Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

«Розробка для додатку для гри в шахи з використанням ігрового штучного інтелекту» 122

Керівник курсової роботи
ст. викладач Борозенний С. О.
_____
(підпис)
“____”_____2023 р.

Виконав студент 3 р. н.
Кондратенко Д. Д.
“____”_____2023 р.

Київ 2023

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
Зав. Кафедри мультимедійних систем,
доцент, к.ф-м.н
_____Жежерун О. П.
(підпис)
„_____”_____2023 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Кондратенку Дмитру Денисовичу

3-го курсу факультету інформатики

ТЕМА: Розробка для додатку для гри в шахи з використанням ігрового штучного інтелекту

Вихідні дані:

Зміст ТЧ до курсової роботи

Вступ

Анотація

1. Аналіз предметної області
2. Вибір технологій та засобів для розробки
3. Концепція ігрового штучного інтелекту в шахах
4. Реалізація застосунку

Висновки

Список використаних джерел

Дата видачі „___”_____2023 р. Керівник _____
Завдання отримав _____

Календарний план виконання курсової роботи

Тема: Розробка для додатку для гри в шахи з використанням ігрового штучного інтелекту

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу	Жовтень 2022 р.	
2.	Аналіз предметної області	Листопад 2022 р.	
3.	Вибір концепції додатку	Січень-лютий 2023 р.	
4.	Розробка додатку	Березень 2023 р.	
5.	Написання текстової частини курсової	квітень 2023 р.	
6.	Оформлення презентації для захисту	травень 2023 р.	
7.	Здача роботи для перевірки на плагіат	15 травня 2023 р.	
8.	Захист курсової роботи	22 травня 2023 р.	

Студент: Кондратенко Д.Д.

Керівник: Борозенний С. О.

“_____” _____

Зміст

Вступ

На даний момент можна сміливо стверджувати, що шахи – одна з найпопулярніших і водночас найдавніших ігор у світі. На турнірах з цієї дисципліни розігруються призові фонди з 7-значним числом доларів, а прямі трансляції з місця подій переглядають одночасно десятки тисяч вболівальників по всьому світу. Наприклад, на нещодавньому чемпіонаті світу з шахів, що проходив з 7 квітня по 1 травня 2023 року в Астані між росіянином Яном Непомнящим та представником Китаю Дін Ліженем розігрувалися не тільки звання найсильнішого шахіста планети, а також і 2 мільйони доларів, 60 відсотків з яких забирав переможець, а решту – інший учасник. За цією баталією одночасно в середньому слідкувало близько 150 тисяч глядачів по всьому світу, а пік онлайну сягнув позначки у 572 тисячі.

До речі, у напруженій боротьбі титул виборов представник КНР, ставши першим в історії китайським гравцем, що завоював шахову корону, замінивши на п'ятдесталі ще одного видатного гравця – норвежця Магнуса Карлсена.

А що, якщо я Вам скажу, що найсильніші шахісти – зовсім не люди, і навіть ніхто з вище згаданих. У шахи набагато краще за людей давно вже навчились грати роботи. А саме: ігрові рушії з елемнтами штучного інтелекту, такі як Stockfish, Leela Chess Zero, Ethereal, Houdini і так далі. Якщо порівняти машини відносно людей, застосувавши певну ретингову систему (в даному випадку, найбільш відповідною буде система рейтигу Ело, що обчислює відносний рівень гравців у різні види дисциплін), то спостерігатимемо відчутну різницю в рейтингу: вже станом на 2021 рік піковий рейтинг штучного інтелекту майже на 700 пунктів випереджав піковий рейтинг людини.

Ігровий рушій	Рейтинг станом на 2021 р.	Шахіст	Рейтинг станом на 2023 р.
Stockfish	3584	Магнус Карлсен	2847
Leela Chess Zero	3459	Фабіано Каруана	2828
Houdini	3417	Дін Ліжень	2805
Komodo	3407	Левон Аронян	2787
Ethereal	3355	Веслі Со	2785

То ж не дивно, що стала популярною гра людини проти комп'ютера, а також аналіз своєї гри за домогою штучного інтелекту. Провідні шахові платформи, такі як lichess.org та chess.com, давно запровадили інструментарій для гри з комп'ютером і аналізу будь-яких позицій. Саме останній пункт є найважливішим у цій курсовій роботі.

Її метою є реалізація певного функціоналу комп'ютерного шахового аналізу за допомогою доступного набору інструментів і даних, а також дослідження основних методів підрахунку оцінки шахової позиції. Робота складається з чотирьох розділів.

Перший включає в себе аналіз предметної області, а саме функціоналу платрформ для гри в шахи, виокремлення важливих функцій, які згодом буде реалізовано у застосунку.

У другому розділі буде проведений опис та обґрунтований вибір

інструментів для розробки додатку для аналізу позицій.

Третій розділ присвячений дослідженню основних методів розрахунку оцінки шахових партій на основі коду ігрового рушія Stockfish у вільному доступі.

У четвертому розділі увага приділятиметься саме реалізації графічного інтерфейсу користувача та опису основних методів бібліотеки для роботи з шаховим штучним інтелектом.

Анотація

Курсова робота присвячена розробці застосунку для аналізу шахової позиції за допомогою Python 3.9 та вбудованих бібліотек: Pygame – для реалізації логіки поведінки шахових фігур та дошки і їх візуалізації; Stockfish – для підключення до проєкту відповідного шахового рушія.

Додаток пропонує такі основні функції: введення певної, відмінної від початкової, позиції шахових фігур на дошці, її подальший аналіз та пропонування найефективнішого ходу і відображення оцінки позиції власне ігровим рушієм.

1. Аналіз предметної області

1.1 Огляд додатків для гри в шахи

Найпопулярнішими шаховими платформами є lichess.org та chess.com.

Обидві платформи включають в себе майже ідентичні функції, такі як:

1) Гра з іншими людьми в режимі реального часу онлайн
2) Гра з комп'ютером різного рівня складності
3) Створення турнірів і запрошення гравців з усього світу
4) Аналіз партій, який допомагає користувачам досліджувати свої помилки та вдалі ходи
5) Тренування своїх шахових навичок за допомогою різних режимів вирішення задач

Але є одна особливість, яка вирізняє lichess.org серед конкурентів:

саме цей сервіс надає доступ до статистики користувачам абсолютно безкоштовно, за рахунок чого вони можуть відстежувати свій прогрес.

В свою чергу, chess.com за рахунок системи платних підписок, активніше впроваджує новинки та покращення, а також має більшу аудиторію, і, відповідно, більшу кількість контенту безпосередньо на сайті платформи.

1.2 Виокремлення важливих функцій

Для якомога найефективнішої реалізації додатку-аналізатора шахових партій важливо виділити функції платформ-аналогів. Дослідивши дві найпопулярніші, було визначено, що партії ширяться з-поміж користувачів за допомогою спеціальних більш компактних форматів запису поточної позиції на шаховій дошці (PGN і FEN формати), а пропоновані ходи та загальна оцінка партії

висвітлюється на екрані простим списком декількох найкращих варіантів продовження. Саме ці функції і було обрано для їх реалізації і написання відповідного застосунку.

1.3 Постановка задачі

Зважаючи на мету розробки додатку та предметну область, можна сформулювати задачу, поставлену переді мною, як розробника, і функції, які цей додаток виконуватиме:

- Реалізація логіки поведінки шахових фігур (з дотриманням усіх чинних шахових правил)
- Побудова шахової позиції фігур за заданою FEN нотацією
- Вивід найкращого продовження у заданій позиції на екран
- Графічний інтерфейс користувача

2. Вибір технологій та засобів для розробки

2.1. Python для візуалізації та реалізації шахової логіки

Коли перетинаються теми програмування і певні обрахункові задачі, то вибір мови програмування майже завжди припадає на Python по ряду об'єктивних причин.

1) Легкість у вивченні: Python відома своєю простотою та легкістю у використанні, що робить її ідеальним вибором для початківців, які хочуть вивчати програмування. Синтаксис мови простий і зрозумілий, що полегшує розуміння і написання коду.

2) Велика та активна спільнота: Python має велику та активну спільноту розробників, які створюють бібліотеки та інструменти, що можуть бути використані для реалізації шахової логіки та візуалізації шахових ігор. Це означає, що ви можете знайти безліч ресурсів в Інтернеті, таких як навчальні посібники, документація та відкритий код, які допоможуть вам розпочати роботу.

3) Потужні бібліотеки: Python має декілька потужних бібліотек, які можна використовувати для програмування шахів, зокрема Pygame, PyChess та Chessnut. Ці бібліотеки пропонують широкий спектр функціональних можливостей, від візуалізації шахових партій до реалізації ігрової логіки та алгоритмів штучного інтелекту.

4) Крос-платформна підтримка: Python є кросплатформенною мовою, що означає, що код, написаний на одній платформі, може бути легко перенесений на іншу платформу без змін. Це дозволяє легко створювати шахові програми, які можна запускати на різних операційних системах, таких як Windows, macOS і Linux.

5) Легка інтеграція з іншими технологіями: Python можна легко інтегрувати з іншими технологіями, такими як бази даних та веб-фреймворки, що дозволяє легко створювати веб-додатки для гри в шахи або зберігати дані про ігри в базі даних.

2.2. Stockfish у якості ігрового рушія

У якості ігрового рушія було обрано Stockfish. Stockfish - це шаховий рушій, який був вперше випущений у 2008 році. Він був розроблений групою волонтерів і поширюється під ліцензією GNU General Public License. Рушій написаний на C++ і оптимізований під сучасні процесори, що робить його надзвичайно швидким та ефективним.

Stockfish - це програма командного рядка, яку можна запускати на різних платформах, включаючи Windows, macOS, Linux і Android. Вона також підтримує універсальний шаховий інтерфейс (UCI) і протокол зв'язку шахового рушія (CECP), що дозволяє інтегрувати її з іншим шаховим програмним забезпеченням, таким як графічні інтерфейси користувача (GUI) або шахові сервери в Інтернеті.

Stockfish використовує різноманітні алгоритми для оцінки шахових позицій та ходів. Він використовує форму альфа-бета-обрізки, яка називається пошуком головної варіації (PVS), щоб дослідити дерево гри і знайти найкращі ходи. Він також використовує різні евристики для оцінки позицій, включаючи матеріальний баланс, рухливість фігур, структуру пішаків і безпеку короля. Ці алгоритми та евристики постійно вдосконалювалися протягом багатьох років, що призвело до постійного вдосконалення Stockfish і домінування в комп'ютерних шахах.

Stockfish використовується для різних цілей, окрім гри в шахи. Її використовували в дослідженнях штучного інтелекту і машинного навчання, а також для створення шахових головоломок і задач. Він також використовувався при розробці шахових варіантів та іншого програмного забезпечення, пов'язаного з шахами.

2.3 Переваги та недоліки Stockfish в порівнянні з аналогами

Але Stockfish – не єдиний варіант в даному випадку. То чому ж було обрано саме цю опцію? Порівняємо Stockfish з аналогами, такими як LeelaChess Zero та Houdini.

Stockfish vs LeelaChess Zero

Архітектура: Stockfish - це традиційний шаховий рушій, який використовує підхід грубої сили для пошуку найкращого ходу. Він оцінює позиції на основі евристик та алгоритмів, розроблених людьми-програмістами. Leela, з іншого боку, є нейромережевим рушієм, який використовує машинне навчання для оцінки позицій. Вона вчиться, граючи в ігри проти себе, і з часом покращує свою гру.

Стиль гри: Stockfish відома своєю тактичною та позиційною силою, і її часто хвалять за здатність знаходити найкращі ходи в складних позиціях. Ліла, з іншого боку, відома своїм більш творчим і неординарним стилем гри, який іноді може призвести до несподіваних і дивовижних ходів.

Відкрите джерело: І Stockfish, і Leela мають відкритий вихідний код і вільно доступні для завантаження. Однак Leela має більш децентралізовану модель розробки, з внесками від багатьох людей та організацій, тоді як Stockfish має більш централізовану команду розробників.

Вимоги до ресурсів: Для повноцінної роботи Leela потребує більше обчислювальних ресурсів, ніж Stockfish. В той час як Stockfish можна запустити на одному ядрі процесора, Leela потребує відеокарти високого класу (GPU), щоб досягти максимальної ігрової потужності.

Інтеграція з іншим програмним забезпеченням: Stockfish широко інтегрована з іншим шаховим програмним забезпеченням, таким як графічні інтерфейси та шахові сервери. Leela також сумісна з багатьма графічними інтерфейсами та онлайн-серверами, але може потребувати додаткового налаштування для належної роботи.

Stockfish vs Houdini

Тут взагалі дуже просто: Houdini, на відміну від Stockfish, є комерційним продуктом, а тому вільно і безкоштовно користуватися послугами цього рушія не можливо.

Таким чином, проаналізувавши ринок шахових рушіїв, Stockfish виявляється найкращим варіантом для реалізації додатку з вимогами, вказаними вище у цій курсовій роботі, і з можливостями, які має студент Києво-Могилянської академії: швидкий, ефективний open-source продукт, який задовільняє фактично будь-які потреби пересічного користувача.

3. Концепція ігрового штучного інтелекту в шахах

3.1 Розрахункова шахова функція

Для того, щоб комп'ютер почав грати в шахи, нам необхідно, що він мав змогу робити хоча б ті ходи, що не суперечать правилам гри. Тобто, запрограмувати комп'ютер здійснювати випадкові легальні ходи. Але проблема в тому, що, в середньому, на кожному ході білих або чорних існує до 50 можливих ходів. Серед них, за власним досвідом можу сказати, існує до 5 ходів, які суттєво не змінюють становище на дошці у бік суперника. За умови, що комп'ютер робитиме ходи випадковим чином, ймовірність того, що хід, зроблений ним, хоча б не зробить позицію значно гіршою скаладає близько 10 відсотків. Ймовірність повторення такого ж успіху, за умови, що перший хід був зроблений правильно сягає взагалі 1 відсотку. Для цього потрібно якось виокремити оптимальні ходи, що ведуть до мату ворожого короля, від усіх інших.

Перше, що спадає на думку, це повний перебір усіх можливих ходів до того моменту, поки один із варіантів не призведе до мату. Але, як зазначалося вище, в середньому на кожному ході існує 50 варіантів (і їхнє число зростає з кожним ходом в геометричній прогресії, що результує у безперечне збільшення об'єму обчислювальних потужностей для перебору), серед яких лише мала частина призводять хоча б не до погіршення ситуації на дошці. У даній ситуації логічним кроком буде обмежити перебір усіх можливих розгалужень до, наприклад, 5 ходів у глибину. У силу того, що в більшості позицій мату в найближчі 5 ходів зазвичай не існує, то потрібно визначити дещо складнішу міру оцінки оптимальності ходу. Тут на допомогу приходить оціночна функція.

Функція оцінки може бути простим індикатором того, чи був поставлений мат даній стороні, але для пошуку з обмеженою глибиною це не є корисним. Більш досконала оцінка полягає у

зваженому підрахунку фігур на кожній стороні. Наприклад, якщо у білих немає ферзя, а у чорних є, то позиція незбалансована, і білі перебувають у скрутному становищі. Якщо у білих на три пішаки менше, але у них є зайвий слон, позиція, скоріше за все, збалансована. Функція оцінки використовує таблицю цінності кожної фінури у пішаковому еквіваленті

При програмуванні комп'ютера для гри в шахи використання функції оцінки має вирішальне значення. Ця функція дозволяє алгоритму слідувати загальним правилам, таким як захоплення важливих фігур, по типу тур та ферзів, та уникнення жертвування важливими фігурами заради виведення пішаків. Однак більш просунутий метод підрахунку очок враховує не лише фігури, але й їхнє розташування на дошці. Присвоюючи значення розташуванню пішаків і коней, а також визнаючи силу тур у ворожих рядах і стратегічну позицію короля, алгоритм може розвинути дивовижну позиційну інтуїцію. Враховуючи тонкі закономірності, такі як структура пішаків, сполучені тури та рухливість короля, оцінка може бути ще більш досконалою.

Рис. 3.1.1 - Оцінка позиції до помилки суперника – скріншот власної партії

Нижче наведений приклад, як один неточний хід, і, здавлося б, лише втрата коня перетворює становище білих на катастрофу. Далі намагатимемося розібратись, чому комп'ютер оціню цю позицію саме так.

Рис. 3.1.2 – Оцінка позиції після помилки суперника – скріншот власної партії

3.2. Мінімакс пошук найкращих ходів

Щоб визначити найкращий хід для гри, в шахових системах використовується алгоритм пошуку. Завдяки функціям оцінки алгоритми пошуку можуть обмежувати глибину пошуку, але при цьому приймати обґрунтовані рішення. Дерево пошуку використовується для представлення дошки і всіх можливих ходів, які можна зробити з певної позиції на дошці. Вихідна дошка є кореневим вузлом дерева, а всі вузли, що розгалужуються від нього, є досяжними позиціями.

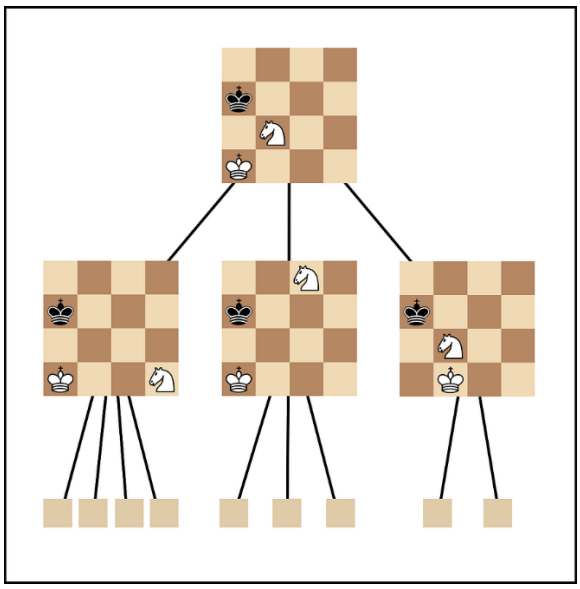

Рис 3.2.1 - Представлення шахових розгалень у вигляді дерева – [1]

Для порівняння можливих ходів у шахових рушіях використовується алгоритм мінімаксу. Оскільки шахи - це гра з нульовою сумою, максимізація своїх шансів на перемогу еквівалентна мінімізації шансів на перемогу суперника. Кожен хід гравець намагається максимізувати функцію оцінки, а інший - мінімізувати її. У дереві пошуку це означає вибір дочірніх вузлів з найкращими або найгіршими оцінками.

Хоча алгоритм мінімаксу є теоретично оптимальним, він неймовірно повільний і непрактичний через високий коефіцієнт розгалуження дерева пошуку. Для вирішення цієї проблеми була розроблена варіація мінімаксу Шеннона типу B. Він розглядає лише декілька найкращих ходів у вузлі замість того, щоб перебирати всі можливі ходи. Ходи-кандидати визначаються оціночною функцією або евристикою на кшталт "перевірки-захоплення-загрози". Цей підхід зменшує коефіцієнт розгалуження дерева пошуку, що дозволяє здійснювати пошук на більшу глибину. Однак він схильний пропускати важливі тактики і потрапляти в пастки.

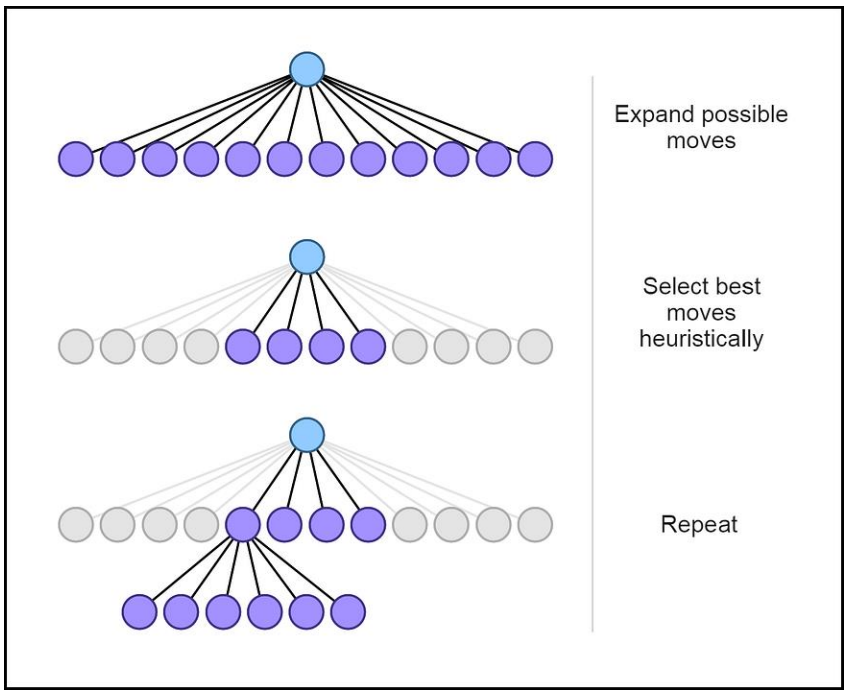

Рис. 3.2.2 - Візуалізація мінімакс пошуку - [1]

У 1970-х роках було відкрито оптимізацію мінімаксу, відому як альфа-бета обрізка. Цей підхід зробив підхід типу А до мінімаксу життєздатним. Вибір ходів-кандидатів для алгоритму типу В був порівняно дорогим в обчислювальному плані, через що він втратив популярність.

3.3. Алфа-бета обрізання

Шахісти знають, що деякі ходи просто погані, тому що дають іншому гравцеві значну перевагу. Однак стандартні алгоритми мінімаксу розглядають всі можливі ходи, включаючи погані. А це, як виявилося вище, істотна частина усіх можливих ходів. Це спричиняє невипрадану ресурсозатратність для реалізації мінімакс пошуку.

Альфа-бета обрізка - це оптимізація мінімаксу, яка покращує його швидкість, ігноруючи нерелевантні вузли в дереві пошуку. Це

досягається додаванням додаткової інформації до кожної вершини у вигляді "альфа" і "бета" значень, що представляють найгірший можливий результат для кожного гравця з цієї вершини. Гравець, який максимізує виграш, може не розглядати ходи, які дозволяють гравцю, що мінімізує виграш, погіршити свою позицію, тому вони відсікаються від дерева.

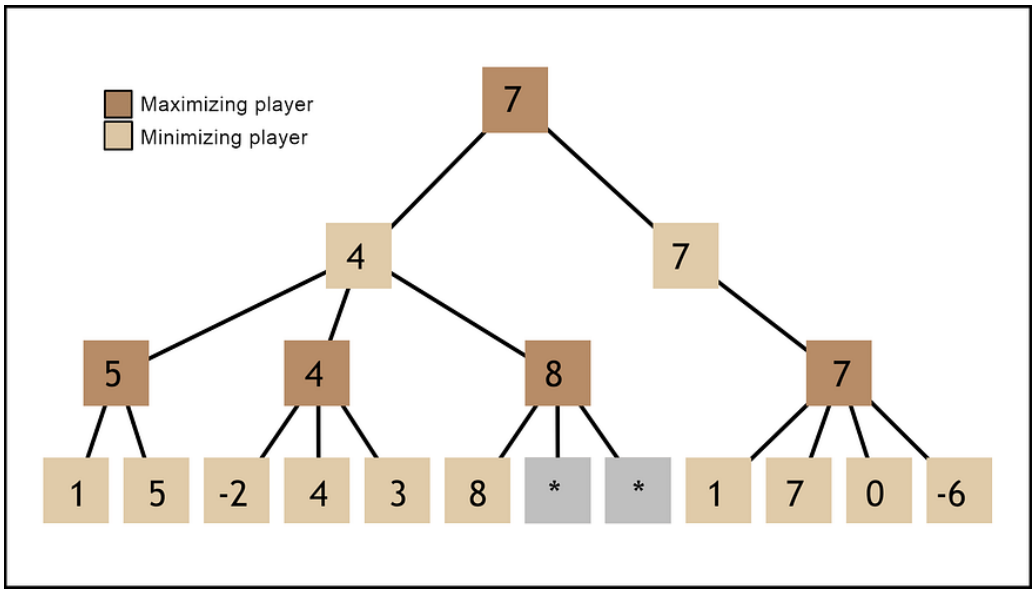

Рис. 3.3.1 - Гравцеві, що мінімізує позицію не має сенсу розглядати продовження варіанту з оцінкою 8, адже це суперечитиме його умові вибору ході. Відповідно, повний вузол зі значенням 8 можна відкидати – [1]

Альфа-бета обрізання - це коректний і ефективний алгоритм пошуку в дереві, який дає такі ж результати, як і стандартний мінімакс. Однак, його покращення продуктивності не завжди є суттєвим. Алгоритм не має системи визначення пріоритетності вузлів для відвідування, що може призвести до того, що неоптимальні ходи будуть досліджуватися раніше, ніж кращі.

Природнім кроком для вирішення цієї проблеми виглялає сортування можливих продовжень за оцінкою позицій після них, і розгляд відповідного (в залежності від того, чи ви максимізуючий гравець, чи мінімізуючий) у першу чергу. Таким чином, варіантів, що не розглядатимуться занадто глибоко, стане менше.

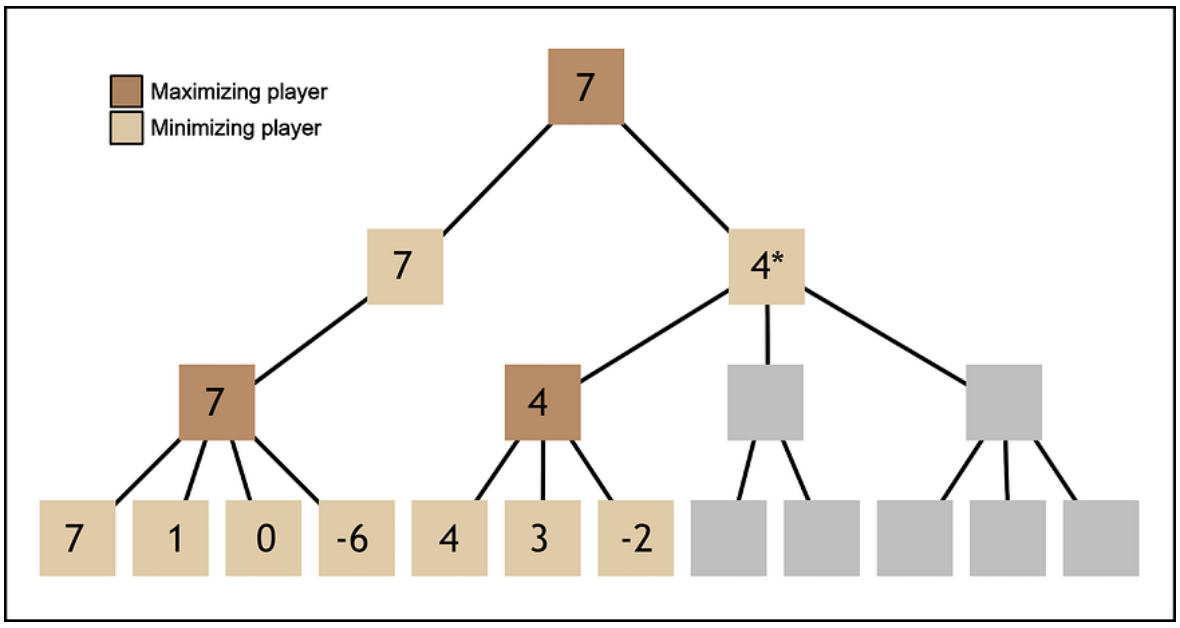

Рис. 3.3.2 - При сортуванні ходів за оцінкою у тій самій позиції мінімізуючому гравцеві достатньо прорахувати лише одну гілку дерева замість трьох – [1]

3.4. Цінність кожної клітинки на дошці

Також при розрахунку оцінки позиції безумовно треба враховувати і цінність клітинок, на яких знаходяться фігури, для ще більш ефективного перформансу. Нижче наведено відносну цінність клітинок. На цифри в цьому випадку не слід звертати увагу, адже головне в цій «системі координат» - це порядок їх розташування у ієрархії.

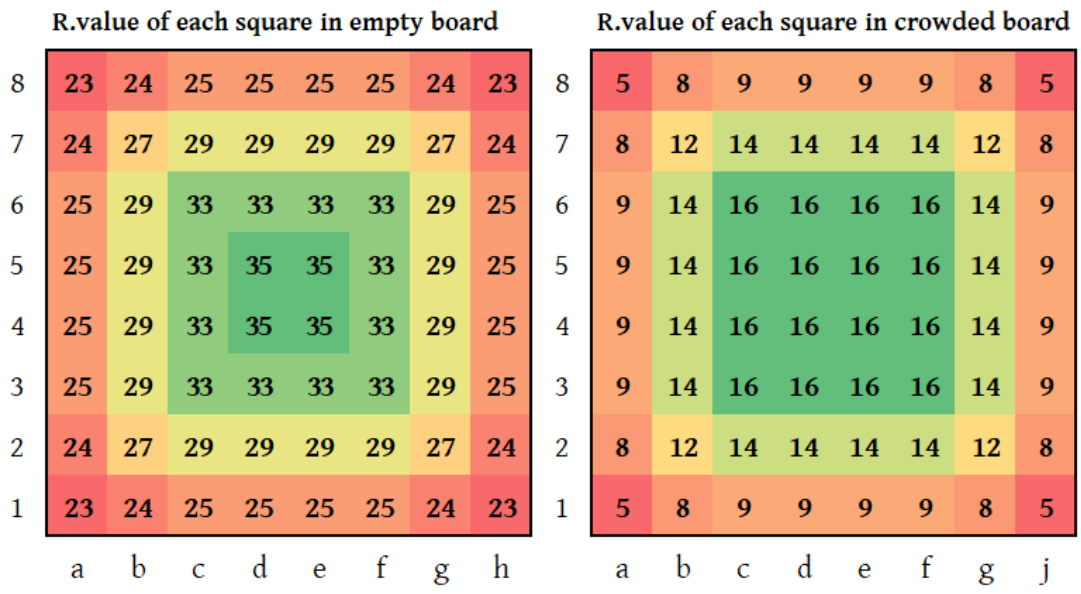

Рис 3.4 - Одна з варіацій представлення цінностей кожної шахової клітини на дошці – [2]

І дійсно, такий розподіл виглядає природнім: принаймні, якщо порівнювати клітинки, що атаковані фігурою, то, відповідно, ті фігури, які стоять у центрі дошки, атакують більше клітинок. З цього випливає, що займаючи центральні клітини, фігура автоматично стає більш цінною, аніж та сама, але яка стоїть у куті. І боротьба за ці клітини повинна пріоритизуватися алгоритмом. Саме тому одними з головних задач шахового дебюту є виведення фігур у центр дошки, а також боротьба (атака) за центральні клітини за допомогою своїх пішаків.

3.5. Цінність фігур у різних видах позицій

Для фіналізації базових вхідних даних, необхідних для достатньо ефективного розрахунку оцінки позиції, потрібно також зазначити, що одні і ті самі фігури в різних видах позицій можуть «коштувати» по-різному. Але для початку наведемо базову цінність кожної фігури на першому ході партії

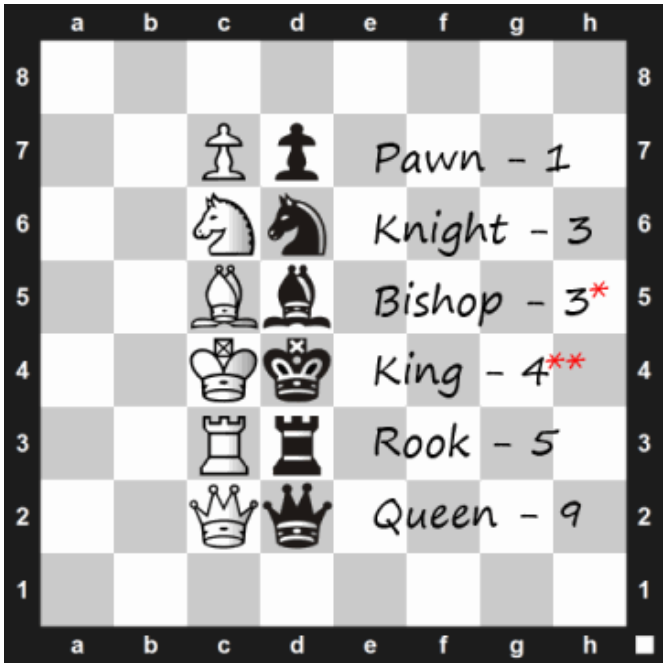

Рис 3.5.1 – Загальноприйнята таблиця цінностей шахових фігур у пішаковому еквіваленті – [3]

Зірочка поряд з позначенням «вартості» слона означає те, що при розрахунку слон коштує трохи дорожче за коня, адже за своєю

сутністю він банально атакує більше ворожих (і, відповідно, захищає більше своїх клітинок), що дуже сильно впливає на цінність слона в багатьох позиціях (особливо у відкритих, де на дошці достатньо мало фігур). Єдиним винятком є закриті позиції з великою кількістю пішаків, де кінь своїм унікальним перестрибуванням через фігури отримує перевагу.

Дві зірочки біля короля, очевидно, означають, що його ліквідація є основним завданням опонента, і власне цінність короля у 4 умовних пішаки ніяк не впливає на розрахунок оптимального ходу. Дана кореляція присутня тут радше для того, щоб продемонструвати силу/слабкість короля відносно інших фігур, якби унеможливлення варіантів захисту від атаки на короля не було головною метою обох гравців.

Звернувшись до вихідного коду Stockfish, можна також побачити, що цінність деяких фігур змінюється відповідно до умов, що наявні на дошці в даний час. Наприклад, слон чи кінь, що стоять на захищенному своїм пішаком полі (форпості), алгоритмом визначаються як більш цінні в порівнянні зі своїм початковим станом.

```
constexpr Score Outpost[] = { S(54, 34), S(31, 25) };
```

Рис. 3.5.2 - Фрагмент коду Stockfish, що враховує перебування легкої фігури на форпості при розрахунку її цінності в данній позиції – [4]

Дана змінна показує, що кінь (перше значення пари) або слон (друге значення), що стоять на форпості (клітинці, яку захищає власний пішак), отримують бонус до власної вартості, знаходячись на 4/5 лінії (перша пара) та 3/6 лінії відповідно. Також варто зазначити, що в підрахунках завжди використовується симетричність шахової дошки, тому значень удвічі менше, адже вони попарно рівні.

Так само можна помітити, що цінність пішака неминучо

зростає зі зближенням його до останнього рядка, де він перетворюється на будь-яку іншу фігуру, окрім короля.

```
// PassedRank[Rank] contains a bonus according to the rank of a passed pawn
constexpr Score PassedRank[RANK_NB] = {
  S(0, 0), S(2, 38), S(15, 36), S(22, 50), S(64, 81), S(166, 184), S(284, 269)
};
```

Рис. 3.5.3 - Фрагмент коду Stockfish, що враховує перебування пішака відносно початкової позиції при розрахунку його цінності в данній позиції – [4]

Тури також зазнають змін за різних обставин на дошці. Найбільш впливовим чинником є наявність відкритих вертикалей та наявність на них власне тур. Нижче наведена різниця в умовній вартості тури на закритій вертикалі та на відкритій відповідно:

```
constexpr Score RookOnClosedFile = S(10, 5);
constexpr Score RookOnOpenFile[] = { S(18, 8), S(49, 26) };
```

Рис. 3.5.4 Фрагмент коду Stockfish, що враховує перебування тури на відкритій вертикалі при розрахунку її цінності в данній позиції – [4]

Підсумовуючи цей розділ, варто зазначити, що алгоритм розрахунку оцінки шахової позиції тільки на перший погляд здається складним. Якщо ж поринутись ширше в шахову теорію, а саме прийняти той факт, що фігури, які тримають під контролем більше клітинок, знаходячись при цьому на важливих полях, вважаються ціннішими за ті, що нічого з цього не виконують, то навіть процес оцінки шахової позиції в режимі реального часу власним мозком стане набагато простішим та ефективнішим.

4. Реалізація застосунку

4.1 Графічний інтерфейс

При запуску користувач бачить перед собою вікно з текстовим інпутом, який зчитує введений користувачем текст, а саме відповідну шахову позицію у FEN-нотації (див. Примітка). Зроблено це для того, щоб аналізувати будь-яку шахову позицію, не переставлючи безпосередньо фігури по дошці.

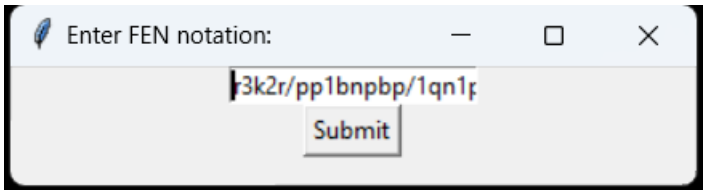

Рис 4.1.1 – Вікно інпуту позиції у FEN нотації

При натисканні на кнопку підтвердження, користувач потрапляє на віртуальну шахову дошку зі згенерованою за допомогою нотації FEN (див. Примітка) позицією фігур.

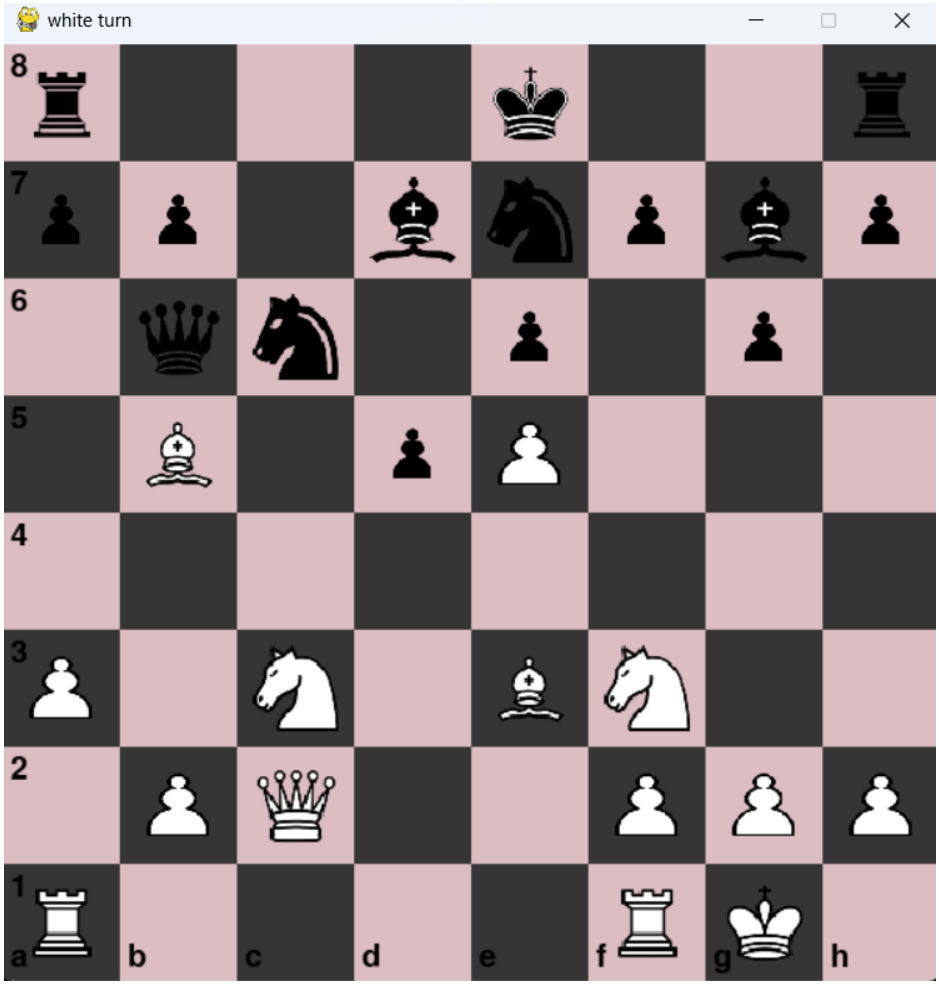

Рис 4.1.2 – Віртуальна дошка для аналізу позицій

(Примітка): Нотація FEN - це стандартна нотація для опису певної шахової позиції за допомогою символів ASCII. FEN розшифровується як нотація Форсайта-Едвардса, названа на честь її винахідників, англійця Девіда Форсайта та австралійського журналіста Стівена Едвардса. Нотація складається з шести полів, розділених пробілом, причому поля розташовані в наступному порядку:

- Розміщення фігури на дошці (з точки зору білих)
- Активний колір (або "w", або "b" для білих або чорних відповідно)
- Доступність рокіровки (KQkq для рокіровки на ферзевому/королівському фланзі для білих/чорних, відповідно; '-', якщо рокіровка недоступна)
- Прохідна клітина цілі (в алгебраїчній нотації, '-', якщо вона недоступна)
- Лічильник півходів (кількість півходів з моменту останнього взяття або ходу пішака, використовується для визначення нічиєї за правилом 50 ходів)
- Кількість повних ходів (кількість повних ходів, зіграних у грі на даний момент)

І після кожного зробленого ходу, у консоль виводиться наступний накращий, на думку шахового рушія, хід у такому форматі:

```
best move e3b6 for white
best move a7b6 for black
best move a7b6 for black
best move c2e2 for white
best move b5c6 for white
best move a8a5 for black
best move e8g8 for black
best move b5c6 for white
best move b5c6 for white
best move d7c6 for black
```

Рис. 4.1.3 – Вивід найкращого продовження у консоль

4.2 Структура проєкту

Даний застосунок реалізований за принципами ООП, де логіка кожного об'єкту реалізована в окремому класі.

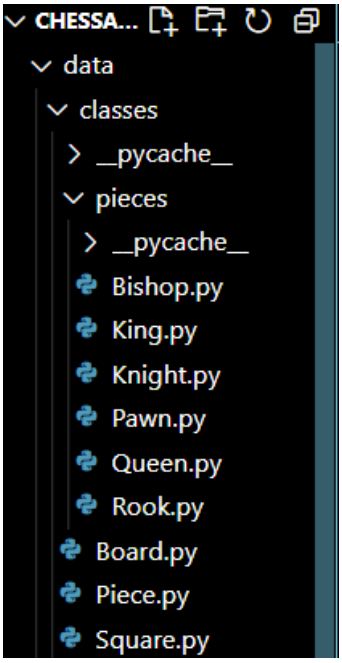

Рис. 4.2 – Структура класів застосунку

У перших шести файлах реалізована логіка поведінки на шаховій дошці відповідних фігур.

Square.py відповідає за логіку і відображення окремої клітинки (перевірка, чи стоїть на ній фігура, чи може конкретна фігура в даний хід ступити на неї та її абсолютні координати)

У Board.py реалізована логіка та відображення шахової дошки залежно від її стану (чи на даний момент королю шах або мат)

Piece.py є файлом, де описана загальна логіка поведінка фігури, незалежно від її варіанту. Тут імплементовані методи перевірки, чи можна в даний момент кокретну фігури побити іншою відповідною фігурою, також «підвищення» пішака і процес рокіровки короля.

Висновки

Результатом виконання цієї роботи став додаток-аналізатор шахових позицій з мінімально необхідним для цього функціоналом для дослідження різних шахових позицій без безпосереднього переставляння фігур по дошці, як би це було у реальному житті.

Застосунок може стати у нагоді людям, які вимагають швидкого і достатньо ефективного аналізу, наприклад, своєї нещодавно зіграної партіїї, коли у пам'яті лишилась тільки певна позиція тої партії.

В ході розробки було досліджено мову програмування Python та її бібліотеки для візуалізації Pygame і Tkinter, а також бібліотеку Stockfish, що дозволяє швидко та легко під'єднати відповідний ігровий рушій до вашого проєкту і комунікувати з ним.

Також було розібрано алгоритм підрахунку оцінки шахової позиції, який так само використовує Stockfish для своєї роботи, і розібрано деякі деталі його роботи саме «під капотом»

Перспектив у цього додатку достатньо. Насамперед, це виведення декількох найкращих продовжень на екран з відповідною оцінкою позиції, а також можливість пересуватись між цими варіантами безпосередньо у застосунку.

Список використаної літератури

1) The Anatomy of Chess AI [електронний ресурс] https://medium.com/@SereneBiologist/the-anatomy-of-a-chess-ai-2087d0d565
2) The relative values of the chess squares, files, ranks and layers of the chessboard [електронний ресурс] https://www.chess.com/blog/HasanElias/the-relative-value-of-the-squares-files-ranks-and-the-layers-of-the-chessboard
3) Chess pieces value chart [електронний ресурс] https://chessfox.com/chess-pieces-value-chart/
4) Official Stockfish github [електронний ресурс] https://github.com/official-stockfish/Stockfish/tree/master
5) Chess using Pygame tutorial [електронний ресурс] https://github.com/x4nth055/pythoncode-tutorials/tree/master/gui-programming/chess-game - тут надихнувся структурою класів проєкту