профілю дозволяє користувачам змінювати нікнейм, вибирати колір аватара або завантажувати фотографію з галереї як аватар. Він складається з таких елементів:

1. Аватар профілю. Вгорі екрану розташовано зображення аватара, яке можна змінити, вибравши одну з запропонованих кольорових іконок або завантаживши фото з галереї. Іконки кольорів дозволяють вибрати фон для аватара, а кнопка «Вибрати фото» дозволяє вибрати фотографію з пристрою.

2. Вибір кольору аватара. Декілька кольорових іконок, розташованих у двох горизонтальних рядах, дають змогу користувачеві вибрати один з кольорів для аватара. Коли користувач натискає на певну іконку, вона змінює свій вигляд на виділений, що вказує на вибір.

3. Редагування нікнейму. Користувач може змінити свій нікнейм за допомогою текстового поля. Під полем для введення нікнейму є текст, що показує кількість введених символів, а також попередження, якщо нікнейм перевищує 15 символів.

4. Кнопка «Зберегти зміни». У нижній частині екрану розташована кнопка для збереження змін. При натисканні зберігаються нові налаштування профілю, і

користувач отримує підтвердження успішного оновлення.

На екрані вибору складності натиснувши на іконку шестерні ви попадаєте на екран налаштування гри з ботом.

На екрані вибору рівня складності, натиснувши на іконку шестерні, ви переходите до налаштувань гри з ботом.

Цей екран дозволяє вибрати, хто робитиме перший хід (ви, комп'ютер чи випадковий вибір), а також який символ (X або O) буде у користувача. Екран містить такі елементи:

1. Вибір першого ходу. Перший блок дає можливість обрати того, хто розпочне гру. Є три варіанти: ви, комп'ютер або випадковий вибір. Кожен варіант представлений рядком із фоном, аватаркою та назвою. При натисканні на номер варіанту він підсвічується, а попередній вибір скасовується.

2. Вибір символу гравця. Наступний блок дозволяє обрати символ, яким гратиме користувач – X, O або випадковим. Варіанти оформлені аналогічно до попереднього блоку й мають свої кольори та іконки. Активний варіант візуально підсвічується.

3. Кнопки керування. Внизу екрана знаходяться кнопки «Скасувати» і «Зберегти». Кнопка «Скасувати» повертає користувача назад без збереження змін. Кнопка «Зберегти» зберігає обрані параметри в локальне сховище й показує повідомлення про успішне оновлення.

Після вибору рівня складності бот-гри, відкривається основний екран гри з комп'ютером. Цей екран містить поле гри 3x3, інформацію про гравця та суперника, і візуально відображає хід гри. Екран складається з таких частин:

У верхній частині екрана розміщено дві картки – ваша та бота. Кожна з них містить ім'я, аватарку, символ (X або O) та індикатор активності. Поточний гравець підсвічується, а неактивний – затухає.

У центрі екрана - квадратна дошка з 9 клітинок. Вона реагує на дотики: при натисканні вільної клітинки встановлюється символ гравця, після чого бот автоматично робить свій хід (затримка мінімальна, залежить від складності).

Хід бота обчислюється через алгоритм Minimax, який адаптується до рівня складності. Після кожного ходу перевіряється стан гри: перемога, нічия або продовження. У разі завершення гри з'являється екран результату з детальним підсумком.

Аватар гравця завантажується з профілю або генерується. Бот отримує аватар відповідно до рівня (легкий, середній, складний). Символи X і O вибираються за попередніми налаштуваннями. На рисунку 3.1 зображено структурну схему гри

«Хрестики-нулики».

Рисунок 3.1 – Структурна схема гри «Хрестики-нулики»

Інтерфейс побудований на базі ViewBinding, а всі процеси – асинхронні. Користувач бачить плавну анімацію переходів, чіткі візуальні стани та мінімум зайвих елементів. Після натиснення на кнопку з надписом «Гра у двох» відкривається екран локальної гри, де двоє гравців змагаються на одному пристрої. Цей екран поєднує у собі інтуїтивну візуалізацію стану гри, чітке представлення гравців та плавний ігровий процес. Складові елементи екрану:

Угорі розміщено дві картки. Ліворуч - гравця, праворуч - суперника. Обидві містять нікнейм, аватар, символ (X або O) та індикатор активності. Той, чия черга ходити - підсвічується, інший - неактивний.

У центрі – кастомна дошка BoardView розміром 3x3, з адаптивною анімацією клітинок. Гравці по черзі натискають на клітинки, й символ автоматично з’являється.

Перший хід визначається випадково. Після кожного ходу перевіряється стан гри: перемога одного з гравців, нічия або продовження. Якщо гра завершується, автоматично відкривається екран з підсумками, де відображаються імена, символи та аватари учасників.

Аватар гравця завантажується з локального профілю або генерується у вигляді ініціалів. Суперник має стандартний вигляд і червоний фон. Символи гравців (X або O) обираються випадково на початку гри. Усе побудовано з використанням ViewBinding та кастомних елементів інтерфейсу (EntityCardView, BoardView). Після завершення партії (перемога, поразка чи нічия) відкривається екран результату гри. Цей екран відображає підсумки матчу у зрозумілому й візуально привабливому форматі. Він складається з наступних компонентів:

Фоновий ефект падаючого снігу (snowfall_background) створює атмосферу завершеності. Уся розмітка побудована на ConstraintLayout з використанням guideline для чіткого позиціонування.

У верхній частині – дві вертикально вирівняні картки з інформацією про учасників. Вони показують ім’я, аватар, символ і опис типу гравця. Дані для карток передаються через Bundle під час створення фрагмента.

Центральна частина екрана – кастомний елемент, що відображає фінальний рахунок, хто переміг, або що гра завершилась нічиєю. Має квадратну форму (співвідношення сторін 1:1), адаптується до ширини екрана.

У нижній частині екрана – кнопка «нова гра». При натисканні видаляє два останні фрагменти зі стеку (гру і результат) і викликає колбек onRefreshClick, який дозволяє запустити нову гру.

Натискання кнопки «назад» також видаляє обидва фрагменти, забезпечуючи послідовне повернення до головного меню без збереження попереднього стану.

3.6 Власні компоненти інтерфейсу гри

У проєкті використовуються кілька власних віджетів (custom views), що дозволяють створити унікальні елементи інтерфейсу з налаштовуваною логікою та зовнішнім виглядом. Серед них можна виділити:

BoardView – віджет для відображення ігрового поля, що реагує на дії користувача та візуально відображає хід гри. Він відповідає за малювання сітки та оновлення стану клітинок.

EntityCardView – картка для представлення гравця або бота з інформацією про їх статус (наприклад, який символ використовують, хто ходить). Картка також містить аватарки та підсвітлення активного гравця.

ResultGameView – віджет, що відображає результат гри, показуючи, хто переміг або чи була нічия. Він має спеціальний дизайн для чіткої ілюстрації підсумків гри.

Ці власні елементи дозволяють гнучко налаштовувати інтерфейс і логіку відображення, забезпечуючи зручний і інтуїтивно зрозумілий досвід для користувача.

У проєкті компонент BoardView реалізований як кастомний елемент інтерфейсу з використанням компонування через тег <merge> у XML-розмітці. Це дозволяє ефективно об’єднувати декілька ImageView, які представляють клітинки ігрового поля, всередині контейнера без створення зайвого рівня вкладеності. Компонент побудований на базі ConstraintLayout і інкапсулює всю логіку взаємодії з клітинками, обробку кліків, зміну стану гри та анімацію виграшної лінії.

У проєкті компонент EntityCardView реалізований як кастомний елемент інтерфейсу з використанням компонування через тег <merge> у XML-розмітці. Це дозволяє інтегрувати вміст картки без додаткових обгорток, що оптимізує ієрархію в'юшок. Компонент побудований на базі ConstraintLayout і інкапсулює в собі всю логіку відображення гравця або бота: ім’я, аватар, маркер та опис. Підтримується динамічна зміна стану (активний/неактивний), завантаження даних з моделі EntityCard, адаптація розміру тексту до ширини екрана та збереження стану щоб був на початку гри.

У проєкті компонент ResultGameView реалізований як повністю власний View, який

відображає результат гри з аватаром переможця та відповідним текстом. Він не використовує XML-компонування, а повністю рендериться вручну в onDraw, що дає повний контроль над виглядом. Компонент малює заокруглений фон, динамічні декоративні трикутники по краях і аватар всередині кола. Підтримується адаптивний розмір тексту, збереження та відновлення стану, а також завантаження аватара як з ресурсу, так і з Bitmap. Все оформлення витримане в одному стилі за допомогою Paint-об'єктів і кастомних Path.

3.7 Графічні ресурси та стилізація інтерфейсу

У застосунку використовується градієнт, визначений у drawable-ресурсі як фон прямокутної форми. Він створює плавний перехід кольорів від насиченого темно-фіолетового #1e0e4a до яскравішого фіолетового #3f31a5 під кутом 45 градусів. Центр зміщено до лівого боку (centerX= «0.40»), що надає фону динамічного вигляду. Такий градієнт візуально підсилює атмосферу застосунку, особливо в інтерфейсах з темною кольоровою палітрою.

Повний код градієнту:

<?xml version="1.0" encoding="utf-8"?>

<shape xmlns:android="http://schemas.android.com/apk/res/android"

android:shape="rectangle">

<gradient

android:angle="45"

android:centerX="0.40"

android:endColor="#3f31a5"

android:startColor="#1e0e4a"

android:type="linear" />

</shape>

У кастомному компоненті EntityCard використовуються два drawable-ресурси у формі прямокутників із заокругленими кутами (<corners android:radius=«18dp» />) та фоном кольору dark_indigo. Один із них має білу рамку (<stroke>), інший – без рамки. Це дозволяє візуально відрізняти активну або вибрану картку від неактивної, змінюючи фон програми в залежності від стану.

Код заокругленого прямокутника з рамкою:

<shape xmlns:android="http://schemas.android.com/apk/res/android"

android:shape="rectangle">

<solid android:color="@color/dark_indigo" />

<corners android:radius="18dp" />

<stroke

android:width="2dp"

android:color="@color/white" />

</shape>

Код заокругленого прямокутника:

<shape xmlns:android="http://schemas.android.com/apk/res/android"

android:shape="rectangle">

<solid android:color="@color/dark_indigo" />

<corners android:radius="18dp" />

</shape>

На екрані вибору складності кожна кнопка має власний кольоровий стиль із закругленими кутами та контуром. Їхній фон визначений у окремих XML-файлах через selector, який змінює колір при натисканні. Ось приклад одного з таких файлів:

<?xml version="1.0" encoding="utf-8"?>

<selector xmlns:android="http://schemas.android.com/apk/res/android">

<item android:state_pressed="true">

<shape android:shape="rectangle">

<solid android:color="@color/indigo_violet_pressed" />

<stroke android:width="2dp" android:color="@color/pink" />

<corners android:radius="40dp" />

</shape>

</item>

<item android:state_pressed="false">

<shape android:shape="rectangle">

<solid android:color="@color/indigo_violet" />

<stroke android:width="2dp" android:color="@color/pink" />

<corners android:radius="40dp" />

</shape>

</item>

</selector>

Інші кнопки використовують аналогічну структуру, відрізняючись лише значеннями кольорів у solid та stroke. Такий підхід забезпечує узгоджений вигляд і дозволяє легко змінювати стилі для різних рівнів складності.

Кнопки на головному меню мають фони з градієнтним переходом кольорів, які змінюються при натисканні. Це реалізовано через selector, який визначає окремий стан для звичайного вигляду та стану натиснення. Нижче приклад одного з таких XML-файлів:

<?xml version="1.0" encoding="utf-8"?>

<selector xmlns:android="http://schemas.android.com/apk/res/android">

<item android:state_pressed="true">

<shape android:shape="rectangle">

<corners android:radius="40dp" />

<gradient

android:angle="45"

android:centerX="0.35"

android:centerColor="#A822B2"

```
android:startColor="#B92072"

android:endColor="#8733DF"

android:type="linear"

/>

</shape>

</item>

<item>

<shape android:shape="rectangle">

<corners android:radius="40dp" />

<gradient

android:angle="45"

android:centerX="0.35"

android:centerColor="#C22FB9"

android:startColor="#D92982"

android:endColor="#A733EF"

android:type="linear" />

</shape>

</item>

</selector>
```

Інші кнопки в головному меню оформлені аналогічно, з тією ж структурою selector, але з різними значеннями кольорів у градієнтах. Такий підхід дозволяє витримати єдиний стиль і водночас зробити кожну кнопку візуально унікальною.

Іконки хрестика та нулика були створені у Figma у векторному форматі для чіткості на різних екранах. Також у Figma були реалізовані іконки слабкого, середнього та сильного бота - кожна з унікальним стилем, що відповідає рівню складності.

Стрілка, яка візуалізує випадковий вибір, була взята з сайту безкоштовних іконок IconScout. З цього ж ресурсу були використані: іконка профілю, іконка «Нова гра» (refresh), іконка робота, а також шестерня, що символізує самі налаштування. Усі іконки були підібрані для збереження єдиного стилю інтерфейсу. Ось XML-файл, який задає фон дошки:

<?xml version="1.0" encoding="utf-8"?>

<shape xmlns:android="http://schemas.android.com/apk/res/android"

android:shape="rectangle">

<solid android:color="@color/board_background" />

<corners android:radius="16dp" />

</shape>

На екрані налаштувань для виділення вибраних пунктів використовується спеціальний фон елемента. Він задається у вигляді прямокутника з заокругленими кутами, темно-синім фоном та блакитною рамкою. Приблизний вигляд такого фону представлений у цьому файлі:

<shape xmlns:android="http://schemas.android.com/apk/res/android"

android:shape="rectangle">

<solid android:color="#333f98" />

<corners android:radius="16dp" />

<stroke

android:width="2dp"

android:color="#3a80f3" />

</shape>

Інші подібні елементи оформлені аналогічно, лише з відмінностями у кольорах.

Кольори в проекті зберігаються у файлі colors.xml, який знаходиться в папці res/values. Цей файл містить всі кольори, що використовуються у додатку, і дозволяє централізовано керувати кольоровою палітрою застосунку. Кожен колір визначається за допомогою унікального імені, яке потім використовується у ресурсах, таких як фони,

текст, кнопки тощо.

Цей підхід є важливим для підтримки консистентності дизайну в межах застосунку та спрощує зміну кольорової теми, оскільки всі зміни можна внести лише в цьому файлі.

У процесі реалізації графічного інтерфейсу всі текстові рядки проєкту були винесені у ресурсний файл strings.xml, що відповідає стандартам Android-розробки та забезпечує легку локалізацію. У проєкті використано два таких файли: один для мови за замовчуванням (англійської), інший – для української. Це дозволяє динамічно змінювати мову інтерфейсу залежно від мовних налаштувань пристрою користувача.

У проєкті частково використовуються розміри, визначені у файлі dimens.xml, який містить стандартні значення відступів, розмірів тексту та інших параметрів інтерфейсу. Це дозволяє централізовано керувати візуальним виглядом елементів, спрощуючи підтримку та адаптацію під різні екрани. Оскільки українські текстові рядки зазвичай довші за англійські, у проєкті також використовується окремий файл dimens.xml для української локалізації (values-uk/dimens.xml), в якому задаються змінені розміри для забезпечення коректного відображення інтерфейсу.

У проєкті використовується файл themes.xml, однак теми як такі не застосовуються – файл служить для зберігання окремих стилів елементів інтерфейсу. Зокрема, в ньому описано зовнішній вигляд кнопок головного меню, елементів складності, налаштувань тощо. Ці стилі включають параметри розміру, кольору, шрифту та відступів, що дозволяє досягти єдиного дизайну без створення повноцінної теми.

У проєкті також використовується файл attrs.xml, у якому визначено набір кастомних атрибутів для власних компонентів інтерфейсу. Зокрема, описано атрибути для EntityCardView та ResultGameView, що дозволяють задавати назву, опис, маркер, аватар та стан активності. Це спрощує створення повторно використовуваних UI-елементів із налаштовуваними властивостями.

<?xml version="1.0" encoding="utf-8"?>

<resources>

<attr name="avatar" format="reference" />

<declare-styleable name="EntityCardView">

<attr name="name" format="string" />

<attr name="mark" format="reference" />

<attr name="description" format="string" />

<attr name="active" format="boolean" />

<attr name="avatar" />

</declare-styleable>

<declare-styleable name="ResultGameView">

<attr name="text" format="string" />

<attr name="avatar" />

</declare-styleable>

</resources>

Усі файли та XML-розмітка, що були розглянуті вище, наведені у додатку А.

3.8 Ігрова логіка та алгоритм для пошуку оптимальних стратегій

У проєкті реалізація ігрової логіки чітко структурована через окремі компоненти та інтерфейси. Зокрема, існує пакет contract, який містить набір контрактів для навігації між екранами, а також керування заголовком і кастомною дією на тулбарі. Інтерфейс Navigator визначає основні переходи між фрагментами: запуск гри проти бота, відкриття екрану вибору складності, редагування профілю, налаштувань, гри удвох, екрану завершення гри та інше. Це дозволяє фрагментам бути незалежними від конкретної реалізації навігації та логіки переходів, зберігаючи чисту архітектуру.

Крім того, інтерфейси HasCustomAction і HasCustomTitle надають фрагментам можливість змінювати дії та заголовки тулбара в залежності від контексту. Таким чином, ці контракти разом забезпечують гнучку й контрольовану інтеграцію користувацького інтерфейсу з логікою гри. У додатку Б наведено повний код інтерфейсів навігації, які відповідають за керування переходами між екранами, кастомними діями та заголовками в інтерфейсі додатку.

У проєкті головний клас MainActivity реалізує інтерфейс Navigator, що дозволяє керувати переходами між різними екранами застосунку. Він відповідає за ініціалізацію інтерфейсу, запуск фрагментів, а також за оновлення тулбара залежно від контексту. У цьому класі обробляються переходи на фрагменти, такі як головне меню, гра з ботом, гра удвох, екран складності, налаштування, редагування профілю та екран результату гри. Крім того, MainActivity динамічно змінює заголовок тулбара та додає кастомну дію, якщо активний фрагмент реалізує інтерфейси HasCustomTitle або HasCustomAction. Це забезпечує гнучкість інтерфейсу та зручність у навігації між екранами. У додатку В

наведений повний код MainActivity.

BoardView – це кастомний компонент, що відповідає за візуалізацію і логіку ігрового поля "Хрестики-нулики". Він реалізує взаємодію з користувачем, обробляє кліки по клітинках, відображає хід гравця або бота, визначає стан гри (перемога, нічия, продовження) та виконує анімації переможної комбінації. Компонент підтримує збереження та відновлення свого стану при зміні конфігурації пристрою, а також надає API для підписки на зміну стану гри та хід гравця.

EntityCardView – це кастомний візуальний компонент, створений на основі ConstraintLayout, який відображає картку гравця або бота. Компонент забезпечує можливість показу імені, аватару, опису та позначки (іконки хрестика або нулика). Підтримується як ресурсне, так і бітмапове зображення. EntityCardView дозволяє відображати активний стан гравця шляхом зміни фону й видимості опису. Крім цього, реалізоване масштабування тексту відповідно до ширини екрана та підтримка збереження й відновлення стану через Parcelable, що гарантує коректну роботу при зміні конфігурації. Компонент також підтримує ініціалізацію атрибутів із XML.

ResultGameView – це кастомний Android-компонент, який наслідує View і використовується для візуалізації результату гри. Він відображає фоновий прямокутник із заокругленнями, декоративні трикутники по краях, круглу аватарку гравця та текст результату (наприклад, «Перемога» чи «Нічия»). Компонент підтримує завантаження аватарки як з ресурсу, так і з об'єкта Bitmap. Всі розміри та координати розраховуються динамічно на основі ширини в’ю. Є метод loadResultGame, який дозволяє оновити дані, а також реалізовано збереження стану через серіалізацію ResultGame. Малювання відбувається в onDraw, а адаптація до розміру – в onSizeChanged. Внутрішньо використовуються Paint, Path, Canvas і ClipPath для створення візуального оформлення.

Код ResultGameView, EntityCardView, BoardView наведений у додатку Г.

У проєкті використовуються власні класи даних, що зберігаються в пакеті com.example.tictactoe.data. Серед них є:

EntityCard – модель, яка містить ім’я, опис, аватар (у вигляді Bitmap) та оцінку. Підтримує серіалізацію через Parcelable.

Point – проста структура для збереження координат (x, y) у вигляді Float.

Profile – клас для профілю гравця з полями для нікнейму, аватарки (Uri) та обраного кольору (ColorStateList). Має метод default, який створює профіль за замовчуванням.

ResultGame – модель результату гри з аватаркою переможця (Bitmap) і рядком з текстом результату. Також реалізує Parcelable.

Settings – конфігураційний клас, який зберігає налаштування гри: випадковий вибір хрестика/нулика, перший хід, та чи гравець починає.

Ці структури забезпечують серіалізацію, передачу даних між компонентами та кастомну логіку збереження графічних даних. У додатку Е наведено повний код цих класів.

У проєкті реалізовано кілька фрагментів, кожен з яких відповідає за окремий функціональний блок: головне меню, профіль користувача, налаштування, вибір складності, поле гри, результати та інші. Фрагменти взаємодіють між собою через навігацію та спільні дані, забезпечуючи зручний і логічний користувацький інтерфейс.

MenuFragment – це фрагмент, що відповідає за головне меню гри. Він містить кнопки для запуску різних режимів: одиночної гри, локального мультиплеєра та (у перспективі) мережевого мультиплеєра. Також реалізована підтримка кастомної дії в тулбарі - перехід до редагування профілю користувача. Навігація між екранами виконується через контракт navigator().

DifficultyFragment – це фрагмент, що дозволяє користувачу обрати рівень складності гри проти бота: легкий, середній або складний. Для кожного рівня реалізовано окрему дію, яка відкриває відповідний ігровий екран. Також фрагмент має власний заголовок для тулбара та кастомну дію - перехід до налаштувань гри.

BotGameFragment – це фрагмент, у якому реалізована гра проти бота з різними рівнями складності. Тут обробляється логіка гри, рендеряться гравці (користувач і бот), вибираються відповідні аватари, позначки й порядок ходів згідно з налаштуваннями. Для гри використовується алгоритм Minimax, який приймає рішення за бота. Після завершення гри фрагмент переходить до екрана результату.

EditProfileFragment – це фрагмент для редагування профілю користувача. Він дозволяє:

змінити нікнейм (з перевіркою на довжину до 15 символів);

вибрати фото з галереї або один із кольорових аватарів;

переглянути попередній перегляд аватара;

зберегти зміни у SharedPreferences.

Також фрагмент має кнопку «Готово» в тулбарі, яка виконує ту ж дію, що й кнопка Save.

SettingsFragment дозволяє користувачеві налаштовувати параметри гри, такі як:

1. Вибір мітки для гравця (Tic або Tac або випадковий вибір).

2. Вибір, хто робить перший хід (людина або комп'ютер, або випадковий вибір). Основні функціональні можливості:

При виборі мітки чи ходу оновлюється інтерфейс та зберігаються зміни в налаштуваннях.

Є кнопка для збереження змін та кнопка для скасування.

Профіль користувача зображується на екрані завдяки AvatarManager.

Настройки зберігаються в SharedPreferences, а після збереження користувач отримує повідомлення через SnackBar.

LocalMultiplayerFragment є фрагментом для гри в хрестики-нулики в режимі двох гравців на одному пристрої. Ось основні моменти цього фрагмента: Профіль користувача та профіль супротивника отримуються з SharedPreferences. Профіль користувача має аватар, яке відображається за допомогою AvatarManager.

За допомогою випадкового вибору визначається, хто буде грати «X» (TIC), а хто «O» (TAC). Гравці по черзі роблять ходи на полі, що оновлюється у відповідь на кожен хід. Гравець може робити хід, натискаючи на клітинки поля, після чого хід передається супротивнику.

При кожному ході оновлюється стан гри, і активність між гравцями чергується. Візуально це відображається через позначку активного гравця на картках профілю. Коли гра закінчується (виграш, програш або нічия), показується екран з результатами гри. Також на ньому відображаються аватари обох гравців і результат (виграш, програш або нічия).

Використовуються Glide та AvatarManager для завантаження аватарок гравців. Якщо аватарка відсутня, генерується текстовий аватар. Цей фрагмент реалізує всі основні функції для гри в локальний мультиплеєр, включаючи зміни стану гри, візуальні оновлення та збереження результатів.

GameOverFragment – це фрагмент, який відображає результати гри після її завершення. Він отримує три об'єкти з аргументів: картки гравця та супротивника, а також результат гри. Ці дані відображаються в UI, використовуючи методи loadEntityCard та loadResultGame. Фрагмент також обробляє натискання кнопки "Refresh", викликаючи передану функцію onRefreshClick, що дозволяє перезапустити гру або повернутись до попереднього екрану. Також, при натисканні на кнопку «Назад», виконується два рази popBackStack(), що дозволяє відкотити стек фрагментів і повернутися до попереднього екрану. Фрагмент сумісний з різними версіями Android завдяки використанню методу getParcelableCompat для отримання об'єктів типу Parcelable з Bundle. Повний код

фрагментів наведений в додатку Ж.

У проєкті є пакет utils з файлами AvatarManager, BitmapConverter, SnackBarUtils. Клас AvatarManager відповідає за керування аватарками користувачів, використовуючи їхній профіль. Основні функції:

setAvatar(imageView: ImageView). Ця функція встановлює аватар користувача в переданий ImageView. Якщо профіль не має URI для аватара, створюється текстовий аватар, що містить першу літеру імені користувача (або пробіл, якщо ім'я порожнє). Якщо URI присутній, використовується бібліотека Glide для завантаження аватара з цього URI;

createTextDrawable(letter: String, color: Int, imageView: ImageView). Створює текстовий аватар у вигляді кола з літерою в середині. Аватар малюється на бітмапі, використовуючи надані параметри (літера та колір);

createTextBitmap(). Створює бітмап із текстовим аватаром, використовуючи першу літеру імені користувача, або знак питання, якщо ім'я порожнє.

Файл BitmapConverter.kt містить клас з функціями для конвертації зображень між різними форматами. До них слід віднести:

bitmapToByteArray(bitmap: Bitmap). Конвертує зображення у форматі Bitmap в масив байтів;

byteArrayToBitmap(byteArray: ByteArray). Конвертує масив байтів назад у Bitmap;

uriToBitmap(contentResolver: ContentResolver, fileUri: Uri?). Перетворює URI на Bitmap. Використовуються різні методи в залежності від версії Android (зокрема, використовується ImageDecoder для Android 9+);

vectorDrawableToBitmap(vectorDrawable: VectorDrawable). Конвертує векторне зображення (VectorDrawable) в бітмап.

Файл SnackBarUtils.kt містить клас, який допомагає створювати спеціальні спливаючі повідомлення (що з'являються внизу екрана, як SnackBar). Метод цього класу showCustomSnackBar використовується для того, щоб показати таке повідомлення. Він дозволяє налаштувати, вигляд:

Вказати, де саме на екрані показати повідомлення (view).

Задати сам текст повідомлення (message).

Вибрати колір фону для цього спливаючого вікна (backgroundColor).

Клас AvatarManager дозволяє обробляти аватари користувачів: завантажувати їх за URI або генерувати текстовий аватар.

Клас BitmapConverter містить функції для роботи з зображеннями в різних форматах (від бітмапів до URI та зворотно).

Клас SnackBarUtils спрощує створення кастомних SnackBar для покращення UX, даючи можливість налаштовувати вигляд повідомлень.

Файли з цими класами допомагають організувати зручні інструменти для роботи з аватарами, зображеннями та повідомленнями в додатку. У додатку Д наведено повний код пакету utils.

У пакеті ai реалізовано логіку штучного інтелекту для гри «Хрестики-нулики». Основні компоненти цього модуля:

Перелічення Mark – це enum, що містить три значення: TIC (хрестик), TAC (нулик) та EMPTY (порожня клітинка). Він використовується для позначення стану кожної клітинки на ігровому полі.

Клас Board представляє ігрове поле розміром 3×3. Він інкапсулює масив типу Mark і забезпечує зручний доступ до клітинок через оператори get і set. Основні методи класу Board:

getEmptyCells() – повертає список координат усіх порожніх клітинок;

getGrid() – повертає поточну сітку з мітками;

Board.empty() – створює нову порожню дошку.

Клас Minimax реалізує алгоритм Мінімаксу для вибору оптимального ходу комп’ютера. Він приймає такі параметри: поточну дошку (Board), маркер користувача (Mark), максимальну глибину пошуку (maxDepth). Основні методи класу Minimax:

evaluate() – оцінює поточний стан гри: повертає +10, якщо виграв комп’ютер, -10, якщо виграв користувач, або `0`, якщо нічия чи гра ще не завершена;

minimax() – рекурсивний метод, що реалізує сам алгоритм пошуку оптимального ходу з урахуванням глибини;

findBestMove() – запускає алгоритм Мінімакс і повертає найкращий хід для комп’ютера у вигляді пари координат.

У додатку К наведено повний код пакету ai. На рисунку 3.2 візуально зображено основні компоненти пакету ai.

Рисунок 3.2 – Основні компоненти пакету ai

У проєкті також є пакет extensions, який спрощує роботу з SharedPreferences, дозволяючи зберігати і витягувати цілі об'єкти, а не тільки примітиви. Він використовує Gson з кастомним адаптером для Uri, щоб коректно серіалізувати й десеріалізувати URI-поля, наприклад, аватари профілів. Це дозволяє легко зберігати дані профілю користувача в локальне сховище без зайвого коду.

3.9 Тестування програми на Android-пристрої

Для перевірки працездатності та коректності роботи розробленої програми було проведено тестування на реальному фізичному Android-пристрої. В якості тестового пристрою використовувався особистий смартфон «Poco x5 pro 5g».

Тестування включало наступні етапи:

Запуск та налагодження. Програма компілювалася та встановлювалася на підключений через USB смартфон безпосередньо з середовища розробки Android Studio.

Моніторинг логів. Активно використовувалася вкладка Logcat в Android Studio для відстеження системних повідомлень та повідомлень самої програми в режимі реального часу. Це дозволяло моніторити ключові події життєвого циклу активностей, реакцію на дії користувача та виявляти потенційні помилки чи неочікувану поведінку.

Використання точок останова (Breakpoints). Для детального аналізу виконання коду та перевірки значень змінних на критичних ділянках встановлювалися точки останова. Це дозволило покроково пройти виконання функцій, перевірити логіку роботи алгоритмів та переконатися у коректності обробки даних.

Такий підхід дозволив не лише перевірити функціональність програми в умовах, наближених до реального використання, але й ефективно виявити та виправити помилки на етапі розробки.

Приклад виводу логів під час роботи програми наведено на рисунку 3.3.

Рисунок 3.3 – Вивід логів

3.10 Візуалізація гри

На рисунку 3.4 зображено головне меню програми. Це стартовий екран, який 61 надає користувачеві доступ до основних функцій застосунку, таких як початок нової гри в

різних режимах, перехід до редагування профілю.

Рисунок 3.4 – Головне меню програми

На рисунку 3.5 показано інтерфейс екрану редагування профілю користувача. В цьому розділі гравець може змінити своє ім'я або аватар, що використовуються для ідентифікації в грі.

Рисунок 3.5 – Екран редагування профілю користувача

На рисунку 3.6 представлено екран вибору рівня складності. Цей екран з'являється перед початком гри проти комп'ютерного супротивника (бота) і дозволяє користувачеві обрати бажаний рівень майстерності штучного інтелекту.

Рисунок 3.6 – Екран вибору рівня складності гри

На рисунку 3.7 продемонстровано основний ігровий екран у режимі гри для двох гравців. Тут відображається ігрове поле, поточний стан гри та елементи керування, необхідні для взаємодії двох користувачів на одному пристрої.

Рисунок 3.7 – Екран ігрового процесу для двох гравців

На рисунку 3.8 показано екран налаштуваня гри. У цьому розділі користувач може керувати різними параметрами гри, такими як вибір ходу, вибір ігрового символа хрестика або нулика.

Рисунок 3.8 – Екран налаштування гри

На рисунку 3.9 представлено екран відображення результатів завершеної партії гри. На цьому екрані показується переможець, фінальний рахунок або повідомлення про нічию, а також є опція для початку нової гри або повернення в головне меню.

Рисунок 3.9 – Екран результатів гри

4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ

Завданням дипломного проєкту є розробка 55 програмного забезпечення для пошуку оптимальних стратегій у грі «Хрестики-нулики» з використанням методу Minimax. 2 Ефект від використання розробленого рішення полягає у практичній реалізації алгоритмів прийняття рішень у середовищі з повною інформацією, що дає змогу моделювати поведінку раціонального гравця. Економічний ефект полягає в спрощенні процесу тестування та демонстрації стратегічних моделей, а також у можливості використання додатку як навчального інструменту для ознайомлення з основами теорії

ігор та штучного інтелекту.

4.1 Розрахунок витрат на розробку та впровадження проєктного рішення.

Витрати на розробку та впровадження програмних засобів (К) включають:

K=K1+K2, (4.1)

де К1 – витрати на розробку програмного продукту, грн.; К2 – витрати на налагодження та дослідну експлуатацію програмного засобу на ПК, грн.

Витрати на розробку програмного засобу включають в себе:

Витрати на оплату праці розробників (З);

Нарахування на зарплату (НЗ);

Витрати на куповані витрати (ВК);

Накладні витрати (НВ);

Інші витрати (Інші).

Для розробки програмного продукту потрібні чотири спеціалісти-розробники, а саме:

Керівник проєкту (К);

Студент дипломник (СД);

Консультант з охорони праці (КОП);

Консультант з економічної частини (КЕ).

Згідно з штатним розписом Відокремленого структурного підрозділу «Фахового коледжу інформаційних технологій Національного університету «Львівська політехніка» місячною стипендією студента-дипломника є 1510 грн, а 1 година робочого навантаження становить для:

Керівника проєкту – 118,13 грн;

Консультанта з економічної частини – 118,13 грн;

Консультанта з охорони праці – 103,48 грн.

Денна оплата студента дипломника визначається:

1510/173 = 8,73 (грн),

де 173- 2 місячний фонд робочого часу, годин.

2 Розрахунок витрат на оплату праці всіх спеціалістів проєкту визначається за формулою:

;(4.2)

де 2 – чисельність розробників проєкту 1-ої спеціальності чол.; 28 – час, витрачений на розробку проєкту працівником 1-ої спеціальності, дні; ЗП – погодинна 2 заробітна плата розробника 1-ої 62 спеціальності, грн;

62 Таким чином, 28 витрати 2 на оплату праці розробників 62 складають:

62 Зк = 1* 14 * 118,13= 1653,82 (грн);

Зке = 1 * 1 * 118,13 = 118,13 (грн) ;

Зкоп = 1 * 1 *103,48 = 103,48(грн).

Зст = 1 * 180 * 8,73 = 1571,40(грн).

Сумарні 1 витрати на оплату праці:

1 Воп= Зк+ Зке+ Зст+ Зкоп.

Відповідно,

Воп=1653,82 +118,13+ 103,48+ 1571,40=3446,83 (грн.)

1 Розрахунок витрат на оплату праці розробників наведено 1 у таблиці 4.1.

39 Таблиця 4.1 39 – 1 Розрахунок витрат на оплату праці

1 Спеціальність розробника

17 Кількість розробників роб.

Час роботи, год.

1 Погодинна 1 заробітна плата розробника, грн.

1 Витрати на оплату праці, грн.

1 Керівник 17 проєкту

17 1

14

118,13

1653,82

39 Консультант з економічної частини

39 1

1

118,13

118,13

4 Консультант з охорони праці

4 1

4 1

103,48

103,48

Студент-дипломник

1

180

8,73

1 571,40

Всього

4

196

-

3446,83

Нарахування на зарплату становить згідно з нормативом 22% 1 від фонду оплати праці:

1 Нз= 1 Воп•22,0/100,(4.3)

1 де Воп – 1 витрати на оплату праці, тис.грн.; 44 22 – норматив нарахувань на зарплату,%.

Нз= 1875,43* 0,22 = 412,59 (грн).

4.2 1 Розрахунок витрат на куповані 18 вироби

18 Витрати 4 на куповані вироби (папір, друк) визначаються за їх 1 фактичними цінами з врахуванням найменування, номенклатури та необхідної кількості в проєкті. Транспортно-заготівельні витрати становлять 10% 17 від суми витрат на куповані вироби. 1 Розрахунок витрат на куповані 18 вироби наведено 18 в табл. 4.2.

39 Таблиця 4.2 39 – 1 Розрахунок витрат на куповані 18 вироби

18 Найменування 17 купованих виробів

17 Марка, тип

17 Кількість на розробку, 1 шт.

1 Ціна за оди- ницю, 17 грн.

17 Сума витрат, грн.

17 Папка для проєкту

Формат А4

1

150

150

Папір, пачок

Формат А4

1

200

200

Роздрук поясню-вальної записки

Формат А4

150

3

450

Разом

4 –

4 –

4 –

4 800

Транспортно-заготі-вельні витрати (10%)

4 –

4 –

4 –

4 65

Всього

4 –

4 –

4 –

4 865

4 Витрати на куповані вироби становлять:

Вк = 150+ 200 + 450 + 65 = 865,00 (грн.).

4.3 Розрахунок накладних та інших витрат

Накладні витрати (Нв) розраховуються за встановленими відсотками (30%) до витрат на оплату праці:

Нв= 3446,83* 30/100 = 1034,04 (грн).

Інші витрати розраховуються по їх питомій вазі у структурі собівартості (10%):

Він = (Воп+Нз+Нв+Вк)•10/100;(4.4)

Він = (3446,83+ 412,59+ 1034,04+865,00)*10/90 = 639,82 (грн);

Витрати на розробку проєктного рішення визначаються за формулою:

К1=Воп+Нз+Нв+Вк+Він;(4.5)

К1= 3446,83+ 412,59+ 1034,04+ 865,00+ 639,82= 6398,28 (грн);

4.4 Розрахунок витрат на налагодження проєктного рішення

Програма була розроблена протягом 50 днів із розрахунком 3.6 год на день (t=180 год.).

Потужність комп'ютерної техніки (Р) включає ноутбук, який споживає 0,12 кВт*год. Отже, вартість однієї машино-години роботи визначається за формулою:

Sm.r.= Р•Тф, (4.6)

де Р – потужність комп'ютерної техніки, кВт; Тф - вартість 1 кВт-год електроенергії, грн. (Тф=7,50).

Sm.r=0,12* 7,50= 0,9 (грн/год).

Витрати на налагодження та дослідну експлуатацію програмного продукту на ПК визначаються за формулою:

К2= Sm.r • t, (4.7)

де Sm.r. – вартість однієї машино-години роботи, грн\год; t – машинний час, витрачений на налагодження та дослідну експлуатацію програмного продукту, год.

К2 = 0,9 * 180 = 162,00 (грн).

Таким чином, витрати на розробку та впровадження програмного продукту становлять:

К = 6398,28 + 162,00 = 6560,28 (грн).

1 Кошторис витрат на розробку та впровадження проєктного рішення наведений в таблиці 4.3.

Таблиця 4.3 Кошторис витрат розробки та реалізації програмного продукту.

1 Найменування елементів витрат

1 Сума витрат, грн.

1 Витрати на оплату праці

1 3446,83

Нарахування на зарплату

412,59

4 Витрати на куповані вироби

4 865,00

Накладні витрати

1034,04

Інші витрати

639,82

Витрати на налагодження та дослідну експлуатацію

162,00

Всього (К=К1+К2)

6560,28

Таким чином, загальні 1 витрати на розробку та впровадження проєктного рішення становлять 6560,28 (грн.).

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ

5.1 Загальні вимоги 14 охорони праці при роботі з комп’ютером

Робота з персональним комп'ютером (ПК) та іншою сучасною офісною технікою

пов'язана з низкою потенційно шкідливих та небезпечних факторів виробничого середовища. Дотримання загальних вимог охорони праці є обов'язковим для всіх працівників, діяльність яких передбачає 18 використання комп'ютерної техніки, з метою запобігання професійним захворюванням, нещасним випадкам та забезпечення комфортних і безпечних умов праці.

До основних загальних вимог 14 охорони праці при роботі з комп'ютером 14 належать:

14 Допуск до роботи. 14 До самостійної роботи з персональним комп'ютером допускаються особи, які досягли 18-річного віку, 9 пройшли медичний огляд відповідно до чинного законодавства, вступний інструктаж 4 з охорони праці, інструктаж 9 на робочому місці та навчання безпечним методам і прийомам роботи.

9 Дотримання правил внутрішнього розпорядку. Працівник 9 зобов'язаний дотримуватися правил внутрішнього трудового розпорядку підприємства (організації), режиму праці та відпочинку, встановленого 39 для даного виду робіт.

Використання обладнання за призначенням. Використовувати комп'ютерну техніку та периферійні пристрої слід суворо за їх прямим призначенням, відповідно до інструкцій з експлуатації.

Підтримання порядку 9 на робочому місці. Робоче 29 місце необхідно утримувати в чистоті та порядку, не захаращувати сторонніми предметами. Слідкувати за справністю обладнання та кабелів живлення.

Повідомлення про несправності. У разі виявлення будь-яких несправностей комп'ютерної техніки, пошкодження кабелів, розеток, виникнення 37 запаху гару або диму, 65 необхідно негайно припинити роботу, відключити обладнання від електромережі та повідомити безпосереднього керівника або відповідальну особу.

Заборона самостійного ремонту. Категорично забороняється самостійно розбирати та ремонтувати комп'ютерну техніку та інше електрообладнання. Ремонтні роботи мають виконувати лише кваліфіковані спеціалісти.

Дотримання особистої гігієни. 9 Перед початком роботи та після її завершення, а також після відвідування санвузла необхідно мити 37 руки з милом.

37 Заборона прийому їжі та напоїв. Не допускається прийом їжі та напоїв безпосередньо 9 на робочому місці біля 9 комп'ютерної техніки.

37 Дії 37 в аварійних ситуаціях. Працівник повинен знати порядок дій 26 у разі виникнення аварійних ситуацій (пожежа, 18 ураження електричним струмом тощо) та

вміти надавати першу домедичну допомогу потерпілим.

Дотримання цих загальних вимог є основою для створення безпечного робочого середовища та мінімізації ризиків 29 при роботі з комп'ютерною 29 технікою.

20 5.2 Вимоги до 20 організації робочого місця користувача ПК

Правильна організація робочого місця користувача персонального комп'ютера є ключовим фактором для збереження здоров'я, запобігання втомі та підвищення продуктивності праці. Неправильно облаштоване робоче місце 49 може призвести до розвитку захворювань опорно-рухового апарату, погіршення зору та загального дискомфорту. Основні вимоги до 20 організації робочого місця користувача ПК включають:

Площа та об'єм робочого місця. 26 Площа на одне робоче місце з ПК повинна становити не менше 6,0 26 $м^2$, а об'єм – не менше 20,0 $м^3$. Робочі місця слід розташовувати таким чином, щоб забезпечити вільний доступ та достатній простір для руху.

7 Робочий стіл. Конструкція робочого столу повинна забезпечувати оптимальне 20 розміщення обладнання (монітор, системний блок, клавіатура, миша) та документів. 30 Висота робочої поверхні столу має бути в межах 680-800 мм, регульованою або підібраною під зріст користувача. Під столом має бути достатньо 7 простору для ніг: не менше 600 мм у висоту, 500 мм у ширину та 650 мм у глибину на рівні колін. Поверхня столу повинна мати матове покриття для уникнення відблисків.

Робоче крісло (стілець). Крісло 7 має бути підйомно-поворотним, регульованим за висотою сидіння та висотою і кутом нахилу 20 спинки, а також відстанню спинки від 7 переднього краю сидіння. Конструкція крісла повинна забезпечувати підтримку раціональної робочої 20 пози та 30 дозволяти змінювати позу для 7 зниження статичного напруження м'язів. За бажанням 33 крісло може бути оснащене підлокітниками та підголівником. Сидіння та спинка мають бути покриті неслизьким, повітропроникним матеріалом.

Розміщення монітора. Екран монітора слід розташовувати на відстані 600-700 мм 7 від очей користувача, але не ближче 500 мм. Верхній край екрана має знаходитись на рівні очей або трохи нижче. Монітор слід встановлювати так, щоб уникнути відблисків від джерел світла на екрані.

Розміщення клавіатури та миші. 43 Клавіатуру слід розташовувати на поверхні столу на відстані 100-300 30 мм від краю, що звернутий до користувача. Має бути достатньо простору для опори передпліч. Мишу слід розміщувати поряд з клавіатурою, на одному

рівні з нею, у зоні легкого доступу.

Підставка для документів (за потреби). Якщо робота вимагає частого звернення до паперових документів, рекомендується використовувати підставку для документів, розміщену 21 в одній площині та на одній висоті з екраном.

21 Дотримання цих вимог дозволяє створити ергономічне робоче місце, що сприяє збереженню здоров'я та 33 забезпечує комфортні умови для тривалої роботи за комп'ютером.

33 5.3 Основні небезпечні фактори 30 під 4 час 1 роботи на ПК

1 Тривала робота за персональним комп'ютером може супроводжуватися впливом низки шкідливих та небезпечних факторів виробничого середовища, які здатні негативно впливати на здоров'я та працездатність користувача. Розуміння цих факторів є важливим для впровадження ефективних заходів профілактики. До основних таких факторів належать:

Підвищене зорове навантаження. Виникає через необхідність тривалого фокусування погляду 21 на екрані монітора, сприйняття текстової та графічної інформації. Характеристики зображення (яскравість, контрастність, мерехтіння), 21 відблиски на екрані та неправильна відстань до монітора можуть призводити до втоми очей, сухості, почервоніння, зниження гостроти зору.

Статичне навантаження та вимушена робоча поза. Довготривале перебування у фіксованому сидячому положенні спричиняє статичне напруження м'язів спини, шиї, плечового пояса та рук. Це може призвести до болю, дискомфорту, порушень постави та розвитку захворювань опорно-рухового апарату (остеохондроз, сколіоз тощо).

Гіподинамія. Малорухливий характер роботи за комп'ютером 44 призводить до зниження загальної фізичної активності, що негативно впливає 56 на серцево-судинну, дихальну та інші системи організму.

56 Психоемоційне напруження. Робота розробника часто пов'язана з високим рівнем відповідальності, 66 необхідністю обробки великих обсягів інформації, монотонністю деяких операцій, можливими збоями в роботі програмного забезпечення або обладнання, що може призводити до стресу, нервового перенапруження та втоми.

Електромагнітне випромінювання. Хоча сучасні монітори та системні блоки мають значно нижчі рівні випромінювання порівняно зі старішими моделями, тривалий вплив електромагнітних полів низької частоти від роботи комп'ютерної техніки все ще розглядається як потенційно шкідливий фактор.

Несприятливі параметри мікроклімату. Невідповідність температури, вологості та швидкості руху повітря на 32 робочому місці встановленим санітарним нормам може спричиняти дискомфорт та зниження працездатності. Робота комп'ютерної техніки також може призводити до підвищення температури та зниження вологості повітря.

Нераціональне освітлення. Недостатня освітленість робочої зони, наявність прямих та відбитих відблисків на екрані та робочих поверхнях від джерел світла створюють додаткове 32 навантаження на зір.

32 Підвищений рівень шуму. Шум від роботи системних блоків, принтерів, вентиляції може перевищувати допустимі рівні, викликаючи роздратування та підвищену втомлюваність.

32 Небезпека 18 ураження електричним струмом. Існує 18 ризик 18 ураження електричним струмом у разі використання несправного обладнання, пошкодженої ізоляції кабелів живлення, відсутності заземлення або порушення правил експлуатації електроустановок.

Усвідомлення цих факторів та їх потенційного впливу є першим кроком до організації 53 безпечних умов праці для користувачів ПК.

5.4 Рекомендації щодо режиму праці та відпочинку

Для запобігання розвитку втоми, перенапруження та професійних захворювань при роботі 53 з персональним комп'ютером велике значення має раціональна організація режиму праці та відпочинку. Неперервна робота за екраном монітора призводить до значного навантаження на зоровий аналізатор, опорно-руховий апарат та нервову систему, тому важливо дотримуватись регламентованих перерв та виконувати профілактичні заходи.

Основні рекомендації щодо режиму праці та відпочинку:

Регламентовані перерви. Встановлення регулярних перерв є обов'язковим. Рекомендується влаштовувати короткі перерви тривалістю 5-10 хвилин через кожну годину роботи за комп'ютером або 15-хвилинні перерви через кожні дві години роботи. Під час обідньої перерви слід уникати роботи за ПК.

Зміна виду діяльності. Під час регламентованих перерв необхідно змінювати вид діяльності. Рекомендується встати з робочого місця, пройтися, виконати легкі фізичні вправи. Не слід використовувати перерви для роботи з мобільними пристроями або читання з екрана.

Вправи для очей. Для зняття зорової втоми під час коротких перерв або навіть під час

роботи (кожні 20-30 хвилин) рекомендується виконувати спеціальні вправи для очей: переведення погляду на віддалені предмети за вікном (на 10-15 секунд); часте моргання протягом 10-15 секунд для зволоження очей; кругові рухи очима за та проти годинникової стрілки; легкий масаж закритих повік.

Фізичні вправи (виробнича гімнастика): під час довших перерв (15 хвилин) доцільно виконувати комплекс вправ для розвантаження м'язів спини, шиї, плечового пояса та рук; нахили та повороти голови; обертання плечима; потягування; вправи для кистей рук (стискання-розтискання кулаків, обертання зап'ястями); нахили тулуба.

Організація робочого процесу. За можливості слід чергувати роботу за комп'ютером з іншими видами діяльності, що не потребують постійного зорового контакту з екраном (наприклад, робота з документами, обговорення робочих питань).

Дотримання раціонального режиму праці та відпочинку, регулярне виконання профілактичних вправ дозволяє суттєво знизити негативний вплив шкідливих факторів, зберегти високу працездатність та запобігти проблемам зі здоров'ям у довгостроковій перспективі.

5.5 Пожежна безпека та електробезпека в приміщенні розробника

Забезпечення належного рівня пожежної та електробезпеки є критично важливим аспектом охорони праці в будь-якому робочому приміщенні, особливо там, де активно використовується комп'ютерна техніка та інше електрообладнання. Для приміщення розробника ці вимоги набувають особливого значення з огляду на концентрацію електронних пристроїв та потенційні ризики, пов'язані з їх експлуатацією.

Пожежна безпека в приміщенні розробника передбачає комплекс заходів, спрямованих на запобігання виникненню пожеж, обмеження їх поширення та створення умов для безпечної евакуації людей у разі загоряння. До основних вимог належать:

Дотримання правил експлуатації електрообладнання. Забороняється використовувати пошкоджені електроприлади, саморобні подовжувачі, залишати без нагляду ввімкнені пристрої, що нагріваються.

Правильне зберігання легкозаймистих матеріалів. Папір, документація та інші горючі матеріали повинні зберігатися у спеціально відведених місцях, подалі від джерел тепла та електроприладів.

Забезпечення вільного доступу до первинних засобів пожежогасіння. Вогнегасники повинні бути розташовані у легкодоступних місцях, їх кількість та тип мають відповідати нормам для даного типу приміщень. Всі працівники повинні знати

місцезнаходження вогнегасників та вміти ними користуватися.

Приміщення площею до 20 м², у яких розміщено комп'ютерну та офісну техніку, відповідно до норм повинні бути оснащені переносним газовим вогнегасником типу ВВК-2

Розробка та вивішування плану евакуації. На видному місці має бути розміщений план евакуації на випадок пожежі із зазначенням шляхів евакуації, евакуаційних виходів та місць розташування засобів пожежогасіння.

Проведення інструктажів 27 з пожежної безпеки. Всі 27 працівники повинні проходити регулярні 27 інструктажі щодо правил пожежної безпеки та дій у разі виникнення пожежі.

Утримання евакуаційних шляхів вільними. Проходи, коридори, сходові клітки не повинні захаращуватися меблями, обладнанням чи іншими предметами.

Електробезпека в приміщенні розробника спрямована на запобігання ураженню електричним струмом та забезпечення надійної роботи електромережі та електрообладнання. Основні вимоги включають:

Використання справного та сертифікованого електрообладнання. Все обладнання, що підключається до електромережі, повинно відповідати стандартам безпеки та не мати видимих пошкоджень ізоляції, шнурів чи вилок.

Належне заземлення обладнання. Корпуси комп'ютерів, моніторів та іншого електрообладнання повинні бути надійно заземлені для уникнення ураження електричним струмом у разі виникнення несправності.

Уникнення перевантаження електромережі. Не слід підключати до однієї розетки надмірну кількість електроприладів, особливо через трійники чи подовжувачі без відповідного захисту від перевантаження.

Дотримання правил підключення та відключення електроприладів. Вмикати та вимикати прилади слід за допомогою передбачених для цього вимикачів або вилок, не витягуючи шнур з розетки ривком.

Заборона самостійного ремонту електрообладнання чи електропроводки. Усі роботи з ремонту або модифікації електрообладнання та електропроводки повинні виконуватися виключно кваліфікованими фахівцями.

Регулярна перевірка стану електропроводки та електрообладнання. Необхідно проводити періодичні огляди електромережі, розеток, вимикачів та електроприладів

на наявність пошкоджень чи ознак перегріву.

Дотримання вищезазначених вимог пожежної та електробезпеки є запорукою створення безпечних умов праці для розробників, мінімізації ризиків виникнення надзвичайних ситуацій та збереження майна.

ВИСНОВКИ

У результаті виконання дипломного проєкту було реалізовано повнофункціональний Android-додаток «Хрестики-нулики» з графічним інтерфейсом, який підтримує гру з ботом і між двома гравцями. Основною інновацією проєкту стала інтеграція алгоритму Minimax, що дозволяє забезпечити оптимальні стратегії для комп'ютерного супротивника.

Результати роботи включають:

створення Android-застосунку з використанням мови програмування Kotlin і XML;

реалізацію алгоритму Minimax для гри з ботом з трьома рівнями складності;

побудову архітектури з однією активністю і кількома фрагментами;

реалізацію адаптивного графічного інтерфейсу з урахуванням користувацького досвіду;

тестування додатку на реальному Android-пристрої.

Практичне використання результатів. Розроблений додаток може бути використаний як навчальний приклад для ознайомлення з теорією ігор, розробкою Android-додатків, реалізацією AI-алгоритмів у мобільних іграх.

Техніко-економічна ефективність. Реалізація гри не потребує фінансових витрат, окрім часу на розробку, а застосування алгоритму Minimax підвищує інтелектуальну цінність гри без збільшення вимог до ресурсів пристрою. Проєкт не вимагає додаткового обладнання, а весь процес розробки здійснювався з використанням відкритого ПЗ.

Розроблене програмне забезпечення може слугувати основою для розширення функціоналу або комерційного використання після подальшого доопрацювання.

ПЕРЕЛІК ПОСИЛАНЬ

Останкова Л. А. Теорія ігор [Текст] : довідковий метод. посіб. для студ. екон. спец. / Л. А. Останкова, Н. Ю. Шевченко. – Донецьк : ДДМА, 2007. – 103 с

https://kotlinlang.org/docs/basic-syntax.html#conditional-expressions

https://developer.android.com/develop/ui/views/layout/declaring-layout

https://iconscout.com/icons/avatar?price=free

https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/

https://www.youtube.com/watch?v=3QyxdM0g_IM

https://clouddevs.com/kotlin/android-user-interfaces/

https://en.wikipedia.org/wiki/Minimax

https://en.wikipedia.org/wiki/Strategy_(game_theory)

https://kotlinlang.org/docs/coroutines-basics.html#extract-function-refactoring

https://medium.com/google-developer-experts/exploring-kotlin-initialization-with-android-custom-views-cde06e915e8d

https://developer.android.com/develop/ui/views/layout/custom-views/custom-components

https://bumptech.github.io/glide/

https://developer.android.com/guide/fragments/saving-state

https://github.com/JetradarMobile/android-snowfall

https://kotlinlang.org/docs/serialization.html#serialize-and-deserialize-json

https://m2.material.io/components/snackbars/android#using-snackbars

ДОДАТОК А

Код для реалізації графічного інтерфейсу

Файл res/layout/activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"


xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"
```

```
android:id="@+id/main"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="@drawable/bg_nightfall"
android:orientation="vertical"
tools:context=".MainActivity">
<com.google.android.material.appbar.MaterialToolbar
android:id="@+id/toolbar"
android:layout_width="0dp"
android:layout_height="?android:attr/actionBarSize"
android:background="@color/toolbar_background"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:navigationIconTint="@color/white"
app:title="@string/toolbar_menu"
app:titleTextColor="@color/white"
tools:title="Screen Title" />
<androidx.fragment.app.FragmentContainerView
android:id="@+id/fragmentContainer"
android:layout_width="0dp"
android:layout_height="0dp"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
```

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@id/toolbar"

tools:layout="@layout/fragment_menu" />

</androidx.constraintlayout.widget.ConstraintLayout>

Файл res/layout/snowfall_background.xml:

<?xml version="1.0" encoding="utf-8"?>

<merge xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

android:layout_width="match_parent"

android:layout_height="match_parent">

<com.jetradarmobile.snowfall.SnowfallView

android:layout_width="match_parent"

android:layout_height="match_parent"

android:background="@drawable/bg_nightfall"

app:snowflakeAlphaMax="255"

app:snowflakeAlphaMin="180"

app:snowflakeAngleMax="4"

app:snowflakeSizeMax="8dp"

app:snowflakeSizeMin="1dp"

app:snowflakeSpeedMax="5"

app:snowflakeSpeedMin="1"

app:snowflakesAlreadyFalling="true"

app:snowflakesFadingEnabled="true" />

</merge>

Файл res/layout/fragment_menu.xml:

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:id="@+id/main"

android:layout_width="match_parent"

android:layout_height="match_parent"

android:orientation="vertical"

tools:context=".MainActivity">

<include layout="@layout/snowfall_background" />

<TextView

android:id="@+id/tv_game_title"

android:layout_width="0dp"

android:layout_height="0dp"

android:layout_centerInParent="true"

android:fontFamily="@font/main_title"

android:gravity="center"

android:lines="3"

android:maxLines="3"

android:shadowColor="@color/text_shadow"

android:shadowDx="-30"

android:shadowDy="0"
```

```
android:shadowRadius="10"

android:text="@string/game_title"

android:textSize="110sp"

android:textStyle="bold"

app:layout_constraintBottom_toTopOf="@id/bt_single_player"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintHeight_percent="0.55"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toTopOf="parent" />

<Button

android:id="@+id/bt_single_player"

style="@style/main_menu_button"

android:layout_marginTop="35dp"

android:background="@drawable/bt_glow_magenta"

android:text="@string/single_player"

app:layout_constraintTop_toBottomOf="@id/tv_game_title" />

<Button

android:id="@+id/bt_multiplayer"

style="@style/main_menu_button"

android:layout_marginTop="20dp"

android:background="@drawable/bt_sunset_glow"

android:text="@string/multiplayer"

app:layout_constraintTop_toBottomOf="@id/bt_single_player" />

<Button
```

android:id="@+id/bt_local_multiplayer"

style="@style/main_menu_button"

android:layout_marginTop="20dp"

android:background="@drawable/bt_ocean_sunrise"

android:text="@string/local_multiplayer"

app:layout_constraintTop_toBottomOf="@id/bt_multiplayer" />

</androidx.constraintlayout.widget.ConstraintLayout>

Файл res/layout/fragment_difficulty.xml:

<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout_width="match_parent"

android:layout_height="match_parent">

<include layout="@layout/snowfall_background" />

<View

android:layout_width="0dp"

android:layout_height="8dp"

android:background="@drawable/v_line"

app:layout_constraintStart_toStartOf="@id/tv_select_difficulty"

app:layout_constraintTop_toBottomOf="@id/tv_select_difficulty"

app:layout_constraintWidth_percent="0.55" />

<TextView

```
android:id="@+id/tv_select_difficulty"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_marginTop="75dp"

android:lines="1"

android:text="@string/select_difficulty"

android:textColor="@color/white"

android:textSize="@dimen/difficulty_tv"

android:textStyle="bold"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toTopOf="parent" />

<Button

android:id="@+id/bt_easy"

style="@style/bt_difficulty"

android:background="@drawable/bt_easy_bot"

android:text="@string/easy_bot"

app:layout_constraintBottom_toTopOf="@id/bt_medium"

app:layout_constraintTop_toBottomOf="@id/tv_select_difficulty"

tools:ignore="MissingConstraints" />

<ImageView

style="@style/ic_difficulty"

android:contentDescription="@string/easy_difficulty_level_image"

android:src="@drawable/easy_bot"
```

```
app:layout_constraintBottom_toBottomOf="@id/bt_easy"

app:layout_constraintEnd_toStartOf="@id/bt_easy"

app:layout_constraintTop_toTopOf="@id/bt_easy" />

<Button

android:id="@+id/bt_medium"

style="@style/bt_difficulty"

android:background="@drawable/bt_medium_bot"

android:text="@string/medium_bot"

app:layout_constraintBottom_toBottomOf="parent"

app:layout_constraintTop_toBottomOf="@id/tv_select_difficulty"

tools:ignore="MissingConstraints" />

<ImageView

style="@style/ic_difficulty"

android:contentDescription="@string/medium_difficulty_level_image"

android:src="@drawable/medium_bot"

app:layout_constraintBottom_toBottomOf="@id/bt_medium"

app:layout_constraintEnd_toStartOf="@id/bt_medium"

app:layout_constraintTop_toTopOf="@id/bt_medium" />

<Button

android:id="@+id/bt_difficult"

style="@style/bt_difficulty"

android:background="@drawable/bt_difficult_bot"

android:text="@string/difficult_bot"

app:layout_constraintBottom_toBottomOf="parent"
```

app:layout_constraintTop_toBottomOf="@id/bt_medium"

tools:ignore="MissingConstraints" />

<ImageView

style="@style/ic_difficulty"

android:contentDescription="@string/difficult_level_image"

android:src="@drawable/difficult_bot"

app:layout_constraintBottom_toBottomOf="@id/bt_difficult"

app:layout_constraintEnd_toStartOf="@id/bt_difficult"

app:layout_constraintTop_toTopOf="@id/bt_difficult" />

</androidx.constraintlayout.widget.ConstraintLayout>

Файл res/layout/fragment_local_multiplayer.xml:

<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

android:layout_width="match_parent"

android:layout_height="match_parent">

<include layout="@layout/snowfall_background" />

<androidx.constraintlayout.widget.Guideline

android:id="@+id/begin_guide_line"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:orientation="vertical"

app:layout_constraintGuide_begin="32dp" />

```
<androidx.constraintlayout.widget.Guideline

android:id="@+id/end_guide_line"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:orientation="vertical"

app:layout_constraintGuide_end="32dp" />

<com.example.tictactoe.customViews.BoardView

android:id="@+id/cv_board"

android:layout_width="0dp"

android:layout_height="0dp"

android:layout_marginTop="160dp"

android:padding="10dp"

app:layout_constraintBottom_toBottomOf="parent"

app:layout_constraintDimensionRatio="1:1"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toTopOf="parent" />

<com.example.tictactoe.customViews.EntityCardView


android:id="@+id/human_entity_card"

android:layout_width="150dp"

android:layout_height="0dp"

app:description="@string/you_turn"

app:layout_constraintBottom_toTopOf="@id/cv_board"
```

app:layout_constraintStart_toStartOf="@id/begin_guide_line"

app:layout_constraintTop_toTopOf="parent" />

<com.example.tictactoe.customViews.EntityCardView

android:id="@+id/opponent_entity_card"

android:layout_width="150dp"

android:layout_height="0dp"

app:description="@string/opponent_turn"

app:layout_constraintBottom_toTopOf="@id/cv_board"

app:layout_constraintEnd_toStartOf="@id/end_guide_line"

app:layout_constraintTop_toTopOf="parent"

app:mark="@drawable/tac"

app:name="Opponent" />

</androidx.constraintlayout.widget.ConstraintLayout>

Файл res/layout/fragment_edit_profile.xml:

<?xml version="1.0" encoding="utf-8"?><androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android" xmlns:app="http://schemas.android.com/apk/res-auto" xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent" android:layout_height="match_parent" android:background="@color/profile_background"> <LinearLayout android:layout_width="wrap_content" android:layout_height="wrap_content" android:orientation="vertical" app:layout_constraintBottom_toTopOf="@id/bt_save_changes" app:layout_constraintEnd_toEndOf="parent" app:layout_constraintStart_toStartOf="parent" app:layout_constraintTop_toBottomOf="@id/et_nickname"> <LinearLayout android:layout_width="match_parent" android:layout_height="wrap_content" android:layout_marginBottom="15dp" android:gravity="center"> <ImageView android:id="@+id/iv_red" android:layout_width="80dp" android:layout_height="80dp" android:layout_marginEnd="15dp" android:background="@drawable/v_color" android:backgroundTint="@color/background_red" android:contentDescription="@null" /> <ImageView android:id="@+id/iv_teal" android:layout_width="80dp" android:layout_height="80dp" android:layout_marginEnd="15dp"

```
android:background="@drawable/v_color"
android:backgroundTint="@color/background_teal" android:contentDescription="@null" />
<ImageView android:id="@+id/iv_blue" android:layout_width="80dp"
android:layout_height="80dp" android:layout_marginEnd="15dp"
android:background="@drawable/v_color"
android:backgroundTint="@color/background_blue" android:contentDescription="@null" />
<ImageView android:id="@+id/iv_green" android:layout_width="80dp"
android:layout_height="80dp" android:background="@drawable/v_color"
android:backgroundTint="@color/background_green" android:contentDescription="@null" />
</LinearLayout> <LinearLayout android:layout_width="match_parent"
android:layout_height="wrap_content" android:gravity="center"
app:layout_constraintBottom_toTopOf="@id/bt_save_changes"
app:layout_constraintEnd_toEndOf="parent" app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@id/et_nickname"> <ImageView
android:id="@+id/iv_yellow" android:layout_width="80dp" android:layout_height="80dp"
android:layout_marginEnd="15dp" android:background="@drawable/v_color"
android:backgroundTint="@color/background_yellow" android:contentDescription="@null"
/> <ImageView android:id="@+id/iv_pink" android:layout_width="80dp"
android:layout_height="80dp" android:layout_marginEnd="15dp"
android:background="@drawable/v_color"
android:backgroundTint="@color/background_pink" android:contentDescription="@null" />
<ImageView android:id="@+id/iv_purple" android:layout_width="80dp"
android:layout_height="80dp" android:layout_marginEnd="15dp"
android:background="@drawable/v_color"
android:backgroundTint="@color/background_purple" android:contentDescription="@null"
/> <ImageView android:id="@+id/iv_orange" android:layout_width="80dp"
android:layout_height="80dp" android:background="@drawable/v_color"
android:backgroundTint="@color/background_orange" android:contentDescription="@null"
/>

</LinearLayout> </LinearLayout> <ImageView android:id="@+id/iv_human_avatar"
android:layout_width="130dp" android:layout_height="130dp"
android:layout_marginTop="48dp" android:contentDescription="@string/player_avatar"
app:layout_constraintEnd_toEndOf="parent" app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" tools:background="@drawable/profile_avatar"
tools:tint="@color/background_red" /> <TextView android:id="@+id/tv_avatar_letter"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:saveEnabled="true" android:text="" android:textColor="@color/white"
android:textSize="40sp" android:textStyle="bold"
app:layout_constraintBottom_toBottomOf="@id/iv_human_avatar"
app:layout_constraintEnd_toEndOf="@id/iv_human_avatar"
```

app:layout_constraintStart_toStartOf="@id/iv_human_avatar"
app:layout_constraintTop_toTopOf="@id/iv_human_avatar" tools:text="P" /> <TextView
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:layout_marginStart="20dp" android:layout_marginBottom="10dp"
android:text="@string/nickname" android:textColor="@color/white" android:textSize="18sp"
android:textStyle="bold" app:layout_constraintBottom_toTopOf="@id/et_nickname"
app:layout_constraintStart_toStartOf="@id/et_nickname" /> <TextView
android:id="@+id/tv_nickname_size" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_marginTop="10dp"
android:layout_marginEnd="20dp" android:text="@string/char_count_tv"
android:textColor="#a5b4fc" android:textSize="14sp" android:textStyle="bold"
app:layout_constraintEnd_toEndOf="@id/et_nickname"
app:layout_constraintTop_toBottomOf="@id/et_nickname" /> <EditText
android:id="@+id/et_nickname" android:layout_width="match_parent"
android:layout_height="50dp" android:layout_marginStart="15dp"
android:layout_marginTop="100dp" android:layout_marginEnd="15dp"
android:autofillHints="username" android:background="@drawable/et_profile"
android:hint="@string/enter_your_nickname" android:imeOptions="actionDone"
android:inputType="text" android:selectAllOnFocus="true" android:singleLine="true"
android:textColor="@color/white" android:textSize="24sp"
app:layout_constraintEnd_toEndOf="parent" app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@id/iv_human_avatar" tools:text="Player" /> <Button
android:id="@+id/bt_save_changes" android:layout_width="match_parent"
android:layout_height="wrap_content" android:layout_marginStart="15dp"
android:layout_marginEnd="15dp" android:layout_marginBottom="50dp"
android:background="@drawable/bt_gradient_save_changes"
android:text="@string/save_changes" android:textColor="@color/white"
android:textStyle="bold" app:backgroundTint="@null"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent" app:layout_constraintStart_toStartOf="parent"
/> <Button android:id="@+id/bt_select_photo" android:layout_width="50dp"
android:layout_height="50dp" android:background="@drawable/ic_photo_picker"
android:contentDescription="@string/select_photo" app:backgroundTint="@null"
app:layout_constraintBottom_toBottomOf="@id/iv_human_avatar"
app:layout_constraintEnd_toEndOf="@id/iv_human_avatar"
/></androidx.constraintlayout.widget.ConstraintLayout>

Файл res/layout/fragment_settings.xml:

<?xml version="1.0" encoding="utf-8"?><androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
android:layout_height="match_parent" android:background="@color/dark_violet_blue"
tools:ignore="MissingConstraints"> <androidx.constraintlayout.widget.Guideline
android:id="@+id/begin_guide_line" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:orientation="vertical"
app:layout_constraintGuide_begin="16dp" /> <androidx.constraintlayout.widget.Guideline
android:id="@+id/end_guide_line" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:orientation="vertical"
app:layout_constraintGuide_end="16dp" /> <TextView style="@style/tv_settings"
android:id="@+id/tv_who_goes_first" android:text="@string/who_goes_first"
app:layout_constraintStart_toEndOf="@id/begin_guide_line"
app:layout_constraintTop_toTopOf="parent" /> <View style="@style/item_settings"
android:id="@+id/v_item_human"
app:layout_constraintTop_toBottomOf="@id/tv_who_goes_first" /> <View
style="@style/ic_bg_settings" android:id="@+id/v_human_avatar_bg"
android:background="@drawable/v_line"
app:layout_constraintBottom_toBottomOf="@id/v_item_human"
app:layout_constraintStart_toStartOf="@id/v_item_human"
app:layout_constraintTop_toTopOf="@id/v_item_human" /> <ImageView
style="@style/ic_settings" android:id="@+id/iv_human_avatar"
android:contentDescription="@null"
app:layout_constraintBottom_toBottomOf="@id/v_human_avatar_bg"
app:layout_constraintEnd_toEndOf="@id/v_human_avatar_bg"
app:layout_constraintStart_toStartOf="@id/v_human_avatar_bg"
app:layout_constraintTop_toTopOf="@id/v_human_avatar_bg"
tools:src="@drawable/profile_avatar" /> <TextView style="@style/item_tv_settings"
android:text="@string/you" app:layout_constraintBottom_toBottomOf="@id/v_item_human"
app:layout_constraintStart_toEndOf="@id/v_human_avatar_bg"
app:layout_constraintTop_toTopOf="@id/v_item_human" /> <View
style="@style/item_settings" android:id="@+id/v_item_computer"
app:layout_constraintTop_toBottomOf="@id/v_item_human" /> <View
style="@style/ic_bg_settings" android:id="@+id/v_computer_bg"
android:background="@drawable/v_line"
app:layout_constraintBottom_toBottomOf="@id/v_item_computer"
app:layout_constraintStart_toStartOf="@id/v_item_computer"
app:layout_constraintTop_toTopOf="@id/v_item_computer" /> <ImageView
style="@style/ic_settings" android:contentDescription="@null"
android:src="@drawable/ic_robot"

app:layout_constraintBottom_toBottomOf="@id/v_computer_bg"
app:layout_constraintEnd_toEndOf="@id/v_computer_bg"
app:layout_constraintStart_toStartOf="@id/v_computer_bg"
app:layout_constraintTop_toTopOf="@id/v_computer_bg" /> <TextView
style="@style/item_tv_settings" android:text="@string/computer"
app:layout_constraintBottom_toBottomOf="@id/v_item_computer"
app:layout_constraintStart_toEndOf="@id/v_computer_bg"
app:layout_constraintTop_toTopOf="@id/v_item_computer" /> <View
style="@style/item_settings" android:id="@+id/v_item_random_move"
app:layout_constraintTop_toBottomOf="@id/v_item_computer" /> <View
style="@style/ic_bg_settings" android:id="@+id/v_rand_move_bg"
android:background="@drawable/v_line"
app:layout_constraintBottom_toBottomOf="@id/v_item_random_move"
app:layout_constraintStart_toStartOf="@id/v_item_random_move"
app:layout_constraintTop_toTopOf="@id/v_item_random_move" /> <ImageView
style="@style/ic_settings" android:contentDescription="@null"
android:src="@drawable/ic_random"
app:layout_constraintBottom_toBottomOf="@id/v_rand_move_bg"
app:layout_constraintEnd_toEndOf="@id/v_rand_move_bg"
app:layout_constraintStart_toStartOf="@id/v_rand_move_bg"
app:layout_constraintTop_toTopOf="@id/v_rand_move_bg" /> <TextView
style="@style/item_tv_settings" android:text="@string/random"
app:layout_constraintBottom_toBottomOf="@id/v_item_random_move"
app:layout_constraintStart_toEndOf="@id/v_rand_move_bg"
app:layout_constraintTop_toTopOf="@id/v_item_random_move" /> <TextView
style="@style/tv_settings" android:id="@+id/tv_your_symbol"
android:text="@string/your_symbol"
app:layout_constraintStart_toEndOf="@id/begin_guide_line"
app:layout_constraintTop_toBottomOf="@id/v_item_random_move" /> <View
style="@style/item_settings" android:id="@+id/v_item_tac"
app:layout_constraintTop_toBottomOf="@id/tv_your_symbol" /> <View
style="@style/ic_bg_settings" android:id="@+id/v_tac_bg"
android:background="@drawable/v_line"
android:backgroundTint="@color/raspberry_purple"
app:layout_constraintBottom_toBottomOf="@id/v_item_tac"
app:layout_constraintStart_toStartOf="@id/v_item_tac"
app:layout_constraintTop_toTopOf="@id/v_item_tac" /> <ImageView
style="@style/ic_settings" android:contentDescription="@null" android:src="@drawable/tac"
app:layout_constraintBottom_toBottomOf="@id/v_tac_bg"
app:layout_constraintEnd_toEndOf="@id/v_tac_bg"

app:layout_constraintStart_toStartOf="@id/v_tac_bg"
app:layout_constraintTop_toTopOf="@id/v_tac_bg" /> <TextView
style="@style/item_tv_settings" android:text="@string/play_as_x"
app:layout_constraintBottom_toBottomOf="@id/v_item_tac"
app:layout_constraintStart_toEndOf="@id/v_tac_bg"
app:layout_constraintTop_toTopOf="@id/v_item_tac" /> <View style="@style/item_settings"
android:id="@+id/v_item_tic" app:layout_constraintTop_toBottomOf="@id/v_item_tac" />
<View style="@style/ic_bg_settings" android:id="@+id/v_tic_bg"
android:background="@drawable/v_line" android:backgroundTint="@color/smoky_brown"
app:layout_constraintBottom_toBottomOf="@id/v_item_tic"
app:layout_constraintStart_toStartOf="@id/v_item_tic"
app:layout_constraintTop_toTopOf="@id/v_item_tic" /> <ImageView
style="@style/ic_settings" android:contentDescription="@null" android:src="@drawable/tic"
app:layout_constraintBottom_toBottomOf="@id/v_tic_bg"
app:layout_constraintEnd_toEndOf="@id/v_tic_bg"
app:layout_constraintStart_toStartOf="@id/v_tic_bg"
app:layout_constraintTop_toTopOf="@id/v_tic_bg" /> <TextView
style="@style/item_tv_settings" android:text="@string/play_as_o"
app:layout_constraintBottom_toBottomOf="@id/v_item_tic"
app:layout_constraintStart_toEndOf="@id/v_tic_bg"
app:layout_constraintTop_toTopOf="@id/v_item_tic" /> <View style="@style/item_settings"
android:id="@+id/v_item_random_mark"
app:layout_constraintTop_toBottomOf="@id/v_item_tic" /> <View
style="@style/ic_bg_settings" android:id="@+id/v_rand_mark_bg"
android:background="@drawable/v_line"
app:layout_constraintBottom_toBottomOf="@id/v_item_random_mark"
app:layout_constraintStart_toStartOf="@id/v_item_random_mark"
app:layout_constraintTop_toTopOf="@id/v_item_random_mark" /> <ImageView
style="@style/ic_settings" android:contentDescription="@null"
android:src="@drawable/ic_random"
app:layout_constraintBottom_toBottomOf="@id/v_rand_mark_bg"
app:layout_constraintEnd_toEndOf="@id/v_rand_mark_bg"
app:layout_constraintStart_toStartOf="@id/v_rand_mark_bg"
app:layout_constraintTop_toTopOf="@id/v_rand_mark_bg" /> <TextView
style="@style/item_tv_settings" android:text="@string/random"
app:layout_constraintBottom_toBottomOf="@id/v_item_random_mark"
app:layout_constraintStart_toEndOf="@id/v_rand_mark_bg"
app:layout_constraintTop_toTopOf="@id/v_item_random_mark" /> <Space
android:id="@+id/center" android:layout_width="16dp"
android:layout_height="wrap_content" app:layout_constraintBottom_toBottomOf="parent"

app:layout_constraintEnd_toStartOf="@id/end_guide_line"
app:layout_constraintStart_toEndOf="@id/begin_guide_line"
app:layout_constraintTop_toBottomOf="@id/v_item_random_mark" /> <Button
style="@style/bt_settings" android:id="@+id/bt_cancel"
android:backgroundTint="@color/profile_background" android:text="@string/cancel"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toStartOf="@id/center"
app:layout_constraintStart_toEndOf="@id/begin_guide_line"
app:layout_constraintTop_toBottomOf="@id/v_item_random_mark" /> <Button
style="@style/bt_settings" android:id="@+id/bt_save"
android:backgroundTint="@color/evergreen_depths" android:text="@string/save"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toStartOf="@id/end_guide_line"
app:layout_constraintStart_toEndOf="@id/center"
app:layout_constraintTop_toBottomOf="@id/v_item_random_mark"
/></androidx.constraintlayout.widget.ConstraintLayout>

Файл res/layout/fragment_bot_game.xml:

<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

android:layout_width="match_parent"

android:layout_height="match_parent">

<include layout="@layout/snowfall_background" />

<androidx.constraintlayout.widget.Guideline

android:id="@+id/begin_guide_line"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:orientation="vertical"

app:layout_constraintGuide_begin="32dp" />

```
<androidx.constraintlayout.widget.Guideline

android:id="@+id/end_guide_line"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:orientation="vertical"

app:layout_constraintGuide_end="32dp" />

<com.example.tictactoe.customViews.BoardView

android:id="@+id/cv_board"

android:layout_width="0dp"

android:layout_height="0dp"

android:layout_marginTop="160dp"

android:padding="10dp"

app:layout_constraintBottom_toBottomOf="parent"

app:layout_constraintDimensionRatio="1:1"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toTopOf="parent" />

<com.example.tictactoe.customViews.EntityCardView

android:id="@+id/human_entity_card"

android:layout_width="150dp"

android:layout_height="0dp"

app:description="@string/you_turn"

app:layout_constraintBottom_toTopOf="@id/cv_board"

app:layout_constraintStart_toStartOf="@id/begin_guide_line"
```

app:layout_constraintTop_toTopOf="parent" />

<com.example.tictactoe.customViews.EntityCardView

android:id="@+id/bot_entity_card"

android:layout_width="150dp"

android:layout_height="0dp"

app:description="@string/bot_turn"

app:layout_constraintBottom_toTopOf="@id/cv_board"

app:layout_constraintEnd_toStartOf="@id/end_guide_line"

app:layout_constraintTop_toTopOf="parent"

app:mark="@drawable/tac" />

</androidx.constraintlayout.widget.ConstraintLayout>

Файл res/layout/fragment_local_multiplayer.xml:

<?xml version="1.0" encoding="utf-8"?><androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android" xmlns:app="http://schemas.android.com/apk/res-auto" android:layout_width="match_parent" android:layout_height="match_parent"> <include layout="@layout/snowfall_background" /> <androidx.constraintlayout.widget.Guideline android:id="@+id/begin_guide_line" android:layout_width="wrap_content" android:layout_height="wrap_content" android:orientation="vertical" app:layout_constraintGuide_begin="32dp" /> <androidx.constraintlayout.widget.Guideline android:id="@+id/end_guide_line" android:layout_width="wrap_content" android:layout_height="wrap_content" android:orientation="vertical" app:layout_constraintGuide_end="32dp" /> <com.example.tictactoe.customViews.BoardView android:id="@+id/cv_board" android:layout_width="0dp" android:layout_height="0dp" android:layout_marginTop="160dp" android:padding="10dp" app:layout_constraintBottom_toBottomOf="parent" app:layout_constraintDimensionRatio="1:1" app:layout_constraintEnd_toEndOf="parent" app:layout_constraintStart_toStartOf="parent" app:layout_constraintTop_toTopOf="parent" /> <com.example.tictactoe.customViews.EntityCardView android:id="@+id/human_entity_card" android:layout_width="150dp" android:layout_height="0dp" app:description="@string/you_turn" app:layout_constraintBottom_toTopOf="@id/cv_board"

app:layout_constraintStart_toStartOf="@id/begin_guide_line"
app:layout_constraintTop_toTopOf="parent" />
<com.example.tictactoe.customViews.EntityCardView
android:id="@+id/opponent_entity_card" android:layout_width="150dp"
android:layout_height="0dp" app:description="@string/opponent_turn"
app:layout_constraintBottom_toTopOf="@id/cv_board"
app:layout_constraintEnd_toStartOf="@id/end_guide_line"
app:layout_constraintTop_toTopOf="parent" app:mark="@drawable/tac"
app:name="Opponent" /></androidx.constraintlayout.widget.ConstraintLayout>

Файл res/layout/fragment_game_over.xml:

<?xml version="1.0" encoding="utf-8"?><androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
android:layout_height="match_parent"> <include layout="@layout/snowfall_background" />
<androidx.constraintlayout.widget.Guideline android:id="@+id/begin_guide_line"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:orientation="vertical" app:layout_constraintGuide_begin="32dp" />
<androidx.constraintlayout.widget.Guideline android:id="@+id/end_guide_line"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:orientation="vertical" app:layout_constraintGuide_end="32dp" />
<com.example.tictactoe.customViews.ResultGameView android:id="@+id/result_game"
android:layout_width="0dp" android:layout_height="0dp"
android:layout_marginLeft="@dimen/result_game_margin"
android:layout_marginTop="160dp"
android:layout_marginRight="@dimen/result_game_margin"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintDimensionRatio="1:1" app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent" app:layout_constraintTop_toTopOf="parent"
/> <com.example.tictactoe.customViews.EntityCardView
android:id="@+id/human_entity_card" android:layout_width="150dp"
android:layout_height="0dp" app:description="@string/you_turn"
app:layout_constraintBottom_toTopOf="@id/result_game"
app:layout_constraintStart_toStartOf="@id/begin_guide_line"
app:layout_constraintTop_toTopOf="parent" tools:avatar="@drawable/medium_bot"
tools:name="Player" /> <com.example.tictactoe.customViews.EntityCardView
android:id="@+id/bot_entity_card" android:layout_width="150dp"
android:layout_height="0dp" app:description="@string/you_turn"
app:layout_constraintBottom_toTopOf="@id/result_game"

app:layout_constraintEnd_toStartOf="@id/end_guide_line"
app:layout_constraintTop_toTopOf="parent" tools:avatar="@drawable/easy_bot"
tools:mark="@drawable/tic" tools:name="bot" /> <ImageButton
android:id="@+id/bt_refresh" android:layout_width="@dimen/bt_refresh"
android:layout_height="@dimen/bt_refresh" android:background="@drawable/bt_red_glow"
android:contentDescription="@string/button_refresh"
android:padding="@dimen/bt_refresh_padding" android:scaleType="fitXY"
android:src="@drawable/ic_refresh"
app:layout_constraintBottom_toBottomOf="@id/result_game"
app:layout_constraintEnd_toEndOf="parent" app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@id/result_game"
/></androidx.constraintlayout.widget.ConstraintLayout>

Файл res/layout/board.xml:

<?xml version="1.0" encoding="utf-8"?><merge
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
android:layout_height="match_parent"
tools:parentTag="androidx.constraintlayout.widget.ConstraintLayout"> <View
android:id="@+id/v_board" android:layout_width="0dp" android:layout_height="0dp"
android:background="@drawable/v_board"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintDimensionRatio="1:1" app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent" app:layout_constraintTop_toTopOf="parent"
/> <ImageView android:id="@+id/cell_2_2" android:layout_width="0dp"
android:layout_height="0dp" android:contentDescription="@null"
android:src="@drawable/cell_empty"
app:layout_constraintBottom_toBottomOf="@id/v_board"
app:layout_constraintDimensionRatio="1:1"
app:layout_constraintEnd_toEndOf="@id/v_board"
app:layout_constraintStart_toStartOf="@id/v_board"
app:layout_constraintTop_toTopOf="@id/v_board"
app:layout_constraintWidth_percent="@dimen/board_cell" /> <ImageView
android:id="@+id/cell_1_1" android:layout_width="0dp" android:layout_height="0dp"
android:contentDescription="@null" android:src="@drawable/cell_empty"
app:layout_constraintBottom_toTopOf="@id/cell_2_1"
app:layout_constraintDimensionRatio="1:1"
app:layout_constraintEnd_toStartOf="@id/cell_1_2"

app:layout_constraintStart_toStartOf="@id/v_board"
app:layout_constraintTop_toTopOf="@+id/v_board"
app:layout_constraintWidth_percent="@dimen/board_cell" /> <ImageView
android:id="@+id/cell_1_2" android:layout_width="0dp" android:layout_height="0dp"
android:contentDescription="@null" android:src="@drawable/cell_empty"
app:layout_constraintBottom_toTopOf="@id/cell_2_2"
app:layout_constraintDimensionRatio="1:1" app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="@id/v_board"
app:layout_constraintWidth_percent="@dimen/board_cell" /> <ImageView
android:id="@+id/cell_1_3" android:layout_width="0dp" android:layout_height="0dp"
android:contentDescription="@null" android:src="@drawable/cell_empty"
app:layout_constraintBottom_toTopOf="@id/cell_2_3"
app:layout_constraintDimensionRatio="1:1"
app:layout_constraintEnd_toEndOf="@id/v_board"
app:layout_constraintStart_toEndOf="@id/cell_1_2"
app:layout_constraintTop_toTopOf="@id/v_board"
app:layout_constraintWidth_percent="@dimen/board_cell" /> <ImageView
android:id="@+id/cell_2_1" android:layout_width="0dp" android:layout_height="0dp"
android:contentDescription="@null" android:src="@drawable/cell_empty"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintDimensionRatio="1:1"
app:layout_constraintEnd_toStartOf="@id/cell_2_2"
app:layout_constraintStart_toStartOf="@id/v_board"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintWidth_percent="@dimen/board_cell" /> <ImageView
android:id="@+id/cell_2_3" android:layout_width="0dp" android:layout_height="0dp"
android:contentDescription="@null" android:src="@drawable/cell_empty"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintDimensionRatio="1:1"
app:layout_constraintEnd_toEndOf="@id/v_board"
app:layout_constraintStart_toEndOf="@id/cell_2_2"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintWidth_percent="@dimen/board_cell" /> <ImageView
android:id="@+id/cell_3_1" android:layout_width="0dp" android:layout_height="0dp"
android:contentDescription="@null" android:src="@drawable/cell_empty"
app:layout_constraintBottom_toBottomOf="@id/v_board"
app:layout_constraintDimensionRatio="1:1"
app:layout_constraintEnd_toStartOf="@id/cell_3_2"
app:layout_constraintStart_toStartOf="@id/v_board"

app:layout_constraintTop_toBottomOf="@id/cell_2_1"
app:layout_constraintWidth_percent="@dimen/board_cell" /> <ImageView
android:id="@+id/cell_3_2" android:layout_width="0dp" android:layout_height="0dp"
android:contentDescription="@null" android:src="@drawable/cell_empty"
app:layout_constraintBottom_toBottomOf="@id/v_board"
app:layout_constraintDimensionRatio="1:1" app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@id/cell_2_2"
app:layout_constraintWidth_percent="@dimen/board_cell" /> <ImageView
android:id="@+id/cell_3_3" android:layout_width="0dp" android:layout_height="0dp"
android:contentDescription="@null" android:src="@drawable/cell_empty"
app:layout_constraintBottom_toBottomOf="@id/v_board"
app:layout_constraintDimensionRatio="1:1"
app:layout_constraintEnd_toEndOf="@id/v_board"
app:layout_constraintStart_toEndOf="@id/cell_3_2"
app:layout_constraintTop_toBottomOf="@id/cell_2_3"
app:layout_constraintWidth_percent="@dimen/board_cell" /></merge>

Файл res/layout/entity_card.xml:

<?xml version="1.0" encoding="utf-8"?>

<merge xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout_width="match_parent"

android:layout_height="match_parent">

<androidx.constraintlayout.widget.Guideline

android:id="@+id/top_guideline"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:orientation="horizontal"

app:layout_constraintGuide_percent="0.15" />

```
<androidx.constraintlayout.widget.Guideline

android:id="@+id/bottom_guideline"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:orientation="horizontal"

app:layout_constraintGuide_percent="0.85" />

<androidx.constraintlayout.widget.ConstraintLayout

android:id="@+id/cl_entity"

android:layout_width="0dp"

android:layout_height="0dp"

android:background="@drawable/bg_status_inactive"

app:layout_constraintBottom_toBottomOf="@id/bottom_guideline"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toTopOf="@id/top_guideline">

<TextView

android:id="@+id/tv_name"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:textColor="@color/white"

android:textStyle="bold"

app:layout_constraintBottom_toBottomOf="parent"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="parent"
```

```
app:layout_constraintTop_toTopOf="parent"

tools:text="Player" />

<ImageView

android:id="@+id/iv_avatar"

android:layout_width="0dp"

android:layout_height="0dp"

android:contentDescription="@string/avatar"

app:layout_constraintBottom_toTopOf="@id/tv_name"

app:layout_constraintDimensionRatio="1:1"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toTopOf="parent"

app:layout_constraintWidth_percent="@dimen/iv_entity_card"

tools:src="@drawable/easy_bot" />

<ImageView

android:id="@+id/iv_mark"

android:layout_width="0dp"

android:layout_height="0dp"

android:contentDescription="@string/avatar"

app:layout_constraintBottom_toBottomOf="parent"

app:layout_constraintDimensionRatio="1:1"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@id/tv_name"
```

app:layout_constraintWidth_percent="@dimen/iv_entity_card"

tools:src="@drawable/tac" />

</androidx.constraintlayout.widget.ConstraintLayout>

<TextView

android:id="@+id/tv_description"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_marginTop="10dp"

android:textColor="@color/white"

android:textSize="20sp"

android:textStyle="bold"

android:visibility="invisible"

app:layout_constraintBottom_toBottomOf="parent"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toTopOf="@id/bottom_guideline"

tools:text="Winner"

tools:visibility="visible" />

</merge>

Файл tictactoe\customView\ResultGameView.kt:

package com.example.tictactoe.customViewsimport android.content.Contextimport android.graphics.Bitmapimport android.graphics.Canvasimport android.graphics.Paintimport android.graphics.Pathimport android.graphics.RectFimport android.graphics.Typefaceimport android.os.Buildimport android.os.Parcelimport android.os.Parcelableimport android.util.AttributeSetimport android.view.Viewimport androidx.appcompat.content.res.AppCompatResourcesimport

androidx.core.content.withStyledAttributesimport androidx.core.graphics.drawable.toBitmapimport androidx.core.graphics.withSaveimport com.example.tictactoe.Rimport com.example.tictactoe.data.Pointimport com.example.tictactoe.data.ResultGameimport kotlin.math.maxclass ResultGameView @JvmOverloads constructor(context: Context, attrs: AttributeSet? = null, defStyleAttr: Int = 0, defStyleRes: Int = 0) : View(context, attrs, defStyleAttr, defStyleRes) { private val backgroundClipPath = Path() private val avatarClipPath = Path() private var avatarRadius = 0f private var avatarMargin = 0f private var backgroundCornerRadius = 0f private var sizeTriangle = 0f private lateinit var backgroundRect: RectF private lateinit var avatarRect: RectF private lateinit var center: Point private lateinit var topLeft: Point private lateinit var bottomLeft: Point private lateinit var bottomRight: Point private lateinit var topSecond: Point private lateinit var bottomSecond: Point private lateinit var leftSecond: Point private lateinit var rightSecond: Point var text = "" set(value) { field = value invalidate() } var avatarResId: Int = 0 set(value) { field = value if (value != 0 && width > 0 && height > 0) { updateAvatarBitmap() } } var avatarBitmap: Bitmap? = null set(value) { field = value invalidate() } init { if (attrs != null) { initAttrs(attrs, defStyleAttr, defStyleRes) } } private fun initAttrs(attrs: AttributeSet?, defStyleAttr: Int, defStyleRes: Int) { if (attrs == null || avatarBitmap == null) { return } context.withStyledAttributes(attrs, R.styleable.ResultGameView, defStyleAttr, defStyleRes) { text = getString(R.styleable.ResultGameView_text) ?: "" avatarResId = getResourceId(R.styleable.ResultGameView_avatar, R.drawable.profile_avatar) } } override fun onMeasure(widthMeasureSpec: Int, heightMeasureSpec: Int) { val minWidth = suggestedMinimumWidth + paddingLeft + paddingRight val desiredWidth = max(minWidth, MeasureSpec.getSize(widthMeasureSpec) + paddingLeft + paddingRight) val desiredHeight = desiredWidth setMeasuredDimension(resolveSize(desiredWidth, widthMeasureSpec), resolveSize(desiredHeight, heightMeasureSpec)) } override fun onSizeChanged(w: Int, h: Int, oldw: Int, oldh: Int) { super.onSizeChanged(w, h, oldw, oldh) sizeTriangle = width / 10f backgroundCornerRadius = width / 20f avatarRadius = width / 3.5f avatarMargin = width / 40f backgroundRect = RectF(0f, 0f, width.toFloat(), height.toFloat()) center = Point(width / 2f, height / 3f) topLeft = Point(0f, 0f) bottomLeft = Point(0f, height.toFloat()) bottomRight = Point(width.toFloat(), height.toFloat()) topSecond = Point(sizeTriangle, 0f) bottomSecond = Point(sizeTriangle, height.toFloat()) leftSecond = Point(0f, height.toFloat() - sizeTriangle) rightSecond = Point(width.toFloat(), height.toFloat() - sizeTriangle) val locale = context.resources.configuration.locales[0] textPaint.textSize = if (locale.language == "uk") width / 8f else width / 6f backgroundClipPath.reset() backgroundClipPath.addRoundRect(backgroundRect, backgroundCornerRadius, backgroundCornerRadius, Path.Direction.CW) avatarClipPath.reset() avatarClipPath.addCircle(center.x, center.y, avatarRadius - avatarMargin, Path.Direction.CW) if (avatarResId != 0) { updateAvatarBitmap() } avatarBitmap?.let { bitmap -> val bitmapWidth = bitmap.width val bitmapHeight = bitmap.height val scale = if (bitmapWidth > 0 && bitmapHeight > 0) { (avatarRadius * 2) / minOf(bitmapWidth, bitmapHeight).toFloat() } else { 0f } val left = center.x - (bitmapWidth *

scale / 2) val top = center.y - (bitmapHeight * scale / 2) avatarRect = RectF(left, top, left + bitmapWidth * scale, top + bitmapHeight * scale) } ?: run { avatarRect = RectF(0f, 0f, 0f, 0f) } } private val paintMediumPurple = createPaint(R.color.medium_purple) private val paintRoyalPurple = createPaint(R.color.royal_purple) private val paintBackground = createPaint(R.color.royal_purple) private val paintAvatar = createPaint(R.color.dark_indigo) private val textPaint = Paint().apply { color = context.getColor(android.R.color.white) textAlign = Paint.Align.CENTER style = Paint.Style.FILL_AND_STROKE isAntiAlias = true typeface = Typeface.create(Typeface.DEFAULT, Typeface.BOLD) } override fun onDraw(canvas: Canvas) { super.onDraw(canvas) canvas.drawRoundRect(backgroundRect, backgroundCornerRadius, backgroundCornerRadius, paintBackground) canvas.withSave { clipPath(backgroundClipPath) // draw top triangle drawTriangleRow(this, center, topLeft, topSecond, sizeTriangle, true) // draw bottom triangle drawTriangleRow(this, center, bottomLeft, bottomSecond, sizeTriangle, false) // draw left triangle drawTriangleColumn(this, center, bottomLeft, leftSecond, sizeTriangle, true) // draw right triangle drawTriangleColumn(this, center, bottomRight, rightSecond, sizeTriangle, false) } canvas.drawText(text, width / 2f, height / 2f + avatarRadius, textPaint) canvas.drawCircle(center.x, center.y, avatarRadius, paintAvatar) canvas.withSave { clipPath(avatarClipPath) avatarBitmap?.let { canvas.drawBitmap(it, null, avatarRect, null) } } } fun loadResultGame(resultGame: ResultGame) { text = resultGame.result avatarBitmap = resultGame.avatarBitmap requestLayout() invalidate() } private fun createPaint(color: Int) = Paint().apply { this.color = context.getColor(color) style = Paint.Style.FILL isAntiAlias = true } private fun updateAvatarBitmap() { val drawable = AppCompatResources.getDrawable(context, avatarResId) if (drawable != null) { val size = (avatarRadius * 2).toInt() avatarBitmap = if (size > 0) { drawable.toBitmap(width = size, height = size) } else { drawable.toBitmap(width = 100, height = 100) } } } private fun drawTriangle(x: Point, y: Point, z: Point, paint: Paint, canvas: Canvas) { val path = Path() path.moveTo(x.x, x.y) path.lineTo(y.x, y.y) path.lineTo(z.x, z.y) path.close() canvas.drawPath(path, paint) } private fun drawTriangleRow(canvas: Canvas, center: Point, start: Point, second: Point, size: Float, startWithMedium: Boolean) { var y = start var z = second for (i in 1..10) { val useMediumColor = (i % 2 != 0 && startWithMedium) || (i % 2 == 0 && !startWithMedium) val paint = if (useMediumColor) paintMediumPurple else paintRoyalPurple drawTriangle(center, y, z, paint, canvas) y = z z = Point(z.x + size, z.y) } } private fun drawTriangleColumn(canvas: Canvas, center: Point, start: Point, second: Point, size: Float, startWithMedium: Boolean) { var y = start var z = second for (i in 1..10) { val useMediumColor = (i % 2 != 0 && startWithMedium) || (i % 2 == 0 && !startWithMedium) val paint = if (useMediumColor) paintMediumPurple else paintRoyalPurple drawTriangle(center, y, z, paint, canvas) y = z z = Point(z.x, z.y - size) } } override fun onSaveInstanceState(): Parcelable? { val superState = super.onSaveInstanceState() val savedState = SavedState(superState) savedState.resultGame = ResultGame(avatarBitmap, text) return savedState } override fun onRestoreInstanceState(state: Parcelable?) { val savedState = state as SavedState

super.onRestoreInstanceState(savedState.superState) avatarBitmap = savedState.resultGame?.avatarBitmap text = savedState.resultGame?.result ?: "" } class SavedState : BaseSavedState { var resultGame: ResultGame? = null constructor(superState: Parcelable?) : super(superState) constructor(parcel: Parcel) : super(parcel) { resultGame = if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) { parcel.readParcelable(ResultGame::class.java.classLoader, ResultGame::class.java) } else { @Suppress("DEPRECATION") parcel.readParcelable(ResultGame::class.java.classLoader) } } override fun writeToParcel(out: Parcel, flags: Int) { super.writeToParcel(out, flags) out.writeParcelable(resultGame, flags) } companion object { @JvmField val CREATOR: Parcelable.Creator<SavedState> = object : Parcelable.Creator<SavedState> { override fun createFromParcel(source: Parcel): SavedState { return SavedState(source) } override fun newArray(size: Int): Array<SavedState?> { return Array(size) { null } } } } }}

Файл res/values/colors.xml:

<?xml version="1.0" encoding="utf-8"?>

<resources>

<color name="black">#FF000000</color>

<color name="white">#FFFFFFFF</color>

<color name="blue">#3963ed</color>

<color name="red">#eb1751</color>

<color name="yellow">#fed032</color>

<color name="orange">#df6437</color>

<color name="orange_pressed">#f08060</color>

<color name="text_shadow">#2a1c53</color>

<color name="board_background">#6649c4</color>

<color name="board_cell_background">#332167</color>

<color name="indigo_violet">#552fd2</color>

<color name="indigo_violet_pressed">#7a5ff5</color>

<color name="pink">#ad53e7</color>

<color name="ocean_blue">#2f88b1</color>

<color name="ocean_blue_pressed">#55a8cc</color>

<color name="dark_indigo">#27175d</color>

<color name="dark_violet_blue">#312e81</color>

<color name="toolbar_background">#2a1c75</color>

<color name="profile_background">#1e1b4b</color>

<color name="background_red">#FF6B6B</color>

<color name="background_teal">#4ECDC4</color>

<color name="background_blue">#45B7D1</color>

<color name="background_green">#96CEB4</color>

<color name="background_yellow">#FFEEAD</color>

<color name="background_pink">#D4A5A5</color>

<color name="background_purple">#9B59B6</color>

<color name="background_orange">#E67E22</color>

<color name="gray">#5A5A68</color>

<color name="success_color">#22c55e</color>

<color name="error_color">#DC2626</color>

<color name="evergreen_depths">#297a6f</color>

<color name="midnight_blue">#354cab</color>

<color name="deep_navy">#24306d</color>

<color name="earthy_brown">#5A4B46</color>

<color name="smoky_brown">#473A3E</color>

<color name="deep_amethyst">#74378a</color>

<color name="raspberry_purple">#74378a</color>

<color name="medium_purple">#6648c4</color>

<color name="royal_purple">#5e41b8</color>

</resources>

Файл res/values/strings.xml (англійська мова, мова за замовчуванням):

<resources>

<string name="app_name">Tic Tac Toe</string>

<string name="single_player">Single Player</string>

<string name="multiplayer">Multiplayer</string>

<string name="local_multiplayer">Local Multiplayer</string>

<string name="game_title">

T

I

C

\n

T

A

C

\n

T

O

E

</string>

<string name="select_difficulty">Select Difficulty</string>

<string name="easy_bot">Easy Bot</string>

<string name="difficult_level_image">Difficult difficulty level icon</string>

<string name="difficult_bot">Difficult Bot</string>

<string name="medium_difficulty_level_image">Medium difficulty level icon</string>

<string name="medium_bot">Medium Bot</string>

<string name="easy_difficulty_level_image">Easy difficulty level icon</string>

<string name="opponent" translatable="false">Opponent</string>

<string name="toolbar_menu">Game Menu</string>

<string name="toolbar_difficulty">Choose Difficulty</string>

<string name="toolbar_bot">Tic-Tac-Toe: Bot</string>

<string name="toolbar_settings">Tic-Tac-Toe: Settings</string>

<string name="toolbar_game_result">Game Over: Play Again?</string>

<string name="toolbar_local_multiplayer">Tic-Tac-Toe: Local Multiplayer</string>

<string name="player_avatar">Player Avatar</string>

<string name="bot_avatar">Bot Avatar</string>

<string name="player_human">Human Player</string>

<string name="player_bot">Bot Player</string>

<string name="you_turn">Your Turn</string>

<string name="toolbar_edit_profile">Edit Profile</string>

<string name="done">Done</string>

<string name="enter_your_nickname">Enter Your Nickname</string>

<string name="nickname">Nickname</string>

<string name="save_changes">Save Changes</string>

<string name="char_count_tv">%d/15</string>

<string name="settings">Settings</string>

<string name="who_goes_first">Who goes first?</string>

<string name="random">Random</string>

<string name="select_photo">Select Photo</string>

<string name="your_symbol">Your Symbol</string>

<string name="save">Save</string>

<string name="cancel">Cancel</string>

<string name="you">You</string>

<string name="play_as_o">Play as O</string>

<string name="computer">Computer</string>

<string name="play_as_x">Play as X</string>

<string name="avatar">Avatar</string>

<string name="button_refresh">Refresh Button</string>

<string name="you_win">YOU WIN!</string>

<string name="you_lose">YOU LOSE!</string>

<string name="draw">DRAW!</string>

<string name="bot_turn">Bot Turn</string>

<string name="opponent_turn">Opponent Turn</string>

</resources>

Файл res/values-uk/strings.xml (українська мова):

<resources>

<string name="app_name">TicTacToe</string>

<string name="single_player">Один гравець</string>

<string name="multiplayer">Мультиплеєр</string>

<string name="local_multiplayer">Гра удвох</string>

<string name="game_title">

T

I

C

\n

T

A

C

\n

T

O

E

</string>

<string name="select_difficulty">Виберіть складність</string>

<string name="easy_bot">Легкий бот</string>

<string name="difficult_level_image">Зображення складного рівня</string>

<string name="difficult_bot">Складний бот</string>

<string name="medium_difficulty_level_image">Зображення середнього рівня</string>

<string name="medium_bot">Середній бот</string>

<string name="easy_difficulty_level_image">Зображення легкого рівня</string>

<string name="toolbar_menu">Меню гри</string>

<string name="toolbar_difficulty">Оберіть складність</string>

<string name="toolbar_bot">TicTacToe: Гра з ботом</string>

<string name="toolbar_settings">TicTacToe: Налаштування</string>

<string name="toolbar_game_result">Гра завершена: Грати ще?</string>

<string name="toolbar_local_multiplayer">TicTacToe: Гра удвох</string>

<string name="player_avatar">Аватар гравця</string>

<string name="bot_avatar">Аватар бота</string>

<string name="player_human">Гравець</string>

<string name="player_bot">Бот</string>

<string name="you_turn">Ваш хід</string>

<string name="toolbar_edit_profile">Редагувати профіль</string>

<string name="done">Готово</string>

<string name="enter_your_nickname">Введіть ваш нікнейм</string>

<string name="nickname">Нікнейм</string>

<string name="save_changes">Зберегти зміни</string>

<string name="char_count_tv">%d/15</string>

<string name="settings">Налаштування</string>

<string name="who_goes_first">Хто ходить першим?</string>

<string name="random">Випадково</string>

<string name="select_photo">Вибрати фото</string>

<string name="your_symbol">Ваш символ</string>

<string name="save">Зберегти</string>

<string name="cancel">Скасувати</string>

<string name="you">Ви</string>

<string name="play_as_o">Грати за O</string>

<string name="computer">Комп\'ютер</string>

<string name="play_as_x">Грати за X</string>

<string name="avatar">Аватар</string>

<string name="button_refresh">Грати ще</string>

<string name="you_win">ВИ ПЕРЕМОГЛИ!</string>

<string name="you_lose">ВИ ПРОГРАЛИ!</string>

<string name="draw">НІЧИЯ!</string>

<string name="bot_turn">Хід бота</string>

<string name="opponent_turn">Хід суперника</string>

</resources>

Файл res/values/dimens.xml:

<?xml version="1.0" encoding="utf-8"?>

<resources>

<dimen name="bt_difficulty_text">28sp</dimen>

<dimen name="bt_menu_text">28sp</dimen>

<dimen name="menu_title">110sp</dimen>

<dimen name="board_cell">0.3</dimen>

<dimen name="iv_entity_card">0.3</dimen>

<dimen name="avatar_size">20dp</dimen>

<dimen name="bt_refresh">90dp</dimen>

<dimen name="bt_refresh_padding">14dp</dimen>

<dimen name="result_game_margin">10dp</dimen>

<dimen name="difficulty_tv">50sp</dimen>

</resources>

Файл res/values-uk/dimens.xml:

<?xml version="1.0" encoding="utf-8"?>

<resources>

<dimen name="bt_difficulty_text">24sp</dimen>

<dimen name="difficulty_tv">35sp</dimen>

</resources>

Файл res/values/themes.xml:

<resources>

<style name="Base.Theme.TicTacToe" parent="Theme.Material3.DayNight.NoActionBar">

<item name="android:statusBarColor">@color/toolbar_background</item>

</style>

<style name="bt_main_menu">

<item name="android:layout_width">0dp</item>

<item name="android:layout_height">0dp</item>

<item name="android:textColor">@color/white</item>

<item name="android:textSize">@dimen/bt_menu_text</item>

<item name="android:textStyle">bold</item>

<item name="backgroundTint">@null</item>

<item name="layout_constraintHeight_percent">0.085</item>

<item name="layout_constraintWidth_percent">0.75</item>

<item name="layout_constraintStart_toStartOf">parent</item>

<item name="layout_constraintEnd_toEndOf">parent</item>

</style>

<style name="bt_difficulty">

<item name="android:layout_width">0dp</item>

<item name="android:layout_height">0dp</item>

<item name="android:layout_marginStart">60dp</item>

<item name="android:paddingStart">45dp</item>

<item name="android:paddingEnd">0dp</item>

<item name="android:textColor">@color/white</item>

<item name="android:textSize">@dimen/bt_difficulty_text</item>

<item name="backgroundTint">@null</item>

<item name="layout_constraintStart_toStartOf">parent</item>

<item name="layout_constraintEnd_toEndOf">parent</item>

<item name="layout_constraintHeight_percent">0.1</item>

<item name="layout_constraintWidth_percent">0.65</item>

</style>

<style name="ic_difficulty">

<item name="android:layout_width">120dp</item>

<item name="android:layout_height">120dp</item>

<item name="android:layout_marginEnd">-60dp</item>

</style>

<style name="item_settings">

<item name="android:layout_width">0dp</item>

<item name="android:layout_height">70dp</item>

<item name="android:background">@drawable/v_board</item>

<item name="android:backgroundTint">@color/profile_background</item>

<item name="android:layout_marginTop">10dp</item>

<item name="layout_constraintEnd_toStartOf">@+id/end_guide_line</item>

<item name="layout_constraintStart_toEndOf">@+id/begin_guide_line</item>

</style>

<style name="ic_bg_settings">

<item name="android:layout_width">42dp</item>

<item name="android:layout_height">42dp</item>

<item name="android:layout_marginStart">15dp</item>

<item name="android:backgroundTint">@color/deep_navy</item>

</style>

<style name="ic_settings">

<item name="android:layout_width">32dp</item>

<item name="android:layout_height">32dp</item>

<item name="android:padding">2dp</item>

</style>

<style name="item_tv_settings">

<item name="android:layout_width">wrap_content</item>

<item name="android:layout_height">wrap_content</item>

<item name="android:layout_marginStart">10dp</item>

<item name="android:textColor">@color/white</item>

<item name="android:textSize">22sp</item>

</style>

<style name="tv_settings">

<item name="android:layout_width">wrap_content</item>

<item name="android:layout_height">wrap_content</item>

<item name="android:layout_marginTop">35dp</item>

<item name="android:textColor">@color/white</item>

<item name="android:textSize">20sp</item>

<item name="android:textStyle">bold</item>

</style>

<style name="bt_settings">

<item name="android:layout_width">0dp</item>

<item name="android:layout_height">70dp</item>

<item name="android:textColor">@color/white</item>

<item name="android:textSize">20sp</item>

<item name="android:textStyle">bold</item>

<item name="cornerRadius">15dp</item>

</style>

<style name="Theme.TicTacToe" parent="Base.Theme.TicTacToe" />

</resources>

ДОДАТОК Б

Код для побудови контрактів графічного інтерфейсу та навігації

Файл contract/Navigator.kt:

package com.example.tictactoe.contract

import androidx.fragment.app.Fragment

import com.example.tictactoe.data.EntityCard

import com.example.tictactoe.data.ResultGame

import com.example.tictactoe.fragments.OnRefreshClick

```
fun Fragment.navigator(): Navigator {
return requireActivity() as Navigator
}
interface Navigator {
fun showBotGameScreen(level: Int)
fun showDifficultyScreen()
fun showEditProfileScreen()
fun showSettingsMenuScreen()
fun showLocalMultiplayerScreen()
fun showGameOverScreen(
humanEntityCard: EntityCard,
opponentEntityCard: EntityCard,
resultGame: ResultGame,
onRefreshClick: OnRefreshClick
)
fun goBack()
fun goToMenu()
fun createCustomToolbarAction(action: CustomAction)
}
```

Файл contract/ HasCustomAction.kt:

```
package com.example.tictactoe.contract
import androidx.annotation.DrawableRes
import androidx.annotation.StringRes
interface HasCustomAction {
```

```
fun getCustomAction(): CustomAction

}

class CustomAction(

@DrawableRes val iconRes: Int,

@StringRes val textRes: Int,

val onCustomAction: Runnable

)
```

Файл contract/ HasCustomTitle.kt:

```
package com.example.tictactoe.contract

import androidx.annotation.StringRes

interface HasCustomTitle {

@StringRes

fun getTitleRes(): Int }
```

ДОДАТОК В

Код головної активності програми

Файл tictactoe\MainActivity.kt:

```
package com.example.tictactoe

import android.graphics.Color

import android.os.Bundle

import android.view.Menu

import android.view.MenuItem

import androidx.appcompat.app.AppCompatActivity

import androidx.core.content.ContextCompat

import androidx.core.graphics.drawable.DrawableCompat
```

```
import androidx.fragment.app.Fragment

import androidx.fragment.app.FragmentManager

import com.example.tictactoe.contract.CustomAction

import com.example.tictactoe.contract.HasCustomAction

import com.example.tictactoe.contract.HasCustomTitle

import com.example.tictactoe.contract.Navigator

import com.example.tictactoe.data.EntityCard

import com.example.tictactoe.data.ResultGame

import com.example.tictactoe.databinding.ActivityMainBinding

import com.example.tictactoe.fragments.BotGameFragment

import com.example.tictactoe.fragments.DifficultyFragment

import com.example.tictactoe.fragments.EditProfileFragment

import com.example.tictactoe.fragments.GameOverFragment

import com.example.tictactoe.fragments.LocalMultiplayerFragment

import com.example.tictactoe.fragments.MenuFragment

import com.example.tictactoe.fragments.SettingsFragment

class MainActivity : AppCompatActivity(), Navigator {

42 private lateinit var binding: ActivityMainBinding

private val currentFragment: Fragment

get() = supportFragmentManager.findFragmentById(R.id.fragmentContainer)!!

private val fragmentListener = object : FragmentManager.FragmentLifecycleCallbacks() {

override fun onFragmentStarted(fm: FragmentManager, f: Fragment) {

super.onFragmentStarted(fm, f)
```

```
updateUi()
}
}
42 override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
binding = ActivityMainBinding.inflate(layoutInflater).also { setContentView(it.root) }
setSupportActionBar(binding.toolbar)
if (savedInstanceState == null) {
val fragment = MenuFragment()
supportFragmentManager
.beginTransaction()
.setCustomAnimations(
R.anim.slide_in,
R.anim.fade_out,
R.anim.fade_in,
R.anim.slide_out
)
.add(R.id.fragmentContainer, fragment)
.commit()
}
supportFragmentManager.registerFragmentLifecycleCallbacks(fragmentListener, false)
}
override fun onCreateOptionsMenu(menu: Menu?): Boolean {
updateUi()
```

```
return true
}
private fun launchFragment(fragment: Fragment) {
supportFragmentManager
.beginTransaction()
.setCustomAnimations(
R.anim.slide_in,
R.anim.fade_out,
R.anim.fade_in,
R.anim.slide_out
)
.addToBackStack(null)
.replace(R.id.fragmentContainer, fragment)
.commit()
}
override fun onSupportNavigateUp(): Boolean {
goBack()
return true
}
override fun onDestroy() {
super.onDestroy()
supportFragmentManager.unregisterFragmentLifecycleCallbacks(fragmentListener)
}
override fun showDifficultyScreen() {
```

```
launchFragment(DifficultyFragment())
}
override fun showEditProfileScreen() {
launchFragment(EditProfileFragment())
}
override fun showSettingsMenuScreen() {
launchFragment(SettingsFragment())
}
override fun showBotGameScreen(level: Int) {
launchFragment(BotGameFragment.newInstance(level))
}
override fun showLocalMultiplayerScreen() {
launchFragment(LocalMultiplayerFragment())
}
override fun showGameOverScreen(
humanEntityCard: EntityCard,
opponentEntityCard: EntityCard,
resultGame: ResultGame,
onRefreshClick: () -> Unit
) {
val fragment = GameOverFragment.newInstance(
humanEntityCard,
opponentEntityCard,
resultGame
```

```
)

fragment.onRefreshClick = onRefreshClick

launchFragment(fragment)

}

override fun goBack() {

onBackPressedDispatcher.onBackPressed()

}

override fun goToMenu() {

supportFragmentManager.popBackStack(null,
FragmentManager.POP_BACK_STACK_INCLUSIVE)

}

private fun updateUi() {

val fragment = currentFragment

if (fragment is HasCustomTitle) {

binding.toolbar.title = getString(fragment.getTitleRes())

} else {

binding.toolbar.title = getString(R.string.toolbar_menu)

}

if (supportFragmentManager.backStackEntryCount > 0) {

supportActionBar?.setDisplayHomeAsUpEnabled(true)

supportActionBar?.setDisplayShowHomeEnabled(true)

} else {

supportActionBar?.setDisplayHomeAsUpEnabled(false)

supportActionBar?.setDisplayShowHomeEnabled(false)
```

```
}
if (fragment is HasCustomAction) {
createCustomToolbarAction(fragment.getCustomAction())
} else {
binding.toolbar.menu.clear()
}
}
override fun createCustomToolbarAction(action: CustomAction) {
binding.toolbar.menu.clear()
val iconDrawable = DrawableCompat.wrap(ContextCompat.getDrawable(this,
action.iconRes)!!)
iconDrawable.setTint(Color.WHITE)
val menuItem = binding.toolbar.menu.add(action.textRes)
menuItem.setShowAsAction(MenuItem.SHOW_AS_ACTION_ALWAYS)
menuItem.icon = iconDrawable
menuItem.setOnMenuItemClickListener {
action.onCustomAction.run()
return@setOnMenuItemClickListener true
}
}
}
```

ДОДАТОК Г

Код для створення власних компонентів гри

Файл customViews/ BoardView.kt:

```
package com.example.tictactoe.customViews

import android.animation.Animator

import android.animation.AnimatorListenerAdapter

import android.animation.AnimatorSet

import android.animation.ObjectAnimator

import android.animation.PropertyValuesHolder

24 import android.content.Context

24 import android.graphics.Color

import android.os.Parcel

import android.os.Parcelable

24 import android.util.AttributeSet

24 import android.view.LayoutInflater

import android.view.animation.AccelerateDecelerateInterpolator

import android.view.animation.AccelerateInterpolator

import android.view.animation.DecelerateInterpolator

import android.widget.ImageView


import androidx.constraintlayout.widget.ConstraintLayout

import com.example.tictactoe.R

import com.example.tictactoe.ai.Mark

import com.example.tictactoe.databinding.BoardBinding

import kotlinx.coroutines.CoroutineScope

import kotlinx.coroutines.Dispatchers

import kotlinx.coroutines.Job
```

```
import kotlinx.coroutines.cancel

import kotlinx.coroutines.delay

import kotlinx.coroutines.launch

import kotlin.properties.Delegates

enum class GameState {

ONGOING,

HUMAN_WIN,

AI_WIN,

DRAW

}

typealias BoardViewListener = (r: Int, c: Int) -> Unit

typealias GameStateListener = (GameState) -> Unit

class BoardView @JvmOverloads constructor(

context: 24 Context,

24 attrs: AttributeSet? = null,

24 defStyleAttr: Int = 0,

defStyleRes: Int = 0

) : ConstraintLayout(context, attrs, defStyleAttr, defStyleRes) {

private val binding: BoardBinding

private val cells: Array<ImageView>

var humanMark: Int by Delegates.notNull<Int>()

private var boardState = Array(3) { IntArray(3) { 0 } }

private val moveListeners = mutableListOf<BoardViewListener>()

private val gameStateListeners = mutableListOf<GameStateListener>()
```

```
internal var currentGameState: GameState = GameState.ONGOING

set(value) {

if (field != value) {

field = value

notifyGameStateChanged(value)

}

}

init {

val inflater = LayoutInflater.from(context)

inflater.inflate(R.layout.board, this, true)

binding = BoardBinding.bind(this)

cells = getCells()

setupCellClickListeners()

}

private fun setupCellClickListeners() {

for (cell in cells) {

cell.setOnClickListener {

val index = cells.indexOf(it)

val row = index / 3

val col = index % 3

if (boardState[row][col] == 0 && currentGameState == GameState.ONGOING) {

makeHumanMove(row, col)

}

}
```

```
}
}
private fun makeHumanMove(row: Int, col: Int) {
notifyChanges(row, col)
checkGameState()
}
fun setMove(row: Int, column: Int, mark: Mark) {
val cell = cells[row * 3 + column]
when (mark) {
Mark.TAC -> updateCell(cell, TAC)
Mark.TIC -> updateCell(cell, TIC)
else -> return
}
boardState[row][column] = if (mark == Mark.TIC) 1 else -1
checkGameState()
}
private fun checkGameState() {
val humanValue = if (humanMark == TIC) 1 else -1
val aiValue = if (humanMark == TIC) -1 else 1
if (checkWin(humanValue)) {
currentGameState = GameState.HUMAN_WIN
} else if (checkWin(aiValue)) {
currentGameState = GameState.AI_WIN
} else if (isBoardFull()) {
```

```
currentGameState = GameState.DRAW
} else {
currentGameState = GameState.ONGOING
}
}
private fun checkWin(player: Int): Boolean {
for (row in 0 until 3) {
if (boardState[row].all { it == player }) return true
}
for (col in 0 until 3) {
if ((0 until 3).all { boardState[it][col] == player }) return true
}
if ((0 until 3).all { boardState[it][it] == player }) return true
if ((0 until 3).all { boardState[it][2 - it] == player }) return true
return false
}
private fun isBoardFull(): Boolean {
return boardState.all { row -> row.all { it != 0 } }
}
fun addMoveListener(listener: BoardViewListener) {
moveListeners.add(listener)
}
fun addGameStateListener(listener: GameStateListener) {
gameStateListeners.add(listener)
```

```
}
private fun notifyChanges(row: Int, col: Int) {
moveListeners.forEach { it.invoke(row, col) }
}
private val scope = CoroutineScope(Dispatchers.Main + Job())
private fun notifyGameStateChanged(state: GameState) {
scope.launch {
when (state) {
GameState.HUMAN_WIN -> {
val winLine = getWinLine(humanMark)
animateWinLine(winLine)
delay(1000)
}
GameState.AI_WIN -> {
val computerMark = if (humanMark == TIC) TAC else TIC
val winLine = getWinLine(computerMark)
animateWinLine(winLine)
delay(1000)
}
GameState.DRAW -> {
delay(1000)
}
GameState.ONGOING -> {
}
```

```
}

gameStateListeners.forEach { it.invoke(state) }

}

}

override fun onDetachedFromWindow() {

super.onDetachedFromWindow()

scope.cancel()

}

private fun getWinLine(markResId: Int): Array<ImageView> {

val boardValue = if (markResId == TIC) 1 else -1

for (i in 0 until 3) {

if (boardState[0][i] == boardValue && boardState[1][i] == boardValue && boardState[2][i] ==
boardValue) {

return arrayOf(getCells()[i], getCells()[i + 3], getCells()[i + 6])

}

if (boardState[i][0] == boardValue && boardState[i][1] == boardValue && boardState[i][2] ==
boardValue) {

return arrayOf(getCells()[i * 3], getCells()[i * 3 + 1], getCells()[i * 3 + 2])

}

}

if (boardState[0][0] == boardValue && boardState[1][1] == boardValue && boardState[2][2] ==
boardValue) {

return arrayOf(getCells()[0], getCells()[4], getCells()[8])

}

if (boardState[0][2] == boardValue && boardState[1][1] == boardValue && boardState[2][0] ==
```

```
boardValue) {
return arrayOf(getCells()[2], getCells()[4], getCells()[6])

}
return emptyArray()
}
private fun animateWinLine(winLine: Array<ImageView>) {
if (winLine.isEmpty()) return
for (cell in winLine) {
cell.alpha = 1f
cell.scaleX = 1f
cell.scaleY = 1f
}
val duration = 300L
val delay = 100L
for (i in winLine.indices) {
val cell = winLine[i]
val scaleUp = ObjectAnimator.ofPropertyValuesHolder(
cell,
PropertyValuesHolder.ofFloat("scaleX", 1.2f),
PropertyValuesHolder.ofFloat("scaleY", 1.2f)
).apply {
this.duration = duration
this.startDelay = i * delay
```

```
interpolator = DecelerateInterpolator()
}
val scaleDown = ObjectAnimator.ofPropertyValuesHolder(
cell,
PropertyValuesHolder.ofFloat("scaleX", 1f),
PropertyValuesHolder.ofFloat("scaleY", 1f)
).apply {
this.duration = duration
interpolator = AccelerateInterpolator()
}
val highlight = ObjectAnimator.ofArgb(
cell,
"colorFilter",
Color.WHITE,
Color.YELLOW
).apply {
this.duration = duration * 2
this.startDelay = i * delay
interpolator = AccelerateDecelerateInterpolator()
repeatCount = ObjectAnimator.INFINITE
repeatMode = ObjectAnimator.REVERSE
}
val animatorSet = AnimatorSet().apply {
playSequentially(scaleUp, scaleDown)
```

```
}
animatorSet.start()
highlight.start()
}
}
override fun onSaveInstanceState(): Parcelable {
val superState = super.onSaveInstanceState()!!
val savedState = SavedState(superState)
savedState.boardState = boardState.flatMap { it.toList() }.toIntArray()
savedState.gameState = currentGameState.ordinal
return savedState
}
override fun onRestoreInstanceState(state: Parcelable?) {
val savedState = state as SavedState
super.onRestoreInstanceState(savedState.superState)
val flatState = savedState.boardState
for (i in 0 until 3) {
for (j in 0 until 3) {
boardState[i][j] = flatState[i * 3 + j]
when (flatState[i * 3 + j]) {
1 -> updateCell(cells[i * 3 + j], TIC)
-1 -> updateCell(cells[i * 3 + j], TAC)
}
}
```

```
}
currentGameState = GameState.entries.toTypedArray()[savedState.gameState]
}
private fun getCells(): Array<ImageView> {
with(binding) {
return arrayOf(
cell11, cell12, cell13,
cell21, cell22, cell23,
cell31, cell32, cell33
)
}
}
private fun updateCell(imageView: ImageView, newDrawableRes: Int) {
imageView.tag = newDrawableRes
imageView.isClickable = false
val halfDuration = 150L
val flipOut = ObjectAnimator.ofFloat(imageView, "rotationY", 0f, 90f).apply {
duration = halfDuration
interpolator = AccelerateInterpolator()
}
flipOut.addListener(object : AnimatorListenerAdapter() {
override fun onAnimationEnd(animation: Animator) {
imageView.setImageResource(newDrawableRes)
imageView.rotationY = -90f
```

```
val flipIn = ObjectAnimator.ofFloat(imageView, "rotationY", -90f, 0f).apply {
duration = halfDuration
interpolator = DecelerateInterpolator()
}
flipIn.start()
}
})
flipOut.start()
}
companion object {
private val TIC = R.drawable.cell_tic
private val TAC = R.drawable.cell_tac
}
class SavedState : BaseSavedState {
var boardState: IntArray = IntArray(9)
var gameState: Int = 0
constructor(superState: Parcelable) : super(superState)
constructor(parcel: Parcel) : super(parcel) {
boardState = parcel.createIntArray() ?: IntArray(9)
gameState = parcel.readInt()
}
override fun writeToParcel(out: Parcel, flags: Int) {
super.writeToParcel(out, flags)
out.writeIntArray(boardState)
```

```
out.writeInt(gameState)
}
companion object {
@JvmField
val CREATOR = object : Parcelable.Creator<SavedState> {
override fun createFromParcel(parcel: Parcel): SavedState {
return SavedState(parcel)
}
override fun newArray(size: Int): Array<SavedState?> {
return arrayOfNulls(size)
}
}
}
}
}
```

Файл customViews/ EntityCardView.kt:

```
package com.example.tictactoe.customViews
import android.content.Context
import android.content.res.Resources
import android.graphics.Bitmap
import android.graphics.drawable.BitmapDrawable
import android.os.Build
import android.os.Parcel
import android.os.Parcelable
```

```
import android.util.AttributeSet

import android.util.TypedValue

import android.view.LayoutInflater

import android.widget.ImageView

import androidx.constraintlayout.widget.ConstraintLayout

import androidx.core.content.withStyledAttributes

import androidx.core.graphics.drawable.toBitmap

import com.example.tictactoe.R

import com.example.tictactoe.data.EntityCard

import com.example.tictactoe.databinding.EntityCardBinding

class EntityCardView @JvmOverloads constructor(

context: Context, attrs: AttributeSet? = null, defStyleAttr: Int = 0, defStyleRes: Int = 0

) : ConstraintLayout(context, attrs, defStyleAttr, defStyleRes) {

private val binding: EntityCardBinding


var name: String = ""

set(value) {

field = value

binding.tvName.text = value

}

val ivAvatar: ImageView

get() = binding.ivAvatar

var avatarResId: Int = 0

set(value) {
```

```
field = value
binding.ivAvatar.setImageResource(value)
updateAvatarBitmapFromImageView()
}
var avatarBitmap: Bitmap? = null
set(value) {
field = value
if (value != null) {
binding.ivAvatar.setImageBitmap(value)
}
}
var mark: Int = 0
set(value) {
field = value
binding.ivMark.setImageResource(value)
}
var description: String = ""
set(value) {
field = value
binding.tvDescription.text = value
}
var active = false
set(value) {
field = value
```

```
binding.tvDescription.visibility = if (value) VISIBLE else GONE

binding.clEntity.setBackgroundResource(if (value) R.drawable.bg_status_active else
R.drawable.bg_status_inactive)

}

init {

val inflater = LayoutInflater.from(context)

inflater.inflate(R.layout.entity_card, this, true)

binding = EntityCardBinding.bind(this)

binding.ivAvatar.addOnLayoutChangeListener { _, _, _, _, _, _, _, _, _ ->

updateAvatarBitmapFromImageView()

}

adjustTextSize()

initializeAttributes(attrs, defStyleAttr, defStyleRes)

}

fun loadEntityCard(entityCard: EntityCard) {

name = entityCard.name

avatarBitmap = entityCard.avatar

mark = entityCard.mark

description = entityCard.description

invalidate()

}

private fun adjustTextSize() {

val screenWidth = Resources.getSystem().displayMetrics.widthPixels

val textSizeInDp = screenWidth * 0.045f / Resources.getSystem().displayMetrics.density
```

```
binding.tvName.setTextSize(TypedValue.COMPLEX_UNIT_DIP, textSizeInDp)
binding.tvDescription.setTextSize(TypedValue.COMPLEX_UNIT_DIP, textSizeInDp)
}
private fun initializeAttributes(attrs: AttributeSet?, defStyleAttr: Int, defStyleRes: Int) {
if (attrs == null) return
context.withStyledAttributes(
attrs, R.styleable.EntityCardView, defStyleAttr, defStyleRes
) {
name =
getString(R.styleable.EntityCardView_name) ?: context.getString(R.string.easy_bot)
avatarResId = getResourceId(R.styleable.EntityCardView_avatar, R.drawable.easy_bot)
mark = getResourceId(R.styleable.EntityCardView_mark, R.drawable.tac)
description = getString(R.styleable.EntityCardView_description) ?: ""
active = getBoolean(R.styleable.EntityCardView_active, false)
}
}
private fun updateAvatarBitmapFromImageView() {
val drawable = binding.ivAvatar.drawable
avatarBitmap = if (drawable is BitmapDrawable) {
drawable.bitmap
} else {
drawable?.toBitmap()
}
}
```

```
Override fun onSaveInstanceState(): Parcelable {
val superState = super.onSaveInstanceState()
val savedState = SavedState(superState)
savedState.entityCard = EntityCard(name, description, avatarBitmap, mark)
return savedState
}
override fun onRestoreInstanceState(state: Parcelable?) {
if (state !is SavedState) {
super.onRestoreInstanceState(state)
return
}
super.onRestoreInstanceState(state.superState)
state.entityCard?.let {
name = it.name
description = it.description
avatarBitmap = it.avatar
mark = it.mark
}
}
class SavedState : BaseSavedState {
var entityCard: EntityCard? = null
constructor(superState: Parcelable?) : super(superState)
constructor(parcel: Parcel) : super(parcel) {
entityCard = if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
```

```
parcel.readParcelable(EntityCard::class.java.classLoader, EntityCard::class.java)
} else {
@Suppress("DEPRECATION") parcel.readParcelable(EntityCard::class.java.classLoader)
}
}
override fun writeToParcel(out: Parcel, flags: Int) {
super.writeToParcel(out, flags)
out.writeParcelable(entityCard, flags)
}
companion object {
@JvmField
val CREATOR: Parcelable.Creator<SavedState> = object : Parcelable.Creator<SavedState> {
override fun createFromParcel(source: Parcel): SavedState {
return SavedState(source)
}
override fun newArray(size: Int): Array<SavedState?> {
return arrayOfNulls(size)
}
}
}
}
}
}
```

Файл customViews/ ResultGameView.kt:

```
package com.example.tictactoe.customViews
```

```
import android.content.Context
import android.graphics.Bitmap
import android.graphics.Canvas
import android.graphics.Paint
import android.graphics.Path
import android.graphics.RectF
import android.graphics.Typeface
import android.os.Build
import android.os.Parcel
import android.os.Parcelable
import android.util.AttributeSet
import android.view.View
import androidx.appcompat.content.res.AppCompatResources
import androidx.core.content.withStyledAttributes
import androidx.core.graphics.drawable.toBitmap
import androidx.core.graphics.withSave

import com.example.tictactoe.R
import com.example.tictactoe.data.Point
import com.example.tictactoe.data.ResultGame
import kotlin.math.max
class ResultGameView @JvmOverloads constructor(
context: Context, attrs: AttributeSet? = null, defStyleAttr: Int = 0, defStyleRes: Int = 0
) : View(context, attrs, defStyleAttr, defStyleRes) {
```

```
private val backgroundClipPath = Path()

private val avatarClipPath = Path()

private var avatarRadius = 0f

private var avatarMargin = 0f

private var backgroundCornerRadius = 0f

private var sizeTriangle = 0f

private lateinit var backgroundRect: RectF

private lateinit var avatarRect: RectF

private lateinit var center: Point

private lateinit var topLeft: Point

private lateinit var bottomLeft: Point

private lateinit var bottomRight: Point

private lateinit var topSecond: Point

private lateinit var bottomSecond: Point

private lateinit var leftSecond: Point

private lateinit var rightSecond: Point

var text = ""

set(value) {

field = value

invalidate()

}

var avatarResId: Int = 0

set(value) {

field = value
```

```
if (value != 0 && width > 0 && height > 0) {
updateAvatarBitmap()
}
}
var avatarBitmap: Bitmap? = null
set(value) {
field = value
invalidate()
}
init {
if (attrs != null) {
initAttrs(attrs, defStyleAttr, defStyleRes)
}
}
private fun initAttrs(attrs: AttributeSet?, defStyleAttr: Int, defStyleRes: Int) {
if (attrs == null || avatarBitmap == null) {
return
}
context.withStyledAttributes(attrs, R.styleable.ResultGameView, defStyleAttr, defStyleRes) {
text = getString(R.styleable.ResultGameView_text) ?: ""
avatarResId =
getResourceId(R.styleable.ResultGameView_avatar, R.drawable.profile_avatar)
}
}
```

```
override fun onMeasure(widthMeasureSpec: Int, heightMeasureSpec: Int) {
val minWidth = suggestedMinimumWidth + paddingLeft + paddingRight
val desiredWidth =
max(minWidth, MeasureSpec.getSize(widthMeasureSpec) + paddingLeft + paddingRight)
val desiredHeight = desiredWidth
setMeasuredDimension(
resolveSize(desiredWidth, widthMeasureSpec),
resolveSize(desiredHeight, heightMeasureSpec)
)
}
override fun onSizeChanged(w: Int, h: Int, oldw: Int, oldh: Int) {
super.onSizeChanged(w, h, oldw, oldh)
sizeTriangle = width / 10f
backgroundCornerRadius = width / 20f
avatarRadius = width / 3.5f
avatarMargin = width / 40f
backgroundRect = RectF(0f, 0f, width.toFloat(), height.toFloat())
center = Point(width / 2f, height / 3f)
topLeft = Point(0f, 0f)
bottomLeft = Point(0f, height.toFloat())
bottomRight = Point(width.toFloat(), height.toFloat())
topSecond = Point(sizeTriangle, 0f)
bottomSecond = Point(sizeTriangle, height.toFloat())
leftSecond = Point(0f, height.toFloat() - sizeTriangle)
```

```
rightSecond = Point(width.toFloat(), height.toFloat() - sizeTriangle)

val locale = context.resources.configuration.locales[0]

textPaint.textSize = if (locale.language == "uk") width / 8f else width / 6f

backgroundClipPath.reset()

backgroundClipPath.addRoundRect(

backgroundRect, backgroundCornerRadius, backgroundCornerRadius, Path.Direction.CW

)

avatarClipPath.reset()

avatarClipPath.addCircle(

center.x, center.y, avatarRadius - avatarMargin, Path.Direction.CW

)

if (avatarResId != 0) {

updateAvatarBitmap()

}

avatarBitmap?.let { bitmap ->

val bitmapWidth = bitmap.width

val bitmapHeight = bitmap.height

val scale = if (bitmapWidth > 0 && bitmapHeight > 0) {

(avatarRadius * 2) / minOf(bitmapWidth, bitmapHeight).toFloat()

} else {

0f

}

val left = center.x - (bitmapWidth * scale / 2)

val top = center.y - (bitmapHeight * scale / 2)
```

```
avatarRect = RectF(
left, top, left + bitmapWidth * scale, top + bitmapHeight * scale
)
} ?: run {
avatarRect = RectF(0f, 0f, 0f, 0f)
}
}
private val paintMediumPurple = createPaint(R.color.medium_purple)
private val paintRoyalPurple = createPaint(R.color.royal_purple)
private val paintBackground = createPaint(R.color.royal_purple)
private val paintAvatar = createPaint(R.color.dark_indigo)
private val textPaint = Paint().apply {
color = context.getColor(android.R.color.white)
textAlign = Paint.Align.CENTER
style = Paint.Style.FILL_AND_STROKE
isAntiAlias = true
typeface = Typeface.create(Typeface.DEFAULT, Typeface.BOLD)
}
override fun onDraw(canvas: Canvas) {
super.onDraw(canvas)
canvas.drawRoundRect(
backgroundRect, backgroundCornerRadius, backgroundCornerRadius, paintBackground
)
canvas.withSave {
```

```
clipPath(backgroundClipPath)
// draw top triangle
drawTriangleRow(
this, center, topLeft, topSecond, sizeTriangle, true
)
// draw bottom triangle
drawTriangleRow(
this, center, bottomLeft, bottomSecond, sizeTriangle, false
)
// draw left triangle
drawTriangleColumn(

this, center, bottomLeft, leftSecond, sizeTriangle, true
)
// draw right triangle
drawTriangleColumn(
this, center, bottomRight, rightSecond, sizeTriangle, false
)
}
canvas.drawText(text, width / 2f, height / 2f + avatarRadius, textPaint)
canvas.drawCircle(center.x, center.y, avatarRadius, paintAvatar)
canvas.withSave {
clipPath(avatarClipPath)
avatarBitmap?.let {
```

```
canvas.drawBitmap(it, null, avatarRect, null)
}
}
}
fun loadResultGame(resultGame: ResultGame) {
text = resultGame.result
avatarBitmap = resultGame.avatarBitmap
requestLayout()
invalidate()
}
private fun createPaint(color: Int) = Paint().apply {
this.color = context.getColor(color)
style = Paint.Style.FILL
isAntiAlias = true
}
private fun updateAvatarBitmap() {
val drawable = AppCompatResources.getDrawable(context, avatarResId)
if (drawable != null) {
val size = (avatarRadius * 2).toInt()
avatarBitmap = if (size > 0) {
drawable.toBitmap(width = size, height = size)
} else {
drawable.toBitmap(width = 100, height = 100)
}
```

```
}
}
private fun drawTriangle(x: Point, y: Point, z: Point, paint: Paint, canvas: Canvas) {
val path = Path()
path.moveTo(x.x, x.y)
path.lineTo(y.x, y.y)
path.lineTo(z.x, z.y)
path.close()
canvas.drawPath(path, paint)
}
private fun drawTriangleRow(
canvas: Canvas,
center: Point,
start: Point,
second: Point,
size: Float,
startWithMedium: Boolean
) {
var y = start
var z = second
for (i in 1..10) {
val useMediumColor = (i % 2 != 0 && startWithMedium) || (i % 2 == 0 && !startWithMedium)
val paint = if (useMediumColor) paintMediumPurple else paintRoyalPurple
drawTriangle(center, y, z, paint, canvas)
```

```
y = z
z = Point(z.x + size, z.y)
}
}
private fun drawTriangleColumn(
canvas: Canvas,
center: Point,
start: Point,
second: Point,
size: Float,
startWithMedium: Boolean
) {
var y = start
var z = second
for (i in 1..10) {
val useMediumColor = (i % 2 != 0 && startWithMedium) || (i % 2 == 0 && !startWithMedium)
val paint = if (useMediumColor) paintMediumPurple else paintRoyalPurple
drawTriangle(center, y, z, paint, canvas)
y = z
z = Point(z.x, z.y - size)
}
}
override fun onSaveInstanceState(): Parcelable? {
val superState = super.onSaveInstanceState()
```

```
val savedState = SavedState(superState)
savedState.resultGame = ResultGame(avatarBitmap, text)
return savedState
}
override fun onRestoreInstanceState(state: Parcelable?) {
val savedState = state as SavedState
super.onRestoreInstanceState(savedState.superState)
avatarBitmap = savedState.resultGame?.avatarBitmap
text = savedState.resultGame?.result ?: ""
}
class SavedState : BaseSavedState {
var resultGame: ResultGame? = null
constructor(superState: Parcelable?) : super(superState)
constructor(parcel: Parcel) : super(parcel) {
resultGame = if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
parcel.readParcelable(ResultGame::class.java.classLoader, ResultGame::class.java)
} else {
@Suppress("DEPRECATION") parcel.readParcelable(ResultGame::class.java.classLoader)
}
}
override fun writeToParcel(out: Parcel, flags: Int) {
super.writeToParcel(out, flags)
out.writeParcelable(resultGame, flags)
}
```

```
companion object {
@JvmField
val CREATOR: Parcelable.Creator<SavedState> = object : Parcelable.Creator<SavedState> {
override fun createFromParcel(source: Parcel): SavedState {
return SavedState(source)
}
override fun newArray(size: Int): Array<SavedState?> {
return Array(size) { null }
}
}
}
}
}
```

ДОДАТОК Д

Код для створення утиліт програми

Файл utils/ AvatarManager.kt:

```
package com.example.tictactoe.utils
12 import android.graphics.Bitmap
12 import android.graphics.Canvas
12 import android.graphics.Color
12 import android.graphics.Paint
12 import android.graphics.drawable.Drawable
import android.widget.ImageView
import androidx.core.graphics.createBitmap
```

```
import androidx.core.graphics.drawable.toDrawable
import com.bumptech.glide.Glide
import com.bumptech.glide.request.target.CustomTarget
import com.bumptech.glide.request.transition.Transition
import com.example.tictactoe.data.Profile
class AvatarManager(private val profile: Profile) {
fun setAvatar(imageView: ImageView) {
if (profile.avatarUri == null) {
val nickname = profile.nickname
val letter = if (nickname.isEmpty()) {
" "
} else {
nickname[0].toString()
}
val textDrawable = createTextDrawable(
letter,
profile.selectedColor?.defaultColor ?: Color.RED,
imageView
)
imageView.setImageDrawable(textDrawable)
} else {
Glide.with(imageView.context)
.asDrawable()
.load(profile.avatarUri)
```

```
.circleCrop()
.into(object : CustomTarget<Drawable>() {
override fun onResourceReady(

resource: Drawable,
transition: Transition<in Drawable>?
) {
imageView.setImageDrawable(resource)
}
override fun onLoadCleared(placeholder: Drawable?) {}
})
}
}
private fun createTextDrawable(letter: String, color: Int, imageView: ImageView): Drawable {
val size = if (imageView.width > 0) imageView.width else 200
val bitmap = createBitmap(size, size)
val canvas = Canvas(bitmap)
val paint = Paint(Paint.ANTI_ALIAS_FLAG)
paint.color = color
canvas.drawCircle(size / 2f, size / 2f, size / 2f, paint)
paint.color = Color.WHITE
paint.textSize = size * 0.4f
paint.textAlign = Paint.Align.CENTER
val textHeight = paint.descent() - paint.ascent()
```

```
val textOffset = textHeight / 2 - paint.descent()
canvas.drawText(letter, size / 2f, size / 2f + textOffset, paint)
return bitmap.toDrawable(imageView.context.resources)
}
fun createTextBitmap(): Bitmap {
val size = 200
val bitmap = createBitmap(size, size)
val canvas = Canvas(bitmap)
val paint = Paint(Paint.ANTI_ALIAS_FLAG).apply {
color = profile.selectedColor?.defaultColor ?: Color.GRAY
}
canvas.drawCircle(size / 2f, size / 2f, size / 2f, paint)
val letter = profile.nickname.firstOrNull()?.uppercase() ?: "?"
paint.color = Color.WHITE
paint.textSize = size * 0.4f
paint.textAlign = Paint.Align.CENTER
val textHeight = paint.descent() - paint.ascent()
val textOffset = textHeight / 2 - paint.descent()
canvas.drawText(letter, size / 2f, size / 2f + textOffset, paint)
return bitmap
}
}
```

Файл utils/ BitmapConverter.kt:

```
package com.example.tictactoe.utils
```

```
import android.content.ContentResolver
12 import android.graphics.Bitmap
12 import android.graphics.BitmapFactory
import android.graphics.Canvas
import android.graphics.ImageDecoder
import android.graphics.drawable.VectorDrawable
6 import android.net.Uri
6 import android.os.Build
import android.provider.MediaStore
import androidx.annotation.RequiresApi
import androidx.core.graphics.createBitmap
import java.io.ByteArrayOutputStream
import java.io.IOException
internal fun bitmapToByteArray(bitmap: Bitmap): ByteArray {
val stream = ByteArrayOutputStream()
bitmap.compress(Bitmap.CompressFormat.PNG, 100, stream)
return stream.toByteArray()
}
internal fun byteArrayToBitmap(byteArray: ByteArray): Bitmap {
return BitmapFactory.decodeByteArray(byteArray, 0, byteArray.size)
}
@Suppress("DEPRECATION")
private fun getBitmapLegacy(contentResolver: ContentResolver, fileUri: Uri): Bitmap? {
var bitmap: Bitmap? = null
```

```
try {
bitmap = MediaStore.Images.Media.getBitmap(contentResolver, fileUri)
} catch (e: IOException) {
e.printStackTrace()
}
return bitmap
}
@RequiresApi(Build.VERSION_CODES.P)
private fun getBitmapImageDecoder(contentResolver: ContentResolver, fileUri: Uri): Bitmap?
{
var bitmap: Bitmap? = null
try {
bitmap = ImageDecoder.decodeBitmap(ImageDecoder.createSource(contentResolver,
fileUri))
} catch (e: IOException) {
e.printStackTrace()
}
return bitmap
}
internal fun uriToBitmap(contentResolver: ContentResolver, fileUri: Uri?): Bitmap? {
if (fileUri == null) {
return null
}
return if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.P) {
getBitmapImageDecoder(contentResolver, fileUri)
```

```
} else {
getBitmapLegacy(contentResolver, fileUri)
}
}
fun vectorDrawableToBitmap(vectorDrawable: VectorDrawable): Bitmap {
val bitmap = createBitmap(vectorDrawable.intrinsicWidth, vectorDrawable.intrinsicHeight)
val canvas = Canvas(bitmap)
vectorDrawable.setBounds(0, 0, canvas.width, canvas.height)
vectorDrawable.draw(canvas)
return bitmap
}
```

Файл utils/ SnackBarUtils.kt:

```
package com.example.tictactoe.utils
import android.graphics.Typeface
12 import android.util.Log
6 import android.view.View
6 import android.view.ViewGroup
import android.widget.TextView
import androidx.core.content.ContextCompat
import com.example.tictactoe.R
import com.google.android.material.snackbar.Snackbar
object SnackBarUtils {
fun showCustomSnackBar(
view: View, message: String, actionText: String, backgroundColor: Int
```

```
) {
val snackbar = Snackbar.make(view, message, Snackbar.LENGTH_LONG)
.setAction(actionText) { Log.d("Snackbar", "Action clicked") }
.setBackgroundTint(backgroundColor)
.setActionTextColor(ContextCompat.getColor(view.context, R.color.white))
.setTextColor(ContextCompat.getColor(view.context, R.color.white))
val snackbarView = snackbar.view
val textView =
snackbarView.findViewById<TextView>(com.google.android.material.R.id.snackbar_text)
textView.apply {
textSize = 18f
setTypeface(typeface, Typeface.BOLD)
}
val params = snackbarView.layoutParams as ViewGroup.MarginLayoutParams

params.setMargins(30, 30, 30, 50)
snackbarView.layoutParams = params
val actionTextView =
snackbarView.findViewById<TextView>(com.google.android.material.R.id.snackbar_action)
actionTextView.apply {
textSize = 18f
setTypeface(typeface, Typeface.BOLD)
}
snackbar.show()
```

}

}

ДОДАТОК Е

Код для представлення внутрішніх даних

Файл data/ EntityCard.kt:

```
package com.example.tictactoe.data

12 import android.graphics.Bitmap

12 import android.os.Parcel

import android.os.Parcelable

import com.example.tictactoe.utils.bitmapToByteArray

import com.example.tictactoe.utils.byteArrayToBitmap

class EntityCard(val name: String, val description: String, val avatar: Bitmap?, val mark: Int) :

Parcelable {

private val avatarData: ByteArray = avatar?.let { bitmapToByteArray(it) } ?: ByteArray(0)

constructor(parcel: Parcel) : this(

name = parcel.readString() ?: "",

description = parcel.readString() ?: "",

avatar = parcel.createByteArray()

?.let { if (it.isNotEmpty()) byteArrayToBitmap(it) else null },

mark = parcel.readInt()

)

override fun writeToParcel(parcel: Parcel, flags: Int) {

parcel.writeString(name)

parcel.writeString(description)
```

```
parcel.writeByteArray(avatarData)
parcel.writeInt(mark)
}
override fun describeContents(): Int = 0
companion object CREATOR : Parcelable.Creator<EntityCard> {
override fun createFromParcel(parcel: Parcel): EntityCard = EntityCard(parcel)
override fun newArray(size: Int): Array<EntityCard?> = arrayOfNulls(size)
}
}
```

Файл data/ Point.kt:

```
package com.example.tictactoe.data
data class Point(val x: Float, val y: Float)
```

Файл data/Profile.kt:

```
package com.example.tictactoe.data
import android.content.Context
import android.content.res.ColorStateList
6 import android.net.Uri
6 import com.example.tictactoe.R
class Profile(var nickname: String, var avatarUri: Uri?, var selectedColor: ColorStateList?) {
companion object {
fun default(context: Context): Profile = Profile(
"Player", null, ColorStateList.valueOf(context.getColor(R.color.background_red))
)
}
```

```
}
```

Файл data/ResultGame.kt:

```
package com.example.tictactoe.data
12 import android.graphics.Bitmap
12 import android.os.Parcel
import android.os.Parcelable
import com.example.tictactoe.utils.bitmapToByteArray
import com.example.tictactoe.utils.byteArrayToBitmap
class ResultGame(val avatarBitmap: Bitmap?, val result: String) : Parcelable {
private val bitmapData: ByteArray? = avatarBitmap?.let { bitmapToByteArray(it) }
constructor(parcel: Parcel) : this(
parcel.createByteArray()?.let { byteArrayToBitmap(it) },
parcel.readString() ?: ""
)
override fun writeToParcel(parcel: Parcel, flags: Int) {
parcel.writeByteArray(bitmapData)
parcel.writeString(result)
}
override fun describeContents(): Int = 0
companion object CREATOR : Parcelable.Creator<ResultGame> {
override fun createFromParcel(parcel: Parcel): ResultGame = ResultGame(parcel)
override fun newArray(size: Int): Array<ResultGame?> = arrayOfNulls(size)
}
}
```

Файл data/Settings.kt:

```
package com.example.tictactoe.data

class Settings {

private var _randomMark: Boolean

private var _randomMove: Boolean

private var _humanMove: Boolean

private var _humanTic: Boolean

constructor() {

_randomMark = true

_randomMove = true

_humanMove = false

_humanTic = false

}

var randomMark: Boolean

get() = _randomMark

set(value) {

_randomMark = value

_humanTic = false

}

var randomMove: Boolean

get() = _randomMove

set(value) {

_randomMove = value

_humanMove = false
```

```
}
var humanMove: Boolean
get() = _humanMove
set(value) {
_humanMove = value
_randomMove = false
}
var humanTic: Boolean
get() = _humanTic
set(value) {
_humanTic = value
_randomMark = false
}
}
```

ДОДАТОК Ж

Код для реалізації фрагментів інтерфейсу

Файл fragments/ BotGameFragment.kt:

```
package com.example.tictactoe.fragments
import android.content.Context
import android.content.SharedPreferences
12 import android.graphics.Bitmap
12 import android.graphics.BitmapFactory
import android.graphics.drawable.VectorDrawable
6 import android.os.Bundle
```

```
6 import android.view.LayoutInflater
6 import android.view.View
6 import android.view.ViewGroup
import androidx.annotation.DrawableRes
import androidx.annotation.StringRes
import androidx.core.content.ContextCompat
import androidx.fragment.app.Fragment
import androidx.lifecycle.lifecycleScope
import com.bumptech.glide.Glide
import com.example.tictactoe.data.EntityCard
import com.example.tictactoe.customViews.GameState
import com.example.tictactoe.data.Profile
import com.example.tictactoe.R

import com.example.tictactoe.data.ResultGame
import com.example.tictactoe.data.Settings
import com.example.tictactoe.ai.Board
import com.example.tictactoe.ai.Mark
import com.example.tictactoe.ai.Minimax
import com.example.tictactoe.contract.HasCustomTitle
import com.example.tictactoe.contract.navigator
import com.example.tictactoe.databinding.FragmentBotGameBinding
import com.example.tictactoe.extensions.getObject
import com.example.tictactoe.utils.AvatarManager
```

```
import com.example.tictactoe.utils.vectorDrawableToBitmap

import kotlinx.coroutines.Dispatchers

import kotlinx.coroutines.launch

import kotlinx.coroutines.withContext

import kotlin.properties.Delegates

import kotlin.random.Random

class BotGameFragment : Fragment(), HasCustomTitle {

private lateinit var binding: FragmentBotGameBinding

private var levelDifficulty: Int by Delegates.notNull()

private lateinit var profile: Profile

private lateinit var settings: Settings

private lateinit var sharedPref: SharedPreferences

private lateinit var minimax: Minimax

private val board: Board = Board.empty()

private lateinit var humanMark: Mark

private lateinit var computerMark: Mark

private var isComputerThinking = false

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

sharedPref = requireContext().getSharedPreferences(APP_PREFERENCES,
Context.MODE_PRIVATE)

profile = sharedPref.getObject(KEY_PROFILE, Profile.default(requireContext()))

settings = sharedPref.getObject(KEY_SETTINGS, Settings())

}
```

```
11 override fun onCreateView(
11 inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?
11 ): View {
binding = FragmentBotGameBinding.inflate(inflater, container, false)
levelDifficulty = arguments?.getInt(ARG_LEVEL) ?: 1
setupBotAppearance()
setupProfile()
setupSettings()
return binding.root
}
override fun onDestroy() {
super.onDestroy()
isComputerThinking = false
}
private fun setupBotAppearance() {
val (imageRes, infoRes) = when (levelDifficulty) {
1 -> R.drawable.easy_bot to R.string.easy_bot
2 -> R.drawable.medium_bot to R.string.medium_bot
3 -> R.drawable.difficult_bot to R.string.difficult_bot
else -> R.drawable.easy_bot to R.string.easy_bot
}
setImage(imageRes)
setInfo(infoRes)
}
```

```
private fun setupProfile() {
binding.humanEntityCard.name = profile.nickname
AvatarManager(profile).setAvatar(binding.humanEntityCard.ivAvatar)
}
private fun setupSettings() {
humanMark = determineHumanMark()
computerMark = if (humanMark == Mark.TIC) Mark.TAC else Mark.TIC
setupBoard()
startGame()
}
private fun determineHumanMark(): Mark {
return when {
settings.randomMark -> if (Random.nextBoolean()) Mark.TIC else Mark.TAC
settings.humanTic -> Mark.TIC
else -> Mark.TAC
}
}
private fun setupBoard() {
val tic = R.drawable.tic
val tac = R.drawable.tac
with(binding) {
if (humanMark == Mark.TIC) {
humanEntityCard.mark = tic
botEntityCard.mark = tac
```

```
} else {
humanEntityCard.mark = tac
botEntityCard.mark = tic
}
cvBoard.humanMark =
if (humanMark == Mark.TIC) R.drawable.cell_tic else R.drawable.cell_tac
minimax = Minimax(board, computerMark, levelDifficulty * 4)
cvBoard.addMoveListener { r, c ->
if (!isComputerThinking) {
board[r, c] = humanMark
cvBoard.setMove(r, c, humanMark)
computerTurn()
}
}
cvBoard.addGameStateListener { state ->
showGameOverScreen(state)
binding.humanEntityCard.active = false
binding.botEntityCard.active = false
}
}
}
private fun showGameOverScreen(state: GameState) {
val result = when (state) {
GameState.HUMAN_WIN -> getString(R.string.you_win)
```

```
GameState.AI_WIN -> getString(R.string.you_lose)

GameState.DRAW -> getString(R.string.draw)

GameState.ONGOING -> return

}

lifecycleScope.launch {

val humanAvatar = getHumanAvatar()

val botAvatar = getBotAvatar()

val humanEntityCard = EntityCard(

name = profile.nickname,

description = "You",

mark = if (humanMark == Mark.TIC) R.drawable.tic else R.drawable.tac,

avatar = humanAvatar

)

val botEntityCard = EntityCard(

name = when (levelDifficulty) {

1 -> getString(R.string.easy_bot)

2 -> getString(R.string.medium_bot)

3 -> getString(R.string.difficult_bot)

else -> getString(R.string.easy_bot)

},

description = "Opponent",

mark = if (computerMark == Mark.TIC) R.drawable.tic else R.drawable.tac,

avatar = botAvatar

)
```

```
val resultGame = ResultGame(avatarBitmap = humanAvatar, result = result)
navigator().showGameOverScreen(
humanEntityCard,
botEntityCard,
resultGame
) {
navigator().showBotGameScreen(levelDifficulty)

}
}
}
private suspend fun getHumanAvatar(): Bitmap? = withContext(Dispatchers.IO) {
return@withContext if (profile.avatarUri != null) {
try {
Glide.with(requireContext())
.asBitmap()
.circleCrop()
.load(profile.avatarUri)
.submit()
.get()
} catch (_: Exception) {
AvatarManager(profile).createTextBitmap()
}
} else {
```

```
AvatarManager(profile).createTextBitmap()
}
}
private suspend fun getBotAvatar(): Bitmap = withContext(Dispatchers.IO) {
try {
val botResId = when (levelDifficulty) {
1 -> R.drawable.easy_bot
2 -> R.drawable.medium_bot
3 -> R.drawable.difficult_bot
else -> R.drawable.easy_bot
}
val vectorDrawable =
ContextCompat.getDrawable(requireContext(), botResId) as VectorDrawable
vectorDrawableToBitmap(vectorDrawable)
} catch (_: Exception) {
BitmapFactory.decodeResource(resources, R.drawable.profile_avatar)
}
}
private fun startGame() {
if (getHumanMove()) {
humanTurn()
} else {
computerTurn()
}
```

```
}
private fun getHumanMove(): Boolean {
return when {
settings.randomMove -> Random.nextBoolean()
settings.humanMove -> true
else -> false
}
}
private fun humanTurn() {
binding.humanEntityCard.active = true
binding.botEntityCard.active = false
}
private fun computerTurn() {
if (isComputerThinking || isBoardFull()) return
isComputerThinking = true
binding.humanEntityCard.active = false
binding.botEntityCard.active = true
lifecycleScope.launch {
val move = withContext(Dispatchers.Default) {
minimax.findBestMove()
}
if (!isComputerThinking) return@launch
board[move.first, move.second] = computerMark
binding.cvBoard.setMove(move.first, move.second, computerMark)
```

```
binding.botEntityCard.active = false

isComputerThinking = false

if (binding.cvBoard.currentGameState == GameState.ONGOING) {

humanTurn()

}

}

}

private fun isBoardFull(): Boolean {

for (i in 0 until 3) {

for (j in 0 until 3) {

if (board[i, j] == Mark.EMPTY) {

return false

}

}

}

return true

}

private fun setImage(@DrawableRes imageResId: Int) {

binding.botEntityCard.avatarResId = imageResId

}

private fun setInfo(@StringRes stringResId: Int) {

binding.botEntityCard.name = getString(stringResId)

}

override fun getTitleRes(): Int = R.string.toolbar_bot
```

```
companion object {
private const val ARG_LEVEL = "levelDifficulty"
fun newInstance(level: Int): BotGameFragment {
return BotGameFragment().apply {
arguments = Bundle().apply {
putInt(ARG_LEVEL, level)
}
}
}
}
}
```

Файл fragments/ DifficultyFragment.kt:

```
package com.example.tictactoe.fragments
6 import android.os.Bundle
6 import android.view.LayoutInflater
6 import android.view.View
6 import android.view.ViewGroup
import androidx.fragment.app.Fragment
import com.example.tictactoe.R
import com.example.tictactoe.contract.CustomAction
import com.example.tictactoe.contract.HasCustomAction
import com.example.tictactoe.contract.HasCustomTitle
import com.example.tictactoe.contract.navigator
import com.example.tictactoe.databinding.FragmentDifficultyBinding
```

```
class DifficultyFragment : Fragment(), HasCustomTitle, HasCustomAction {

private lateinit var binding: FragmentDifficultyBinding

41 override fun onCreateView(

41 inflater: LayoutInflater,

41 container: ViewGroup?,

41 savedInstanceState: Bundle?

41 ): View {

binding = FragmentDifficultyBinding.inflate(inflater, container, false)

binding.btEasy.setOnClickListener {

navigator().showBotGameScreen(1)

}

binding.btMedium.setOnClickListener {

navigator().showBotGameScreen(2)

}

binding.btDifficult.setOnClickListener {

navigator().showBotGameScreen(3)

}

return binding.root

}

override fun getTitleRes(): Int = R.string.toolbar_difficulty

override fun getCustomAction(): CustomAction = CustomAction(

R.drawable.ic_settings,

R.string.settings

) { navigator().showSettingsMenuScreen() }
```

```
}
```

Файл fragments/ EditProfileFragment.kt:

```
package com.example.tictactoe.fragments

import android.content.Context
6 import android.content.Intent
6 import android.content.SharedPreferences
6 import android.os.Bundle
6 import android.text.Editable
import android.text.TextWatcher
import android.view.LayoutInflater
6 import android.view.View
6 import android.view.ViewGroup
import android.widget.ImageView
import androidx.activity.result.PickVisualMediaRequest
import androidx.activity.result.contract.ActivityResultContracts.PickVisualMedia
import androidx.core.content.ContextCompat
import androidx.fragment.app.Fragment
import com.example.tictactoe.data.Profile
import com.example.tictactoe.R
import com.example.tictactoe.contract.CustomAction

import com.example.tictactoe.contract.HasCustomAction
import com.example.tictactoe.contract.HasCustomTitle
import com.example.tictactoe.databinding.FragmentEditProfileBinding
```

```
import com.example.tictactoe.extensions.getObject
import com.example.tictactoe.extensions.putObject
import com.example.tictactoe.utils.AvatarManager
import com.example.tictactoe.utils.SnackBarUtils
const val APP_PREFERENCES = "APP_PREFERENCES"
const val KEY_PROFILE = "KEY_PROFILE"
class EditProfileFragment : Fragment(), HasCustomTitle, HasCustomAction {
private lateinit var sharedPref: SharedPreferences
private lateinit var profile: Profile
private lateinit var binding: FragmentEditProfileBinding
private val pickMedia = registerForActivityResult(PickVisualMedia()) { uri ->
uri?.let {
requireContext().contentResolver.takePersistableUriPermission(
it,
Intent.FLAG_GRANT_READ_URI_PERMISSION
)
unFocusAllViews()
profile.avatarUri = it
profile.selectedColor = null
setAvatar()
}
}
private val textWatcher = object : TextWatcher {
override fun onTextChanged(s: 51 CharSequence?, start: Int, before: Int, count: Int) {
```

```
updateUi()
}
25 override fun afterTextChanged(p0: Editable?) {}
25 override fun beforeTextChanged(p0: CharSequence?, p1: Int, p2: Int, p3: Int) {}
}
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
sharedPref = requireContext().getSharedPreferences(APP_PREFERENCES,
Context.MODE_PRIVATE)
profile = sharedPref.getObject(KEY_PROFILE, Profile.default(requireContext()))
}
11 override fun onCreateView(
11 inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?
11 ): View {
binding = FragmentEditProfileBinding.inflate(inflater, container, false)
binding.etNickname.addTextChangedListener(textWatcher)
getImageViews().forEach { setupColorSelection(it) }
binding.btSelectPhoto.setOnClickListener {
pickMedia.launch(
PickVisualMediaRequest(
PickVisualMedia.ImageOnly
)
)
}
```

```
binding.btSaveChanges.setOnClickListener { saveChanges() }
setupProfile()
return binding.root
}
private fun setupColorSelection(imageView: ImageView) {
imageView.setOnClickListener {
unFocusAllViews()
imageView.setImageDrawable(
ContextCompat.getDrawable(requireContext(), R.drawable.v_color_selected)
)
profile.selectedColor = imageView.backgroundTintList
profile.avatarUri = null
setAvatar()
}
}
private fun setAvatar() {
AvatarManager(profile).setAvatar(binding.ivHumanAvatar)
}
private fun unFocusAllViews() {
val iconDrawable = ContextCompat.getDrawable(requireContext(), R.drawable.v_color)
getImageViews().forEach { it.setImageDrawable(iconDrawable) }
}
private fun setupProfile() {
binding.etNickname.setText(profile.nickname)
```

```
setAvatar()
profile.selectedColor?.let { selectedColor ->
getImageViews().forEach { imageView ->
if (imageView.backgroundTintList?.defaultColor == selectedColor.defaultColor) {
imageView.performClick()
return
}
}
}
}
private fun getImageViews(): List<ImageView> = listOf(
binding.ivRed,
binding.ivGreen,
binding.ivOrange,
binding.ivPurple,
binding.ivYellow,
binding.ivBlue,
binding.ivPink,
binding.ivTeal
)
private fun saveChanges() {
if (binding.etNickname.error != null) {
SnackBarUtils.showCustomSnackBar(
binding.root, "The profile was not updated!", "OK",
```

```
ContextCompat.getColor(requireContext(), R.color.error_color)
)
return
}
with(sharedPref.edit()) {
putObject(KEY_PROFILE, profile)
apply()
}
SnackBarUtils.showCustomSnackBar(
binding.root, "Profile updated successfully!", "OK",
ContextCompat.getColor(requireContext(), R.color.success_color)
)
}
private fun updateUi() {
val nickname = binding.etNickname.text.toString().trim()
profile.nickname = nickname
binding.etNickname.error = if (nickname.length > 15) "Nickname is too long" else null
binding.tvNicknameSize.text = getString(R.string.char_count_tv, nickname.length)
setAvatar()
}
override fun getTitleRes(): Int = R.string.toolbar_edit_profile
override fun getCustomAction(): CustomAction {
return CustomAction(R.drawable.ic_done, R.string.done) { saveChanges() }
}
```

}

Файл fragments/ GameOverFragment.kt:

package com.example.tictactoe.fragments

38 import android.os.Build

38 import android.os.Bundle

38 import android.os.Parcelable

38 import android.view.LayoutInflater

38 import android.view.View

38 import android.view.ViewGroup

import androidx.activity.OnBackPressedCallback

import androidx.fragment.app.Fragment

import com.example.tictactoe.data.EntityCard

import com.example.tictactoe.R

import com.example.tictactoe.data.ResultGame

import com.example.tictactoe.contract.HasCustomTitle

import com.example.tictactoe.databinding.FragmentGameOverBinding

typealias OnRefreshClick = () -> Unit

class GameOverFragment : Fragment(), HasCustomTitle {

private lateinit var binding: FragmentGameOverBinding

var onRefreshClick: OnRefreshClick? = null

31 override fun onCreateView(

31 inflater: LayoutInflater,

31 container: ViewGroup?,

31 savedInstanceState: Bundle?

31): 31 View {

31 binding = FragmentGameOverBinding.inflate(inflater, 31 container, false)

31 val humanEntityCard =

arguments.getParcelableCompat(HUMAN_ENTITY_CARD_KEY, EntityCard::class.java)

val botEntityCard =

arguments.getParcelableCompat(BOT_ENTITY_CARD_KEY, EntityCard::class.java)

val resultGame = arguments.getParcelableCompat(RESULT_GAME_KEY,
ResultGame::class.java)

humanEntityCard?.let { binding.humanEntityCard.loadEntityCard(it) }

botEntityCard?.let { binding.botEntityCard.loadEntityCard(it) }

resultGame?.let { binding.resultGame.loadResultGame(it) }

binding.btRefresh.setOnClickListener {

parentFragmentManager.popBackStack()

parentFragmentManager.popBackStack()

onRefreshClick?.invoke()

}

return binding.root

}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {

super.onViewCreated(view, savedInstanceState)

requireActivity().onBackPressedDispatcher.addCallback(

viewLifecycleOwner,

object : OnBackPressedCallback(true) {

```
override fun handleOnBackPressed() {
parentFragmentManager.popBackStack()
parentFragmentManager.popBackStack()
}
})
}
private fun <T : Parcelable> Bundle?.getParcelableCompat(key: String, clazz: Class<T>): T? {
return if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
this?.getParcelable(key, clazz)
} else {
@Suppress("DEPRECATION")
this?.getParcelable(key)
}
}
override fun getTitleRes(): Int {
return R.string.toolbar_game_result
}
companion object {
private const val RESULT_GAME_KEY = "resultGame"
private const val HUMAN_ENTITY_CARD_KEY = "humanEntityCard"
private const val BOT_ENTITY_CARD_KEY = "botEntityCard"
fun newInstance(
humanEntityCard: EntityCard,
botEntityCard: EntityCard,
```

```
resultGame: ResultGame,
): GameOverFragment {
val fragment = GameOverFragment()
val args = Bundle().apply {
putParcelable(HUMAN_ENTITY_CARD_KEY, humanEntityCard)
putParcelable(BOT_ENTITY_CARD_KEY, botEntityCard)
putParcelable(RESULT_GAME_KEY, resultGame)
}
fragment.arguments = args
return fragment
}
}
}
```

Файл fragments/ LocalMultiplayerFragment.kt:

```
package com.example.tictactoe.fragments
import android.content.Context
import android.content.SharedPreferences
import android.content.res.ColorStateList
import android.graphics.Bitmap
import android.graphics.BitmapFactory
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
```

```
import 5 androidx.fragment.app.Fragment
import androidx.lifecycle.lifecycleScope
import com.bumptech.glide.Glide
import com.example.tictactoe.data.EntityCard
import com.example.tictactoe.customViews.GameState
import com.example.tictactoe.data.Profile
import com.example.tictactoe.R
import com.example.tictactoe.data.ResultGame
import com.example.tictactoe.ai.Mark
import com.example.tictactoe.contract.HasCustomTitle
import com.example.tictactoe.contract.navigator
import com.example.tictactoe.databinding.FragmentLocalMultiplayerBinding
import com.example.tictactoe.extensions.getObject
import com.example.tictactoe.utils.AvatarManager
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.launch
import kotlinx.coroutines.withContext
import kotlin.random.Random
class LocalMultiplayerFragment : Fragment(), HasCustomTitle {
5 private lateinit var binding: 5 FragmentLocalMultiplayerBinding
5 private lateinit var profile: 5 Profile
5 private lateinit var opponentProfile: 5 Profile
5 private lateinit var sharedPref: SharedPreferences
private var humanMove = Random.nextBoolean()
```

5 private lateinit var humanMark: 5 Mark

5 private lateinit var opponentMark: Mark

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

sharedPref = requireContext().getSharedPreferences(APP_PREFERENCES, Context.MODE_PRIVATE)

profile = sharedPref.getObject(KEY_PROFILE, Profile.default(requireContext()))

opponentProfile = Profile(

"Opponent",

null,

ColorStateList.valueOf(requireContext().getColor(R.color.background_red))

)

}

11 override fun onCreateView(

11 inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?

11): View {

binding = FragmentLocalMultiplayerBinding.inflate(inflater, container, false)

setupProfile()

setupOpponentProfile()

determinateMark()

binding.humanEntityCard.mark = if (humanMark == Mark.TIC) R.drawable.tic else R.drawable.tac

binding.opponentEntityCard.mark =

if (opponentMark == Mark.TIC) R.drawable.tic else R.drawable.tac

```
setupBoard()

startGame()

return binding.root

}

private fun setupProfile() {

binding.humanEntityCard.name = profile.nickname

AvatarManager(profile).setAvatar(binding.humanEntityCard.ivAvatar)

}

private fun setupOpponentProfile() {

binding.opponentEntityCard.name = getString(R.string.opponent)

AvatarManager(opponentProfile).setAvatar(binding.opponentEntityCard.ivAvatar)

}

private fun determinateMark() {

if (Random.nextBoolean()) {

humanMark = Mark.TIC

opponentMark = Mark.TAC

} else {

humanMark = Mark.TAC

opponentMark = Mark.TIC

}

}

private fun setupBoard() {

with(binding) {

cvBoard.humanMark =
```

```
if (humanMark == Mark.TIC) R.drawable.cell_tic else R.drawable.cell_tac
cvBoard.addMoveListener { r, c ->
cvBoard.setMove(r, c, if (humanMove) humanMark else opponentMark)
humanMove = !humanMove
if (humanMove) {
humanTurn()
} else {
opponentTurn()
}
}
cvBoard.addGameStateListener { state ->
showGameOverScreen(state)
binding.humanEntityCard.active = false
binding.opponentEntityCard.active = false
}
}
}
private fun humanTurn() {
binding.humanEntityCard.active = true
binding.opponentEntityCard.active = false
}
private fun opponentTurn() {
binding.humanEntityCard.active = false
binding.opponentEntityCard.active = true
```

```
}

private fun showGameOverScreen(state: GameState) {

val result = when (state) {

GameState.HUMAN_WIN -> getString(R.string.you_win)

GameState.AI_WIN -> getString(R.string.you_lose)

GameState.DRAW -> getString(R.string.draw)

GameState.ONGOING -> return

}

lifecycleScope.launch {

val humanAvatar = getHumanAvatar()

val opponentAvatar = getOpponentAvatar()

val humanEntityCard = EntityCard(

name = profile.nickname,

description = "You",

mark = if (humanMark == Mark.TIC) R.drawable.tic else R.drawable.tac,

avatar = humanAvatar

)

val opponentEntityCard = EntityCard(

name = getString(R.string.opponent),

description = "Opponent",

mark = if (opponentMark == Mark.TIC) R.drawable.tic else R.drawable.tac,

avatar = opponentAvatar

)

val resultGame = ResultGame(avatarBitmap = humanAvatar, result = result)
```

```
navigator().showGameOverScreen(
humanEntityCard,
opponentEntityCard,
resultGame
) {
navigator().showLocalMultiplayerScreen()
}
}
}
private suspend fun getHumanAvatar(): Bitmap? = withContext(Dispatchers.IO) {
return@withContext if (profile.avatarUri != null) {
try {
Glide.with(requireContext())
.asBitmap()
.circleCrop()
.load(profile.avatarUri)
.submit()
.get()
} catch (_: Exception) {
AvatarManager(profile).createTextBitmap()
}
} else {
AvatarManager(profile).createTextBitmap()
}
```

```
}
private suspend fun getOpponentAvatar(): Bitmap = withContext(Dispatchers.IO) {
try {
AvatarManager(opponentProfile)
.createTextBitmap()
} catch (_: Exception) {
BitmapFactory.decodeResource(resources, R.drawable.profile_avatar)
}
}
private fun startGame() {
if (humanMove) {
humanTurn()
} else {
opponentTurn()
}
}
override fun getTitleRes(): Int = R.string.toolbar_local_multiplayer
}
```

Файл fragments/ MenuFragment.kt:

```
package com.example.tictactoe.fragments
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
```

```
import 5 androidx.fragment.app.Fragment
import com.example.tictactoe.R
import com.example.tictactoe.contract.CustomAction
import com.example.tictactoe.contract.HasCustomAction
import com.example.tictactoe.contract.navigator
import com.example.tictactoe.databinding.FragmentMenuBinding
class MenuFragment : Fragment(), HasCustomAction {
5 private lateinit var binding: FragmentMenuBinding
override fun onCreateView(
inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?
): View {
binding = FragmentMenuBinding.inflate(inflater, container, false)
binding.btSinglePlayer.setOnClickListener {
navigator().showDifficultyScreen()
}
binding.btMultiplayer.setOnClickListener { }
binding.btLocalMultiplayer.setOnClickListener {
navigator().showLocalMultiplayerScreen()
}
return binding.root
}
override fun getCustomAction(): CustomAction {
return CustomAction(
```

```
iconRes = R.drawable.ic_profile,
textRes = R.string.player_avatar,
onCustomAction = { navigator().showEditProfileScreen() })
}
}
```

Файл fragments/ SettingsFragment.kt:

```
package com.example.tictactoe.fragments
import android.content.Context
import android.content.SharedPreferences
import android.content.res.ColorStateList
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import 5 androidx.annotation.ColorRes
import androidx.annotation.DrawableRes
import androidx.core.content.ContextCompat
5 import androidx.fragment.app.Fragment
5 import com.example.tictactoe.data.Profile
import com.example.tictactoe.R
import com.example.tictactoe.data.Settings
import com.example.tictactoe.contract.CustomAction
import com.example.tictactoe.contract.HasCustomAction
import com.example.tictactoe.contract.HasCustomTitle
```

```
import com.example.tictactoe.contract.navigator

import com.example.tictactoe.databinding.FragmentSettingsBinding

import com.example.tictactoe.extensions.getObject

import com.example.tictactoe.extensions.putObject

import com.example.tictactoe.utils.AvatarManager

import com.example.tictactoe.utils.SnackBarUtils

const val KEY_SETTINGS = "KEY_SETTINGS"

class SettingsFragment : Fragment(), HasCustomTitle, HasCustomAction {

5 private lateinit var sharedPref: 5 SharedPreferences

5 private lateinit var profile: 5 Profile

5 private lateinit var settings: 5 Settings

5 private lateinit var binding: FragmentSettingsBinding

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

sharedPref = requireContext().getSharedPreferences(APP_PREFERENCES,
Context.MODE_PRIVATE)

profile = sharedPref.getObject(

KEY_PROFILE, Profile.default(requireContext())

)

settings = sharedPref.getObject(KEY_SETTINGS, Settings())

}

override fun onCreateView(

inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?

): View? {
```

```
binding = FragmentSettingsBinding.inflate(inflater, container, false)
AvatarManager(profile).setAvatar(binding.ivHumanAvatar)
binding.btSave.setOnClickListener { save() }
binding.btCancel.setOnClickListener { navigator().goBack() }
binding.vItemRandomMark.setOnClickListener { selectRandomMark() }
binding.vItemTic.setOnClickListener { selectTic() }
binding.vItemTac.setOnClickListener { selectTac() }
binding.vItemRandomMove.setOnClickListener { selectRandomMove() }
binding.vItemComputer.setOnClickListener { selectMoveComputer() }
binding.vItemHuman.setOnClickListener { selectMoveHuman() }
setupSettings()
return binding.root
}
private fun updateView(
view: View, @DrawableRes backgroundRes: Int
) {
view.apply {
backgroundTintList = null
setBackgroundResource(backgroundRes)
}
}
private fun selectTic() {
settings.humanTic = true
unSelectMarkViews()
```

```
with(binding) {
selectView(vItemTic, vTicBg, R.drawable.v_selected_tic, R.color.earthy_brown)
}
}
private fun selectTac() {
settings.humanTic = false
unSelectMarkViews()
with(binding) {
selectView(vItemTac, vTacBg, R.drawable.v_selected_tac, R.color.deep_amethyst)
}
}
private fun selectRandomMark() {
settings.randomMark = true
unSelectMarkViews()
with(binding) {
selectView(
vItemRandomMark, vRandMarkBg, R.drawable.v_selected_random, R.color.midnight_blue
)
}
}
private fun selectRandomMove() {
settings.randomMove = true
unSelectMoveViews()
with(binding) {
```

```
selectView(
vItemRandomMove,
vRandMoveBg,
R.drawable.v_selected_random,
R.color.midnight_blue
)
}
}
private fun selectMoveComputer() {
settings.humanMove = false
unSelectMoveViews()
with(binding) {
selectView(
vItemComputer,
vComputerBg,
R.drawable.v_selected_random,
R.color.midnight_blue
)
}
}
private fun selectMoveHuman() {
settings.humanMove = true
unSelectMoveViews()
with(binding) {
```

```
selectView(
vItemHuman,
vHumanAvatarBg,
R.drawable.v_selected_random,
R.color.midnight_blue
)
}
}
private fun selectView(
view: View, iconView: View, @DrawableRes backgroundRes: Int, @ColorRes iconTintColorRes: Int
) {
updateView(view, backgroundRes)
updateIconTint(iconView, iconTintColorRes)
}
private fun unSelectMarkViews() {
val markViews = arrayOf(binding.vItemRandomMark, binding.vItemTic, binding.vItemTac)
for (markView in markViews) {
restoreView(markView)
}
updateIconTint(binding.vRandMarkBg, R.color.deep_navy)
updateIconTint(binding.vTicBg, R.color.smoky_brown)
updateIconTint(binding.vTacBg, R.color.raspberry_purple)

}
```

```
private fun unSelectMoveViews() {
val firstMoveViews =
arrayOf(binding.vItemRandomMove, binding.vItemComputer, binding.vItemHuman)
for (firstMoveView in firstMoveViews) {
restoreView(firstMoveView)
}
updateIconTint(binding.vRandMoveBg, R.color.deep_navy)
updateIconTint(binding.vComputerBg, R.color.deep_navy)
updateIconTint(binding.vHumanAvatarBg, R.color.deep_navy)
}
private fun restoreView(
view: View,
) {
view.setBackgroundResource(R.drawable.v_board)
view.backgroundTintList = ColorStateList.valueOf(
ContextCompat.getColor(requireContext(), R.color.profile_background)
)
}
private fun updateIconTint(view: View, @ColorRes colorRes: Int) {
view.backgroundTintList = ColorStateList.valueOf(
ContextCompat.getColor(requireContext(), colorRes)
)
}
private fun save() {
```

```
with(sharedPref.edit()) {
putObject(KEY_SETTINGS, settings)
apply()
}
SnackBarUtils.showCustomSnackBar(
binding.root,
"Settings updated successfully!",
"OK",
ContextCompat.getColor(requireContext(), R.color.success_color)
)
}
private fun setupSettings() {
when {
settings.randomMark -> selectRandomMark()
settings.humanTic -> selectTic()
else -> selectTac()
}
when {
settings.randomMove -> selectRandomMove()
settings.humanMove -> selectMoveHuman()
else -> selectMoveComputer()
}
}
override fun getTitleRes(): Int = R.string.toolbar_settings
```

```
override fun getCustomAction(): CustomAction =

CustomAction(R.drawable.ic_done, R.string.done) { save() }

}
```

ДОДАТОК К

Код для пошуку оптимальних стратегій гри

Файл ai/Board.kt:

```
package com.example.tictactoe.ai

@JvmInline

value class Board(private val grid: Array<Array<Mark>>) {

operator fun get(row: Int, col: Int): Mark = grid[row][col]

operator fun set(row: Int, col: Int, mark: Mark) {

if (row !in 0 until 3 || col !in 0 until 3) {

throw IllegalArgumentException("Invalid row or column index")

}

grid[row][col] = mark

}

fun getEmptyCells(): MutableList<Pair<Int, Int>> {

val emptyCells = mutableListOf<Pair<Int, Int>>()

for (r in grid.indices) {

for (c in grid[r].indices) {

if (grid[r][c] == Mark.EMPTY) {

emptyCells.add(Pair(r, c))

}

}
```

```
}
return emptyCells
}
fun getGrid(): Array<Array<Mark>> = grid
companion object {
fun empty(): Board = Board(Array(3) { Array(3) { Mark.EMPTY } })
}
}
```

Файл ai/Minimax.kt:

```
package com.example.tictactoe.ai
import kotlin.math.max
import kotlin.math.min
enum class Mark {
TIC, TAC, EMPTY
}
class Minimax(private val board: Board, private val human: Mark, private val maxDepth: Int) {
private val computer: Mark = if (human == Mark.TIC) Mark.TAC else Mark.TIC
private fun evaluate(): Int {
for (row in board.getGrid()) {
if (row[0] != Mark.EMPTY && row[0] == row[1] && row[1] == row[2]) {
return if (row[0] == computer) 10 else -10
}
}
for (c in 0 until board.getGrid().size) {
```

```
if (board[0, c] != Mark.EMPTY &&
board[0, c] == board[1, c] &&
board[1, c] == board[2, c]
) {
return if (board[0, c] == computer) 10 else -10
}
}
if (board[0, 0] != Mark.EMPTY &&
board[0, 0] == board[1, 1] &&
board[1, 1] == board[2, 2]
) {
return if (board[0, 0] == computer) 10 else -10
}
if (board[0, 2] != Mark.EMPTY &&
board[0, 2] == board[1, 1] &&
board[1, 1] == board[2, 0]
) {
return if (board[0, 2] == computer) 10 else -10
}
return 0
}
private fun minimax(isMaximizing: Boolean, depth: Int): Int {
val score = evaluate()
if (score == 10) return score - depth
```

```
if (score == -10) return score + depth

if (board.getEmptyCells().isEmpty() || depth >= maxDepth) return 0

if (isMaximizing) {

var best = Int.MIN_VALUE

for ((r, c) in board.getEmptyCells()) {

board[r, c] = computer

best = max(best, minimax(false, depth + 1))

board[r, c] = Mark.EMPTY

}

return best

} else {

var best = Int.MAX_VALUE

for ((r, c) in board.getEmptyCells()) {

board[r, c] = human

best = min(best, minimax(true, depth + 1))

board[r, c] = Mark.EMPTY

}

return best

}

}

fun findBestMove(): Pair<Int, Int> {

var bestVal = Int.MIN_VALUE

var bestMove = Pair(-1, -1)

for ((r, c) in board.getEmptyCells()) {
```

```
board[r, c] = computer

val moveVal = minimax(false, 0)

board[r, c] = Mark.EMPTY

if (moveVal > bestVal) {

bestMove = Pair(r, c)

bestVal = moveVal

}

}

return bestMove

}

}
```

КОПІЇ ОБОВ’ЯЗКОВИХ КРЕСЛЕНЬ

Посилання

Це джерела виділених збігів у вашому документі. Кожен збіг позначено темно-зеленим числом, яке відповідає вказаному тут джерелу. Джерела впорядковані за схожістю — чим вищий бал, тим сильніше збіг.

#	Джерело	%
1	repository.lnup.edu.ua	0.7%
2	antibotan.com	0.4%
3	repository.hneu.edu.ua	0.2%
4	ksm-x4.narod.ru	0.2%
5	git.is.ulstu.ru	0.2%
6	tesi.cab.unipd.it	0.2%
7	lib.iitta.gov.ua	0.2%
8	gavial.com.ua	0.2%
9	kltk.com.ua	0.2%
10	er.nau.edu.ua	0.1%
11	developer.android.com	0.1%
12	android.googlesource.com	0.1%
13	it-college.khai.edu	0.1%
14	elartu.tntu.edu.ua	0.1%
15	studfile.net	0.1%
16	itcollege.lviv.ua	0.1%
17	lektsii.org	0.1%
18	studfile.net	0.1%
19	evnuir.vnu.edu.ua	0.1%
20	ir.nmu.org.ua	0.1%
21	8ref.com	0.1%
22	dspace.znu.edu.ua	0.1%
23	files.confscience.webnode.ru	0.1%

#	Джерело	%
24	stackoverflow.com	0.1%
25	stackoverflow.com	0.1%
26	card-file.ontu.edu.ua	0.1%
27	docplayer.net	0.1%
28	ena.lpnu.ua	0.1%
29	ir.lib.vntu.edu.ua	0.1%
30	dspace.nuft.edu.ua	0.1%
31	developer.android.com	0.1%
32	eir.nuos.edu.ua	0.1%
33	os24.com.ua	0.1%
34	eprints.library.odeku.edu.ua	0.1%
35	ami.lnu.edu.ua	0.1%
36	core.ac.uk	0.1%
37	323.kiev.ua	0.1%
38	java2s.com	0.1%
39	socrates.vsau.org	0.1%
40	mmsa.kpi.ua	0.1%
41	developer.android.com	0.0%
42	stackoverflow.com	0.0%
43	ir.stu.cn.ua	0.0%
44	dspace.oneu.edu.ua	0.0%
45	ela.kpi.ua	0.0%
46	ir.nmu.org.ua	0.0%
47	ebib.pp.ua	0.0%
48	ebib.pp.ua	0.0%
49	nsnatura.pl	0.0%
50	oppb.com.ua	0.0%
51	developer.android.com	0.0%
52	dstu.dp.ua	0.0%
53	ua-referat.com	0.0%
54	shipregister.ua	0.0%
55	refdb.ru	0.0%
56	uadocs.exdat.com	0.0%
57	e.ieu.edu.ua	0.0%
58	ela.kpi.ua	0.0%

#	Джерело	%
59	kazedu.com	0.0%
60	tpv.vpi.kpi.ua	0.0%
61	pz.vntu.edu.ua	0.0%
62	antibotan.com	0.0%
63	kbuapa.kharkov.ua	0.0%
64	kpk.edu.ua	0.0%
65	ir.nmu.org.ua	0.0%
66	kpi.kharkov.ua	0.0%
67	ekmair.ukma.edu.ua	0.0%

Plag

Дякуємо, що перевірили
свій документ за допомогою
Plag!