НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих

комп'ютерних систем

До захисту допущено:

Завідувач кафедри

_____Віталій РОМАНКЕВИЧ

« » 2026 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою

«Системне програмування та спеціалізовані комп'ютерні системи»

спеціальності 123 «Комп'ютерна інженерія»

на тему: «Розробка гібридного шахового рушія із інтеграцією оціночного ядра та пояснювального модуля на основі мовної моделі ШІ»

Виконав:
студент IV курсу, групи КВ-21
Гуманіцький Андрій Олександрович _____

Керівник:
Асистент доктор філософії,
Сергієнко Павло Анатолійович _____

Консультант з нормоконтролю:
Доцент, к.т.н., Клятченко Ярослав Михайлович

_____

Рецензент:
_____ _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.

Студент (-ка) Гуманіцький А.

_____

Київ – 2026 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра системного програмування і
спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Системне програмування та спеціалізовані комп'ютерні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Віталій РОМАНКЕВИЧ
« » 2026 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Гуманіцькому Андрію Олександровичу

1. Тема проєкту «Розробка гібридного шахового рушія із інтеграцією оціночного ядра та пояснювального модуля на основі мовної моделі ШІ», керівник проєкту Сергієнко Павло Анатолійович асистент, PhD, затверджені наказом по університету від «___»_____2026 р. №_____

2. Термін подання студентом проєкту _____

3. Вихідні дані до проєкту Технічне завдання

4. Зміст пояснювальної записки

1. Аналіз існуючих рішень та обґрунтування теми дипломного проєкту;
2. Програмні та апаратні засоби для розробки додатку;
3. Опис реалізації розробленого додатку;
4. Тестування розробленого додатку;
5. Аналіз отриманих результатів.

6. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

1. Алгоритм роботи оціночного ядра рушія
2. Алгоритм оцінки бонусів в оціночному ядрі
3. Алгоритм оцінки мобільності та пішакової структури в оціночному ядрі
4. Алгоритм оцінки позиції короля та його можливостей в оціночному ядрі
5. Алгоритм оцінки розвитку фігур, контролю центру та перевірки висячих фігур в оціночному ядрі
6. Алгоритм оцінки ендшпільних можливостей в оціночному ядрі

7. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Доцент, к.т.н., Клятченко Ярослав Михайлович		

8. Дата видачі завдання 02.09.2025

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення літератури за тематикою проєкту	07.01.2026	
2	Розроблення та узгодження технічного завдання	17.02.2026	
3	Аналіз існуючих рішень	02.03.2026	
4	Підготовка матеріалів першого розділу дипломного проєкту	10.03.2026	
5	Підготовка матеріалів другого розділу дипломного проєкту	27.03.2026	
6	Підготовка графічної частини дипломного проєкту	20.04.2026	
7	Оформлення документації дипломного проєкту	25.05.2026	
8	Попередній огляд матеріалів диплому на кафедрі	30.05.2026	

Студент Андрій ГУМАНІЦЬКИЙ

Керівник Павло СЕРГІЄНКО

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (60 с., 49 рис., 2 табл., 7 додатки).

Об’єкт розробки – гібридний шаховий рушій із інтеграцією оціночного ядра та пояснювального модуля на основі мовної моделі штучного інтелекту.

Розроблена система дозволяє виконувати аналіз шахових позицій, знаходити рекомендовані ходи, формувати текстові пояснення щодо вибору ходу та загального характеру позиції, а також адаптувати подачу пояснень до рівня підготовки користувача. Система може бути використана як основа для навчальних, аналітичних та тренувальних шахових застосунків. У процесі розробки було використано засоби програмування C++, алгоритми генерації шахових ходів і пошуку найкращого продовження, підходи до оцінювання шахових позицій, а також методи формування пояснень на основі мовної моделі штучного інтелекту.

В ході розробки:

- проведено аналіз існуючих рішень у сфері шахового аналізу та навчання;
- виконано аналіз підходів і алгоритмів, що застосовуються в шахових рушіях;
- сформульовано вимоги до гібридного шахового рушія з пояснювальним модулем;
- розроблено шахове оціночне ядро для аналізу позицій і вибору ходів;
- розроблено пояснювальний модуль на основі мовної моделі штучного інтелекту;
- розроблено структуру взаємодії між аналітичним і пояснювальним компонентами системи;
- розроблено користувацький інтерфейс для взаємодії з системою.

Упровадження розробленої системи дозволить підвищити зрозумілість шахового аналізу для користувачів, спростити навчання початківців і гравців середнього рівня та покращити ефективність використання шахових програм у навчальному й практичному середовищі.

Ключові слова:

ШАХИ, ШАХОВИЙ РУШІЙ, ГІБРИДНИЙ ШАХОВИЙ РУШІЙ, ШТУЧНИЙ ІНТЕЛЕКТ, МОВНА МОДЕЛЬ, АНАЛІЗ ШАХОВИХ ПОЗИЦІЙ.

ABSTRACT

The qualification paper includes an explanatory note (60 pp., 49 figs., 2 tables, 7 appendices).

The development object is a hybrid chess engine with an integrated evaluation core and an explanatory module based on an AI language model.

The developed system analyzes chess positions, finds recommended moves, generates textual explanations for move choice and position character, and adapts explanations to the user's skill level. It can serve as a foundation for educational, analytical, and training chess applications. Development used programming tools, move generation and search algorithms, position evaluation methods, and explanation generation based on an AI language model.

- The following tasks were completed:
- existing chess analysis and training solutions were analyzed;
- chess engine approaches and algorithms were analyzed;
- hybrid engine requirements were formulated;
- an evaluation core for analysis and move selection was developed;
- an explanatory module based on an AI language model was developed;
- analytical and explanatory component interaction was implemented;
- a user interface was developed.

The developed system improves the understandability of chess analysis, supports training for beginner and intermediate players, and increases the practical value of chess software in educational settings.

Keywords:

CHESS, CHESS ENGINE, HYBRID CHESS ENGINE, ARTIFICIAL INTELLIGENCE, LANGUAGE MODEL, CHESS POSITION ANALYSIS.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045420.002 ТЗ	Розробка гібридного	4		
			шахового рушія			
			із пояснювальним			
			модулем			
			Технічне завдання			
	А4	ІАЛЦ.045420.003 ТП	Розробка гібридного	3		
			шахового рушія			
			із пояснювальним			
			модулем			
			Відомість технічного			
			проєкту			
	А4	ІАЛЦ.045420.004 ПЗ	Розробка гібридного	60		
			шахового рушія			
			із пояснювальним			
			модулем			
			Пояснювальна записка			

Змін.	Арк.	№ докум.	Підпис	Дата	**ІАЛЦ. 045420.001 ОА**		
Розробив		Гуманіцький А.			Розробка гібридного шахового рушія із пояснювальним модулем Опис альбому **Опис альбому**	Літ.	Аркуш 1 / Аркушів 3
Перевірив		Сергієнко П. А.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-21	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045420.005 Д1	Алгоритм роботи	1		
			оціночного ядра рушія			
	А4	ІАЛЦ.045420.006 Д2	Алгоритм оцінки бонусів в	1		
			оціночному ядрі			
	А4	ІАЛЦ.045420.007 Д3	Алгоритм оцінки	1		
			мобільності та пішакової			
			структури в оціночному			
			ядрі			
	А4	ІАЛЦ.045420.008 Д4	Алгоритм оцінки позиції	1		
			короля та його			
			можливостей в			
			оціночному ядрі			
	А4	ІАЛЦ.045420.009 Д5	Алгоритм оцінки	1		
			розвитку фігур, контролю			
			центру та перевірки			
			висячих фігур в			
			оціночному ядрі			

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045420.010 Д6	Алгоритм оцінки	1		
			ендшпільних			
			можливостей в			
			оціночному ядрі			
	А4	ІАЛЦ.045420.011 Д7	Лістинг коду оціночного	1		
			ядра рушія			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045420.001 ОА	Арк.
						3

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Гібридний шаховий рушій із інтеграцією оціночного ядра та пояснювального модуля на основі мовної моделі ШІ».

Галузь застосування: автоматизований аналіз шахових позицій, навчальні та тренувальні шахові системи, програмні засоби підтримки прийняття рішень у шахах.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проектування на здобуття першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проєкту є розробка гібридного шахового рушія, що поєднує оціночне ядро для аналізу шахових позицій із пояснювальним модулем на основі мовної моделі штучного інтелекту.

Призначенням системи є аналіз шахових позицій, визначення рекомендованих ходів та формування зрозумілих текстових пояснень щодо логіки вибору ходу і загального характеру позиції.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами розробки є технічна та науково-технічна література з шахового програмування, алгоритмів пошуку та оцінювання позицій, документація шахових рушіїв і протоколів взаємодії, матеріали щодо застосування мовних моделей

штучного інтелекту, а також публікації у періодичних виданнях і електронні ресурси мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- коректна реалізація правил шахів;
- генерація всіх допустимих ходів для заданої позиції;
- аналіз шахових позицій і визначення рекомендованого ходу;
- оцінювання позиції на основі оціночного ядра;
- формування текстових пояснень щодо запропонованих ходів;
- можливість адаптації пояснень до рівня підготовки користувача;
- модульна структура програмного забезпечення;
- збереження та обробка шахових позицій у стандартному форматі;
- наявність користувацького інтерфейсу для взаємодії з системою.

5.2. Вимоги до апаратного забезпечення

- процесор з тактовою частотою не менше 2,0 ГГц;
- оперативна пам’ять не менше 4 Гб;
- наявність вільного місця на накопичувачі не менше 500 Мб;
- доступ до мережі Internet.

5.3. Вимоги до програмного та системного забезпечення користувача

- операційна система Windows, Linux.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів
1	Вивчення літератури за тематикою проєкту	15.03.2026
2	Розроблення та узгодження технічного завдання	20.03.2026
3	Аналіз існуючих рішень	27.03.2026
4	Підготовка матеріалів першого розділу дипломного проєкту	01.04.2026
5	Підготовка матеріалів другого розділу дипломного проєкту	20.04.2026
6	Підготовка графічної частини дипломного проєкту	07.05.2026
7	Оформлення документації дипломного проєкту	15.05.2026
8	Попередній огляд матеріалів диплому на кафедрі	27.05.2026

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045420.004 ПЗ	Розробка гібридного	60		
			шахового рушія			
			із пояснювальним			
			модулем			
			Пояснювальна записка			
	А4	ІАЛЦ.045420.005 Д1	Алгоритм роботи	1		
			оціночного ядра рушія			
	А4	ІАЛЦ.045420.006 Д2	Алгоритм оцінки бонусів в	1		
			оціночному ядрі			
	А4	ІАЛЦ.045420.007 Д3	Алгоритм оцінки	1		
			мобільності та пішакової			
			структури в оціночному			
			ядрі			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045420.003 ТП		
Розробив		Гуманіцький А.			Розробка гібридного шахового рушія із пояснювальним модулем **Відомість технічного проєкту**	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Сергієнко П. А.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-21	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045420.008 Д4	Алгоритм оцінки позиції	1		
			короля та його			
			можливостей в			
			оціночному ядрі			
	А4	ІАЛЦ.045420.009 Д5	Алгоритм оцінки	1		
			розвитку фігур, контролю			
			центру та перевірки			
			висячих фігур в			
			оціночному ядрі			
	А4	ІАЛЦ.045420.010 Д6	Алгоритм оцінки	1		
			ендшпільних			
			можливостей в			
			оціночному ядрі			
	А4	ІАЛЦ.045420.011 Д7	Лістинг коду оціночного	1		
			ядра рушія			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045420.003 ТП	Арк.
						2

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
		Диск CD-Rom	Текст пояснювальної			
			записки.			
			Графічні матеріали.			
			Повний код розробленого			
			додатку.			
			Відео з демонстрацією			
			роботи додатку.			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045420.003 ТП	Арк. 3

Пояснювальна записка
до дипломного проєкту

на тему: Гібридний шаховий рушій із інтеграцією оціночного ядра та пояснювального модуля на основі мовної моделі ШІ

Київ - 2026

ЗМІСТ

ДОДАТКИ

Додаток 1. Копії графічних матеріалів

ІАЛЦ.045420.005 Д1. Алгоритм роботи оціночного ядра рушія

ІАЛЦ.045420.006 Д2. Алгоритм оцінки бонусів в оціночному ядрі

ІАЛЦ.045420.007 Д3.Алгоритм оцінки мобільності та пішакової структури в оціночному ядрі

ІАЛЦ.045420.008 Д4.Алгоритм оцінки позиції короля та його можливостей в оціночному ядрі

ІАЛЦ.045420.009 Д5.Алгоритм оцінки розвитку фігур, контролю центру та перевірки висячих фігур в оціночному ядрі

ІАЛЦ.045420.010 Д6.Алгоритм оцінки ендшпільних можливостей в оціночному ядрі

Додаток 2. Фрагменти програмного коду

ІАЛЦ.045420.011 Д7. Лістинг коду оціночного ядра рушія

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

C++ — мова програмування загального призначення, що використовується для реалізації основної логіки шахового рушія.

CMake — кросплатформена система автоматизації складання програмних проєктів.

SFML — мультимедійна бібліотека, що використовується для створення вікна додатку, обробки подій та графічного відображення шахової дошки.

ImGui — бібліотека для створення графічного інтерфейсу користувача в режимі immediate mode.

Python — мова програмування, що використовується для реалізації пояснювального модуля системи.

Perft-перевірка — метод тестування коректності генератора ходів у шахових програмах шляхом підрахунку кількості можливих позицій на заданій глибині.

LLM — велика мовна модель, що використовується для генерації або перефразування текстових пояснень.

UI — інтерфейс користувача, тобто сукупність візуальних елементів, за допомогою яких користувач взаємодіє з програмою.

UX — досвід користувача, тобто загальне враження від зручності, зрозумілості та ефективності використання додатку.

Шах — ситуація в шахах, коли король однієї зі сторін перебуває під атакою фігури суперника.

Мат — ситуація, у якій король перебуває під шахом, і сторона, що захищається, не має жодного легального ходу для уникнення атаки.

Бот — програмний суперник, який автоматично обирає ходи на основі алгоритмів шахового рушія.

FEN — стандартний текстовий формат запису шахової позиції, що містить інформацію про розташування фігур, сторону ходу, рокіровку, взяття на проході та лічильники ходів.

SAN — стандартна алгебраїчна нотація запису шахових ходів, зручна для читання людиною.

UCI — універсальний інтерфейс шахових рушіїв, а також формат запису ходів, наприклад e2e4.

Транспозиційна таблиця — структура даних, що зберігає результати вже проаналізованих позицій для їх повторного використання під час пошуку.

Сантипішак — одиниця оцінювання шахової позиції, де 100 сантипішаків приблизно відповідають вартості одного пішака.

Ендшпіль — заключна стадія шахової партії, у якій на дошці залишається відносно невелика кількість фігур.

Ело — рейтингова система оцінювання сили шахіста або шахового бота.

Quiescence Search — додатковий пошук у нестабільних тактичних позиціях, який дозволяє уникнути передчасного оцінювання позиції під час розміну або атаки.

Хешування Зобриста — метод створення унікального числового ключа для шахової позиції, що використовується для швидкого порівняння позицій і роботи транспозиційної таблиці.

Killer moves — евристика впорядкування ходів, за якої ходи, що раніше спричиняли відсікання в дереві пошуку, перевіряються раніше в подібних позиціях.

Null Move Pruning (NMP) — метод оптимізації пошуку, який дозволяє відсікати частину дерева варіантів, якщо навіть умовна передача ходу супернику не погіршує позицію суттєво.

Late Move Reductions (LMR) — метод скорочення глибини аналізу для ходів, які розглядаються пізніше в списку ходів і з меншою ймовірністю є найкращими.

ВСТУП

Шахи протягом тривалого часу залишаються однією з найпоширеніших інтелектуальних ігор у світі, поєднуючи змагальний, навчальний та аналітичний аспекти. У сучасних умовах розвитку цифрових технологій шахи отримали новий рівень популярності завдяки онлайн-платформам, комп’ютерним рушіям та системам автоматизованого аналізу партій. Це дало змогу значно розширити доступ до якісного тренувального процесу для гравців різного рівня підготовки — від початківців до досвідчених шахістів.

На сьогодні існує велика кількість програмних рішень, здатних виконувати точний аналіз шахових позицій, знаходити оптимальні ходи та оцінювати перспективи подальшого розвитку партії. Проте більшість сучасних шахових рушіїв орієнтована насамперед на силу гри та обчислювальну ефективність, а не на зрозуміле пояснення своїх рішень. У результаті користувач часто отримує числову оцінку позиції або варіант продовження, але не завжди може зрозуміти логіку вибору ходу, стратегічну ідею або тактичне підґрунтя рекомендації, особливо якщо йдеться про гравців початкового та середнього рівня.

Паралельно з цим розвиток систем штучного інтелекту, зокрема мовних моделей, відкриває нові можливості для створення інтелектуальних програмних засобів, здатних не лише виконувати обчислення, а й формувати зрозумілі текстові пояснення. Поєднання класичного шахового аналізу з пояснювальними можливостями мовних моделей дозволяє розглядати новий підхід до побудови шахових систем, у яких важливою є не тільки точність оцінювання, а й інтерпретованість отриманих результатів.

Таким чином, актуальною є розробка гібридного шахового рушія, який поєднує оціночне ядро для аналізу позицій із пояснювальним модулем на основі мовної моделі ШІ. Такий підхід дозволить створити систему, орієнтовану не лише на вибір сильного ходу, а й на подання зрозумілого обґрунтування цього

вибору, що може підвищити ефективність навчання, аналізу партій та взаємодії користувача з шаховою програмою.

Метою дипломного проєкту є розробка гібридного шахового рушія із інтеграцією оціночного ядра та пояснювального модуля на основі мовної моделі штучного інтелекту для аналізу шахових позицій і формування зрозумілих пояснень щодо запропонованих ходів та загального характеру позиції.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1. Актуальність проблеми

Сучасні шахові платформи, рушії та навчальні сервіси надають користувачам широкі можливості для гри, аналізу партій і вдосконалення власних навичок. На сьогодні існує велика кількість рішень, які дозволяють оцінювати позиції, знаходити найкращі ходи, виявляти помилки та навіть формувати індивідуальні рекомендації для навчання. Однак, попри значний розвиток таких систем, проблема зрозумілого та адаптованого пояснення шахових рішень залишається актуальною.

Більшість сучасних шахових систем або орієнтована переважно на силу аналізу й точність обчислень, або надає лише спрощене пояснення окремих помилок чи рекомендацій. У багатьох випадках користувач отримує числову оцінку позиції, варіанти продовження або короткий коментар, але не отримує цілісного й зрозумілого пояснення того, чому саме певний хід є сильним, які стратегічні або тактичні ідеї стоять за вибором рушія, а також як це пояснення слід адаптувати до рівня підготовки гравця.

Особливо важливою ця проблема є для початківців і гравців середнього рівня, для яких сам факт отримання найкращого ходу не завжди є достатньо корисним без доступного обґрунтування. Саме тому актуальною є розробка гібридного шахового рушія, який поєднуватиме потужне оціночне ядро з пояснювальним модулем на основі мовної моделі штучного інтелекту. Такий підхід дасть змогу не лише виконувати сильний аналіз позиції, а й формувати змістовні пояснення, адаптовані до рівня користувача, що зробить систему корисною не тільки з аналітичної, а й з навчальної точки зору.

1.2. Огляд існуючих рішень

Серед сучасних рішень у сфері шахового аналізу та навчання можна виділити кілька платформ і систем, які реалізують окремі важливі функції та підходи до взаємодії з користувачем.

1.2.1. Платформа Chess.com

Платформа Chess.com [3] пропонує зручне середовище для гри, навчання та аналізу партій. Особливу увагу привертає режим гри з тренером (рис. 1.1), у якому користувач може обирати складність суперника, а також отримувати короткий розбір партії після її завершення.

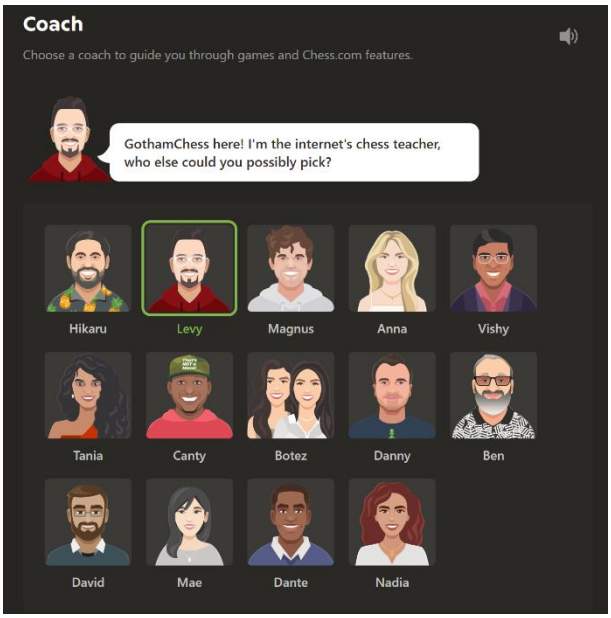


Рисунок 1.1 - Вибірка тренерів на сайті chess.com

Тренерами на вибір являються популярними особистостями у світі шахів, ютуберами, гросмейстерами та декілька штучно створених тренерів. Загалом у кожного з ним є деякі особливості при розборах ігор, свої голосові лінії.

З тренером можна проводити тренувальні ігри (рис. 1.2) при цьому обирати складність, опираючись на свій рівень знань в шахах.

Проте в цьому режимі тренування тренер часто підкреслює свої помилки, які не мають бути типовими для того чи іншого рівня, що дуже помітно. Про грубі помилки гравця часто говориться і під час самої партії. При розборі

(рис. 1.3) ж гри тренер буде коротко пояснювати ходи та казати про опущенні можливості під час партії.

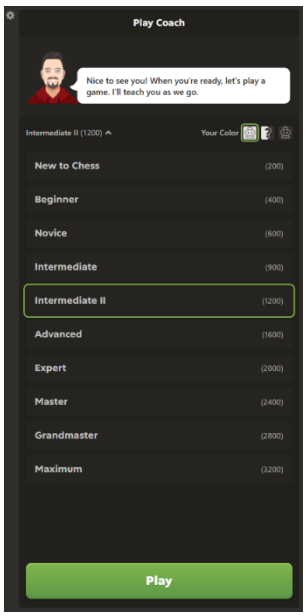
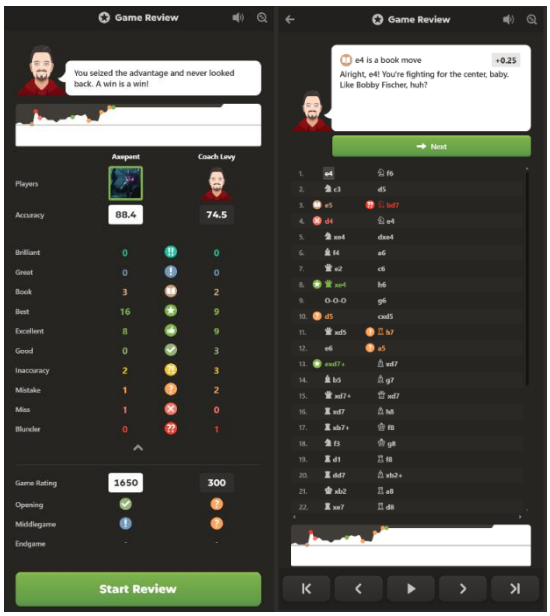


Рисунок 1.2 - Демонстрація можливості вибору складності при грі з тренером



а) б)

Рисунок 1.3 - Аналіз партії після її завершення. а) загальний аналіз гри; б) конкретний аналіз по ходам

1.2.2. Сервіс DecodeChess

Сервіс DecodeChess [6] орієнтований насамперед на пояснення помилок і розбір конкретних позицій. Він дозволяє імпортувати партії з інших платформ, аналізувати окремі критичні моменти гри (рисунки 1.4 – 1.5) та навіть переглядати позицію безпосередньо під час партії.

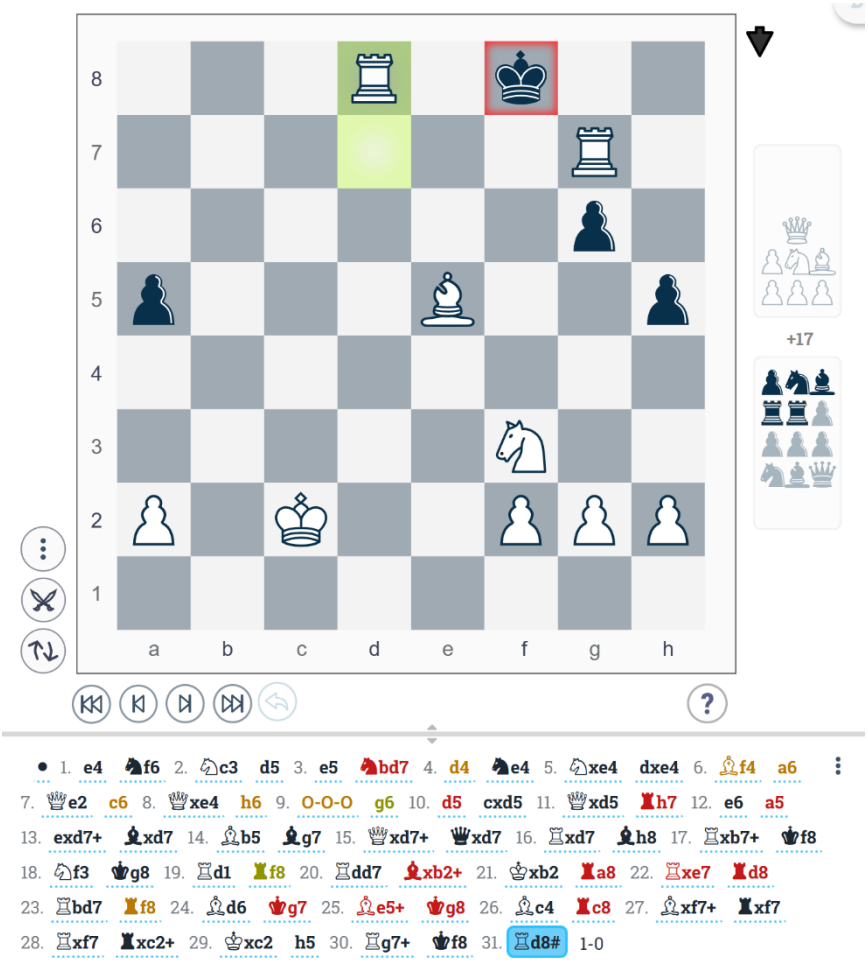


Рисунок 1.4 - Запис самої партії після імпортування

Рисунок 1.5 - Розбір конкретної помилки в партії

Також на сайті є можливість зіграти з ботом, у якого також є вибір складності (рис. 1.6), проте його ранжування не виявляється інтуїтивно зрозумілим.

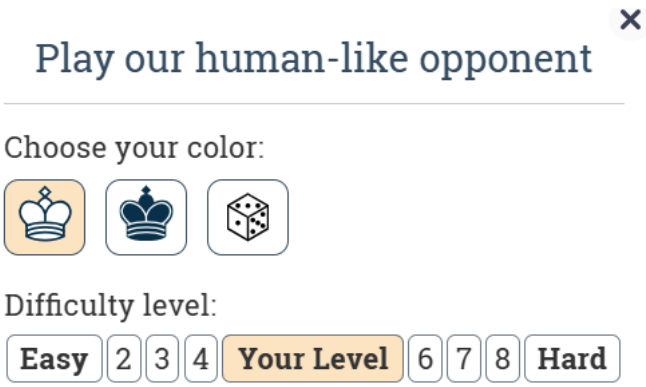
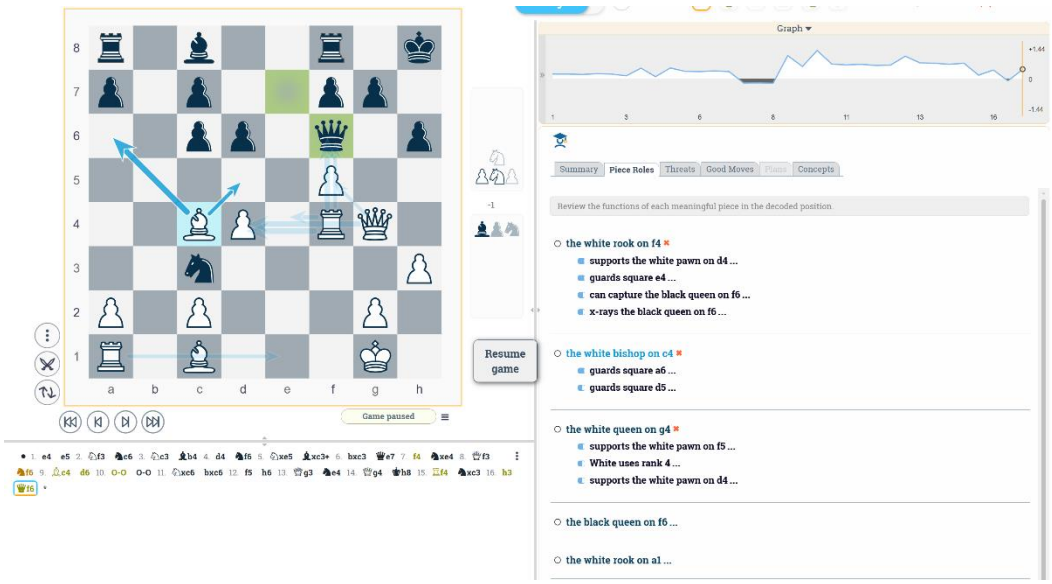


Рисунок 1.6 - Вибір складності на DecodeChess

Цікавою фішкою є можливість будь-який момент гри можна провести аналіз позиції (рис. 1.7), для того щоб зрозуміти, що краще зробити та мати краще уявлення про те як пов’язані між собою фігури в даний момент часу.

а) б)



в)

Рисунок 1.7 - Партія з ботом та розбір позиції під час гри. а) рекомендовані можливості; б) текстовий аналіз; в) загальний вигляд аналізу

На графі аналізу гри (рис. 1.8) видно моменти в яких були допущені помилки (виділені синіми лініями) та момент аналіз позиції (помаранчева лінія на графі).

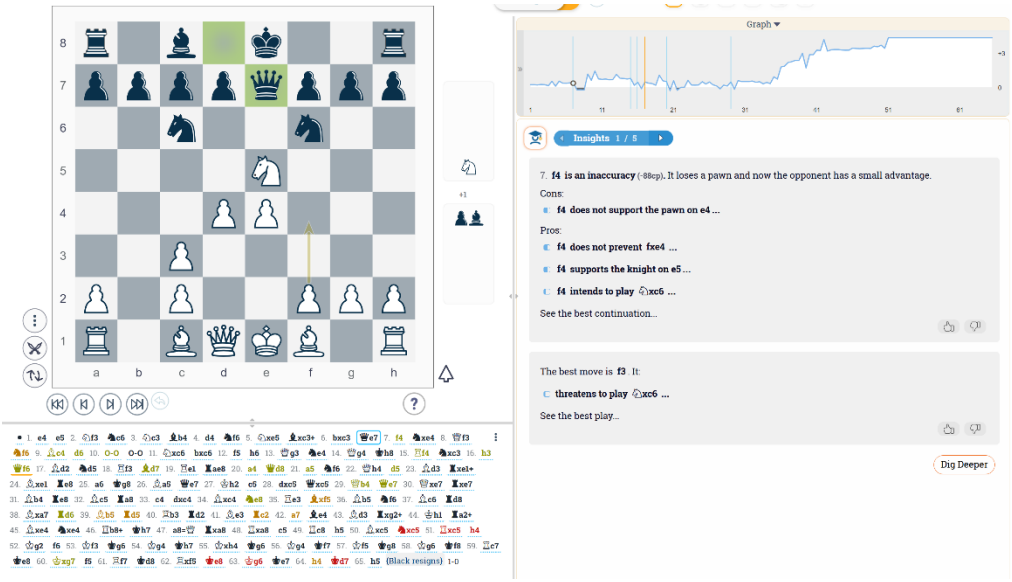


Рисунок 1.8 - Аналіз партії після її закінчення

1.2.3. Проєкт Maia

Проєкт Maia [10] демонструє інший підхід до шахового аналізу, оскільки орієнтується не лише на пошук найсильніших ходів, а й на моделювання людського стилю гри.

Незвичною є функція сайту, “Bot or Not” (рис. 1.9 – 1.11) режим суть якого вгадати хто в партії являється ШІ, а хто реальною людиною. З рис. 1.9 видно, що є можливість продивитися всі ходи партії, проте це вся інформація яка доступна з самого початку.

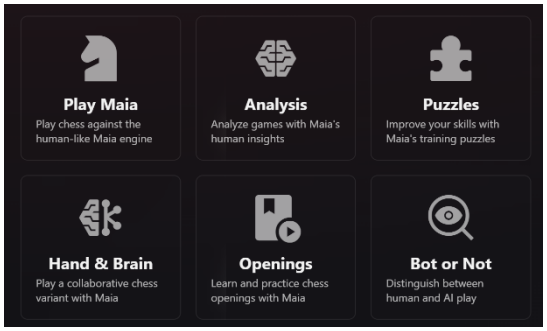


Рисунок 1.9 - Всі режими сайту maiachess.com

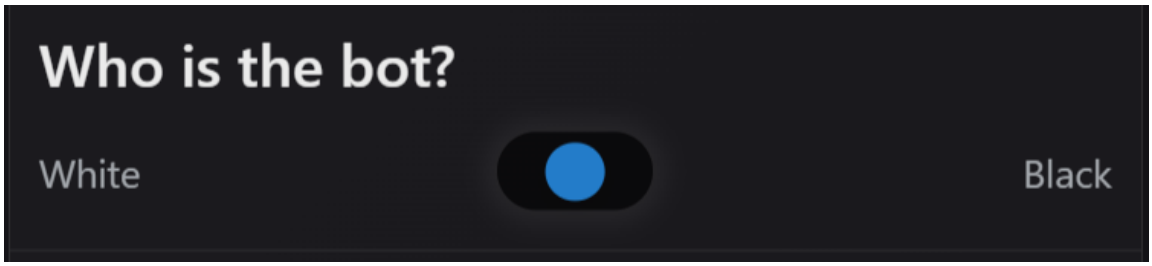


Рисунок 1.10 - Меню для винесення вердикту, який виноситься після закінчення аналізу

Рисунок 1.11 - Інтерфейс режиму “Bot or Not”

Загалом з BOT Maia можна зіграти на lichess.org, цей бот позиціонується як найбільш подібний до людини, з точки зору стилю гри.

Також на сайті присутній режим “Hand & Brain” (рис. 1.12) в якому в момент ходу гравцю видається “Рука” з типом фігури якою можна зробити хід, проте цей режим не є надто цікавим з точки зору пояснення позицій і ходів.

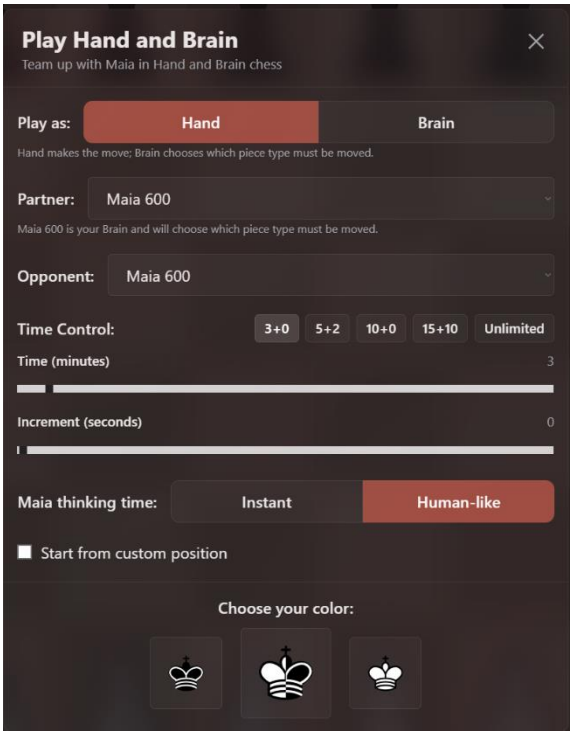


Рисунок 1.12 - Початкові налаштування перед грою в режимі “Hand & Brain”

Цікавість представляє також аналіз, який включає подвійний аналіз від Stockfish 17 та Maia (рис. 1.13), які відповідно представляють найкращі ходи та найбільш людські ходи, ці ходи наглядно показуються стрілочками на ігровому полі та в більш розширеному варіанті таблички в панелі “Analysis”. Також з цікавого – показані процентажі того скільки ходів є помилками і скільки є прийнятних ходів.

Рисунок 1.13 - Режим аналізу на maiachess.com

1.2.4. Платформа Aimchess

Платформа Aimchess [1] спеціалізується на персоналізованому навчанні. Вона аналізує партії користувача (рис. 1.14), виявляє типові помилки та на основі цього формує індивідуальні уроки й рекомендації. Такий підхід є корисним для довгострокового розвитку шахових навичок і демонструє важливість персоналізації в навчальних шахових системах.

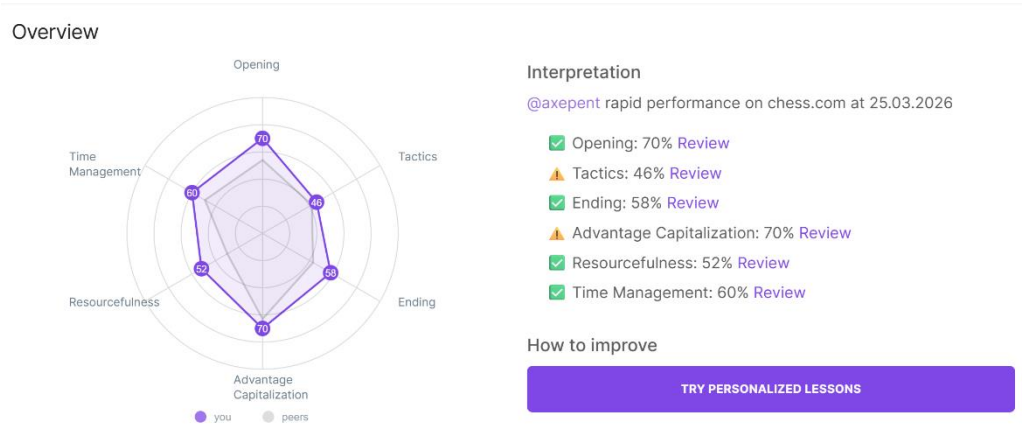


Рисунок 1.14 - Приклад загальна оцінка останніх 40 партій

Аналіз партії та боти (рис. 1.15) не представляють із себе нічого цікавого, порівнюючи з минулими сайтами, оскільки єдине, що є - це шкала оцінки позиції.

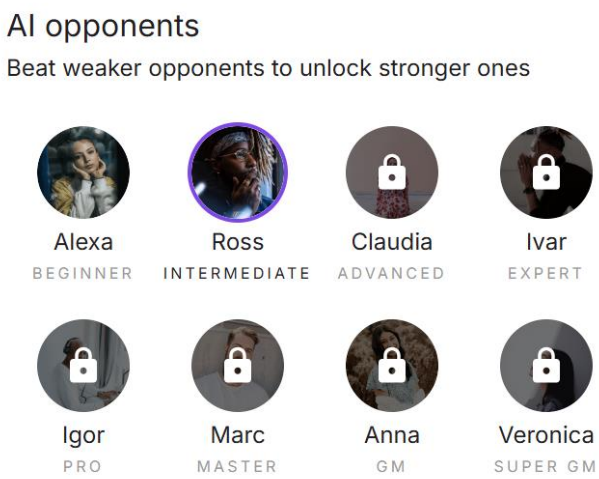


Рисунок 1.15 - Вибір суперника

1.2.5. En Croissant + Stockfish

Окремо варто виділити поєднання En Croissant [8] + Stockfish [15] (GUI + chess engine), яке є прикладом класичного високоточного шахового аналізу. Stockfish виступає як потужне оціночне ядро, здатне проводити глибокий аналіз позиції, тоді як En Croissant надає графічний інтерфейс для зручної взаємодії з рушієм та дозволяє налаштовувати різні технічні параметри аналізу.

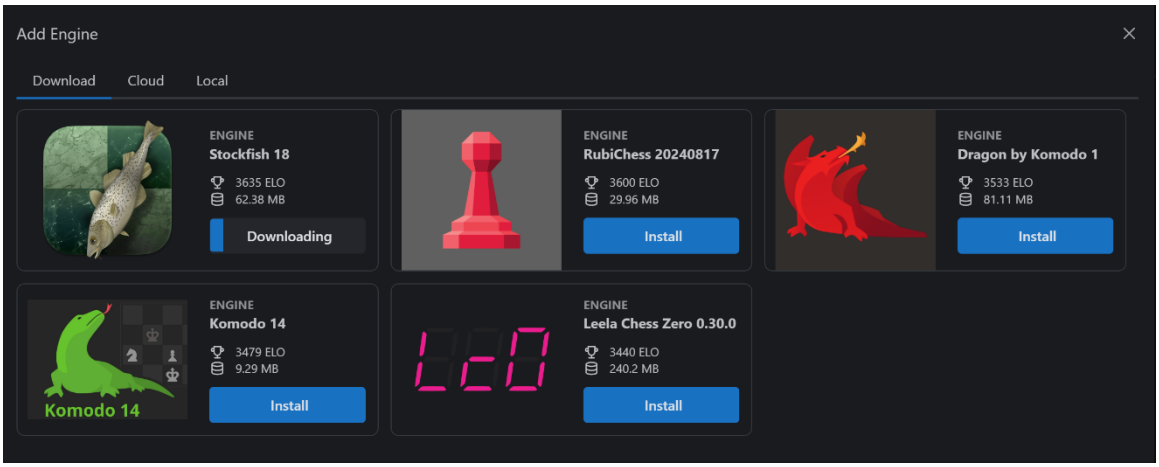


Рисунок 1.16 - Вибір рушіїв, які можливо завантажити

Для використання GUI необхідно завантажувати датасети та відповідно рушії і після всіх налаштувань можна перейти до розгляду режимів гри (рис. 1.17).

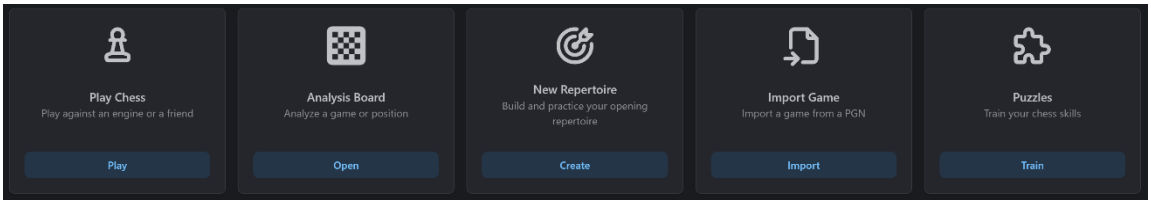
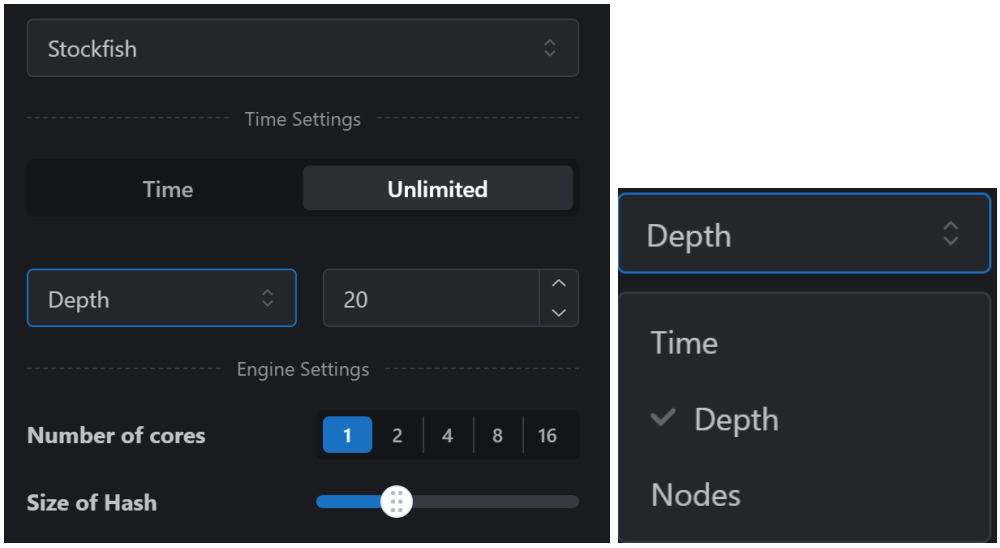


Рисунок 1.17 - Всі режими, які є стандартними в En Croissant

Після завантаження відповідного рушія при початку гри його можна налаштувати.

Отже можна налаштувати: час (для більш давніх моделей це більш суттєво), глибину аналізу (на скільки ходів наперед буде аналізувати рушій), nodes (те на скільки велику кількість розвитку гри буде проаналізовано) та більш технічні налаштування (рис. 1.18).



а) б)

Рисунок 1.18 - Налаштування рушія при початку гри. а) загальні налаштування; б) спливаючий список налаштувань

1.3. Аналіз недоліків розглянутих рішень

Розглянуті платформи та шахові системи реалізують корисні й у деяких випадках доволі функції аналізу, навчання або моделювання стилю гри, проте жодне з них не забезпечує повного поєднання тих властивостей, які є важливими для створення дійсно універсального та сучасного шахового помічника для початківців та гравців середнього рівня.

Платформа Chess.com, попри свою популярність, зручність та добре організоване навчальне середовище, має низку обмежень саме з точки зору глибини пояснення. Режим аналізу гри є цікавим для початкового знайомства, однак коментарі після партії зазвичай мають короткий і дещо спрощений характер. Користувачу вказують на помилки або втрачені можливості, проте пояснення часто не розкриває повною мірою, чому саме конкретний хід був слабшим, яка загальна ідея стояла за кращим продовженням та як це рішення пов’язане із планом у позиції. Також, у тренувальному режимі помітним недоліком є те, що бот іноді припускається помилок, які виглядають штучними та не зовсім відповідають заявленому рівню складності. У результаті формується враження, що адаптація сили гри реалізована не як природне відтворення стилю шахіста певного рівня, а скоріше як навмисне спрощення поведінки рушія, що скоріше за все зроблено для підвищення впевненості користувача, проте така штучна впевненість не може бути корисною в довгостроковій перспективі.

Сервіс DecodeChess вигідно вирізняється прагненням саме пояснювати помилки й аналізувати окремі позиції. Пояснювальні можливості здебільшого зосереджені на конкретних епізодах, тобто користувач може зрозуміти, де було зроблено неточність або який хід виглядав сильнішим у конкретний момент. В сервісі є можливість гри з ботом, сам підхід до ранжування рівнів складності не виглядає достатньо інтуїтивним і прозорим для користувача. Це зменшує зручність використання системи як адаптивного навчального інструмента. У

підсумку DecodeChess радше виконує роль аналітичного помічника або надбудови для розбору вже зіграних партій, ніж повноцінної інтегрованої системи, яка могла б супроводжувати гравця під час навчання, гри та подальшого розуміння логіки прийнятих рішень.

Проєкт Maia є дуже цікавим з наукової та практичної точки зору, оскільки він демонструє підхід до моделювання людського стилю гри. Це важлива перевага, адже порівняння “найкращого” ходу від класичного рушія та “найбільш людського” ходу дозволяє краще зрозуміти різницю між об’єктивно сильним і типовим людським рішенням. Проте саме в контексті даної роботи основним недоліком Maia є відсутність розгорнутого текстового пояснення. Система показує рекомендовані ходи, статистику, стрілки на дошці, співвідношення помилок і прийнятних рішень, але практично не пояснює словесно, чому певний хід є типовим для людини, у чому полягає його логіка та які позиційні або тактичні мотиви за ним стоять. Додаткові режими, такі як “Bot or Not” або “Hand & Brain”, безумовно, роблять платформу цікавішою, однак вони не вирішують головної проблеми — відсутності адаптованого, змістовного пояснення шахових ідей у доступній формі. Таким чином, Maia є сильним прикладом моделювання людської гри, але не прикладом повноцінної пояснювальної системи.

Платформа Aimchess демонструє інший напрям розвитку шахових сервісів — персоналізацію навчання. Це корисний підхід, оскільки система аналізує серію партій користувача, виявляє типові слабкі місця та на основі цього формує індивідуальні рекомендації й уроки. Проте якщо оцінювати її саме як засіб пояснення шахових рішень у межах окремої партії або позиції, можливості виглядають обмеженими. Безпосередній аналіз гри подається досить узагальнено і не містить глибокого пояснення конкретних ходів. Наявність лише шкали оцінки позиції або загальних висновків не дозволяє користувачеві повною мірою зрозуміти внутрішню логіку аналізу. Тобто Aimchess краще працює як інструмент довгострокового статистичного

спостереження за прогресом гравця, ніж як система, яка здатна детально пояснювати кожне окреме рішення на дошці. Це створює певний розрив між персоналізацією навчання та реальним поясненням шахового змісту позиції.

Поєднання En Croissant + Stockfish є, мабуть, найсильнішим із розглянутих рішень у суто технічному аспекті. Воно дозволяє здійснювати глибокий аналіз позиції, отримувати точні оцінки, варіанти продовження та гнучко налаштовувати параметри роботи рушія. Однак саме ця технічна сила одночасно виявляється і джерелом обмежень для широкого кола користувачів. Для повноцінного використання такої системи потрібно не лише окремо встановлювати рушії та додаткові компоненти, а й мати розуміння технічних параметрів, таких як глибина аналізу, кількість вузлів, часові обмеження та інші налаштування. Для підготовленого користувача це є перевагою, але для початківця — суттєвою перешкодою. Найголовніше ж полягає в тому, що така зв'язка фактично не надає зрозумілого текстового пояснення. Користувач отримує сильний результат аналізу, але змушений самостійно інтерпретувати оцінки, варіанти та технічні дані. Отже, рішення чудово виконує функцію оціночного ядра, але не забезпечує навчального та пояснювального супроводу.

Якщо розглядати всі проаналізовані рішення в сукупності (табл. 1.1), можна виділити кілька спільних недоліків. Насамперед це майже або повна відсутність зрозумілого текстового пояснення ходів і позицій. Другою суттєвою проблемою є слабка адаптація пояснення до рівня підготовки користувача: навіть якщо система надає коментарі, вони зазвичай не враховують, чи є користувач початківцем, гравцем середнього рівня чи більш досвідченим шахістом. Ще одним недоліком є розрив між аналітичною та навчальною складовими: одні рішення добре аналізують, але майже не навчають, інші допомагають у навчанні, але не забезпечують достатньої глибини шахового аналізу. Також спостерігається відсутність єдиного середовища, у якому користувач міг би отримати і сильний рушійний аналіз, і доступне пояснення його результатів, і певну персоналізацію подачі матеріалу.

Таблиця 1. Порівняння розглянутих рішень

	БОТи для різних рівнів гри	Аналіз позиції під час гри	Пояснення позиції	Аналіз повної гри	Аналіз загальних помилок	Адаптивність пояснень
Chess.com	+	-+	-	+	-	-
DecodeChess	+	+	+	+-	-	-
Maia	+	+-	-	+	-	-
Aimchess	+	-	-	+	+	+-
En Croissant + Stockfish	-	-	-	-+	-+	-

Отже, проведений аналіз недоліків показує, що сучасні рішення хоча й досягли значного рівня розвитку, усе ще не забезпечують повною мірою того балансу між силою аналізу та зрозумілістю пояснення, який є особливо важливим для навчального й практичного використання. Саме тому доцільною є розробка гібридного шахового рушія, у якому потужне оціночне ядро поєднуватиметься з пояснювальним модулем на основі мовної моделі штучного інтелекту. Такий підхід дозволить не лише знаходити сильні ходи, а й формувати логічні, змістовні та адаптовані до користувача пояснення, що безпосередньо усуває виявлені недоліки розглянутих систем.

2. ПРОГРАМНІ ТА АПАРАТНІ ЗАСОБИ ДЛЯ РОЗРОБКИ ДОДАТКУ

2.1. Мова програмування та стандарти розробки

Основна частина розробленого додатку AxepEngine реалізована мовою програмування C++ із використанням стандарту C++17. Вибір цієї мови зумовлений тим, що шаховий рушій потребує високої швидкодії, ефективної роботи з пам’яттю та можливості реалізації низькорівневих структур даних.

C++17 забезпечує достатній набір сучасних мовних можливостей для реалізації рушія: роботу зі стандартними контейнерами, файловою системою, багатопотоковими задачами та іншими засобами, які використовуються для організації логіки системи, графічного інтерфейсу та асинхронної взаємодії з модулем пояснень.

У проєкті стандарт C++17 задається на рівні системи складання, що забезпечує однакові вимоги до компіляції всіх компонентів системи.

2.2. Система складання та структура програмних компонентів

Для складання проєкту використовується CMake [5], що дозволяє описати структуру програмної системи незалежно від конкретного середовища розробки. CMake забезпечує створення окремих цільових компонентів, підключення бібліотек, налаштування параметрів компіляції та керування залежностями.

Архітектурно проєкт поділено на декілька основних програмних компонентів:

- AxepCore —яка містить основну логіку шахового рушія;
- AxepEngine — консольний виконуваний файл для взаємодії з рушієм через текстові команди;
- AxepEngineUI — графічний інтерфейс користувача, побудований на основі SFML 3 та ImGui-SFML;

- Python-модуль пояснень — окремий модуль, який формує текстові пояснення на основі даних, отриманих від рушія.

2.3. Консольний варіант інтерфейсу додатку

Консольний виконуваний файл AxepEngine використовується для тестування базової взаємодії з шаховим рушієм без графічного інтерфейсу. Він дозволяє завантажувати стартову позицію, встановлювати позицію у форматі FEN, виконувати ходи у форматі UCI, запускати пошук ходу рушієм, запускати автоматичну гру та виконувати perft-перевірку генератора ходів [4].

Наявність консольного режиму є важливою для розробки, оскільки він дозволяє швидко перевіряти коректність роботи окремих компонентів рушія без залежності від графічної оболонки. Він використовувся для налагодження ігрової логіки: перевірки мату, пату, нічиєї через недостатній матеріал, повторення позиції та правила п’ятдесяти ходів. Це дозволяло ізолювати помилки саме в логіці рушія.

2.4. Засоби реалізації графічного інтерфейсу

Графічний інтерфейс AxepEngineUI реалізований із використанням бібліотеки SFML 3 та надбудови ImGui-SFML [7]. SFML використовується для створення вікна, обробки подій, відображення шахової дошки, фігур, підсвічування ходів та інших візуальних елементів. ImGui-SFML застосовується для створення панелі інтерфейсу, елементів керування, блоків вибору профілю бота, пояснень та допоміжних інструментів.

У CMake графічна частина винесена в окрему опцію AXEP_BUILD_UI, що дозволяє збирати консольну версію рушія незалежно від графічного інтерфейсу.

У графічному інтерфейсі передбачено відображення шахової дошки, фігур, тем оформлення, аватарів профілів ботів, статусу партії та текстового блоку пояснення позиції.

Графічний інтерфейс не замінює логіку рушія, а лише надає користувачу зручний спосіб взаємодії. Усі ключові шахові операції - генерація ходів, перевірка правил, пошук найкращого ходу та оцінювання позиції - виконуються через бібліотеку AxepCore.

2.5. Засоби реалізації пояснювального модуля та апаратні вимоги

Окремою частиною системи є модуль пояснень, реалізований мовою Python. Його призначення полягає у формуванні текстових пояснень позиції та останнього ходу на основі даних, які передаються з шахового рушія.

У запиті до пояснювального модуля передаються такі дані, як FEN-позиція, активний профіль бота, режим пояснення, останній хід, найкращий хід, оцінка позиції, тип позиції, можлива загроза суперника, наявність незахищеної фігури, можливість безпечного взяття та інші позиційні ознаки. Такий підхід дозволяє відокремити шаховий аналіз від мовного оформлення: рушій відповідає за фактичну частину, а модуль пояснень - за перетворення цих даних у зрозумілий текст.

Також передбачено використання локального LLM-перефразатора через Ollama [11] [12]. Мовна модель не аналізує шахову позицію самостійно, а лише перефразовує вже сформований текст, що прибирає ризик появи некоректних шахових пояснень, оскільки джерелом шахових фактів залишається власний рушій AxepEngine.

3. ОПИС РОЗРОБЛЕНОГО ДОДАТКУ

3.1. Загальна структура

Розроблений додаток AxepEngine є програмним комплексом для гри в шахи з підтримкою різних рівнів сили бота та поясненням шахових рішень у текстовій формі. Система складається з декількох взаємопов'язаних частин: шахового ядра, консольного застосунку, графічного інтерфейсу та окремого модуля пояснень.

Основною частиною системи є бібліотека AxepCore, у якій зосереджена логіка шахового рушія. До неї належать наступні модулі: представлення дошки, завантаження позицій у форматі FEN, генерація ходів, оцінювання позиції, пошук найкращого ходу, транспозиційна таблиця, дебютна книги та профілі ботів.

Використаний підхід дозволяє розділити внутрішню логіку рушія та інтерфейси взаємодії з користувачем. Консольна версія використовувалась для тестування, налагодження й перевірки роботи шахового ядра, а графічна версія - для зручної гри та демонстрації функцій системи.

3.2. Реалізація шахового ядра AxepCore

Шахове ядро AxepCore відповідає за повну внутрішню логіку гри. Воно зберігає поточний стан шахової дошки, визначає сторону ходу, контролює правила переміщення фігур, перевіряє легальність ходів, обробляє спеціальні шахові правила та передає позицію до пошукового модуля. У межах ядра реалізовано такі основні функції:

- збереження та оновлення стану шахової дошки;
- підтримка стандартної початкової позиції;
- завантаження позицій у форматі FEN;
- генерація псевдолегальних і легальних ходів;

- виконання ходів;
- перевірка шаху, мату, пату та нічийних ситуацій;
- оцінювання позиції;
- пошук найкращого ходу для поточного профілю бота.

Окрему роль у системі відіграє модуль FEN. Він дозволяє швидко завантажувати довільну шахову позицію, що є важливим як для тестування рушія, так і для подальшого аналізу конкретних позицій. Завдяки цьому система може працювати не лише зі стандартною початковою позицією, а й з будь-якою можливою ситуацією.

3.3. Представлення шахової позиції та генерація ходів

Для ефективної роботи рушія шахова позиція подається у внутрішньому форматі, орієнтованому на швидку обробку даних. У системі використовується bitboard-підхід, за якого розташування фігур на дошці може бути представлене через 64-бітові маски. Такий спосіб є зручним для шахових програм, оскільки дозволяє швидко виконувати операції над множинами клітинок дошки.

Генератор ходів відповідає за побудову списку можливих ходів для поточної сторони. При цьому враховуються базові правила руху всіх типів фігур, взяття, рокіровка, перетворення пішака, а також ситуації, пов’язані із шахом королю. Після генерації можливих ходів система відсіює ті ходи, які залишають власного короля під шахом, тобто формує список саме легальних ходів.

Коректність генерації ходів є критично важливою для роботи шахового рушія, оскільки, при варіанті того, що генератор ходів працює неправильно, пошуковий алгоритм може аналізувати некоректні варіанти або пропускати допустимі ходи. Тому в системі передбачено можливість запуску perft-перевірки, яка використовувалась на етапі розробки для підрахунку кількості позицій на заданій глибині та порівняння результатів із відомими тестовими

значеннями. Консольний застосунок підтримує команду perft <n>, що дозволяє запускати таку перевірку без графічного інтерфейсу.

3.4. Пошук найкращого ходу та оцінювання позиції

Пошуковий модуль відповідає за вибір найкращого ходу для шахового бота. Його робота базується на аналізі дерева можливих ходів: рушій розглядає варіанти гри на певну глибину, оцінює отримані позиції та обирає хід, який приводить до найвигіднішого результату.

Для керування пошуком, в консольному варіанті, використовуються параметри, що визначають максимальну глибину аналізу або обмеження часу на хід. У консольній версії передбачені команди «depth <n>» (команда дає можливість задати глибину пошуку) та «movetime <min>» (команда для обмеження часу на аналіз). Також під час виконання команди «go» система виводить інформацію про знайдений хід, оцінку позиції, досягнуту глибину, кількість переглянутих вузлів, швидкість пошуку, кількість звернень до транспозиційної таблиця та кількість відсічень.

Оцінювання позиції виконується у вигляді числового значення в сантипішаках. Додатне значення означає перевагу білих, від’ємне — перевагу чорних. Під час оцінювання враховуються матеріальні і позиційні фактори: співвідношення фігур, активність фігур, розвиток, контроль центру, безпека короля.

Оскільки система орієнтована не лише на гру, а й на навчання, оцінка позиції використовується не тільки для вибору ходу, але й для формування пояснень. Пояснювальний модуль орієнтується, на оцінку для видання рекомендації, яка буде підходити під ситуації, яка відбувається на шаховій дошці.

3.5. Профілі ботів та керування рівнем гри

У системі реалізовано чотири профілі ботів, які відрізняються силою гри та формулюванням пояснень. Це дозволяє використовувати AxepEngine не лише як звичайний шаховий рушій, а як навчальну систему, адаптовану до користувачів різного рівня підготовки.

Кожен профіль може мати власні обмеження пошуку. Для слабших ботів використовується менша глибина аналізу та спеціальна логіка, орієнтована на шахові принципи.

Такий підхід дозволяє створити систему поступового навчання. Користувач може почати гру проти слабшого профілю, а потім переходити до сильніших ботів.

3.6. Пояснювальний модуль системи

Однією з ключових особливостей AxepEngine є наявність пояснювального модуля. Його завдання полягає у створенні текстових пояснень для поточної позиції, останнього ходу бота та рекомендованого фокусу в позиції.

Для взаємодії між рушієм і модулем пояснень використовується структура ExplanationRequest. Вона містить FEN-позицію, ідентифікатор і назву профілю, стиль пояснення, режим пояснення, сторону ходу, останній хід, найкращий хід, рекомендований хід, тип позиції, загрозу суперника, оцінку позиції, глибину пошуку, інформацію про шах, нічию, мат, незахищені фігури, безпечні взяття та інші ознаки.

Запит формується на стороні графічного інтерфейсу. Система визначає тип позиції, можливу загрозу суперника, наявність висячих фігур, можливість взяття фігур суперника, а також додаткові причини переваги, наприклад матеріал, розвиток, контроль центру або безпеку короля. Після цього дані передаються до Python-модуля пояснень.

Python-модуль не виконує самостійний шаховий аналіз позиції. Його завдання - перетворити вже підготовлені дані з позиції в текст. Він визначає фазу гри, інтерпретує оцінку позиції, формує пояснення переваги, описує тактичний стан, пояснює останній хід і дає коротку пораду щодо подальшого плану. Також передбачена можливість використання локального LLM-перефразування через Ollama. У такому режимі мовна модель не приймає шахових рішень і не аналізує позицію самостійно, а лише перефразовує вже сформований текст пояснення. Це дозволяє зробити пояснення більш природними, але зберегти контроль над шаховою коректністю, оскільки всі фактичні дані надходять від рушія.

3.7. Опис консольного додатку

На рисунку 3.1 видно наступні команди та пояснення до них:

- help – показати команди;
- startpos – завантажити початкову позицію;
- fen <FEN> – завантажити позицію в форматі FEN;
- board – показати дошку;
- legal – вивести легальні ходи;
- move <uci> - зіграти хід;
- go – дати хід рушію (обраному боту);
- auto – повна автоматична гра від рушія;
- depth <n> – задати глибину пошуку для рушія (обраного бота);
- movetime <min> - часовий ліміт на хід (актуально лише для найсильнішої версії рушія);
- bots – показати всіх ботів;
- bot <id> - обрати бота за його ID;
- rename <id> <name> - перейменувати бота;
- profile – показати активний профіль бота;

- perft <n> - виконати перевірку perft;
- quit – вихід з програми.

```
Commands:
  help                  - show commands
  startpos              - load standard starting position
  fen <FEN>             - load position from FEN
  board                 - print board
  legal                 - print legal moves
  move <uci>            - play move manually, example: e2e4
  go                    - engine makes one move
  auto                  - engine plays both sides until game over
  depth <n>             - set fixed depth mode for max profile
  movetime <min>        - set time per move in minutes for max profile
  bots                  - show all bot profiles
  bot <id>              - select active bot profile
  rename <id> <name>    - rename bot profile
  profile               - show current bot profile
  perft <n>             - run perft divide for depth n
  quit                  - exit program

Promotion examples:
  move c2c1q            - promote to queen
  move c2c1             - defaults to queen if promotion exists
```

Рисунок 3.1 - Повний лог команд для роботи з консольною версією додатку

На рисунку 3.2 показані всі доступні боти в наступному форматі: <ID> <Ім’я> | <Глибина пошуку>, а на рисунки 3.3 показаний уже інформація про обраного бота.

```
Available bots:
    [1] Hatsune Miku | depth 1-1
    [2] Matsushita Chiaki | depth 2-3
    [3] Oreki Houtarou | depth 3-4
  * [4] Axepent Prime | maximum strength
```

Рисунок 3.2 - Всі боти доступні для гри

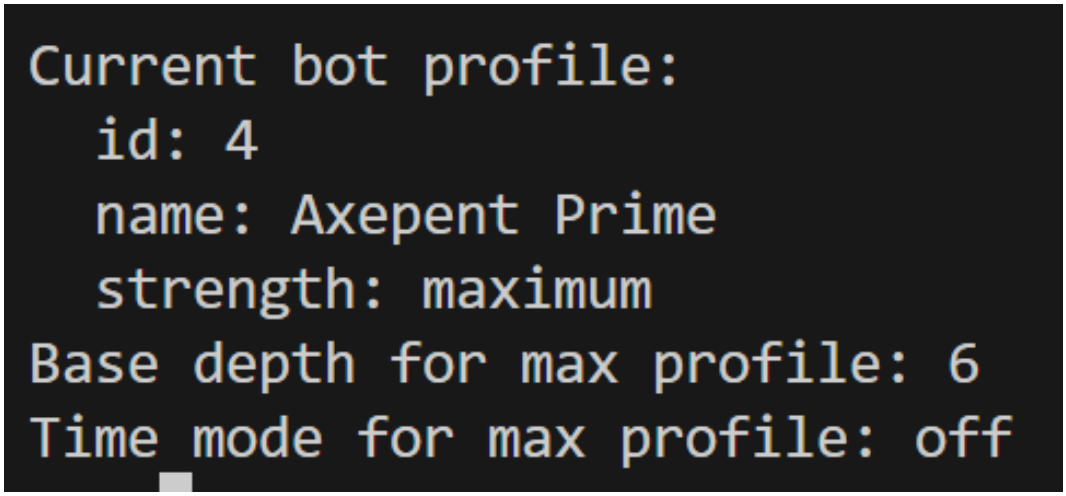


Рисунок 3.3 - Приклад інформації про обраного бота

На рисунку 3.4 важливо помітити різницю між білими та чорними фігурами, а саме великі та малі літери з наступними літерними позначеннями:

- P/p – Pawn (пішак);
- R/r – Rook (тура);
- N/n – Knight (кінь);
- B/b – Bishop (слон);
- Q/q – Queen (королева);
- K/k – King (король).

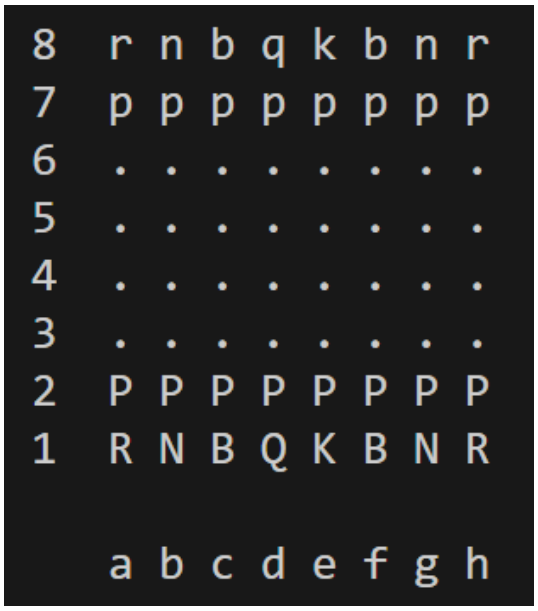


Рисунок 3.4 - Представлення дошки в консольному варіанті додатку

На рисунках 3.5 – 3.8 показана робота частини інструментарії в консольному додатку.

```
Bot: Axepent Prime [4] | depth 6
Book move: e7e6

8  r n . q k b n r
7  p p . . . p p p
6  . . p . p . . .
5  . . . p P b . .
4  . . . P . . . .
3  . . . . . N . .
2  P P P . . P P P
1  R N B Q K B . R

   a b c d e f g h

Side: white
Castling: KQkq
En passant: -
```

Рисунок 3.5 - Приклад виконаного ботом ходу

```
Bot: Axepent Prime [4] | depth 6
Best move: b8c6
Score: -104
Depth: 6
Nodes: 3658816
NPS: 63386
TT hits: 48909
Cutoffs: 1890246
Time ms: 57722
```

Рисунок 3.6 - Приклад виконаного ботом ходу

```
>>> movetime 0.1
Time per move for max profile set to 0.1 minute(s)
Max depth is ignored while time mode is active
```

Рисунок 3.7 - Задання ліміту часу на хід

```
>>> fen rn1qkbnr/pp3ppp/4p3/2ppPb2/3P4/5N2/PPP1BPPP/RNBQK2R w KQkq - 0 1
FEN loaded.

8  r n . q k b n r
7  p p . . . p p p
6  . . . . p . . .
5  . . p p P b . .
4  . . . P . . . .
3  . . . . . N . .
2  P P P . B P P P
1  R N B Q K . . R

   a b c d e f g h
```

Рисунок 3.8 - Приклад задання позиції за допомогою FEN формату

3.8. Опис UI

Більшу частину вікна додатку займає шахова дошка, основний же функціонал представлено на правій панелі додатку, на якій присутнє кольорове розділення логічних блоків (вибір профілю бота, пояснювальний модуль, інструменти для гри) (рис. 3.9).

Рисунок 3.9 - Повний знімок розробленого додатку

Отже, почерговий розбір функціональної панелі, перший блок вибір профілю бота (рис. 3.10 – 3.11):

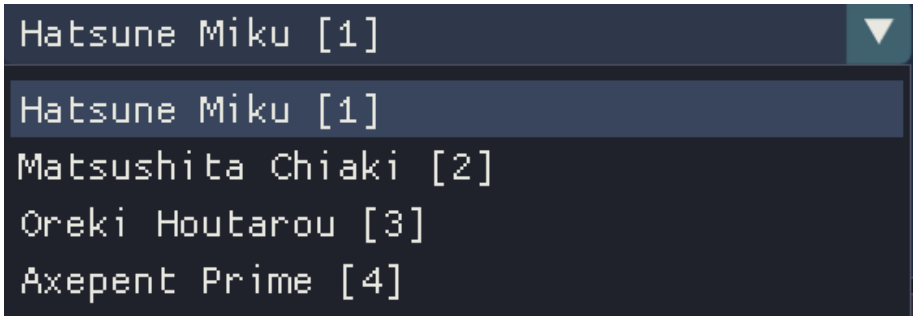
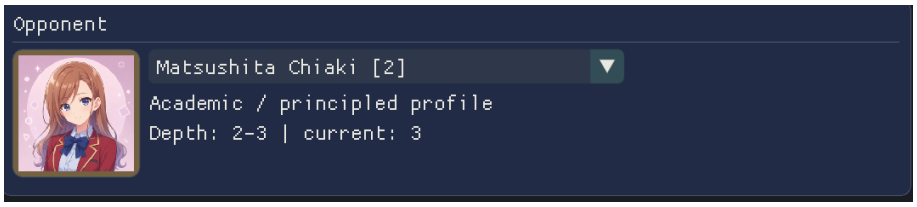
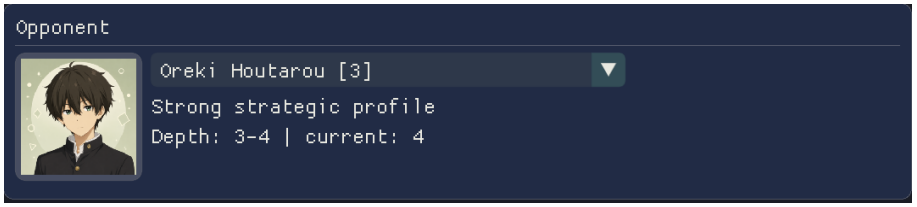


Рисунок 3.10 Випадний список усіх доступних ботів

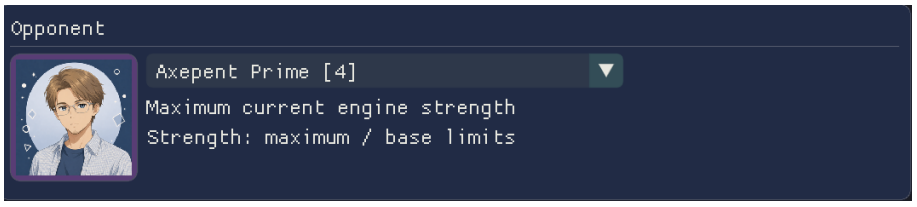
а)



б)



в)



г)

Рисунок 3.11 Усі боти в порядку зростання ігрових навичок (сили). а) БОТ Miku; б) БОТ Matsushita; в) БОТ Houtarou; г) БОТ Axepent Prime

В другому блоці (рис. 3.12), пояснювальному модулі, присутній лише один спосіб взаємодії, а саме «Rewrite explanation in character style», при виборі цього варіанту для перефразування пояснення позиції, останнього ходу та рекомендації фокусу буде використана модель Ollama gemma3:4b.

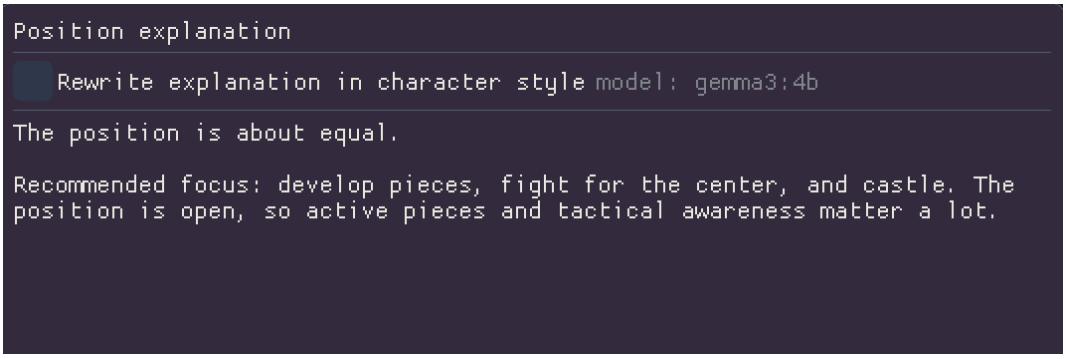


Рисунок 3.12 - Пояснювальний модуль у початковому стані

Третій блок, інструментарій, на рисунку 3.13 видно наступні інструменти:

- Evaluation - оціночна шкала для поточної позиції;
- Auto bot reply - плажка для автоматичної відповіді бота на хід гравця;
- Hint from bot – підказка хода від бота;
- Bot move - одиночний хід бота;
- Undo last move - відміна останнього ходу;
- Undo player + bot - відміна ходу гравця та бота;
- Flip board - візуальна зміна сторони гравця;
- New game - повернення до початкової позиції;
- Depth – налаштування глибину пошуку (актуально лише для максимально рівня складності).

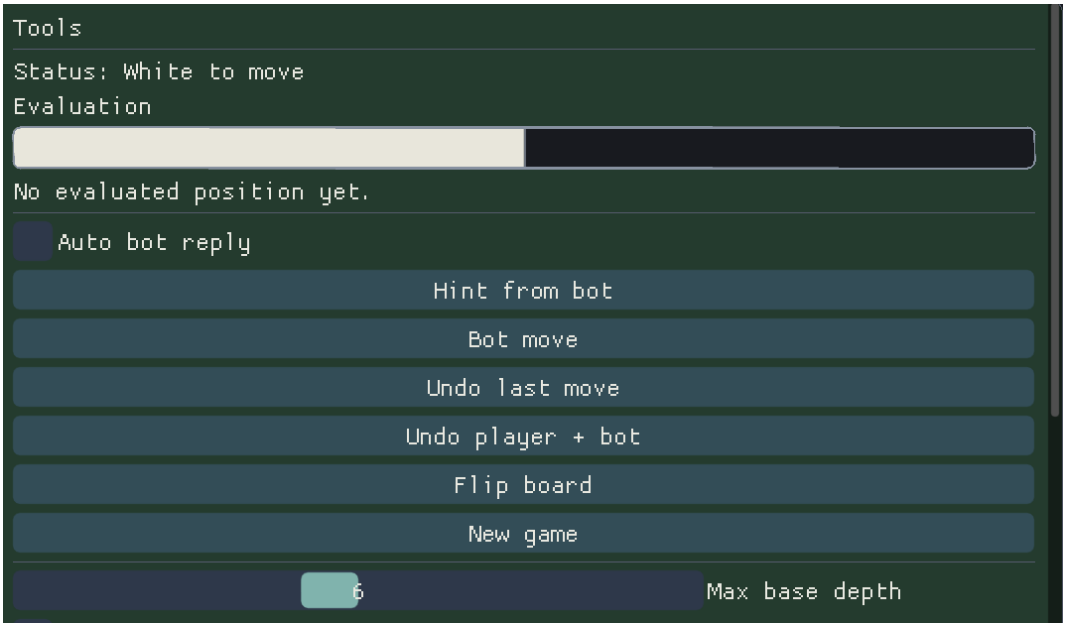


Рисунок 3.13 - Перша частина блоку інструментарію

На рисунку 3.14 видно наступні інструменти:

- Use time limit – плажка, що дає можливість виставити боту ліміт часу на хід (актуальна лише для «Axepent Prime», оскільки інші боти ходять майже моментально);
- Search history - історія ходів, створюється автоматично відповідно до стандарту SAN.

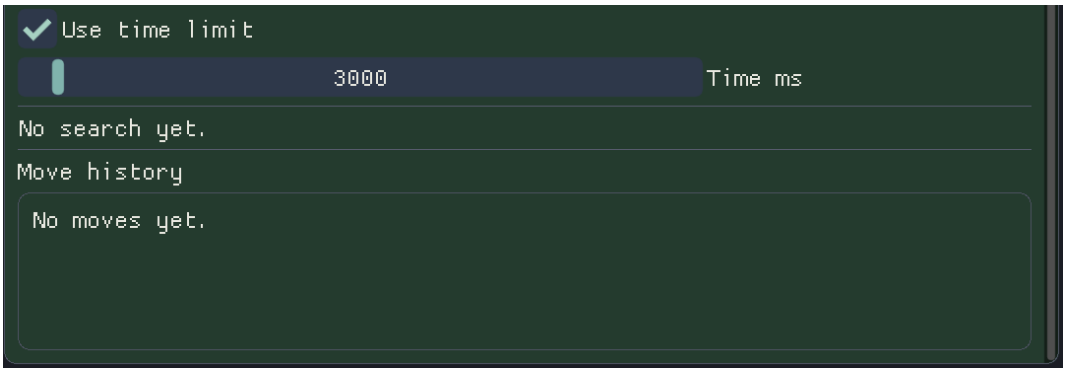
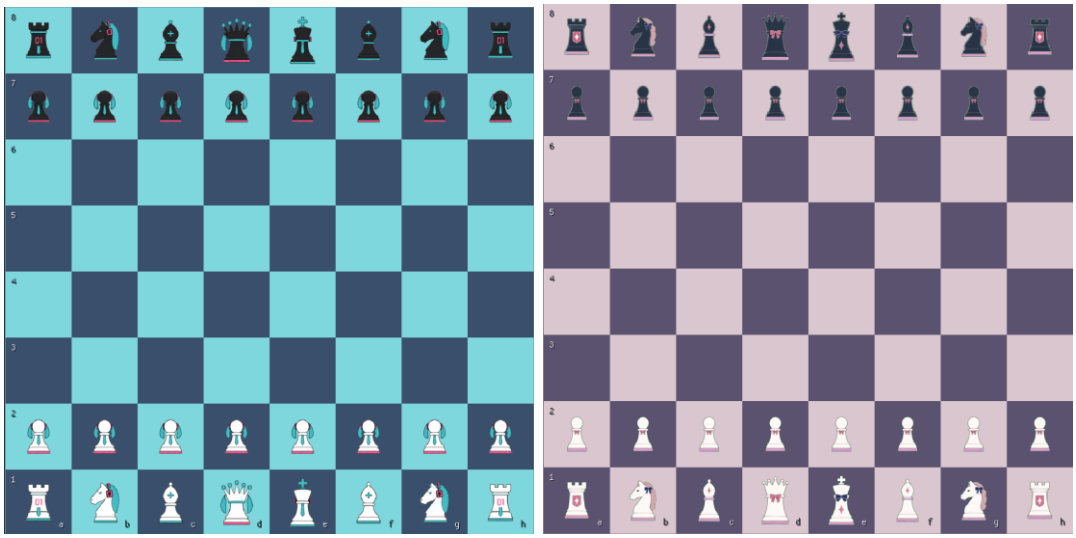


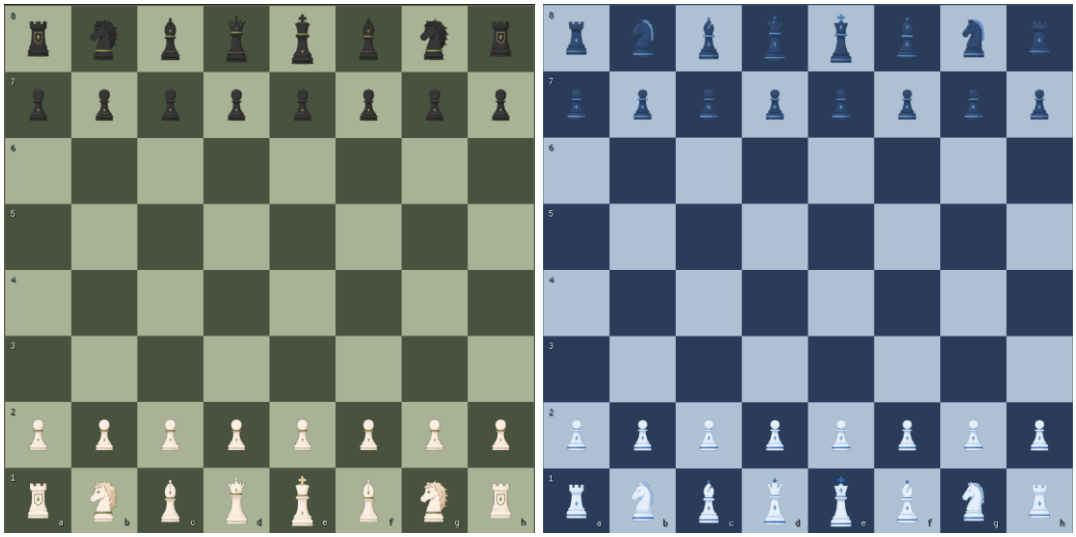
Рисунок 3.14 - Продовження блоку інструментарію

Останнім елементом UI являється нижня панель, яка є косметичною частиною (3.15) з можливістю вибору зміни візуального представлення дошки та фігур під стилі персонажів, які були обрані для ботів (рис 3.16).

Рисунок 3.15 - Панель вибору візуального представлення дошки та фігур



а) б)



в) г)

Рисунок 3.16 - Підібрані під стилістику персонажів стилі фігур та зміна кольорів шахової дошки. а) тема Miku; б) тема Matsushita; в) тема Houtarou; г) тема Axepent Prime

4. ТЕСТУВАННЯ НА РІЗНИХ ЕТАПАХ РОЗРОБКИ

4.1. Загальний підхід до тестування

Тестування розробленого додатку AxepEngine виконувалося поетапно протягом усього процесу розробки. Оскільки система складається з декількох частин - шахового рушія, профілів ботів, консольного режиму, графічного інтерфейсу та пояснювального модуля - перевірка працездатності проводилася не лише після завершення реалізації, а після кожного суттєвого оновлення.

Основним методом перевірки на перших двох етапах розробки були шахові партії проти ботів визначеного рівня на сайті “chess.com”. Такий підхід дозволяв оцінювати не тільки формальну коректність роботи програми, але й практичну якість її гри: стабільність у дебюті, здатність знаходити тактичні ходи, якість оцінювання позиції та поведінку в ендшпілі [9] [13] [14].

Під час тестування фіксувалися виявлені проблеми, після чого вносилися зміни до алгоритмів пошуку, оцінювальної функції, дебютної книги або логіки профілів ботів.

На подальших етапах розробки додавалися нові функції для UI/UX та проводився суб’єктивний аналіз для оцінки пояснювального модуля.

4.2. Перший етап: тестування шахового рушія

На першому етапі основна увага приділялася перевірці базової сили гри та коректності роботи шахового ядра. Початкові тестові партії показали, що рушій програвав через недостатню глибину прорахунку та занадто просту оцінювальну функцію. Це дозволило визначити перший напрям покращення: збільшення глибини аналізу та ускладнення позиційної оцінки.

Після однієї з перших партій було реалізовано автоматичне перетворення пішака у ферзя при досягненні останньої горизонталі, цей момент не був

надважливим, проте дав можливість трохи зручніше користуватися консольною версією додатку.

Наступним важливим кроком стало додавання Quiescence Search та збільшення глибини аналізу. Це було зроблено для зменшення ефекту «горизонту», коли рушій оцінює позицію на нестабільному тактичному моменті та не бачить очевидного продовження. Також на цьому етапі було додано модуль Perft, який використовується для перевірки правильності генерації ходів.

Під час однієї з тестових партій рушій отримав виграну позицію, однак допустив триразове повторення та завершив партію внічию (рис. 4.1). Після цього було додано перевірку повторення позицій, що дозволило уникати подібних ситуацій у майбутньому. Також були реалізовані хешування Зобриста та транспозиційна таблиця, що прискорило пошук за рахунок повторного використання вже проаналізованих позицій.

Рисунок 4.1 - Позиція нічийної гри після трьох разового повторення позиції та відповідна історія ходів

Окремо та поступово покращувалась оцінювальна функція. До неї поступово додавалися такі фактори, як мобільність фігур, пішакова структура, безпека короля, фазова інтерполяція, оцінка ендшпілю, слабші бонуси за прохідних пішаків, оцінка турових ендшпілів, пошук оптимального завершення партії матом, виявлення незахищених фігур та тиск на короля.

4.3. Тестування алгоритмів пошуку та швидкодії

Окремим напрямом тестування була перевірка швидкодії рушія. Після базових покращень сили гри виникла потреба зменшити час пошуку ходу без значної втрати якості. Для цього було оновлено порядок перебору ходів: спочатку аналізувався хід із транспозиційної таблиці, потім взяття, killer moves.

Також були додані або налаштовані алгоритми NMP та LMR. Їх використання дало змогу суттєво скоротити дерево пошуку, однак тестування показало, що надто агресивне відсікання може погіршити якість гри, особливо на ранніх етапах партії. Після цього NMP і LMR були послаблені, щоб рушій не відкидав важливі варіанти занадто рано.

Після оптимізації швидкість прийняття рішень стала значно кращою: ходи почали виконуватися приблизно в межах декількох секунд, що зробило гру комфортнішою. Водночас тестові партії показали, що прискорення не повинно досягатися за рахунок різкого падіння якості (рис. 4.2). Тому подальші зміни були спрямовані на пошук балансу між швидкістю та силою гри.

На цьому ж етапі було розширено дебютну книгу. Спочатку вона містила мінімальний набір систем: для білих — Віденська партія, Ферзевий гамбіт та Лондонська система, для чорних — Захист Каро-Канн, Французький захист та Скандинавський захист. Після тестових партій було виявлено, що рушій іноді отримує погану позицію вже після перших ходів, тому дебютна книга була розширена, а оцінка дебютної стадії додатково покращена.

а) б)

Рисунок 4.2 - Порівняння рівня гри до та після застосування алгоритмів для пришвидшення прийняття рішень. а) результат до впрожадження алгоритмів; б) результат після впровадження алгоритмів

Результатом цього етапу стало отримання більш стабільної версії рушія, яка краще поєднувала якість гри та швидкість пошуку (рис. 4.3). У контрольних партіях рушій почав демонструвати прийнятний баланс між глибиною аналізу, тактичною уважністю та часом на хід.

а) б)

Рисунок 4.3 - Результуючі оцінки фінальної ітерації покращення рушія. а) перша гра фінальної версії рушія; б) друга гра фінальної версії рушія

4.4. Тестування ігрових правил та нічийних ситуацій

Під час тестування окремо перевірялися спеціальні шахові правила та завершення партії. На ранніх етапах було виявлено проблему з повторенням позиції, через яку рушій міг погодитися на нічию навіть у виграшній позиції. Після цього було додано механізм виявлення повторення позицій.

Пізніше в одній із партій була зафіксована нічия через недостатній матеріал (рисунок 4.4), після чого стало зрозуміло, що цей механізм також необхідно реалізувати в системі. Це дозволило коректно обробляти позиції, у яких жодна зі сторін уже не може поставити мат.

У фінальній версії консольного режиму передбачено перевірку основних варіантів завершення гри: мат, пат, нічия через недостатній матеріал, повторення позиції та правило п’ятдесяти ходів. Це робить поведінку рушія відповідною до реальних шахових правил.

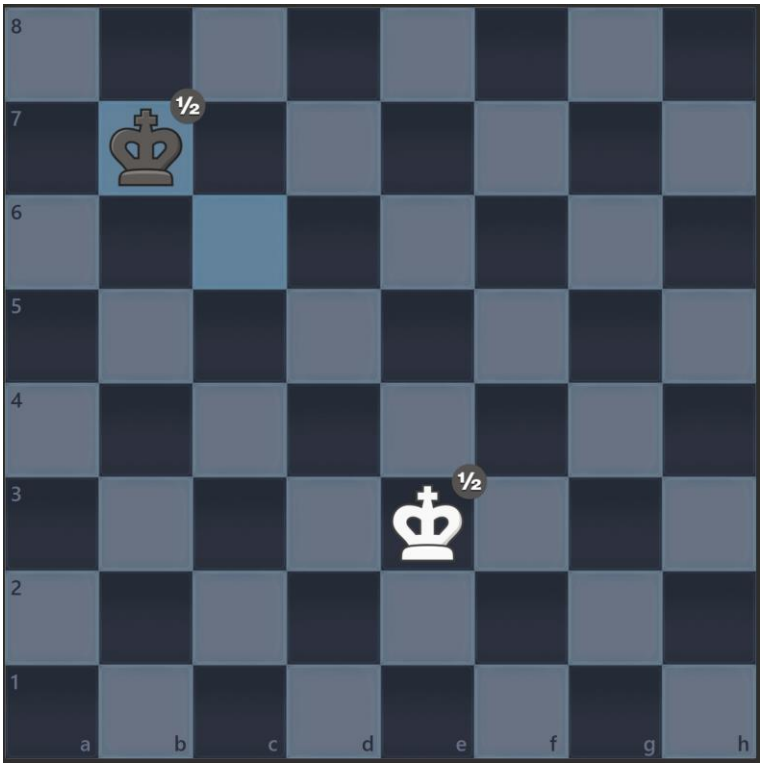


Рисунок 4.4 - Нічийна гра через недостатній матеріал

4.5. Другий етап: тестування та калібрування профілів ботів

Після стабілізації основного рушія почався другий етап розробки - розділення системи на рівні гри. Було створено чотири профілі ботів:

- Hatsune Miku – початковий рівень;
- Chiaki Matsushita – низький середній рівень;
- Oreki Houtarou – високий середній рівень;
- Axepent Prime – максимальна сила рушія (початковий рівень майстрів).

Метою цього етапу було не просто створити різні назви для ботів, а зробити так, щоб кожен профіль мав приблизно відповідний рівень гри. Для цього проводилися серії тестових партій, після яких змінювалися глибина пошуку, принципи вибору ходів та поведінкові обмеження слабших ботів.

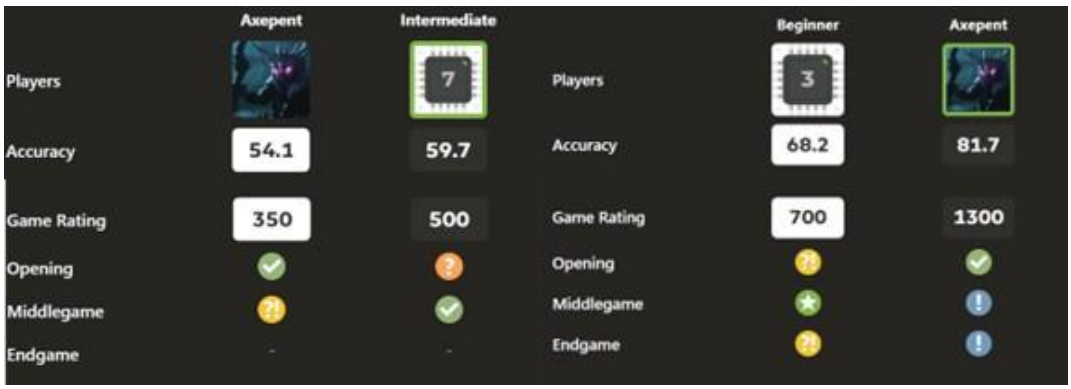
Перша спроба персоналізації слабкого бота виявилася невдалою: бот почав ігнорувати значну кількість загроз заради ходів, які формально відповідали заданим принципам. Після цього було прийнято рішення повернутися до простішої моделі ослаблення: зменшення глибини аналізу та додавання базових принципів гри для початкових рівнів.

4.5.1. Калібрування Miku

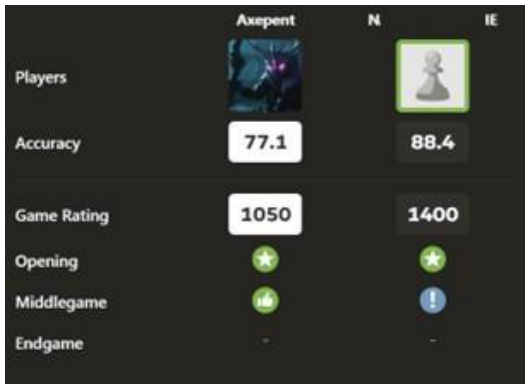
Під час перших перевірок Miku або грала занадто сильно для заявленого рівня, або після надмірного ослаблення починала пропускати очевидні взяття. Після декількох ітерацій було знайдено більш прийнятний варіант: бот дотримувався базових принципів, міг забирати незахищені фігури, але при цьому іноді допускав очевидні помилки, що відповідало задуму слабшого навчального профілю, її приблизна оцінка сили, як шахового бота ~800 ело.

На рисунках 4.5 можна помітити деяку нестабільність в оцінці рівня Miku, проте це є типовим для шахових гравців. Також на рисунку 4.10 була проведена

гра з реальним новачком в шахах і результат цієї гри показав, що рівень Miku прийнятний для бота початківця.



а) б)



в)

Рисунок 4.5 Фінальні ігри калібрування Miku. а) перша гра фінальної версії БОТа; б) друга гра фінальної версії БОТа; в) третя гра фінальної версії БОТа;

4.5.2. Калібрування Matsushita

Для Matsushita також проводилося окреме калібрування. Під час тестів проти ботів сильнішого рівня було встановлено, що цей профіль грає сильніше за Miku, але все ще має помітні обмеження. У результаті Miku та Matsushita були налаштовані як два різні рівні гри, які помітно відрізняються між собою.

Рисунки 4.6 доцільно порівняти з рисунками 4.8 – 4.10 для розуміння різного рівня гри між двома початковими ботами, що і було головним

завданням: відкалібрувати їх з однаковими вказівками “академічного” стилю, проте зробити їх відмінними по “силі”.

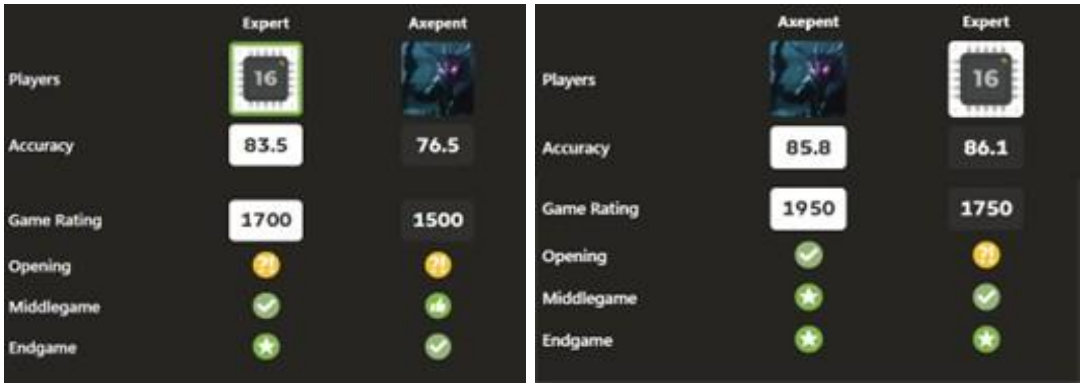
а) б)

в)

Рисунок 4.6 Фінальні ігри калібрування Matsushita. а) перша гра фінальної версії БОТа; б) друга гра фінальної версії БОТа; в) третя гра фінальної версії БОТа;

4.5.3. Калібрування Houtarou

Профіль Oreki Houtarou тестувався як сильніший бот, однак під час партій були виявлені проблеми в ендшпілі. Зокрема, у виграних або перспективних позиціях бот міг не знаходити достатньо точного продовження. Порівняння з Axepent Prime показало, що найсильніший профіль завдяки більшій глибині аналізу міг уникати деяких помилок, які допускав Houtarou. Після цього калібрування ботів було завершено.



а) б)

Рисунок 4.7 Фінальні ігри калібрування Houtarou. а) перша гра фінальної версії БОТа; б) друга гра фінальної версії БОТа

Рисунки 4.7 знову ж таки доцільно порівняти з рисунками 4.6 для розуміння різного рівня гри між ботами Matsushita та Houtarou, особливо показним є те, що Matsushita завжди програвала проти експертного рівня рушія на chess.com, в той час як Houtarou часто зводив ігри до нічиєї, або перемоги.

4.6. Третій етап: тестування UI/UX-частини

На третьому етапі тестування було зосереджено на графічному інтерфейсі. Спочатку до проєкту було підключено SFML для створення ігрового поля, а також створено окрему папку external для зовнішніх бібліотек. Після цього консольну та графічну версії було розділено на рівні CMake, щоб UI можна було збирати окремо від базового рушія.

Перша версія графічної дошки була створена для перевірки самого факту підключення SFML та коректного відображення шахового поля (рис. 4.8). Далі поступово додавалися функції взаємодії з дошкою: відображення фігур, виконання ходів, візуалізація останнього ходу, можливість взяття фігур та підсвічування ігрових дій (рис. 4.9 – 4.10).

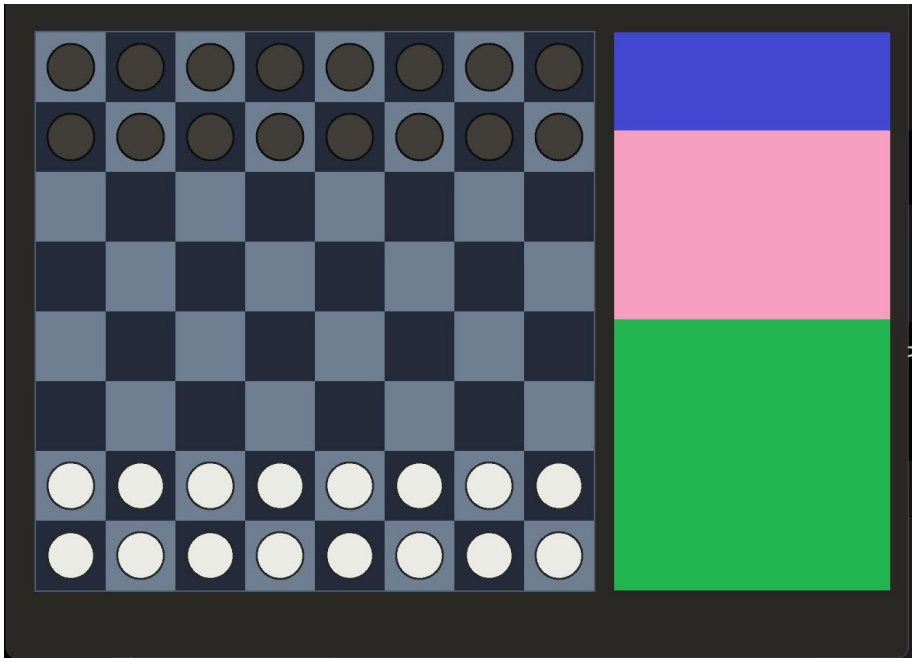


Рисунок 4.8 Перший прототип UI

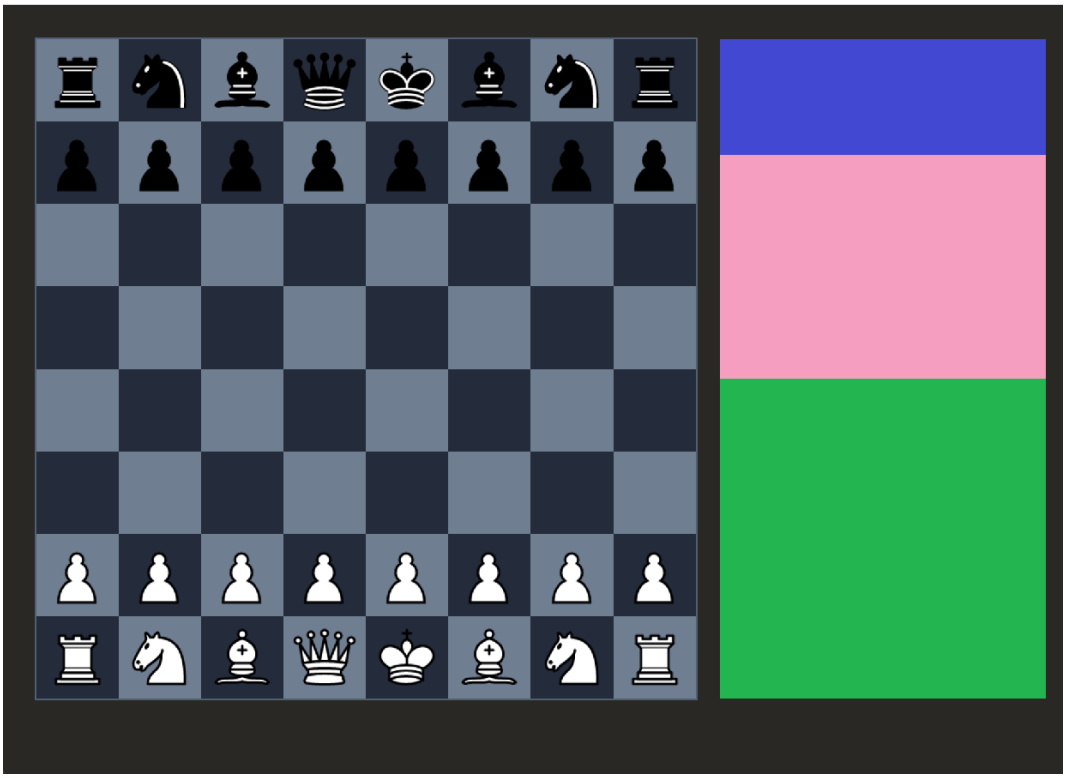
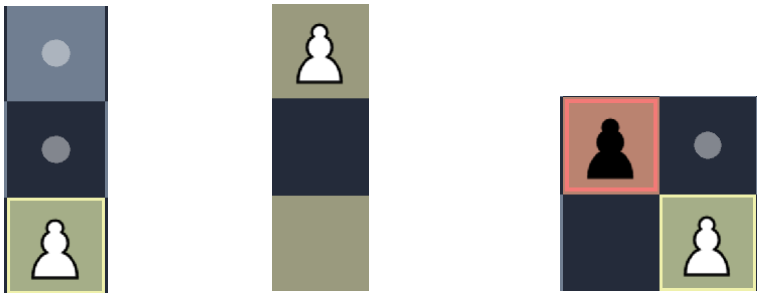


Рисунок 4.9 Прототип UI з уже доданими ассетами фігур



а) б) в)

Рисунок 4.10 Виділення можливих ходів, візуальне останнього ходу та візуальне відображення можливості взяття фігури. а) візуалізація можливих ходів; б) візуалізація ходу; в) візуалізація можливості захоплення

Також додалась права панель інтерфейсу (рис. 4.11). До неї було додано інформацію про активного бота, історію ходів, інструменти керування, оцінка позиції, режим обмеження часу на хід, аватари ботів та теми оформлення. Окремо тестувалося масштабування вікна, збільшення шрифту, додавання координат клітинок a-h та 1-8, а також перемикання тем фігур і кольорів дошки.

Рисунок 4.11 Перший прототип правої панелі

Після базової роботи дошки було додано кнопку отримання підказки від активного бота (рис. 4.12). Це дозволило перевірити зв’язок між графічним інтерфейсом і шаховим рушієм не лише в режимі автоматичного ходу бота, а й у режимі навчальної допомоги користувачу.

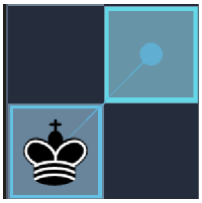


Рисунок 4.12 Візуальний вигляд підказки від бота

У результаті UI/UX-тестування було підтверджено, що графічна версія може використовувати основні можливості рушія (рис. 4.13): гру проти бота, отримання підказки, відображення оцінки позиції, історії ходів та візуального стану партії.

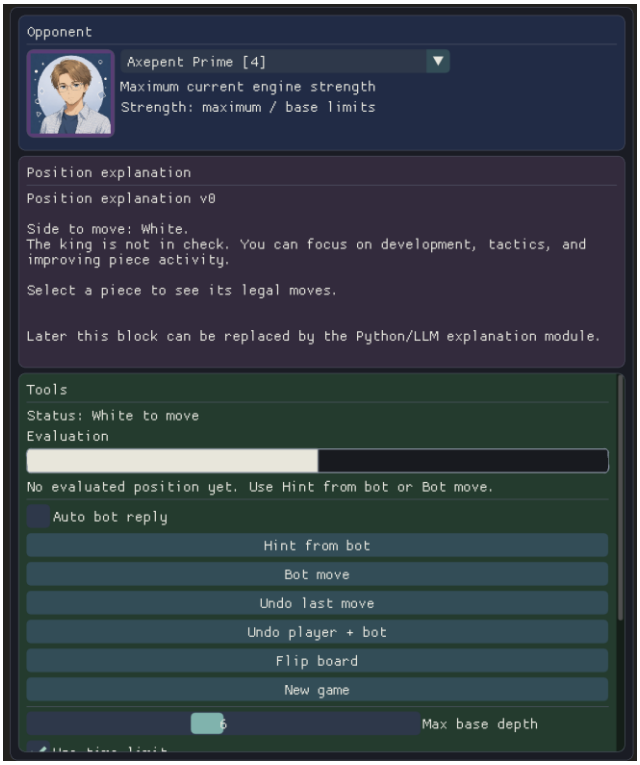


Рисунок 4.13 Фінальний вигляд правої панелі на цьому етапі розробки

4.7. Четвертий етап: тестування пояснювального модуля

Початкова реалізація модуля мала детермінований характер: текст пояснення формувався на основі фактичних даних, отриманих безпосередньо від шахового рушія. До таких даних належали числова оцінка позиції, запропонований хід, стадія партії. У ході тестування було прийнято рішення додати до інформації для обробки: тип позиції, наявність безпосередніх загроз, висячих фігур, можливість безкоштовного взяття, а також ознаки, пов’язані з розвитком фігур, контролем центру та безпекою короля.

Після редагування та аналізу заготовок для пояснень було додано локальний перефразатор на основі Ollama (рис. 4.14). Його призначенням не шахове аналізування позиції, а виключно стилістичне переписування вже сформованого пояснення у більш природній та персоналізованій формі.

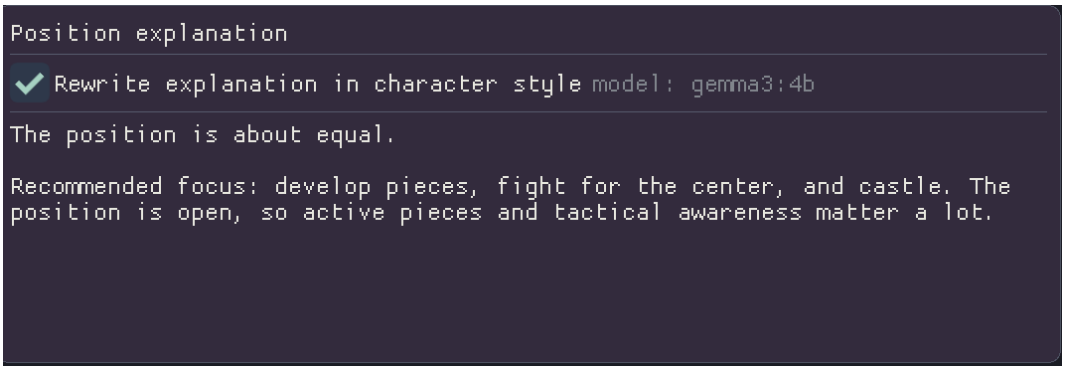
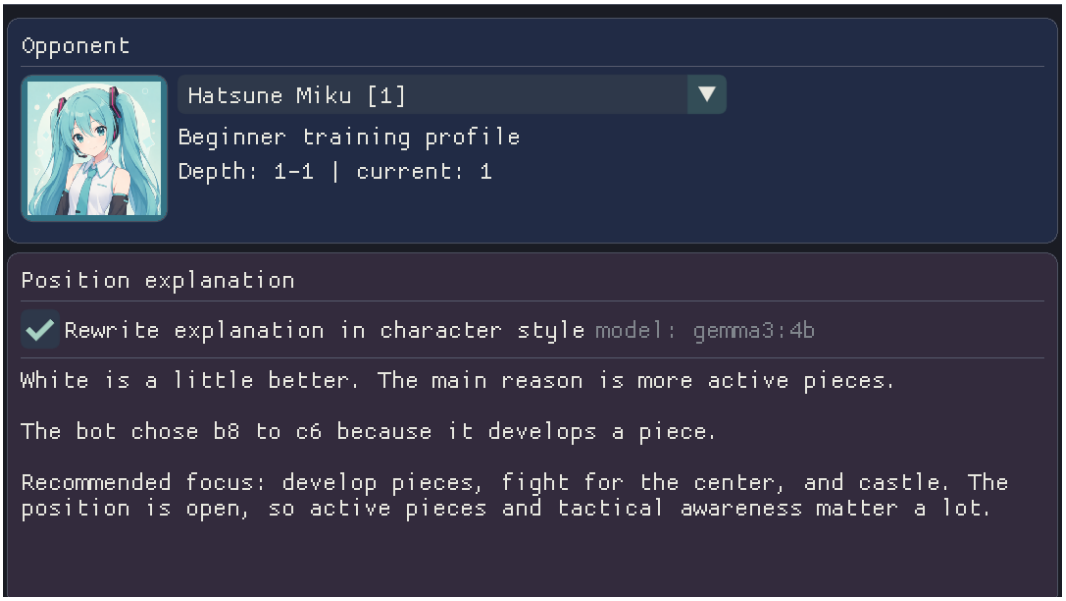
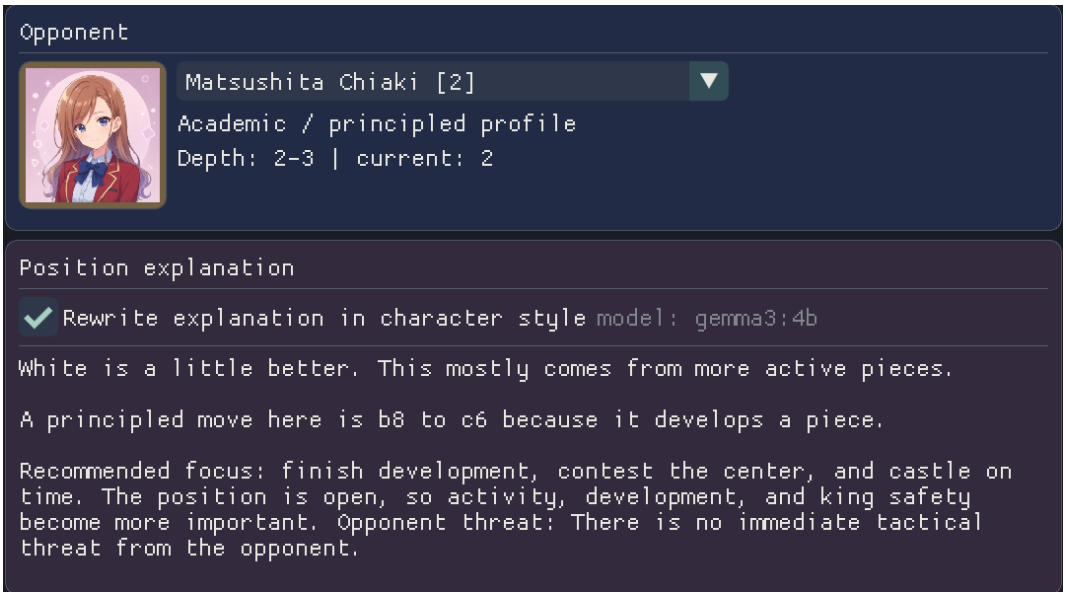


Рисунок 4.14 Фінальний вигляд пояснювального модуля на початку партії

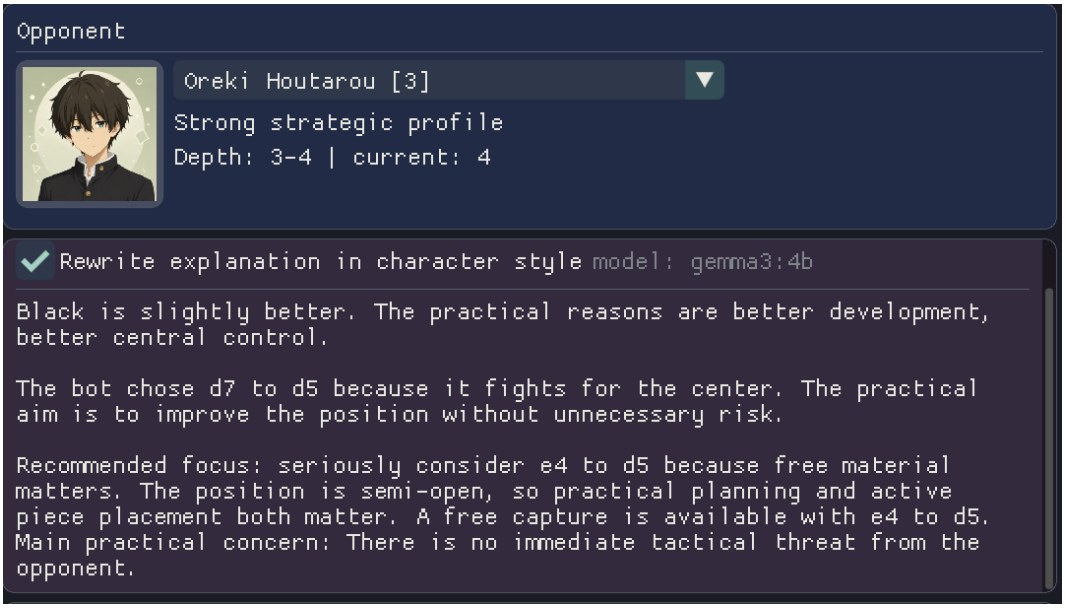
На рисунку 4.15-а та 4.15-б показана різниця пояснення однієї позиції від Miku та Matsushita, а на рисунках 4.15-в та 4.15-г уже різниця з сильнішими профілями Houtarou та Axepent Prime.


Opponent
Hatsune Miku [1]
Beginner training profile
Depth: 1-1 | current: 1
Position explanation
Rewrite explanation in character style model: gemma3:4b
White is a little better. The main reason is more active pieces.
The bot chose b8 to c6 because it develops a piece.
Recommended focus: develop pieces, fight for the center, and castle. The position is open, so active pieces and tactical awareness matter a lot.

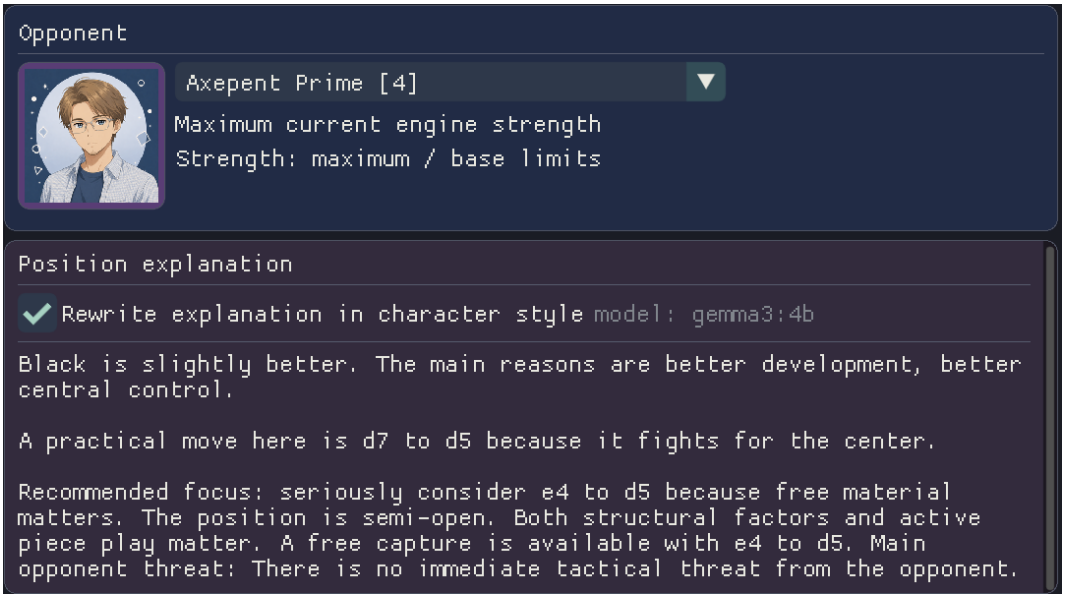
а)


Opponent
Matsushita Chiaki [2]
Academic / principled profile
Depth: 2-3 | current: 2
Position explanation
Rewrite explanation in character style model: gemma3:4b
White is a little better. This mostly comes from more active pieces.
A principled move here is b8 to c6 because it develops a piece.
Recommended focus: finish development, contest the center, and castle on time. The position is open, so activity, development, and king safety become more important. Opponent threat: There is no immediate tactical threat from the opponent.

б)



в)



г)

Рисунок 4.15 Приклад зміни стилістики пояснення залежно від обраного бота.

а) пояснювальний модуль Miku; б) пояснювальний модуль Matsushita;

в) пояснювальний модуль Houtarou; г) пояснювальний модуль Axepent Prime

5. АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

5.1. Порівняння з попередньо розглянутими рішеннями

Таблиця 2 Порівняння розглянутих рішень та розробленого проєктку

	БОТи для різних рівнів гри	Аналіз позиції під час гри	Пояснення позиції	Аналіз повної гри	Аналіз загальних помилок	Адаптивність пояснень
Chess.com	+	-+	-	+	-	-
DecodeChess	+	+	+	+-	-	-
Maia	+	+-	-	+	-	-
Aimchess	+	-	-	+	+	-+
En Croissant + Stockfish	-	-	-	-+	-+	-
AxepEngine	+	+	+	-	-	+

Наведене порівняння показує (табл. 2), що кожне з розглянутих рішень має власну сильну сторону. Chess.com забезпечує зручне середовище для гри з ботами різного рівня та аналізу партії після її завершення. DecodeChess вирізняється можливістю пояснення окремих позицій і аналізу під час гри. Maia зосереджується на моделюванні людського стилю гри, а Aimchess — на довгостроковому аналізі типових помилок користувача. Поєднання En Croissant + Stockfish демонструє сильний технічний аналіз, однак майже не має навчального пояснювального шару для користувача.

На цьому фоні результатом розробки AxepEngine стало поєднання декількох функцій в одному середовищі. Серед розглянутих рішень саме AxepEngine одночасно підтримує гру з ботами різного рівня, аналіз позиції під час гри, пояснення позиції та адаптивність пояснень. Це дозволяє розглядати розроблений додаток не лише як шаховий рушій або графічну оболонку для гри,

а як навчально-аналітичну систему, орієнтовану на користувачів різного рівня підготовки.

Особливо важливим результатом є реалізація адаптивного пояснення. У більшості розглянутих систем пояснення або відсутнє, або має однаковий характер незалежно від рівня користувача. У AxepEngine пояснення пов’язане з профілем бота та стилем подачі інформації, тому одна й та сама позиція може бути пояснена більш просто або більш детально залежно від обраного рівня. Це є суттєвою якісною відмінністю розробленого додатку.

5.2. Значення адаптивного пояснювального модуля

Ключовим результатом розробки є пояснювальний модуль, який формує текстові пояснення на основі фактичних даних шахового рушія. До таких даних належать FEN-позиція, активний профіль бота, режим пояснення, останній хід, найкращий хід, оцінка позиції, тип позиції, загрози суперника, наявність незахищених фігур, можливість безпечного взяття та інші позиційні ознаки.

Адаптивність пояснення проявляється в тому, що система може подавати інформацію з різним рівнем деталізації. Для слабшого профілю пояснення може акцентувати увагу на базових принципах: розвитку фігур, безпеці короля, контролі центру або простих тактичних загрозах. Для сильнішого профілю пояснення може бути більш аналітичним і враховувати складніші позиційні фактори. Таким чином, користувач отримує не просто текстовий коментар, а пояснення, яке краще відповідає його поточному рівню розуміння шахів.

Саме ця властивість відрізняє AxepEngine від більшості розглянутих аналогів. Частина систем має сильний рушійний аналіз, але не пояснює його достатньо зрозуміло. Інші рішення надають навчальні рекомендації, але не забезпечують гнучкої інтеграції пояснення безпосередньо з процесом гри. AxepEngine поєднує ці підходи: користувач може грати, отримувати оцінку

позиції, бачити рекомендований хід і одразу отримувати пояснення, адаптоване до обраного профілю.

5.3. Навчальна цінність розробленої системи

Розроблений додаток має навчальну цінність завдяки поєднанню декількох компонентів: ботів різного рівня, аналізу позиції під час гри, оцінки позиції та текстових пояснень. Це дозволяє користувачу не лише зіграти партію, а й краще зрозуміти логіку окремих рішень без необхідності самостійно інтерпретувати лише числову оцінку рушія.

На відміну від класичних шахових рушіїв, які зазвичай орієнтовані на пошук найсильнішого ходу, AxepEngine також враховує навчальний аспект. Користувач може обрати бота відповідного рівня, поступово переходити до сильніших профілів і отримувати пояснення, які супроводжують процес гри. Це робить систему корисною для початківців і гравців середнього рівня, для яких важливо не тільки дізнатися, який хід є кращим, а й зрозуміти причину цього вибору.

Таким чином, AxepEngine займає проміжну нішу між шаховим рушієм, навчальною платформою та пояснювальною системою. Його результатом є не максимальна сила гри сама по собі, а створення середовища, у якому аналіз позиції поєднується з доступним поясненням і адаптацією до рівня користувача.

5.4. Обмеження поточної версії та напрями подальшого розвитку

Порівняльна таблиця 2 також дозволяє визначити обмеження поточної версії AxepEngine. Найбільш помітними з них є відсутність аналізу повної партії після її завершення та відсутність модуля виявлення загальних

повторюваних помилок користувача. Ці функції реалізовані або частково реалізовані в окремих розглянутих рішеннях, зокрема в Chess.com та Aimchess.

Подальший розвиток системи доцільно спрямувати на реалізацію повного аналізу партії. Такий модуль міг би автоматично визначати ключові моменти гри, класифікувати ходи за якістю, знаходити критичні помилки, неточності, втрачені можливості та вдалі рішення. Це дозволило б перейти від пояснення окремої позиції до цілісного розбору всієї партії.

Ще одним перспективним напрямом є персоналізований аналіз стабільних помилок користувача. Система могла б накопичувати інформацію про зіграні партії та виявляти повторювані проблеми: слабку гру в дебюті, тактичні прорахунки, недостатній розвиток фігур, передчасні атаки, неточності в ендшпілі або слабку реалізацію переваги. На основі цього можна було б формувати індивідуальні рекомендації для навчання.

Також доцільним є подальше посилення шахового рушія, розширення дебютної книги, покращення ендшпільної гри та додавання окремої вкладки для вивчення дебютів. Це дозволило б розширити AxepEngine від системи гри й пояснення позицій до більш повноцінного навчального шахового середовища.

ВИСНОВКИ

У межах дипломного проєкту було отримано такі результати:

1. Розроблено програмний додаток AxepEngine, який поєднує шаховий рушій, графічний інтерфейс користувача, консольний режим роботи та пояснювальний модуль.

2. Реалізовано шахове ядро AxepCore, яке забезпечує представлення шахової позиції, генерацію легальних ходів, перевірку правил гри, оцінювання позиції та пошук рекомендованого ходу.

3. Реалізовано підтримку основних шахових правил і ситуацій завершення партії, зокрема шаху, мату, пату, повторення позиції, недостатнього матеріалу та правила п’ятдесяти ходів.

4. Створено оцінювальне ядро, яке враховує матеріальні, позиційні, тактичні та ендшпільні фактори: активність фігур, розвиток, контроль центру, пішакову структуру, безпеку короля, тиск на короля та наявність незахищених фігур.

5. Реалізовано профілі ботів різної сили гри, що дозволяє використовувати систему як навчальний інструмент для користувачів різного рівня підготовки.

6. Розроблено пояснювальний модуль, який формує текстові пояснення на основі фактичних даних, отриманих від шахового рушія.

7. Забезпечено адаптивність пояснень відповідно до обраного профілю бота, що дозволяє подавати шахову інформацію з різним рівнем складності.

8. Реалізовано графічний інтерфейс AxepEngineUI, який забезпечує гру на шаховій дошці, вибір профілю бота, перегляд історії ходів, оцінки позиції та текстових пояснень.

9. Проведено тестування шахового ядра, алгоритмів пошуку, профілів ботів, графічного інтерфейсу та пояснювального модуля.

10. Проведено порівняння з розглянутими аналогами показало, що AxepEngine поєднує гру з ботами різного рівня, аналіз позиції під час гри, пояснення позиції та адаптивність пояснень у межах одного додатку.

11. Подальший розвиток системи доцільно спрямувати на реалізацію аналізу повної партії, виявлення типових помилок користувача, посилення шахового рушія та розширення навчальних можливостей.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. *Aimchess*. URL: https://aimchess.com/home.
2. Category:SVG chess pieces - Wikimedia Commons. *Wikimedia Commons*. URL: https://commons.wikimedia.org/wiki/Category:SVG_chess_pieces.
3. *Chess.com*. URL: http://chess.com.
4. *Chessprogramming wiki*. URL: https://www.chessprogramming.org/Main_Page.
5. *CMake - Upgrade Your Software Build System*. URL: https://cmake.org.
6. *DecodeChess*. URL: https://decodechess.com.
7. Documentation for SFML 3.1.0 - Simple and Fast Multimedia Library. *Home - Simple and Fast Multimedia Library*. URL: https://www.sfml-dev.org/documentation/3.1.0/.
8. En Croissant. *En Croissant - The Ultimate Chess Toolkit | En Croissant*. URL: https://encroissant.org.
9. Gavriel T. The Complete Guide to Chess Tactics. *Udemy*.
10. *Maia Chess*. URL: https://www.maiachess.com.
11. Mouzouni C. Ollama Python Tutorial: Complete 2026 Guide (with Code Examples). *Cohorte – AI Bootcamp + Courses for Professionals Who Ship Systems*. URL: https://cohorte.co/blog/using-ollama-with-python-step-by-step-guide.
12. *Ollama*. URL: https://ollama.com.
13. Rozman L. How to win at chess: the ultimate guide for beginners and beyond. Ten Speed Press, 2023. 272 p.
14. Rozman L. *YouTube*. URL: https://www.youtube.com/@GothamChess.
15. *Stockfish*. URL: https://stockfishchess.org.

ДОДАТОК 1

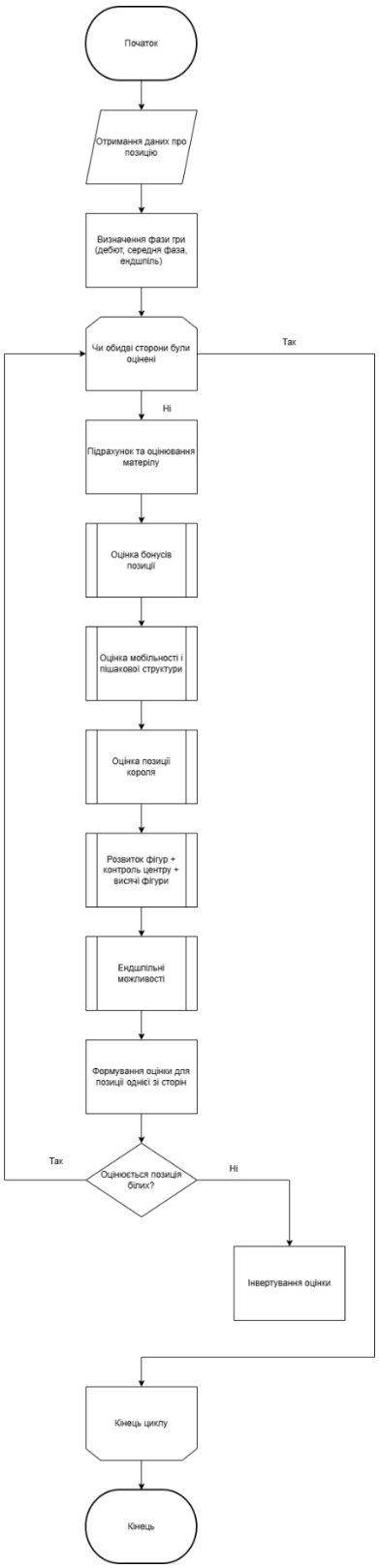
Розробка гібридного шахового рушія із інтеграцією оціночного ядра та пояснювального модуля на основі мовної моделі ШІ

Алгоритм роботи оціночного ядра рушія

ІАЛЦ.045420.005 Д1

Аркушів 1

Київ 2026р



Зм	Лист	№ докум.	Підп.	Дата	ІАЛЦ. 045420.005 Д1		
Розроб.		Гуманіцький			Розробка гібридного шахового рушія із інтеграцією оціночного ядра та пояснювального модуля на основі мовної моделі ШІ	Літ.	Лист	Листів
Перев.		Сергієнко						
Н. контр.		Клятченко			Алгоритм роботи оціночного ядра рушія	НТУУ «КПІ ім. І. Сікорського», ФПМ, КВ-21		
Затв.		Романкевич						

ДОДАТОК 2

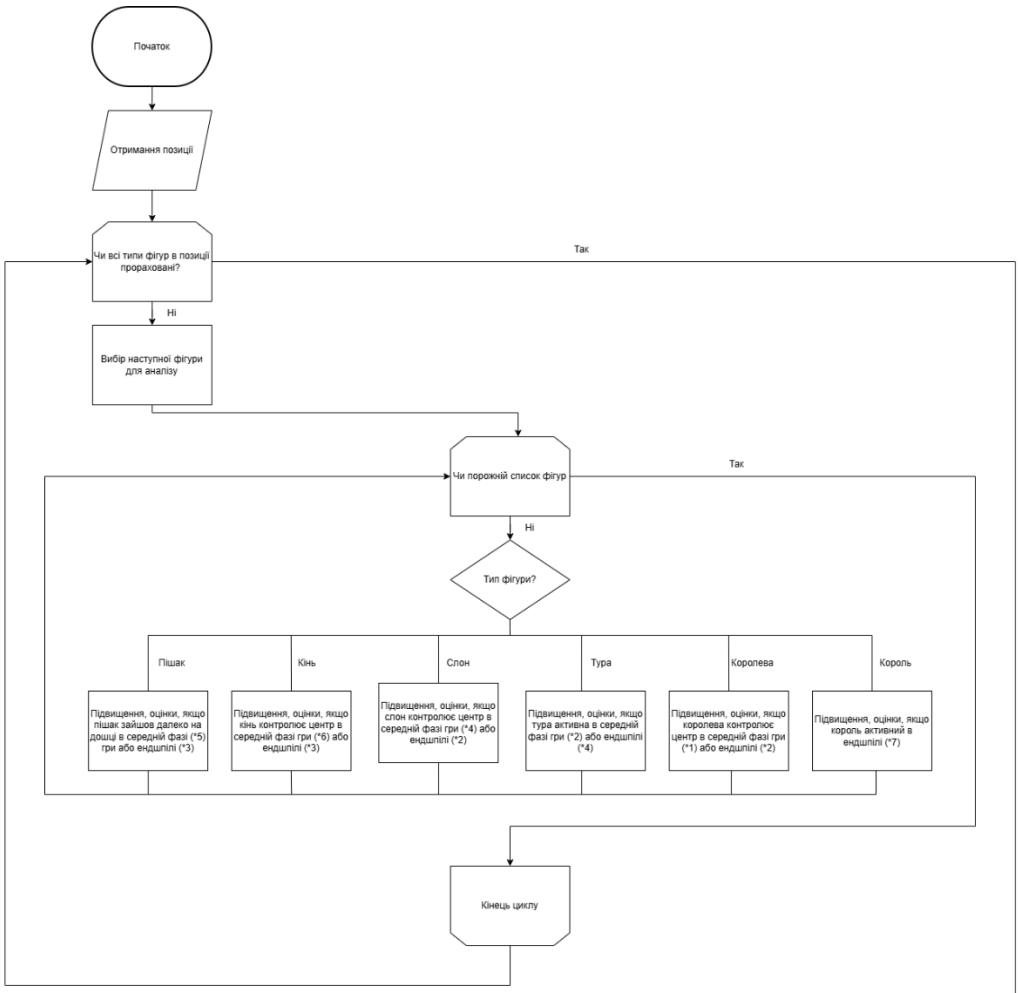
Розробка гібридного шахового рушія із інтеграцією оціночного ядра та пояснювального модуля на основі мовної моделі ШІ

Алгоритм оцінки бонусів в оціночному ядрі

ІАЛЦ.045420.006 Д2

Аркушів 1

Київ 2026р



					ІАЛЦ. 045420.006 Д2			
Зм	***Лист***	***№ докум.***	***Підп.***	***Дата***				
Розроб.		Гуманіцький			Розробка гібридного шахового рушія із інтеграцією оціночного ядра та пояснювального модуля на основі мовної моделі ШІ	***Літ.***	***Лист***	***Листів***
Перев.		Сергієнко						
						НТУУ «КПІ ім. І. Сікорського», ФПМ, КВ-21		
Н. контр.		Клятченко			**Алгоритм оцінки бонусів в оціночному ядрі**			
Затв.		Романкевич						

ДОДАТОК 3

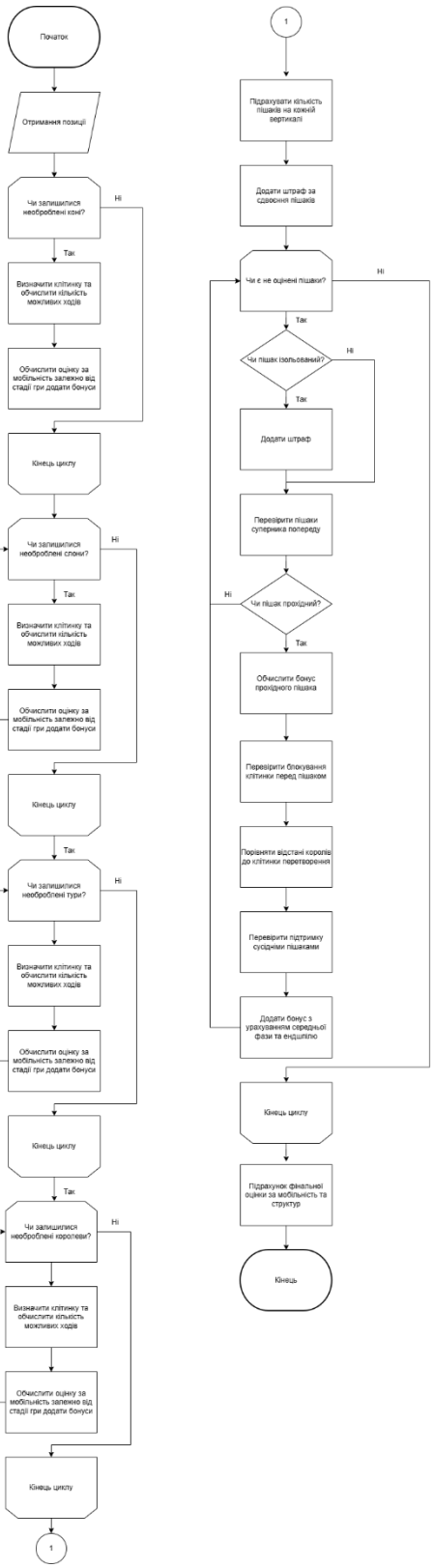
Розробка гібридного шахового рушія із інтеграцією оціночного ядра та пояснювального модуля на основі мовної моделі ШІ

Алгоритм оцінки мобільності та пішакової структури в оціночному ядрі

ІАЛЦ.045420.007 Д3

Аркушів 1

Київ 2026р



Зм	Лист	№ докум.	Підп.	Дата	ІАЛЦ. 045420.007 ДЗ		
Розроб.		Гуманіцький			Розробка гібридного шахового рушія із інтеграцією оціночного ядра та пояснювального модуля на основі мовної моделі ШІ	Літ.	Лист	Листів
Перев.		Сергієнко						
Н. контр.		Клятченко			Алгоритм оцінки мобільності та пішакової структури в оціночному ядрі	НТУУ «КПІ ім. І. Сікорського», ФПМ, КВ-21		
Затв.		Романкевич						

ДОДАТОК 4

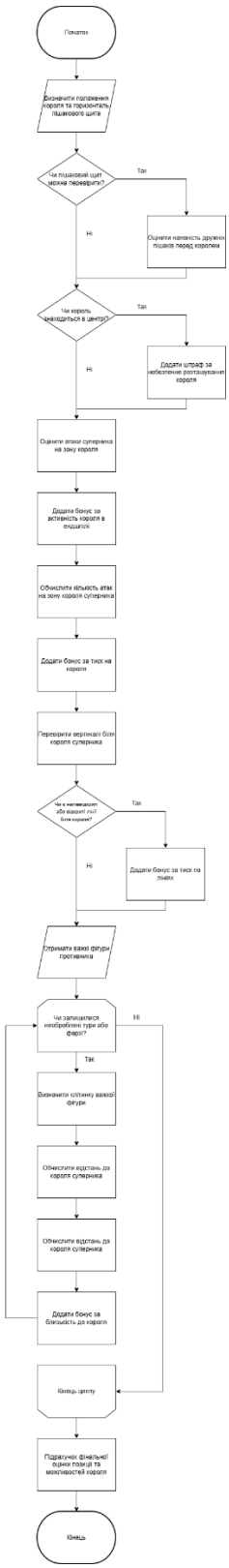
Розробка гібридного шахового рушія із інтеграцією оціночного ядра та пояснювального модуля на основі мовної моделі ШІ

Алгоритм оцінки позиції короля та його можливостей в оціночному ядрі

ІАЛЦ.045420.008 Д4

Аркушів 1

Київ 2026р



Зм	Лист	№ докум.	Підп.	Дата	ІАЛЦ. 045420.008 Д4		
Розроб.		Гуманіцький			Розробка гібридного шахового рушія із інтеграцією оціночного ядра та пояснювального модуля на основі мовної моделі ШІ	Літ.	Лист	Листів
Перев.		Сергієнко						
						НТУУ «КПІ ім. І. Сікорського», ФПМ, КВ-21		
Н. контр.		Клятченко			Алгоритм оцінки позиції короля та його можливостей в оціночному ядрі			
Затв.		Романкевич						

ДОДАТОК 5

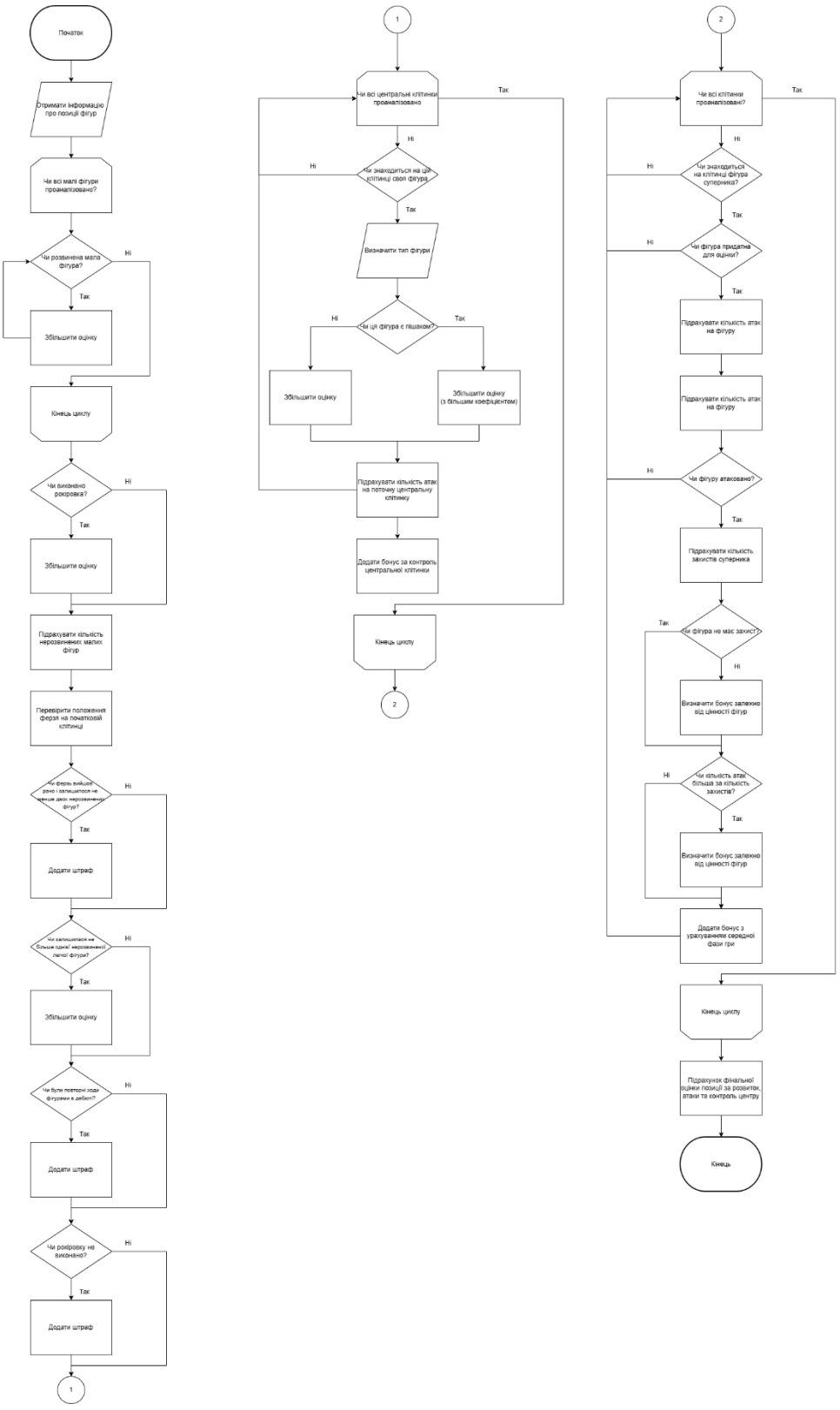
Розробка гібридного шахового рушія із інтеграцією оціночного ядра та пояснювального модуля на основі мовної моделі ШІ

Алгоритм оцінки розвитку фігур, контролю центру та перевірки висячих фігур в оціночному ядрі

ІАЛЦ.045420.009 Д5

Аркушів 1

Київ 2026р



Зм	Лист	№ докум.	Підп.	Дата	ІАЛЦ. 045420.009 Д5		
Розроб.		Гуманіцький			Розробка гібридного шахового рушія із інтеграцією оціночного ядра та пояснювального модуля на основі мовної моделі ШІ	Літ.	Лист	Листів
Перев.		Сергієнко						
Н. контр.		Клятченко			Алгоритм оцінки розвитку фігур, контролю центру та перевірки висячих фігур в оціночному ядрі	НТУУ «КПІ ім. І. Сікорського», ФПМ, КВ-21		
Затв.		Романкевич						

ДОДАТОК 6

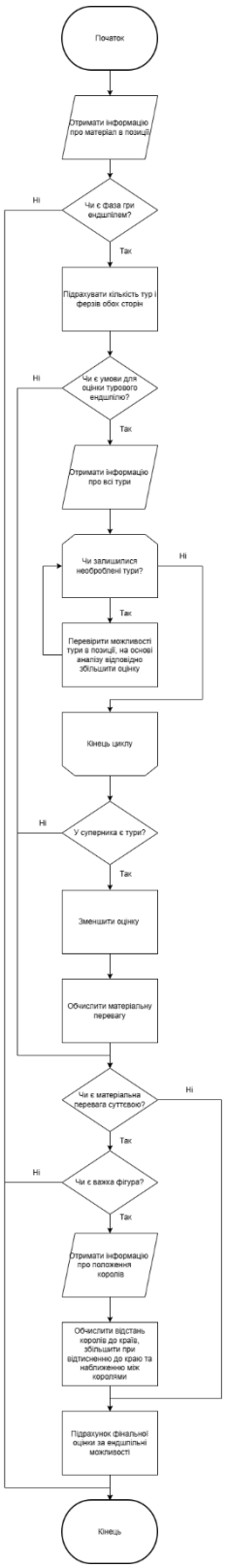
Розробка гібридного шахового рушія із інтеграцією оціночного ядра та пояснювального модуля на основі мовної моделі ШІ

Алгоритм оцінки ендшпільних можливостей в оціночному ядрі

ІАЛЦ.045420.010 Д6

Аркушів 1

Київ 2026р



Зм	Лист	№ докум.	Підп.	Дата	ІАЛЦ. 045420.010 Д6		
Розроб.		Гуманіцький			Розробка гібридного шахового рушія із інтеграцією оціночного ядра та пояснювального модуля на основі мовної моделі ШІ	Літ.	Лист	Листів
Перев.		Сергієнко						
Н. контр.		Клятченко			Алгоритм оцінки ендшпільних можливостей в оціночному ядрі	НТУУ «КПІ ім. І. Сікорського», ФПМ, КВ-21		
Затв.		Романкевич						

ДОДАТОК 7

Розробка гібридного шахового рушія із інтеграцією оціночного ядра та пояснювального модуля на основі мовної моделі ШІ

Лістинг коду оціночного ядра рушія

ІАЛЦ.045420.011 Д7

Аркушів 15

Київ 2026р

```
#include "eval/eval.h"

#include <algorithm>
#include <array>
#include <cmath>
#include "core/bitboard.h"
#include "core/constants.h"
#include "movegen/attacks.h"

namespace axe {

namespace {

constexpr int A2_SQ = 8;
constexpr int F2_SQ = 13;
constexpr int G2_SQ = 14;
constexpr int H2_SQ = 15;

constexpr int A4_SQ = 24;
constexpr int G4_SQ = 30;
constexpr int H4_SQ = 31;

constexpr int A5_SQ = 32;
constexpr int G5_SQ = 38;
constexpr int H5_SQ = 39;

constexpr int A6_SQ = 40;
constexpr int H6_SQ = 47;

constexpr int D4_SQ = 27;
constexpr int E4_SQ = 28;
constexpr int D5_SQ = 35;
constexpr int E5_SQ = 36;

constexpr int A7_SQ = 48;
constexpr int F7_SQ = 53;
constexpr int G7_SQ = 54;
constexpr int H7_SQ = 55;

constexpr int B8_SQ = 57;
constexpr int C8_SQ = 58;
constexpr int D8_SQ = 59;
constexpr int F8_SQ = 61;
constexpr int G8_SQ = 62;

struct PhaseWeights {
    int opening = 0;
    int middlegame = 0;
    int endgame = 0;
};

int scaleTerm(int value, int weight) {
    return (value * weight) / 100;
}

int centerBonus(int square) {
    const int f = fileOf(square);
    const int r = rankOf(square);
```

```
    const int df = std::abs(3 - f);
    const int dr = std::abs(3 - r);
    return 6 - (df + dr);
}

int manhattanDistance(int a, int b) {
    return std::abs(fileOf(a) - fileOf(b)) + std::abs(rankOf(a) - rankOf(b));
}

bool hasFriendlyPawnOnSquare(const Board& board, int side, int square) {
    if (square < 0 || square >= 64) {
        return false;
    }
    return board.colorOn[square] == side && board.pieceOn[square] == PAWN;
}

int totalNonPawnMaterial(const Board& board) {
    int total = 0;

    for (int side = 0; side < COLOR_NB; ++side) {
        total += countBits(board.pieces[side][KNIGHT]) * PIECE_VALUES[KNIGHT];
        total += countBits(board.pieces[side][BISHOP]) * PIECE_VALUES[BISHOP];
        total += countBits(board.pieces[side][ROOK]) * PIECE_VALUES[ROOK];
        total += countBits(board.pieces[side][QUEEN]) * PIECE_VALUES[QUEEN];
    }

    return total;
}

int sideMaterial(const Board& board, int side) {
    int total = 0;
    total += countBits(board.pieces[side][PAWN]) * PIECE_VALUES[PAWN];
    total += countBits(board.pieces[side][KNIGHT]) * PIECE_VALUES[KNIGHT];
    total += countBits(board.pieces[side][BISHOP]) * PIECE_VALUES[BISHOP];
    total += countBits(board.pieces[side][ROOK]) * PIECE_VALUES[ROOK];
    total += countBits(board.pieces[side][QUEEN]) * PIECE_VALUES[QUEEN];
    return total;
}

PhaseWeights computePhaseWeights(const Board& board) {
    PhaseWeights w;

    const int npm = totalNonPawnMaterial(board);
    const int mgBase = std::clamp((npm - 1200) * 100 / 4800, 0, 100);
    const int egBase = 100 - mgBase;

    int openingWeight = 0;
    if (board.fullmoveNumber <= 14 && npm >= 3800) {
        openingWeight = std::clamp((15 - board.fullmoveNumber) * 10, 0, 100);
    }

    w.opening = openingWeight;
    w.middlegame = std::max(mgBase, openingWeight / 2);
    w.endgame = egBase;

    return w;
}
```

```
int countAttackers(const Board& board, int side, int square) {
    int attackers = 0;
    const Bitboard allOcc = board.occupancies[BOTH];

    if (side == WHITE) {
        attackers += countBits(movegen::pawnAttacks[BLACK][square] & board.pieces[WHITE][PAWN]);
    } else {
        attackers += countBits(movegen::pawnAttacks[WHITE][square] & board.pieces[BLACK][PAWN]);
    }

    attackers += countBits(movegen::knightAttacks[square] & board.pieces[side][KNIGHT]);
    attackers += countBits(movegen::kingAttacks[square] & board.pieces[side][KING]);

    const Bitboard bishopLike = board.pieces[side][BISHOP] | board.pieces[side][QUEEN];
    const Bitboard rookLike = board.pieces[side][ROOK] | board.pieces[side][QUEEN];

    attackers += countBits(movegen::getBishopAttacks(square, allOcc) & bishopLike);
    attackers += countBits(movegen::getRookAttacks(square, allOcc) & rookLike);

    return attackers;
}

bool isCastled(const Board& board, int side) {
    const int kingSq = board.kingSquare(side);
    if (side == WHITE) {
        return kingSq == G1 || kingSq == C1;
    }
    return kingSq == G8_SQ || kingSq == C8_SQ;
}

int undevelopedMinorCount(const Board& board, int side) {
    if (side == WHITE) {
        int count = 0;
        if (board.colorOn[B1] == WHITE && board.pieceOn[B1] == KNIGHT) ++count;
        if (board.colorOn[G1] == WHITE && board.pieceOn[G1] == KNIGHT) ++count;
        if (board.colorOn[C1] == WHITE && board.pieceOn[C1] == BISHOP) ++count;
        if (board.colorOn[F1] == WHITE && board.pieceOn[F1] == BISHOP) ++count;
        return count;
    }

    int count = 0;
    if (board.colorOn[B8_SQ] == BLACK && board.pieceOn[B8_SQ] == KNIGHT) ++count;
    if (board.colorOn[G8_SQ] == BLACK && board.pieceOn[G8_SQ] == KNIGHT) ++count;
    if (board.colorOn[C8_SQ] == BLACK && board.pieceOn[C8_SQ] == BISHOP) ++count;
    if (board.colorOn[F8_SQ] == BLACK && board.pieceOn[F8_SQ] == BISHOP) ++count;
    return count;
}

int evaluatePieceSquareBonus(const Board& board, int side, const PhaseWeights& w) {
    int score = 0;

    for (int piece = 0; piece < PIECE_TYPE_NB; ++piece) {
        Bitboard bb = board.pieces[side][piece];

        while (bb) {
            const int square = popLSB(bb);
            const int center = centerBonus(square);
```

```
            switch (piece) {
                case PAWN: {
                    const int advance = (side == WHITE ? rankOf(square) : (7 - rankOf(square)));
                    score += scaleTerm(advance * 5, w.middlegame);
                    score += scaleTerm(advance * 3, w.endgame);
                    break;
                }
                case KNIGHT:
                    score += scaleTerm(center * 6, w.middlegame);
                    score += scaleTerm(center * 3, w.endgame);
                    break;
                case BISHOP:
                    score += scaleTerm(center * 4, w.middlegame);
                    score += scaleTerm(center * 2, w.endgame);
                    break;
                case ROOK: {
                    const int advance = (side == WHITE ? rankOf(square) : (7 - rankOf(square)));
                    score += scaleTerm(advance * 2, w.middlegame);
                    score += scaleTerm(advance * 4, w.endgame);
                    break;
                }
                case QUEEN:
                    score += scaleTerm(center * 1, w.opening);
                    score += scaleTerm(center * 2, w.middlegame);
                    break;
                case KING:
                    score += scaleTerm(center * 7, w.endgame);
                    break;
                default:
                    break;
            }
        }
    }

    if (countBits(board.pieces[side][BISHOP]) >= 2) {
        score += 30;
    }

    return score;
}

int evaluateMobility(const Board& board, int side, const PhaseWeights& w) {
    const Bitboard ownOcc = board.occupancies[side];
    const Bitboard allOcc = board.occupancies[BOTH];
    int score = 0;

    Bitboard knights = board.pieces[side][KNIGHT];
    while (knights) {
        const int sq = popLSB(knights);
        const int mobility = countBits(movegen::knightAttacks[sq] & ~ownOcc);
        score += scaleTerm(mobility * 3, w.opening);
        score += scaleTerm(mobility * 4, w.middlegame);
        score += scaleTerm(mobility * 2, w.endgame);
    }

    Bitboard bishops = board.pieces[side][BISHOP];
    while (bishops) {
        const int sq = popLSB(bishops);
```

```
        const int mobility = countBits(movegen::getBishopAttacks(sq, allOcc) & ~ownOcc);
        score += scaleTerm(mobility * 4, w.opening);
        score += scaleTerm(mobility * 5, w.middlegame);
        score += scaleTerm(mobility * 3, w.endgame);
    }

    Bitboard rooks = board.pieces[side][ROOK];
    while (rooks) {
        const int sq = popLSB(rooks);
        const int mobility = countBits(movegen::getRookAttacks(sq, allOcc) & ~ownOcc);
        score += scaleTerm(mobility * 2, w.opening);
        score += scaleTerm(mobility * 3, w.middlegame);
        score += scaleTerm(mobility * 3, w.endgame);
    }

    Bitboard queens = board.pieces[side][QUEEN];
    while (queens) {
        const int sq = popLSB(queens);
        const int mobility = countBits(movegen::getQueenAttacks(sq, allOcc) & ~ownOcc);
        score += scaleTerm(mobility * 1, w.opening);
        score += scaleTerm(mobility * 2, w.middlegame);
        score += scaleTerm(mobility * 2, w.endgame);
    }

    return score;
}

int evaluatePawnStructure(const Board& board, int side, const PhaseWeights& w) {
    Bitboard pawns = board.pieces[side][PAWN];
    int fileCounts[8]{};
    int score = 0;

    Bitboard copy = pawns;
    while (copy) {
        const int sq = popLSB(copy);
        ++fileCounts[fileOf(sq)];
    }

    for (int file = 0; file < 8; ++file) {
        if (fileCounts[file] > 1) {
            score -= fileCounts[file] > 1 ? (fileCounts[file] - 1) * 14 : 0;
        }
    }

    copy = pawns;
    while (copy) {
        const int sq = popLSB(copy);
        const int file = fileOf(sq);
        const int rank = rankOf(sq);

        const bool leftFriendly = (file > 0 && fileCounts[file - 1] > 0);
        const bool rightFriendly = (file < 7 && fileCounts[file + 1] > 0);

        if (!leftFriendly && !rightFriendly) {
            score -= 12;
        }

        bool passed = true;
```

```
        Bitboard enemyPawns = board.pieces[oppositeColor(side)][PAWN];

        while (enemyPawns) {
            const int enemySq = popLSB(enemyPawns);
            const int enemyFile = fileOf(enemySq);
            const int enemyRank = rankOf(enemySq);

            if (std::abs(enemyFile - file) <= 1) {
                if (side == WHITE && enemyRank > rank) {
                    passed = false;
                    break;
                }
                if (side == BLACK && enemyRank < rank) {
                    passed = false;
                    break;
                }
            }
        }

        if (passed) {
            const int advance = (side == WHITE ? rank : (7 - rank));
            int bonus = 8 + advance * 5;

            const int forwardSq = (side == WHITE ? sq + 8 : sq - 8);
            if (forwardSq >= 0 && forwardSq < 64) {
                if (board.pieceOn[forwardSq] != NO_PIECE) {
                    bonus -= 10;
                } else {
                    bonus += 4;
                }
            }

            const int promotionSq = (side == WHITE ? file + 56 : file);
            const int ownKingSq = board.kingSquare(side);
            const int enemyKingSq = board.kingSquare(oppositeColor(side));

            const int ownDist = ownKingSq == -1 ? 7 : manhattanDistance(ownKingSq, promotionSq);
            const int enemyDist = enemyKingSq == -1 ? 7 : manhattanDistance(enemyKingSq, promotionSq);

            if (ownDist < enemyDist) {
                bonus += 8;
            } else if (enemyDist + 1 < ownDist) {
                bonus -= 8;
            }

            if (leftFriendly || rightFriendly) {
                bonus += 4;
            }

            score += scaleTerm(bonus, w.middlegame);
            score += scaleTerm(bonus * 2, w.endgame);
        }
    }

    return score;
}

int countEnemyAttacksOnKingZone(const Board& board, int side) {
```

```
    const int kingSq = board.kingSquare(side);
    if (kingSq == -1) {
        return 0;
    }

    const int enemy = oppositeColor(side);
    const Bitboard kingZone = movegen::kingAttacks[kingSq] | bit(kingSq);
    const Bitboard allOcc = board.occupancies[BOTH];
    int attacks = 0;

    Bitboard knights = board.pieces[enemy][KNIGHT];
    while (knights) {
        const int sq = popLSB(knights);
        if (movegen::knightAttacks[sq] & kingZone) {
            ++attacks;
        }
    }

    Bitboard bishops = board.pieces[enemy][BISHOP];
    while (bishops) {
        const int sq = popLSB(bishops);
        if (movegen::getBishopAttacks(sq, allOcc) & kingZone) {
            ++attacks;
        }
    }

    Bitboard rooks = board.pieces[enemy][ROOK];
    while (rooks) {
        const int sq = popLSB(rooks);
        if (movegen::getRookAttacks(sq, allOcc) & kingZone) {
            ++attacks;
        }
    }

    Bitboard queens = board.pieces[enemy][QUEEN];
    while (queens) {
        const int sq = popLSB(queens);
        if (movegen::getQueenAttacks(sq, allOcc) & kingZone) {
            attacks += 2;
        }
    }

    Bitboard enemyPawns = board.pieces[enemy][PAWN];
    while (enemyPawns) {
        const int sq = popLSB(enemyPawns);
        if (movegen::pawnAttacks[enemy][sq] & kingZone) {
            ++attacks;
        }
    }

    return attacks;
}

int evaluateKingPressure(const Board& board, int side, const PhaseWeights& w) {
    const int enemy = oppositeColor(side);
    const int enemyKingSq = board.kingSquare(enemy);
    if (enemyKingSq == -1) {
        return 0;
```

```
    }

    int score = 0;
    const int kingFile = fileOf(enemyKingSq);

    const int attackUnits = countEnemyAttacksOnKingZone(board, enemy);
    score += scaleTerm(attackUnits * 12, w.middlegame);

    for (int df = -1; df <= 1; ++df) {
        const int file = kingFile + df;
        if (file < 0 || file >= 8) {
            continue;
        }

        bool ownPawnOnFile = false;
        bool enemyPawnOnFile = false;

        Bitboard ownPawns = board.pieces[side][PAWN];
        while (ownPawns) {
            const int sq = popLSB(ownPawns);
            if (fileOf(sq) == file) {
                ownPawnOnFile = true;
                break;
            }
        }

        Bitboard enemyPawns = board.pieces[enemy][PAWN];
        while (enemyPawns) {
            const int sq = popLSB(enemyPawns);
            if (fileOf(sq) == file) {
                enemyPawnOnFile = true;
                break;
            }
        }

        if (!enemyPawnOnFile) {
            score += scaleTerm(8, w.middlegame);
        }
        if (!ownPawnOnFile && !enemyPawnOnFile) {
            score += scaleTerm(10, w.middlegame);
        }
    }

    Bitboard heavy = board.pieces[side][ROOK] | board.pieces[side][QUEEN];
    while (heavy) {
        const int sq = popLSB(heavy);
        const int dist = manhattanDistance(sq, enemyKingSq);
        score += scaleTerm(std::max(0, 7 - dist) * 3, w.middlegame);
    }

    return score;
}

int evaluateKingSafety(const Board& board, int side, const PhaseWeights& w) {
    const int kingSq = board.kingSquare(side);
    if (kingSq == -1) {
        return 0;
    }
```

```
    int score = 0;
    const int file = fileOf(kingSq);
    const int rank = rankOf(kingSq);
    const int forward = (side == WHITE) ? 1 : -1;
    const int shieldRank = rank + forward;

    if (shieldRank >= 0 && shieldRank < 8) {
        for (int df = -1; df <= 1; ++df) {
            const int nf = file + df;
            if (nf < 0 || nf >= 8) {
                continue;
            }

            const int sq = shieldRank * 8 + nf;
            if (hasFriendlyPawnOnSquare(board, side, sq)) {
                score += scaleTerm(10, w.opening + w.middlegame / 2);
            } else {
                score -= scaleTerm(14, w.opening + w.middlegame / 2);
            }
        }
    }

    if (file >= 2 && file <= 5) {
        score -= scaleTerm(centerBonus(kingSq) * 5, w.opening + w.middlegame / 2);
    }

    score -= scaleTerm(countEnemyAttacksOnKingZone(board, side) * 9, w.middlegame);
    score += scaleTerm(centerBonus(kingSq) * 7, w.endgame);

    return score;
}

int evaluateRepeatedPieceMoves(const Board& board, int side) {
    const std::size_t maxPly = std::min<std::size_t>(board.history.size(), 16);
    std::array<bool, 64> reachedSquares{};
    int penalty = 0;

    for (std::size_t i = 0; i < maxPly; ++i) {
        const int movingSide = (i % 2 == 0) ? WHITE : BLACK;
        if (movingSide != side) {
            continue;
        }

        const Move move = board.history[i].move;

        if (move.isNull() || move.piece() == PAWN || move.isCastling()) {
            continue;
        }

        if (reachedSquares[move.from()]) {
            penalty += (move.piece() == QUEEN ? 22 : 14);
        }

        reachedSquares[move.to()] = true;
    }

    return penalty;
```

```
}

int openingPawnDisciplinePenalty(const Board& board, int side) {
    int penalty = 0;

    if (side == WHITE) {
        if (board.colorOn[A2_SQ] != WHITE || board.pieceOn[A2_SQ] != PAWN) penalty += 10;
        if (board.colorOn[H2_SQ] != WHITE || board.pieceOn[H2_SQ] != PAWN) penalty += 12;

        if (board.colorOn[G2_SQ] != WHITE || board.pieceOn[G2_SQ] != PAWN) penalty += 8;
        if (board.colorOn[F2_SQ] != WHITE || board.pieceOn[F2_SQ] != PAWN) penalty += 6;

        if (board.colorOn[A4_SQ] == WHITE && board.pieceOn[A4_SQ] == PAWN) penalty += 10;
        if (board.colorOn[A5_SQ] == WHITE && board.pieceOn[A5_SQ] == PAWN) penalty += 14;
        if (board.colorOn[H4_SQ] == WHITE && board.pieceOn[H4_SQ] == PAWN) penalty += 14;
        if (board.colorOn[H5_SQ] == WHITE && board.pieceOn[H5_SQ] == PAWN) penalty += 18;
        if (board.colorOn[G4_SQ] == WHITE && board.pieceOn[G4_SQ] == PAWN) penalty += 10;
        if (board.colorOn[G5_SQ] == WHITE && board.pieceOn[G5_SQ] == PAWN) penalty += 14;
    } else {
        if (board.colorOn[A7_SQ] != BLACK || board.pieceOn[A7_SQ] != PAWN) penalty += 10;
        if (board.colorOn[H7_SQ] != BLACK || board.pieceOn[H7_SQ] != PAWN) penalty += 12;

        if (board.colorOn[G7_SQ] != BLACK || board.pieceOn[G7_SQ] != PAWN) penalty += 8;
        if (board.colorOn[F7_SQ] != BLACK || board.pieceOn[F7_SQ] != PAWN) penalty += 6;

        if (board.colorOn[A5_SQ] == BLACK && board.pieceOn[A5_SQ] == PAWN) penalty += 10;
        if (board.colorOn[A4_SQ] == BLACK && board.pieceOn[A4_SQ] == PAWN) penalty += 14;
        if (board.colorOn[H5_SQ] == BLACK && board.pieceOn[H5_SQ] == PAWN) penalty += 14;
        if (board.colorOn[H4_SQ] == BLACK && board.pieceOn[H4_SQ] == PAWN) penalty += 18;
        if (board.colorOn[G5_SQ] == BLACK && board.pieceOn[G5_SQ] == PAWN) penalty += 10;
        if (board.colorOn[G4_SQ] == BLACK && board.pieceOn[G4_SQ] == PAWN) penalty += 14;
    }

    return penalty;
}

int evaluateDevelopment(const Board& board, int side, const PhaseWeights& w) {
    int score = 0;

    const bool knightBHome = board.colorOn[side == WHITE ? B1 : B8_SQ] == side
                          && board.pieceOn[side == WHITE ? B1 : B8_SQ] == KNIGHT;
    const bool knightGHome = board.colorOn[side == WHITE ? G1 : G8_SQ] == side
                          && board.pieceOn[side == WHITE ? G1 : G8_SQ] == KNIGHT;
    const bool bishopCHome = board.colorOn[side == WHITE ? C1 : C8_SQ] == side
                          && board.pieceOn[side == WHITE ? C1 : C8_SQ] == BISHOP;
    const bool bishopFHome = board.colorOn[side == WHITE ? F1 : F8_SQ] == side
                          && board.pieceOn[side == WHITE ? F1 : F8_SQ] == BISHOP;

    if (!knightBHome) score += scaleTerm(20, w.opening);
    if (!knightGHome) score += scaleTerm(20, w.opening);
    if (!bishopCHome) score += scaleTerm(18, w.opening);
    if (!bishopFHome) score += scaleTerm(18, w.opening);

    if (isCastled(board, side)) {
        score += scaleTerm(54, w.opening) + scaleTerm(28, w.middlegame);
    }

    const int undeveloped = undevelopedMinorCount(board, side);
```

```
    const int queenStart = (side == WHITE ? D1 : D8_SQ);

    if (board.colorOn[queenStart] != side || board.pieceOn[queenStart] != QUEEN) {
        if (undeveloped >= 2) {
            score -= scaleTerm(26 + undeveloped * 6, w.opening);
        }
    }

    if (undeveloped <= 1) {
        score += scaleTerm(14, w.opening);
    }

    score -= scaleTerm(evaluateRepeatedPieceMoves(board, side), w.opening);

    if (!isCastled(board, side)) {
        score -= scaleTerm(openingPawnDisciplinePenalty(board, side), w.opening);
    }

    return score;
}

int evaluateCentralControl(const Board& board, int side, const PhaseWeights& w) {
    static constexpr int centralSquares[4] = { D4_SQ, E4_SQ, D5_SQ, E5_SQ };
    int score = 0;

    for (int square : centralSquares) {
        if (board.colorOn[square] == side) {
            const int piece = board.pieceOn[square];
            score += scaleTerm(piece == PAWN ? 16 : 10, w.opening);
            score += scaleTerm(piece == PAWN ? 10 : 7, w.middlegame);
        }

        score += scaleTerm(countAttackers(board, side, square) * 6, w.opening);
        score += scaleTerm(countAttackers(board, side, square) * 5, w.middlegame);
    }

    return score;
}

int evaluateHangingPieces(const Board& board, int side, const PhaseWeights& w) {
    const int enemy = oppositeColor(side);
    int score = 0;

    for (int sq = 0; sq < 64; ++sq) {
        if (board.colorOn[sq] != enemy) {
            continue;
        }

        const int piece = board.pieceOn[sq];
        if (piece == NO_PIECE || piece == KING || piece == PAWN) {
            continue;
        }

        const int attacks = countAttackers(board, side, sq);
        if (attacks == 0) {
            continue;
        }
```

```
        const int defends = countAttackers(board, enemy, sq);
        int bonus = 0;

        if (defends == 0) {
            switch (piece) {
                case KNIGHT:
                case BISHOP: bonus = 18; break;
                case ROOK:   bonus = 28; break;
                case QUEEN:  bonus = 40; break;
                default: break;
            }
        } else if (attacks > defends) {
            switch (piece) {
                case KNIGHT:
                case BISHOP: bonus = 10; break;
                case ROOK:   bonus = 16; break;
                case QUEEN:  bonus = 22; break;
                default: break;
            }
        }

        score += scaleTerm(bonus, w.middlegame);
    }

    return score;
}

bool isPassedPawn(const Board& board, int side, int sq) {
    if (board.colorOn[sq] != side || board.pieceOn[sq] != PAWN) {
        return false;
    }

    const int file = fileOf(sq);
    const int rank = rankOf(sq);
    Bitboard enemyPawns = board.pieces[oppositeColor(side)][PAWN];

    while (enemyPawns) {
        const int enemySq = popLSB(enemyPawns);
        const int enemyFile = fileOf(enemySq);
        const int enemyRank = rankOf(enemySq);

        if (std::abs(enemyFile - file) <= 1) {
            if (side == WHITE && enemyRank > rank) {
                return false;
            }
            if (side == BLACK && enemyRank < rank) {
                return false;
            }
        }
    }

    return true;
}

int evaluateRookEndgame(const Board& board, int side, const PhaseWeights& w) {
    if (w.endgame <= 0) {
        return 0;
    }
```

```
const int enemy = oppositeColor(side);
const int ownRooks = countBits(board.pieces[side][ROOK]);
const int enemyRooks = countBits(board.pieces[enemy][ROOK]);
const int ownQueens = countBits(board.pieces[side][QUEEN]);
const int enemyQueens = countBits(board.pieces[enemy][QUEEN]);

if (ownRooks == 0 || ownQueens != 0 || enemyQueens != 0) {
   return 0;
}

int score = 0;
const int enemyKingSq = board.kingSquare(enemy);

Bitboard rooks = board.pieces[side][ROOK];
while (rooks) {
   const int rookSq = popLSB(rooks);

   const int rookRank = rankOf(rookSq);
   if ((side == WHITE && rookRank == 6) || (side == BLACK && rookRank == 1)) {
      score += 16;
   }

   if (enemyKingSq != -1) {
      const int dist = manhattanDistance(rookSq, enemyKingSq);
      score += std::max(0, 7 - dist) * 2;
   }

   for (int sq = 0; sq < 64; ++sq) {
      if (!isPassedPawn(board, side, sq)) {
         continue;
      }

      if (fileOf(sq) != fileOf(rookSq)) {
         continue;
      }

      if (side == WHITE) {
         if (rankOf(rookSq) < rankOf(sq)) {
            score += 18;
         }
      } else {
         if (rankOf(rookSq) > rankOf(sq)) {
            score += 18;
         }
      }
   }

   for (int sq = 0; sq < 64; ++sq) {
      if (!isPassedPawn(board, enemy, sq)) {
         continue;
      }

      if (fileOf(sq) != fileOf(rookSq)) {
         continue;
      }

      if (side == WHITE) {
```

```
            if (rankOf(rookSq) > rankOf(sq)) {
                score += 10;
            }
        } else {
            if (rankOf(rookSq) < rankOf(sq)) {
                score += 10;
            }
        }
    }
}

if (enemyRooks > 0) {
    score /= 2;
}

return scaleTerm(score, w.endgame);
}

int evaluateMopUp(const Board& board, int side, const PhaseWeights& w) {
if (w.endgame <= 0) {
    return 0;
}

const int enemy = oppositeColor(side);
const int ownMaterial = sideMaterial(board, side);
const int enemyMaterial = sideMaterial(board, enemy);
const int advantage = ownMaterial - enemyMaterial;

if (advantage < 300) {
    return 0;
}

const bool hasMajor = countBits(board.pieces[side][ROOK] | board.pieces[side][QUEEN]) > 0;
if (!hasMajor) {
    return 0;
}

const int ownKingSq = board.kingSquare(side);
const int enemyKingSq = board.kingSquare(enemy);

if (ownKingSq == -1 || enemyKingSq == -1) {
    return 0;
}

const int enemyFile = fileOf(enemyKingSq);
const int enemyRank = rankOf(enemyKingSq);
const int distToEdge = std::min(std::min(enemyFile, 7 - enemyFile), std::min(enemyRank, 7 - enemyRank));
const int edgeBonus = (3 - std::min(distToEdge, 3)) * 10;

const int kingDistance = manhattanDistance(ownKingSq, enemyKingSq);
const int kingBonus = std::max(0, 14 - kingDistance) * 3;

return scaleTerm(edgeBonus + kingBonus, w.endgame);
}

}

Score Evaluator::evaluate(const Board& board) const {
```

```
    const PhaseWeights w = computePhaseWeights(board);
    int score = 0;

    for (int side = 0; side < COLOR_NB; ++side) {
        int sideScore = 0;

        for (int piece = 0; piece < PIECE_TYPE_NB; ++piece) {
            sideScore += countBits(board.pieces[side][piece]) * PIECE_VALUES[piece];
        }

        sideScore += evaluatePieceSquareBonus(board, side, w);
        sideScore += evaluateMobility(board, side, w);
        sideScore += evaluatePawnStructure(board, side, w);
        sideScore += evaluateKingSafety(board, side, w);
        sideScore += evaluateKingPressure(board, side, w);
        sideScore += evaluateDevelopment(board, side, w);
        sideScore += evaluateCentralControl(board, side, w);
        sideScore += evaluateHangingPieces(board, side, w);
        sideScore += evaluateRookEndgame(board, side, w);
        sideScore += evaluateMopUp(board, side, w);

        score += (side == WHITE) ? sideScore : -sideScore;
    }

    return score;
}

}
```

Розробка гібридного шахового рушія із інтеграцією оціночного ядра та пояснювального модуля на основі мовної моделі ШІ

ВИКОНАВ СТУДЕНТ IV КУРСУ , ГРУПИ КВ -21

ГУМАНІЦЬКИЙ АНДРІЙ ОЛЕКСАНДРОВИЧ
КЕРІВНИК: АСИСТЕНТ КАФЕДРИ СПІСКС

СЕРГІЄНКО ПАВЛО АНАТОЛІЙОВИЧ

Актуальність

Сучасні шахові рушії здатні точно аналізувати позиції та знаходити сильні ходи, проте їхні результати часто подаються у вигляді числових оцінок або варіантів без зрозумілого пояснення. Для початківців і гравців середнього рівня цього недостатньо, оскільки важливо розуміти логіку ходу, стратегічну ідею та типові помилки. Саме тому актуальною є розробка гібридного шахового рушія, який поєднує оціночне ядро з пояснювальним модулем на основі мовної моделі ШІ.

Мета

Створення шахового рушія, який не лише знаходить сильні ходи, а й пояснює їх користувачу у зрозумілій формі.

Для цього в системі поєднано:

- оціночне шахове ядро;
- профілі ботів різного рівня;
- пояснювальний модуль

Порівняння з існуючими рішеннями

У межах роботи були розглянуті сучасні шахові платформи та сервіси, які використовуються для гри, аналізу партій і навчання:

- Chess.com
- DecodeChess
- Maia
- Aimchess
- En Croissant + Stockfish

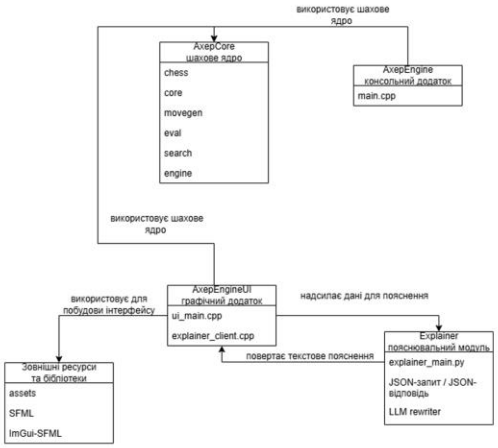
Ці рішення мають корисні функції, однак кожне з них вирішує лише частину задачі: або сильний аналіз, або навчання, або моделювання стилю гри.

Інструменти розробки

Для розробки додатку AxepEngine було використано такі програмні засоби:

- C++17 — основна мова розробки шахового рушія;
- CMake — система складання та організації компонентів проєкту;
- SFML 3 — відображення шахової дошки, фігур та візуальних елементів;
- ImGui-SFML — створення правої панелі інтерфейсу та елементів керування;
- Python — реалізація пояснювального модуля;
- Ollama / локальна LLM — перефразування пояснень у більш природній формі;
- FEN / UCI / SAN — формати для роботи з позиціями та ходами.

Структура проєкту



Опис консольного додатку

```
Commands:
  help                - show commands
  startpos            - load standard starting position
  fen <FEN>           - load position from FEN
  board               - print board
  legal               - print legal moves
  move <uci>          - play move manually, example: e2e4
  go                  - engine makes one move
  auto                - engine plays both sides until game over
  depth <n>           - set fixed depth mode for max profile
  movetime <min>      - set time per move in minutes for max profile
  bots                - show all bot profiles
  bot <id>            - select active bot profile
  rename <id> <name>  - rename bot profile
  profile             - show current bot profile
  perft <n>           - run perft divide for depth n
  quit                - exit program

Promotion examples:
  move c2c1q          - promote to queen
  move c2c1           - defaults to queen if promotion exists
```

```
Available bots:
    [1] Hatsune Miku | depth 1-1
    [2] Matsushita Chiaki | depth 2-3
    [3] Oreki Houtarou | depth 3-4
  * [4] Axepent Prime | maximum strength
```

```
Current bot profile:
  id: 4
  name: Axepent Prime
  strength: maximum
Base depth for max profile: 6
Time mode for max profile: off
```

Опис консольного додатку

```
8  r n b q k b n r
7  p p p p p p p p
6  . . . . . . . .
5  . . . . . . . .
4  . . . . . . . .
3  . . . . . . . .
2  P P P P P P P P
1  R N B Q K B N R

   a b c d e f g h

Side: white
Castling: KQkq
En passant: -
```

```
Bot: Axepent Prime [4] | depth 6
Book move: e7e6

8  r n . q k b n r
7  p p . . . p p p
6  . . p . p . . .
5  . . . p P b . .
4  . . . P . . . .
3  . . . . . N . .
2  P P P . . P P P
1  R N B Q K B . R

   a b c d e f g h

Side: white
Castling: KQkq
En passant: -
```

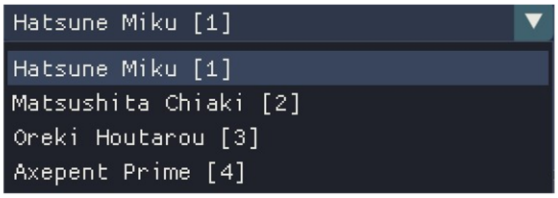
```
Bot: Axepent Prime [4] | depth 6
Best move: b8c6
Score: -104
Depth: 6
Nodes: 3658816
NPS: 63386
TT hits: 48909
Cutoffs: 1890246
Time ms: 57722
```

```
>>> movetime 0.1
Time per move for max profile set to 0.1 minute(s)
Max depth is ignored while time mode is active
```

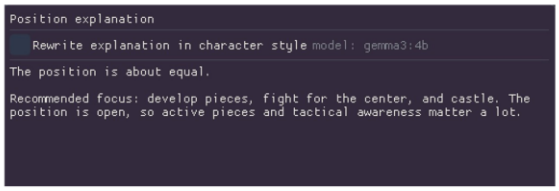
```
>>> fen rn1qkbnr/pp3ppp/4p3/2ppPb2/3P4/5N2/PPP1BPPP/RNBQK2R w KQkq - 0 1
FEN loaded.

8  r n . q k b n r
7  p p . . . p p p
6  . . . . p . . .
5  . . p p P b . .
4  . . . P . . . .
3  . . . . . N . .
2  P P P . B P P P
1  R N B Q K . . R

   a b c d e f g h
```

Опис розробленого додатку

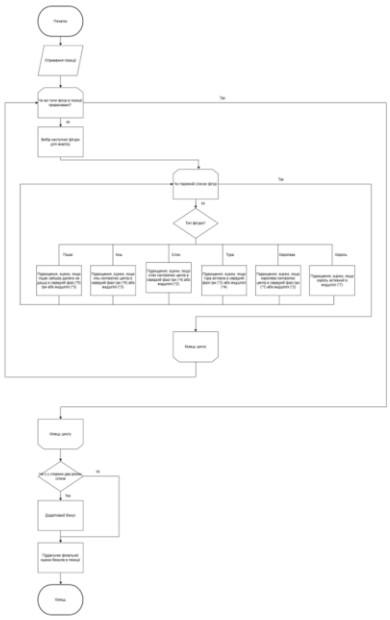
Опис розробленого додатку



Опис розробленого додатку



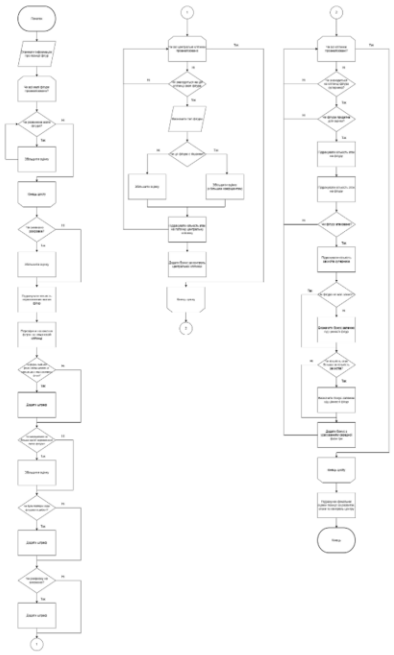
Алгоритм роботи оціночного ядра рушія



Алгоритм оцінки бонусів в оціночному ядрі

Алгоритм оцінки мобільності та пішакової структури в оціночному ядрі

Алгоритм оцінки позиції короля та його можливостей в оціночному ядрі



Алгоритм оцінки розвитку фігур, контролю центру та перевірки висячих фігур в оціночному ядрі

Алгоритм оцінки ендшпільних можливостей в оціночному ядрі

Висновки

У межах роботи було розроблено програмний додаток AxepEngine, який поєднує шаховий рушій, графічний інтерфейс та пояснювальний модуль.

- Основні результати роботи:
- реалізовано шахове ядро AxepCore;
- додано генерацію легальних ходів, оцінювання позиції та пошук найкращого ходу;
- створено консольну та графічну версії додатку;
- реалізовано профілі ботів різного рівня гри;
- додано підказки, оцінку позиції, історію ходів та блок пояснень;
- інтегровано пояснювальний модуль, який формує текстові пояснення на основі даних рушія.

Можливості модернізації

Подальший розвиток системи може бути спрямований на розширення аналітичних і навчальних можливостей додатку.

- Можливі напрями модернізації:
- додавання повного аналізу партії після її завершення;
- автоматичне визначення ключових моментів, помилок і втрачених можливостей;
- персоналізований аналіз стабільних помилок користувача;
- посилення шахового рушія;
- поглиблення дебютної книги;
- покращення ендшпільної гри;
- додавання окремої вкладки для вивчення дебютів.

Дякую за увагу!