Dokumentacja końcowa
Temat

Szachy z wykorzystaniem algorytmu MiniMax

Rozwinięcie tematu i opis

Tematem projektu jest implementacja szachów a dokładniej aplikacji komputerowej umożliwiającej grę w szachy w kilku trybach: użytkowni kontra użytkowniki, użytkownik kontra komputer. Aplikacja powinna posiadać interfejs graficzny prezentujący plansze oraz figury, a także podstawową komunikację z użytkownikiem prezentującą wynik rozgrywki, czy też niepoprawnie wykonany ruch. Silnik szachowy, który będzie sterował komputerem w trybie użytkownik kontra komputer będzie oparty na algorytmie MiniMax, funkcji ewaluacyjnej oraz tabeli transpozycji.

Zadania:

Zadanie	**Czas szacowany**	**Czas rzeczywisty**
Projekt aplikacji (utworzenie dokumentacji oraz szkieletu - powiązań między klasami)	20h	~20h
Przygotowanie środowiska - budowanie, testy, formater, repozytorium, CI/CD.	15h	~5h
GUI konsolowe	3h	~5h
GUI przy pomocy biblioteki SFML	20h	>30h
Reprezentacja szachownicy - klasy Board, MoveContent, Piece	5h	~20h
Generator wszystkich możliwych ruchów	5h	~10h
Generator pseudo legalnych ruchów	6h	~10h
Mechanizm roszady, en passant	3h	~3h
Zasada 50 ruchów, Zasada 3 powtórzeń	3h	~2h
Reprezentacja rozgrywki	3h	-

Funkcja ewaluacyjna	8h	~2h
Klasa wyszukująca właściwy ruch w tym:	15h	>30h
● Algorytm MiniMax	5h	>25h
● Sortowanie ruchów,	5h	zawarte w MiniMax
● Mechanizm Quiescence search	5h	>5h
Zorbist hashing	-	~8h
Suma	106h	145h

Wartości z tabeli są tylko szacunkami, ponieważ bardzo ciężko jest ocenić rzeczywisty czas poświęcony na rozwiązanie zadania. Oprócz samej implementacji głównych elementów, na pewno dużo czasu poświęciliśmy na czytanie na temat tworzenia silników szachowych, implementacji pobocznych funkcjonalności, testowanie oraz zrozumienie algorytmu alpha beta w stopniu umożliwiającym efektywne wykorzystanie oraz ciągłe usprawnianie.

Projekt udało się zrealizować, mimo napotkanych problemów. Nie udało nam się zaimplementować tablicy transpozycji do zapamiętywania wyników przeszukiwania tych samych pozycji i ruchów w celu przyspieszenia działania algorytmu wyszukiwania najlepszego ruchu. Hashowanie nie było problemem, natomiast nie zdążyliśmy zintegrować całego mechanizmu oraz go przetestować.

Poza tym problemy, które częściowo udało się rozwiązać należą do problemów optymalizacyjnych. Na chwilę obecną kod działa zdecydowanie za wolno żeby osiągać wysokie wyniki w rozgrywce w szachy, natomiast miałby na pewno szansę na wygraną z początkującymi graczami. Największą przeszkodą okazał się czas potrzebny na skrupulatne testowanie i inkrementalne wdrażanie usprawnień, które było dość żmudnym procesem. Jednak z uwagi na to, że projekt jest dość ambitny, to uważamy nasze wyniki za sukces i jesteśmy gotowi dalej rozwijać ten projekt.

Funkcjonalności:

Udało się zrealizować wszystkie założone w dokumentacji wstępnej funkcjonalności tj.:

- 1 Etap - implementacja zasad gry w szachy (użytkownik na przemian wykonuje ruchy białymi i czarnymi figurami)

    - Szachownica, na której można wykonywać tylko poprawne ruchy (narazie gracz rusza się białymi i czarnymi)
    - Generator ruchów pseudolosowych
    - Graficzna reprezentacja rozgrywki
- 2 Etap - implementacja algorytmu MiniMax do gry w szachy
    - Funkcja ewaluacyjna
    - MiniMax
    - Optymalizacja wyszukiwania najlepszego ruchu

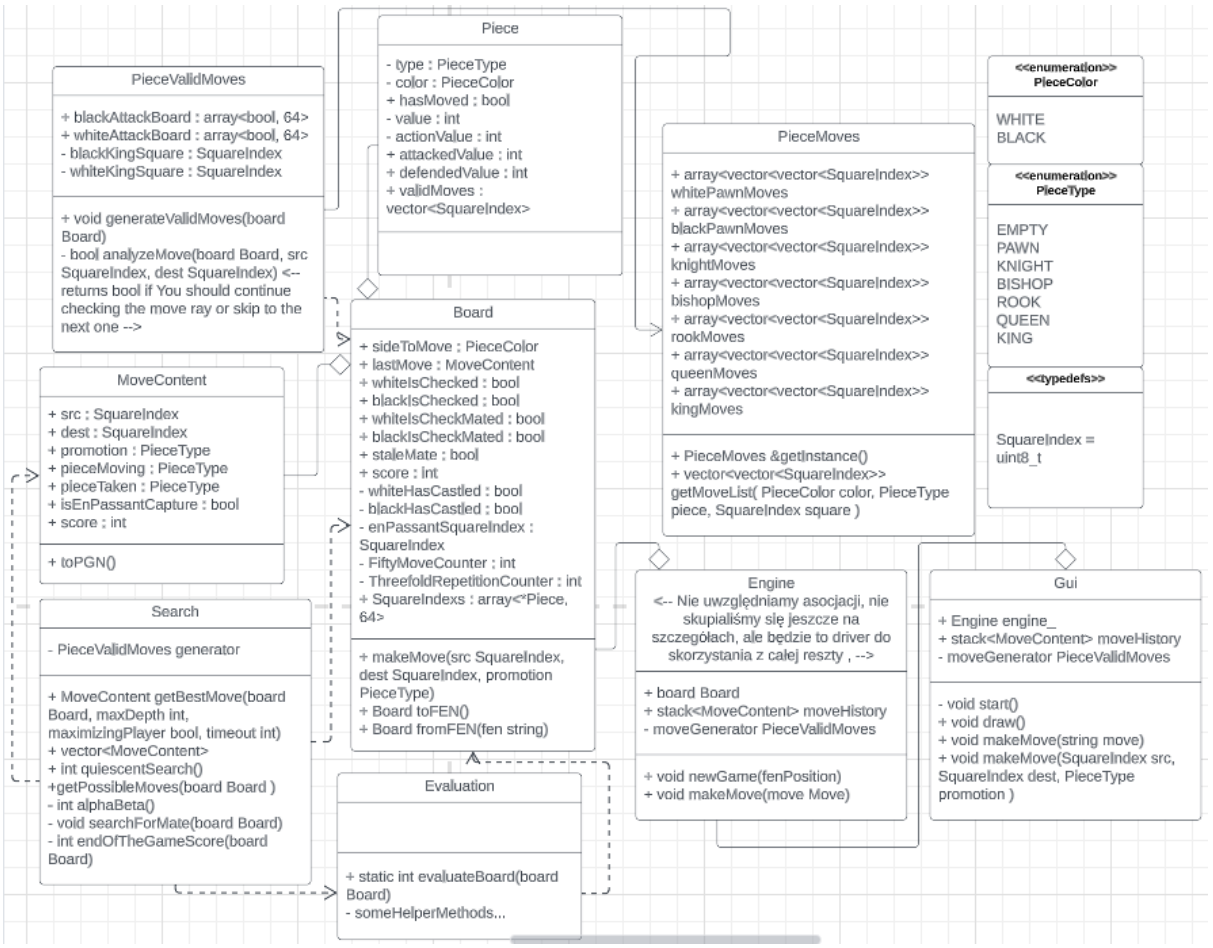
Mimo, że nie udało się osiągnąć efektywności, umożliwiającej wygraną z doświadczonym graczem w szachy, to projekt można nazwać silnikiem szachowym i posiada funkcjonalności wymienione powyżej. Aktualnie myślę, że można go porównać do bota na poziomie około 800 elo na platformie chess.com.

Braki:

- Program ma problemy z wyszukiwaniem wymuszonego mata w kilku ruchach do przodu. Nie udało nam się zidentyfikować przyczyny problemu. Być może jest to pewien błąd związany z logiką algorytmu.

Implementacja:

Poniżej przedstawiono diagram klas zaprojektowany na wstępnym etapie z uzupełnieniem dotyczącym szczegółów klas, które powstały w późniejszym etapie.



- W dokumentacji końcowej nie może zabraknąć opisu funkcjonalności jakie zrealizowano, a także opisu architektury aplikacji, gdzie należy wyjaśnić jakie moduły ze sobą współpracują, a także opisać główne klasy i ich rolę w systemie.

Najważniejsze klasy to klasy:

- Board - klasa reprezentująca szachownicę.
- Search - klasa udostępniająca metodę, które na podstawie obecnego stanu szachownicy jest w stanie podać najlepszy możliwy ruch, wykonując algorytm MinMax i ewaluując szachownicę.
- Gui - klasa odpowiadająca za wyświetlanie, rozpoczynanie i kończenie rozgrywki. Zawiera w sobie instancję klasy Engine, dzięki której ma dostęp do klasy Board oraz możliwość wykonywania ruchów. Zaimplementowano dwie wersję początkowo interfejs graficzny w konsoli, a następnie interfejs okienkowy.

Wnioski:

- Projekt umożliwił praktyczne poznanie języka C++ oraz rozwinięcie praktyk w tworzeniu dużego projektu - projektowanie powiązań między klasami, odpowiednie nazewnictwo klas, zmiennych funkcji itd.
- W przypadku szachów, stworzenie dobrego silnika wymaga zrozumienia zasad gry, zrozumienia sposobów implementacji, a przede wszystkim cierpliwości i iteracyjnej pracy nad optymalnością rozwiązania.