Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

кафедра мультимедійних систем

Розробка android-додатку на прикладі гри "хрестики-нулики"

Текстова частина до курсової роботи за спеціальністю «Інженерія програмного забезпечення»

Керівник курсової роботи

Старший викладач

Борозенний С. О.

_____

(прізвище та ініціали)

_____

(підпис)

“____” _____

2020 р.

Виконав студент Муратов Віктор

(прізвище та ініціали)

“____”

_____ 2020 р.

Київ 2020

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем

ЗАТВЕРДЖУЮ
Зав.кафедри
мультимедійних систем,
доцент

_____О. П. Жежерун
(підпис)
„____”_____2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу

студенту_____ факультету _____ _____курсу

ТЕМА _____

Зміст ТЧ до курсової роботи:

1.Створення концепції гри

2. Огляд самої гри

3. Огляд та порівняння алгоритмів штучного інтелекту

4. План реалізації гри на Андроїд платформі

5. Аналіз створеної гри

6 Висновок

7 Джерела

8 Додатки

Дата видачі „___” _____ 2020 р.

Керівник _____(підпис)

Завдання отримав _____ (підпис)

Тема: _____

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	01.01.2020	
2.	Огляд технічної літератури за темою роботи.	01.01.2020	
3.	Виконати аналіз сучасних методів …	01.01.2020	
3.	Розробка алгоритму …	01.02. 2020	
4.	Програмування розробленого алгоритму	01.02. 2020	
5.	Застосування розробленого алгоритму до …	01.02. 2020	
6.	Написання пояснювальної роботи.	11.05.2020	
7.	Створення слайдів для доповіді та написання доповіді.	11.05.2020	
9.	Остаточне оформлення пояснювальної роботи та слайдів.	11.05.2020	
10.	Захист курсової роботи (проекту)	Після 11.05.2020	

Студент _____ Керівник _____ “_____” _____

Зміст

Анотація:

Ціль курсової роботи: дослідження штучнчого оінтелекту створення мобільної гри зі штучним інтелектом.

Вступ

1 Оцінка сучасного стану ігр зі штучним інтелектом

Станом на сьогодні штучний інтелект у іграх досить розвинений. Як правило існує декілька алгоритмів реалізації однієї задачі та декілька відкритих джерел із прикладом вирішення задачі. Взаємодія NPC у іграх між собою та з гравцем дуже схожа на дії реальних людей. NPC здатні вести повноцінне автономне життя навіть у іграх яким вже півтора-два десятка років, наприклад Stalker та Fallout. Якщо говорити про наприклад шашки чи крестки-нолики то на сьогодняшній день створено багато різніх варіацій штучного інтелекту з різними рівнями складності та навіть із динамічним рівнем складності. Проте не дивлячись на це, штучний інтелект у іграх не припиняє свій динамічний розвиток. Створюють нові алгоритми та покращують старі та заробляють на цьому гроші, очевидна справа.

2 Обґрунтування необхідності виконання роботи

Виконання роботи допоможе отримати практику з актуальної теми та створити щось своє, кількість ігор та попит на них підтверждує той факт, що тема актуальна. Як автор роботи, хочу внести щось своє у цю галузь.

3 Новизна теми

Тема не є новою, проте це не відміняє того факту що вона актуальна та затребувана. Першого бота створили у 1951 році, за 5 років до створення самої концепції.

4 Зв’язок з іншими науковими работами

Обираючи тему та створюючи проект орієнтувався я лише на статті на тематичних сайтах використовуючи їх як джерело знань необхідних для створення проекту та його опису. На тему штучного інтелекту було створено багато научних робіт.

РОЗДІЛ 1: Створення самої концепції гри.

1.1 Що будемо розробляти?

Для початку треба обрати що саме я буду розробляти. Перед мною стоїть задача створити мобільну гру зі штучним інтелектом. Я ще початківець і тому маю тверезо оцінювати свої сили. Були думки зробити шахмати шашки та навіть шутер або щось пов’язане з машинами. Проте вирішив для початку створити крестики нулики бо це перевірена та популярна тема яка має попит та дозволяє реалізувати у собі усе що необіхідно при цьому та не є надскладною у реалізації. Головне дати користувачу щось своє та особливе.

1.2 Назва

1.2.1 Аналіз методів створення назв для проекту

Правильна назва значно підвищує шанси на успіх. Назва – це перший крок до успіху, або до невдачі. Існують наступні підходи до створення назви: Абстракція – деяка абстракція над котрою задумуючись потенціальний гравець отримає інтерес до проекту, внутрішня назва – назва яку можно зрозуміти лише програв у гру, геймплейні назви – назви які описують геймплей, інструкція – назва яка деє вказівку по ігровому процесу, назва на честь єлементів ігрового процесу – наприклад на честь персонажа гри або якогось регіону у якому проходить подія гри. Назва це перше враження про гру. Ми створюємо гру крестики нолики де гравець грає у крестики нолики зі штучним інтелектом. Найбільш відомі проекти зі схожою концепцією у Play market: «Крестики нолики», «Сияющие крестики нолики», «Крестики нолики 2», «Tic tac toe – крестики нолики».

Це якщо гуглити російською. Якщо гуглити англійскою: «Tic tac toe – крестики нолики», «Сияющие крестики нолики», «Tic tac toe», «Tic tac toe glow», «Tic tac toe free». З цього можна зробити виводи: Видача йде тих програм які відповідають суті запиту, а не сугубо збіг по тексту. Можна зробити вивід що назва є важливим та непростим у обранні єлементом який значно впливає на маркетинг проекту я останнє буде розписано у наступному пункті.

Вплив назви на комерційний успіх

Назва яка демонструє ігровий процес з цікавої сторони, назва на честь успішної франшизи, назва яка спонукає клієнта задуматись о проекті та інше очевидно впливає позитивно на успіх – це перше враження та бренд. У той час як не цікава типова або безглузда назва відпугує або гравець просто листає плей маркет далі.

1.2.2 Обрання назви проекта

Зроблено виводи що назва має:

1)Демонструвати суть гри

2)Виділятись на фоні конкруентів

3)Демонструвати сильні сторони гри

4)Закладатись у голові юзера

На основі цих умозаключень я обрав назву «Tic tac toe quick dynamic and complexity» ця назва демонструє суть гри, демонструє сильні сторони та відносно унікальна та закладається тим у голові. Отже назву обрано!

1.3 Відмінність від подібних ігор

1.3.1 Аналіз конкурентів

Ринок крестиків-ноликів є дуже насищеним. Існує багато подібних ігор і досить складно придумати щось чого немає у конкурентів. Аналіз основних конкурентів:

1.3.1.1 Аналіз Tic tac toe

Назва проекту є краткою та зрозумілою проте не виділяє проект серед інших. У головну меню нас зустрічає меню з опціями Single player Muliti player Level. Саме меню має яскравий «неоновий» дизайн. Це одразу попадає до голови. У ігровому режимі ми маємо цікаві анімації та бачимо з ким ми граємо. Рівень для одиночної гри ми можемо обирати у головному меню. Рівнів багато. Існують різні розміри поля та складність проходження. У цілому цей

проект виглядає цікавим та є на мій погляд лідером ринку. Проте не усім потрібна така нафарширована начинка та складний геймплей.

1.3.1.2 Аналіз Tic tac toe glow

Назва проекту є краткою та зрозумілою та виділяється за рахунок «glow». Назва не є унікальною. Не всі знають переклад цього слова проте гра не орієнтована на ринок СНД наприклад. Проте створюючи проект я би так само використовував англоязичні назви бо це дає більш широкий та вигідний охват. Стиль проекту схожий на попередній. Неоновий дизайн. У головному меню ми одразу бачимо опції для налаштування ігрового реживу та кнопку старт. Ми можимо обирати розмірність поля складність та інше. У грі ми маємо яскравий дизайн та кнопку вихода зі гри. Це йе більш проста гра ніж попередня.

1.3.1.3 Аналіз Крестики-нолики

Гра має можливість вибору режиму (одиночна або гра з живим гравцев) Можна обрати поле. Назва та інтерфейс є стандартними дизайн – класичний білий фон(зошит). Особливих відмінностей немає, спочатку ми обираємо режим гри потім налаштовуємо гру.

1.3.1.4 Висновок

Неоновий дизайн вже є типовим. Існує два шляхи – насичений та складний геймплей та інтерфейс та швидкий та простий. Я обираю другий варіант бо його легше реалізувати проте він так само має шанси на успіх. Краще створити мінімалистичний та зручний інтерфейс ніж багато але не так як треба.

1.4 Платформа

Для нас доступно дві платформи – Android та IOS. Андроїд є найпопулярнішою кросплатформеною мобільною платформою. Заснована на лінукс. Структура рівнів андроїд:

1)Рівень програм та застосунків

2)Рівень каркаса програм

3)Рівень бібліотек

4)Рівень середовища

5)Рівень лінукс

IOS використовує ядро XNU і воно дуже схоже на ядро MacOS. IOS є спеціальною операційною системою для пристроїв компанії Apple. На відміну від Андроїд система є закритою та має багато обмежень.

1.4.1 Огляд інструментів розробки

1.4.1.1 XCode

Створено у 2013 році компанією Apple. Уся розробка у XCode відбувається у єдиному вікні. Apple LLVM ловить помилки «на ходу». Також ми маємо зручну панель переміщення між файлами. Мови програмування Swift та Objective C. Середовище є розвиненим. Розробник буде мати доступ до розвиненої документації. Ком'юніті є розвиненим і допоможе вирішити складні питання. Interface Builder пришвидшує розробку проекта. Проблемою є необіхідність пристрою з MacOS але це питання можно вирішити використанням віртуальних машин проте це незручно. У плані публікації недоліками є вартість 99 долларів та складність. Засоби розробки мають достатні можливості для покриття більшої кількості питань.

1.4.1.2 Android studio

Створено 2013 року компанією Google. Середовище є розвиненим та безкоштовним. Підходить середовище як і для розробки командної, так і самостійної. Ми маємо розробляти у данному середовищі під Android, Android TV, Android Wear, Android Auto, Glass. Э можливість зручної роботи з GIT або іншими системами керування версіями. Мови програмування – Java, Kotlin та C++. Розробник буде мати доступ до розвиненої документації.

Ком'юніті є розвиненим і допоможе вирішити складні питання. Зручний фреймворк та багатий функціонал. Багато плагінів. Великий вибір пристроїв проте якщо розробляти під Андроід є складність зробити програму вдалою для кожного девайсу. Недоліком є те що для роботи з Андроід студіо потрібен потужний пк бо середовище вимогливе.

1.4.2 Огляд комерційної частини

Існує точка зору що під IOS розробляти є більш вигідним чим під Андроід. Про те що то ринок що то і кожен надає певний потенціал. Існує деяка статистика з цього приводу: Середній дохід топ 100 компаній-розробників составив 84 млн долларів США против 51 млн долларів США у 2019 році, а у 2018 84 млн долларів США проти 45 млн долларів США. Це при тому що кількість користувачів на Андроїді більша. Користувачі IOS у цілому більш схильні до витрат коштів на контент та більше проявляють цікавості до реклами. Андроід аудиторія є більш широкою як і круг тих кого потенціально може зацікавити наша гра.

1.4.3 Висновок

Розробка під кожну з цих ОС має і плюси і мінуси. Під Андроїд простіше розпочати та дешевле у Яблука більше порядку та більш кращі показники по

статистиці. В мене немає пристрою з підходящою ОС тому буду розробляти під Андроід.

Цільова аудиторія

Крестики нолики є грою у котру грають різні люди різного віку з усього світу. Необхідно створити гру яка буде цікава та зрозуміла усім та кожному на цьому земному шарі. Розпишимо по ключовим пунктам:

What? Мобільну гру крестики нолики з ІІ

Who?Усім. Гра для людей любого віку, статі, походження, інтересів, освіти тощо

Where?З клієнтом взаємодіє мій додаток та якщо буде проект опубліковано у плей маркеті то у відгуках

When?Клієнт або побаче рекламу, або буде шукати додаток самостійно у плей маркеті або хтось йому порадить

Why?Гравець може згаїти час та потренувати логіку

1.5 Висновок

Нехай навіть ідея не є новою проте навіть у такому випадку проект може мати успіх через те що він буде мати правильну рекламу, назву, аватар тощо.

Необхідно дати користувачу щось своє і йому має це сподобатись: зробити більш зручний та захоплюючий геймлпей чи атмосферу у грі. Треба детально підійти до кожного пункту при створені проекту і тоді буде успіх. Не треба думати що «крестики-нолики» це не варте уваги. Звичайний пак зброї анімацій для STALKER(Gunslinger mod) який розробляли 10 років вважають зараз найкращим модом він написаний на асемблері та інші відомі моди підстраются зараз кажись під звичайний пак зброї? Чому? Бо якісно. Те саме і з крестиками-ноликами. Якщо зробити якісно то успіх буде. Необхідно чимось зачипити користувача та щоб юзер захопився грою.

Ми обрали назву, проаналізували конкурентів та створили концепцію загалом. На мій погляд та судячи по собі краще зробити швидкі та зручні крестики-нулики зі зручним інтерфейсом. Також є думка зробити можливість зміни рівня складності прямо у грі. Можна переходити до технічної частини.

РОЗДІЛ 2: Огляд самої гри

2.1 Правила гри

Крестики нолики є відомою грою логічного характера на полі кратної трьом розмірності. Згідно правил гравці по черзі ставлять свій знак так щоб по прямій або діагоналі виникла максимальна кількість його знаків. У такому випадку гравець є переможцем.

2.2 Історія гри

Існує не одна версія походжень цієї відомої гри. Існує думка що гра виникла у індії: засновник гри переробив шахматну дошку у поле для крестиків нуликів а потім її доробив римський винахідник. Також існує думка що гру створив математик із Франції під час вирішення складного рівняння. Також гадають що гру створив художник для Папи Римського. Також археологи знаходили дещо схоже. Існує багато різних думок з цього приводу та теорій і це додає грі додаткового антуражу.

2.3 Тактика гри

2.3.1 Гра за крестики

1)Обрати середину ігрового поля – крестики починають перші і це їхній плюс. З цієї ячейки багато можливостей для подальших ходів.

2)Чекаємо на хід суперника. У нього є декілька вариантів:

1) 1, 3, 7 9 – по кутам

2) 2, 4, 6 8 або по вертикальним/горизонтальним центральним лініям

Якщо гравець обере перший варіант – виграшна стратегія відсутня, треба чекати коли суперник «проколится»

Якщо гравець обере другий варіант – необхідно ставити хрестик у клітинки 1, 3, 7 9. Куди б далі не ходив суперник – у нас є 2 варіанти які при встановлені одного крестика забезпечать нам перемогу

3)Подальші ходи крестиків мають бути направлені на створення трьох підряд крестиків та протидію нуликам

2.3.2 Гра за нулики

1)Якщо центральна клітинка вільна – зайняти її. У іншому випадку займайте кутові точки.

2)Подальші ходи мають бути направлені на створення трьох підряд нуликів та протидію крестикам

2.4 Цікаво знати

Існує Китайський аналог «Гумоку». Гра проходить на полі 19х19 або 15х15 з використанням різнокольорового каміння

РОЗДІЛ 3: Огляд та порівняння алгоритмів штучного інтелекту.

3.1 Алгоритм «Мінімакс»

Алгоритм мінімакс є покроковим алгоритмом прийняття рішень і його задача є пошук оптимального рішення на кожному кроці. Принцип роботи алгоритма схожий на те як мислить стратег-інтелектуал.

Алгоритм призначений для для взаємодії двох сторін – мінімізатор та максімізатор. Тобто один вибарає найбільше значення серед множини ходів, а інший – найменше. Розпишемо кроки та пояснимо їх:

1)Побудувати дерево можливих кроків: тобто дерева кроків і після кожного кроку який крок можливий

2)Оцінити кожний крок: дати йому певне значення

3)Тепер ми маємо обрати для мінімізатора мінімальне значення, а для максімізатора максимальне. Кожному значенню відповідає певний крок.

0 – нічия.

Алгоритм мінімакс можливо покращити і те як це зробити буде розписано у пункті «Альфа-бета алгоритм».

3.2 Альфа-бета алгоритм

Олександр Брудно створив альфа-бета алгоритм у 1963 році. Рональд Мур і Дональд Кнут покращили Альфа-бета алгоритм у 1975 році. Альфа-бета алгоритм є покращенням minimax алгоритму – дозволяє припинити аналіз деяких ходів достроково. Minimax є витратним алгоритмом через те що кожен раз алгоритм будує дерево рішень. Альфа-бета алгоритм вирішує це питання. Нехай у нас є дерево. У корені – «територія» мінімізатора, а на

вершинах максимального рівня ход максимізатора. Ми створюємо дві додаткові змінні: одна відповідає за максимальне значення менше котрого максимізатор ніколи не вибере, інша несе в собі мінімальне значення менше за котре мінімізатор ніколи не вибере. Початкове значення кожної рівне +- нескінченності але у процесі роботи алгоритма значення буде змінено.

Коли максимальна змінна стане більше мінімальної, ми повертаємо поточний результат і це нам дає можливість заощадити ресурси. Завдяки використанню цієї оптимізації глибина дерева буде знижена та швидкість роботи ІІ буде значно збільшено. Для реалізації Альфа-бета достатньо двух змінних та функції.

3.3 Готові рішення

Суть цього алгоритма є простою. Ми зберігаємо стан поля та відповідний крок для крестиків та нуликів. Програма просто обирає відповідний крок із множини станів. З точки зору обчислень простий, проте треба зберігати дані.

3.4 Інтуїтивний алгоритм

Інтуітивний алгоритм забезпечує інший підхід до створення ІІ. Він ще більше походить на алгоритм мислення гравця. Коли ми граємо у крестики нолики як правило ми не часто будуємо стратегію на багато кроків уперед. Ми маємо два правила:

1)Створити максимальну довжину нашого елементу

2)Створити перешкоду опоненту

На основі цього ми можемо зробити функцію оцінювання кожного кроку:

$$+F(\text{крок})+= \text{Користь\_нам}(\text{крок}) + + + \text{Шкода\_супернику}(\text{крок})$$

РОЗДІЛ 4: План реалізації гри на Android платформі.

1)Створення інтерфейсу через редактор інтерфейсу

2)Створення клітинки та координат

3)Створення дошки

4)Створення штучного інтелекту для хода суперника

5)Зв’язати усе разом

6)Тестування

РОЗДІЛ 5: Аналіз створеної мобільної гри

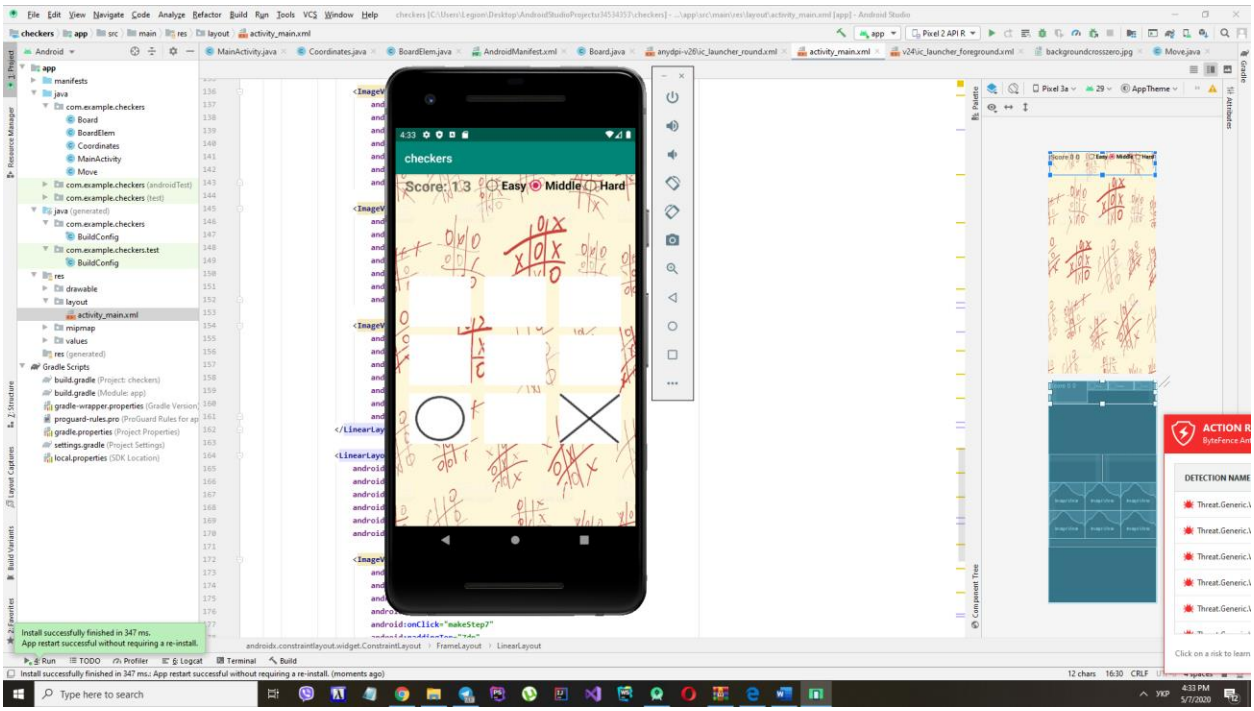

Рисунок 1. Демонстрація ігрового процесу

Далі буде надано пояснення класів та основного файлу розмітки.

5.1 Класс Board

Цей класс по суті є дошкою та основним ігровим полем. Він містить змінні:

hardCoof – коофіцент складності гри. Цей коофіціент впливає на работу мінімакс алгоритму, тобто на його «ідеальність»

SIZE – розмірність ігрового поля по x або y

Game – ця змінна по суті є показником того чи завершена гра(виграш нічія поразка гравця)

Player1 – поточний рахунок гравця

Player2 –поточний рахунок II

numberOfEmpty – поточна кількість пустих клітинок

initBoard() – ініціалізує ігрове поле початковими елементами(пусті клітинки) та встановлює значення змінних за замовчуванням

gameFinish(BoardElem board[][]) – перевіряє чи гра завершена(нявність пустих клітинок, тобто перевіряє нічию)

checkAll(BoardElem b) – перевірити перемогу фігури заданого елементу шляхом обходу ігрового поля. За один цикл одночасно виконується обхід усього ігрового поля.

minimaxBestStep(BoardElem board[][], int depth, Boolean isMax) – Застосовуючи мінімакс алгоритм визначає найкращий крок для нуликів згідно складності гри. Для регулювання складності використано коофіціент складності – він впливає на ймовірність того що II поступить ідеально.

doStepCross(int x, int y) – виконати крок за крестики. За крестики ходить гравець. Після кроку передаємо «естафету» нуликам, якщо гру не було завершено

doStepZero() – виконати крок за нулики. За нулики ходить штучний інтелект. Застосування мінімакс алгоритму. Після кроку передаємо «естафету» крестикам, якщо гру не було завершено

5.2 Класс BoardElem

Класс відповідає за конкретну клітинку чи її стан

Cross – чи є елемент крестиком

Zero – чи є елемент нуликом

positionX – позиція по x ординаті

positionY – позиція по y ординаті

setStateCross() – змінити стан клітинки на клітинку крестиків

setStateZero() – змінити стан клітинки на клітинку нуликів

setStateEmpty() – змінити стан клітинки на пусту клітинку

BoardElem(int x, int y) – конструктор клітинки на основі її координат

BoardElem(BoardElem b) – конструктор клітинки

checkType(BoardElem b) – перевірити чи співпадають типи елементів. Якщо так то повернути true.

5.3 Класс Coordinates

Класс для зручного представлення координат клітинок

X – позиція по x

Y – позиція по y

Coordinates(int x, int y) – конструктор координат по координатам

5.4 Класс MainActivity

Класс який керує усіма процесами гри та поєднує усе разом

Board – ігрове поле. Класс MainActivity маніпулює ним.

B1,b2,b3,b4,b5,b6,b7,b8,b9 – клітинки-зображення

changeImage() – змінити стан клітинки. Наприклад на крестик чи нулик.

makeStep1(View), makeStep2(View), makeStep3(View),makeStep4(View), makeStep5(View),makeStep6(View), makeStep7(View),makeStep8(View), makeStep9(View) – обробити хід гравця

reload() – перезавантажити ігрове поле. Почати гру з початку.

init() – ініціалазація гри

onCreate(Bundle savedInstanceState)

5.5 activity_main.xml

Цей xml файл відповідає за розмітку графічного інтерфейсу.

FrameLayout1 – основний фрейм у якому знаходиться весь контент

LinearLayout0 – компонент у якому знаходяться налаштування рівня складності та очок

Score – поточний рахунок. Зліва гравець, по праву сторону штучний інтелект

rEasy – встановити легку складність (змінити значення коофіцієнту складності)

rMid – встановити середню складність (змінити значення коофіцієнту складності)

rHard – встановити максимальну складність (змінити значення коофіцієнту складності)

linearLayout1 – 1 2 3 клітинки

lnearLayout2 – 4 5 6 клітинки

field1 – 1 клітинка

field2 – 2 клітинка

field3 – 3 клітинка

linearLayout4 – 7 8 9 клітинки

field4 – 4 клітинка

field5 – 5 клітинка

field6 – 6 клітинка

field7 – 7 клітинка

field8 – 8 клітинка

field9 – 9 клітинка

5.7 зображення використані у грі

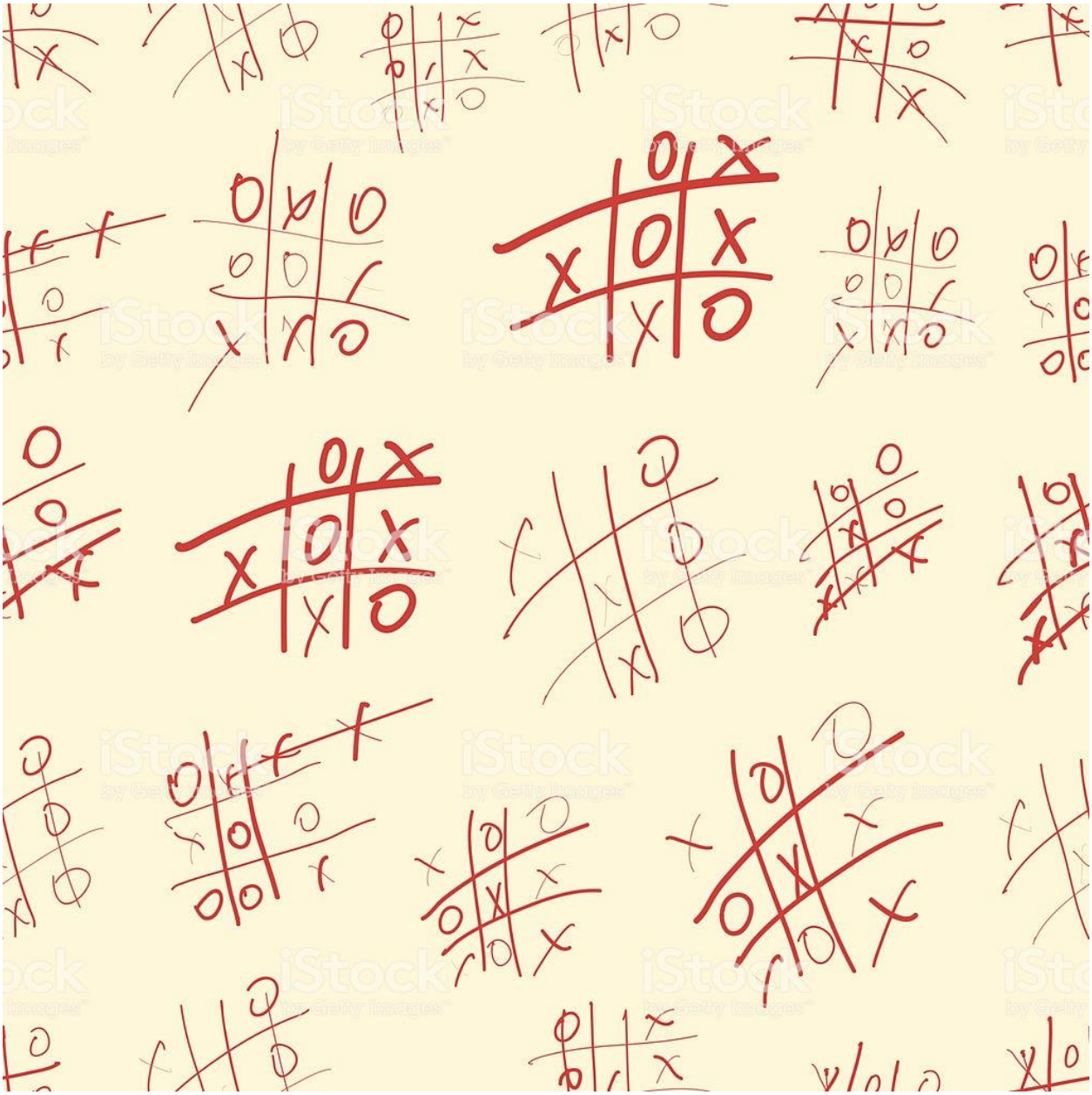

Рис 2. Фон

Джерело: https://www.istockphoto.com/ru

Фонове зображення для ігрового поля. Виглядає дуже тематично.

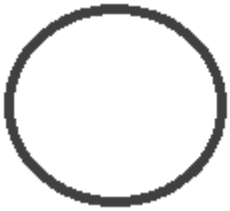

Рис 2,3,4 Фігури

Зображення крестика, пустого поля, та нулика. Намальовані самостійно у графічному редакторі.

РОЗДІЛ 6: Висновок

Було проведено наступні дослідження:

1) створення концепції та дослідження ринку у плей маркет
2) аналіз платформ та засобів розробки мобільних застосунків
3) дослідження Мінімакс та Альфа-бета алгоритму
4) дослідження Інтуїтивного алгоритму
5) дослідження предметної області
6) аналіз створеного продукту

Було отримано досвід з відповідних пунктів та написання значного об’єму текстової частини. Також звісно було отримано досвід зі створення самого

додатку(реалізація гри та штучного інтелекту) та комунікацій, пошуку інформації.

РОЗДІЛ 7: Джерела

https://www.istockphoto.com/ru

https://habr.com/ru/

https://www.google.com/

https://hightech.fm/2019/06/25/ios-vs-android

РОЗДІЛ 8: Додатки

Код програми:

```
package com.example.checkers;

import android.os.Build;
import androidx.annotation.RequiresApi;

import java.util.ArrayList;

public class Board {

    static double hardCoof = 0.98;

    private MainActivity mainActivity;
    private final int SIZE = 3;
    private boolean game = true;
    private int player1 = 0;
    private int player2 = 0;
    private int numberOfEmpty = 9;

    public int getNumberOfEmpty() {
        return numberOfEmpty;
```

```
}

public void setNumberOfEmpty(int numberOfEmpty) {
    this.numberOfEmpty = numberOfEmpty;
}

public BoardElem[][] getBoard() {
    return board;
}

public void setBoard(BoardElem[][] board) {
    this.board = board;
}

public int getPlayer1() {
    return player1;
}

public int getPlayer2() {
    return player2;
}

public boolean isGame() {
    return game;
}

Board(MainActivity mainActivity) {
    this.mainActivity = mainActivity;
    initBoard();
}

protected BoardElem[][] board = new BoardElem[SIZE][SIZE];

public BoardElem getBoardElem(int x, int y) {
    return board[x][y];
}

protected void initBoard() {
    game = true;
    numberOfEmpty = 9;
    for(int i = 0; i < SIZE; i++) {
        for(int j = 0; j < SIZE; j++) {
            board[i][j] = new BoardElem(i, j);
        }
    }
}

protected void setEmptyElem(int x, int y) {
    board[x][y].setStateEmpty();
}

protected void setZeroElem(int x, int y) {
    board[x][y].setStateZero();
}
```

```
protected void setCrossElem(int x, int y) {
    board[x][y].setStateCross();
}

protected ArrayList<Integer> avSp() {
    ArrayList<Integer> resultList = new ArrayList<>();
    int buff = 0;
    for(int i = 0; i < SIZE; i++) {
        for(int j = 0; j < SIZE; j++) {
            if(board[i][j].isEmpty())resultList.add(buff);
            buff++;
        }
    }
    return resultList;
}

Boolean gameFinish(BoardElem board[][]) {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            if (board[i][j].isEmpty()) return true;
        }
    }
    return false;
}

protected boolean checkAll(BoardElem b) {
    int diagX = 0;
    int diagY = SIZE - 1;
    boolean diag1 = true;
    int diag2X = SIZE - 1;
    int diag2Y = SIZE - 1;
    boolean diag2 = true;
    int diagLineX = 0;
    int diagLineY = b.getPositionY();
    boolean line1 = true;
    int diagLine2X = b.getPositionX();
    int diagLine2Y = 0;
    boolean line2 = true;
    for(int i = 0; i < SIZE; i++) {
       if(diag1 && !board[diagX][diagY].checkType(b)) {
            diag1 = false;
        }
        diagX++;
        diagY--;
        if(diag2 && !board[diag2X][diag2Y].checkType(b)) {
            diag2 = false;
        }
        diag2X--;
        diag2Y--;
        if(line1 && !board[diagLineX][diagLineY].checkType(b)) {
            line1 = false;
        }
        diagLineX++;
        if(line2 && !board[diagLine2X][diagLine2Y].checkType(b)) {
            line2 = false;
```

```
            }
            diagLine2Y++;
        }
        game = (diag1 || diag2 || line1 || line2);
        return (diag1 || diag2 || line1 || line2);
    }


    int minimaxBestStep(BoardElem board[][], int depth, Boolean isMax) {
        for (int i = 0; i < 3; i++) {
            if (board[0][i].checkType(board[1][i] )&&
board[1][i].checkType(board[2][i])) {
                if (board[0][i].isZero())
                    return +10;

                else if (board[0][i].isCross())
                    return -10;
            }
            if (board[i][0].checkType(board[i][1]) &&
board[i][1].checkType(board[i][2])) {
                if (board[i][0].isZero())
                    return +10;
                else if (board[i][0].isCross())
                    return -10;
            }
        }

        if (board[0][0].checkType(board[1][1]) && board[1][1].checkType(board[2][2])
&& board[0][0].isZero()) return +10;
        if (board[0][0].checkType(board[1][1]) && board[1][1].checkType(board[2][2])
&& board[0][0].isCross()) return -10;
        if (board[0][2].checkType(board[1][1]) && board[1][1].checkType(board[2][0])
&& board[0][2].isZero()) return +10;
        if (board[0][2].checkType(board[1][1]) && board[1][1].checkType(board[2][0])
&& board[0][2].isCross()) return -10;

        if (gameFinish(board) == false) return 0;

        double rand = Math.random();
        if(rand > hardCoof) {
            if(isMax) isMax = false;
            else isMax = true;
            System.out.println(rand);
        }
        int max;
        if(isMax) max = -1000;
        else max = 1000;
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                if (board[i][j].isEmpty()) {
                    if(isMax) {
                        board[i][j].setStateZero();
                        max = Math.max(max, minimaxBestStep(board, depth + 1,
!isMax));
                    } else {
```

```
                        board[i][j].setStateCross();
                        max = Math.min(max, minimaxBestStep(board, depth + 1,
!isMax));
                    }
                    board[i][j].setStateEmpty();
                }
            }
        }
        return max;
    }

    @RequiresApi(api = Build.VERSION_CODES.N)
    protected void doStepCross(int x, int y) {
        if(board[x][y].isEmpty()) {
            board[x][y].setStateCross();
            numberOfEmpty--;
            if(checkAll(board[x][y])) player1++;
        }
    }

    @RequiresApi(api = Build.VERSION_CODES.N)
    protected Coordinates doStepZero() {
        int bestVal = -1000;
        Coordinates coordinates = new Coordinates(-1, -1);
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                if (board[i][j].isEmpty()) {
                    board[i][j].setStateZero();
                    int moveVal = minimaxBestStep(board, 0, false);
                    board[i][j].setStateEmpty();
                    if (moveVal > bestVal) {
                        coordinates.setX(i);
                        coordinates.setY(j);
                        bestVal = moveVal;
                    }
                }
            }
        }
        if(board[coordinates.getX()][coordinates.getY()].isEmpty()) {
            board[coordinates.getX()][coordinates.getY()].setStateZero();
            numberOfEmpty--;
            if(checkAll(board[coordinates.getX()][coordinates.getY()])) player2++;
            return coordinates;
        }
        return coordinates;
    }
}



package com.example.checkers;

class BoardElem {
    private int positionX;
    private int positionY;
```

```
protected Coordinates toCoordinates() {
    return new Coordinates(positionX, positionY);
}

protected void setPositionX(int positionX) {
    this.positionX = positionX;
}

protected void setPositionY(int positionY) {
    this.positionY = positionY;
}

public int getPositionX() {
    return positionX;
}
public int getPositionY() {
    return positionY;
}
private boolean empty;

public boolean isEmpty() {
    return empty;
}
public boolean isCross() {
    return cross;
}
public boolean isZero() {
    return zero;
}


private boolean cross;

private boolean zero;

public void setStateCross() {
    zero = false;
    cross = true;
    empty = false;
}

public void setStateZero() {
    zero = true;
    cross = false;
    empty = false;
}

public void setStateEmpty() {
    zero = false;
    cross = false;
    empty = true;
}

BoardElem(int x, int y) {
```

```
        positionX = x;
        positionY = y;
        empty = true;
        cross = false;
        zero = false;
    }

    BoardElem(BoardElem b) {
        positionX = b.positionX;
        positionY = b.positionY;
        empty = b.empty;
        cross = b.cross;
        zero = b.zero;
    }

    public boolean checkType(BoardElem b) {
        if(zero == b.zero && cross == b.cross && empty == b.empty) return true;
        return false;
    }
}
```

```
package com.example.checkers;

public class Coordinates {
    private int x;
    private int y;

    Coordinates(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public int getY() {
        return y;
    }

    public void setY(int y) {
        this.y = y;
    }

    public int getX() {
        return x;
    }

    public void setX(int x) {
        this.x = x;
    }
}
```

```
package com.example.checkers;
```

```
import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;

import android.annotation.SuppressLint;
import android.os.Build;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    Board board;
    ImageView b1;
    ImageView b2;
    ImageView b3;
    ImageView b4;
    ImageView b5;
    ImageView b6;
    ImageView b7;
    ImageView b8;
    ImageView b9;
    TextView score;
    private int stepCounter = 0;

    private Coordinates getIndex(int a) {
        int x = 0;
        int y = 0;
        for(int i = 0; i < a; i++) {
            if(y == 3) {
                x++;
                y = 0;
            }
        }
        return new Coordinates(x, y);
    }



    private void changeImage(int x, int y) {
        if(x == 0 && y == 0){
            b1.setImageResource(R.drawable.zerof);
        } else if(x == 1 && y == 0) {
            b2.setImageResource(R.drawable.zerof);
        } else if(x == 2 && y == 0) {
            b3.setImageResource(R.drawable.zerof);
        } else if(x == 0 && y == 1) {
            b4.setImageResource(R.drawable.zerof);
        } else if(x == 1 && y == 1) {
            b5.setImageResource(R.drawable.zerof);
        } else if(x == 2 && y == 1) {
            b6.setImageResource(R.drawable.zerof);
        } else if(x == 0 && y == 2) {
            b7.setImageResource(R.drawable.zerof);
        } else if(x == 1 && y == 2) {
```

```
            b8.setImageResource(R.drawable.zerof);
        } else if(x == 2 && y == 2) {
            b9.setImageResource(R.drawable.zerof);
        }
    }


    @RequiresApi(api = Build.VERSION_CODES.N)
    public void makeStep1(View v){
        if(!board.getBoardElem(0, 0).isEmpty()) return;
            board.doStepCross(0, 0);
            if(board.isGame()) reload();
            b1.setImageResource(R.drawable.crossf);
        stepCounter++;
        if(stepCounter == 9) {
            reload();
            return;
        }
        Coordinates coordinates = board.doStepZero();
        changeImage(coordinates.getX(), coordinates.getY());
        stepCounter++;

        if(board.isGame()) reload();
    }
    @RequiresApi(api = Build.VERSION_CODES.N)
    public void makeStep2(View v){
        if(!board.getBoardElem(1, 0).isEmpty()) return;
        board.doStepCross(1, 0);
        if(board.isGame()) reload();
        b2.setImageResource(R.drawable.crossf);
        stepCounter++;
        if(stepCounter == 9) {
            reload();
            return;
        }
        Coordinates coordinates = board.doStepZero();
        changeImage(coordinates.getX(), coordinates.getY());
        stepCounter++;

        if(board.isGame()) reload();
    }
    @RequiresApi(api = Build.VERSION_CODES.N)
    public void makeStep3(View v){
        if(!board.getBoardElem(2, 0).isEmpty()) return;
        board.doStepCross(2, 0);
        if(board.isGame()) reload();
        b3.setImageResource(R.drawable.crossf);
        stepCounter++;
        if(stepCounter == 9) {
            reload();
            return;
        }
        Coordinates coordinates = board.doStepZero();
        changeImage(coordinates.getX(), coordinates.getY());
        stepCounter++;
```

```
    if(board.isGame()) reload();
}
@RequiresApi(api = Build.VERSION_CODES.N)
public void makeStep4(View v){
    if(!board.getBoardElem(0, 1).isEmpty()) return;
    board.doStepCross(0, 1);
    if(board.isGame()) reload();
    b4.setImageResource(R.drawable.crossf);
    stepCounter++;
    if(stepCounter == 9) {
        reload();
        return;
    }
    Coordinates coordinates = board.doStepZero();
    changeImage(coordinates.getX(), coordinates.getY());
    stepCounter++;

    if(board.isGame()) reload();
}
@RequiresApi(api = Build.VERSION_CODES.N)
public void makeStep5(View v){
    if(!board.getBoardElem(1, 1).isEmpty()) return;
    board.doStepCross(1, 1);
    if(board.isGame()) reload();
    b5.setImageResource(R.drawable.crossf);
    stepCounter++;
    if(stepCounter == 9) {
        reload();
        return;
    }
    Coordinates coordinates = board.doStepZero();
    changeImage(coordinates.getX(), coordinates.getY());
    stepCounter++;

    if(board.isGame()) reload();
}
@RequiresApi(api = Build.VERSION_CODES.N)
public void makeStep6(View v){
    if(!board.getBoardElem(2, 1).isEmpty()) return;
    board.doStepCross(2, 1);
    if(board.isGame()) reload();
    b6.setImageResource(R.drawable.crossf);
    stepCounter++;
    if(stepCounter == 9) {
        reload();
        return;
    }
    Coordinates coordinates = board.doStepZero();
    changeImage(coordinates.getX(), coordinates.getY());
    stepCounter++;

    if(board.isGame()) reload();
}
@RequiresApi(api = Build.VERSION_CODES.N)
```

```
public void makeStep7(View v){
    if(!board.getBoardElem(0, 2).isEmpty()) return;
    board.doStepCross(0, 2);
    if(board.isGame()) reload();
    b7.setImageResource(R.drawable.crossf);
    stepCounter++;
    if(stepCounter == 9) {
        reload();
        return;
    }
    Coordinates coordinates = board.doStepZero();
    changeImage(coordinates.getX(), coordinates.getY());
    stepCounter++;

    if(board.isGame()) reload();
}
@RequiresApi(api = Build.VERSION_CODES.N)
public void makeStep8(View v){
    if(!board.getBoardElem(1, 2).isEmpty()) return;
    board.doStepCross(1, 2);
    if(board.isGame()) reload();
    b8.setImageResource(R.drawable.crossf);
    stepCounter++;
    if(stepCounter == 9) {
        reload();
        return;
    }
    Coordinates coordinates = board.doStepZero();
    changeImage(coordinates.getX(), coordinates.getY());
    stepCounter++;

    if(board.isGame()) reload();
}
@RequiresApi(api = Build.VERSION_CODES.N)
public void makeStep9(View v){
    if(!board.getBoardElem(2, 2).isEmpty()) return;
    board.doStepCross(2, 2);
    if(board.isGame()) reload();
    b9.setImageResource(R.drawable.crossf);
    stepCounter++;
    if(stepCounter == 9) {
        reload();
        return;
    }
    Coordinates coordinates = board.doStepZero();
    changeImage(coordinates.getX(), coordinates.getY());
    stepCounter++;

    if(board.isGame()) reload();
}

@RequiresApi(api = Build.VERSION_CODES.N)
public void easyClick(View v){
    Board.hardCoof = 0.98;
}
```

```
@RequiresApi(api = Build.VERSION_CODES.N)
public void middClick(View v){
    Board.hardCoof = 0.99;
}

@RequiresApi(api = Build.VERSION_CODES.N)
public void hardClick(View v){
    Board.hardCoof = 1;
}

@SuppressLint("WrongViewCast")
private void init() {
    board = new Board(this);
    b1 = findViewById(R.id.field1);
    b2 = findViewById(R.id.field2);
    b3 = findViewById(R.id.field3);
    b4 = findViewById(R.id.field4);
    b5 = findViewById(R.id.field5);
    b6 = findViewById(R.id.field6);
    b7 = findViewById(R.id.field7);
    b8 = findViewById(R.id.field8);
    b9 = findViewById(R.id.field9);
    b1.setImageResource(R.drawable.emptyf);
    b2.setImageResource(R.drawable.emptyf);
    b3.setImageResource(R.drawable.emptyf);
    b4.setImageResource(R.drawable.emptyf);
    b5.setImageResource(R.drawable.emptyf);
    b6.setImageResource(R.drawable.emptyf);
    b7.setImageResource(R.drawable.emptyf);
    b8.setImageResource(R.drawable.emptyf);
    b9.setImageResource(R.drawable.emptyf);
    score = findViewById(R.id.score);
}

private void reload() {
    score.setText("Score: " + board.getPlayer1() + " " + board.getPlayer2());
    board.initBoard();
    b1.setImageResource(R.drawable.emptyf);
    b2.setImageResource(R.drawable.emptyf);
    b3.setImageResource(R.drawable.emptyf);
    b4.setImageResource(R.drawable.emptyf);
    b5.setImageResource(R.drawable.emptyf);
    b6.setImageResource(R.drawable.emptyf);
    b7.setImageResource(R.drawable.emptyf);
    b8.setImageResource(R.drawable.emptyf);
    b9.setImageResource(R.drawable.emptyf);
    stepCounter = 0;
}

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    init();
```

```
    }
}
```