Projekt - Szachy

Programowanie obiektowe grupa 102

Link do repozytorium

Autorzy:
Juliusz Neuman, nr indeksu 292788 e mail: 01132997@pw.edu.pl
Arkadiusz Rybski, nr indeksu 292784 e mail: 01132993@pw.edu.pl
Krystian Szewczak, nr indeksu 292796, e mail: 01133005@pw.edu.pl

Wstęp

Cel projektu

Celem projektu było zaimplementowanie programu do rozgrywki w szachy. Możliwa miała być rozgrywka gracza z komputerem, albo dwóch graczy. Co więcej, aplikacja miała umożliwić zapisywanie rozgrywki.

Założenia projektowe

Środowisko

Kompilacja programu odbywa się z wykorzystaniem narzędzia **Cmake** i kompilatora **gcc**.

Implementacja zasad

W projekcie zaimplementowane zostały podstawowe zasady gry w szachy, takie jak prawidłowy ruch każdej z figur, stan gry podczas szacha, oraz zakończenia gry: mat oraz pat. Zdecydowano się pominąć jednak trzy ruchy, często w literaturze określane magicznymi ruchami w szachach tzn. roszada, bicie w przelocie (*en passant*) oraz promocji piona.

Silnik szachowy

W projekcie zdecydowano się, że silnik szachowy sterujący graczami komputerowymi będzie zwracał losowo wybrany przez siebie ruch. Jest to aspekt do dalszego rozwoju.

Wyświetlanie

Zdecydowano się na wyświetlanie gry w trybie konsoli, aczkolwiek przygotowano generyczny interfejs umożliwiający podmianę konkretnego sposobu wyświetlania, np. na okienkowy.

Interakcja z użytkownikiem

Zdecydowano się, że wszelka komunikacja z użytkownikiem będzie przebiegać z użyciem konsoli.

Opis realizacji projektu

Podział na moduły

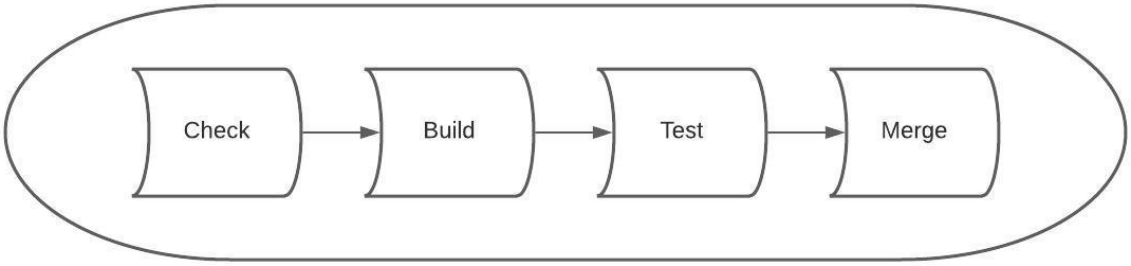
Projekt został podzielony na kilka modułów takich, jak:

- chess - odpowiedzialny za implementację klasy ***board, pieces***
- communicator - odpowiedzialny za implementację klasy ***communicator***
- controller - odpowiedzialny za implementację klasy ***controller***
- engine - odpowiedzialny za implementację klasy ***engine***
- game - odpowiedzialny za implementację klasy ***game***
- loader- odpowiedzialny za implementację klasy ***loader***
- runner - odpowiedzialny za implementację klasy ***runner***
- view - odpowiedzialny za implementację klasy ***view***

Połączenia między modułami zobrazowano na diagramie klas, dołączonym do raportu.

Opis testów i kontroli jakości kodu

W projekcie zastosowano ***Ciągłą Integrację (CI)***, której schemat przedstawiono na poniższym diagramie. Wykorzystano mechanizmy z repozytorium GitLab, kroki potoku realizowano za pomocą Gitlab Runnera działającego na dockerowym obrazie na urządzeniu Raspberry Pi.



1. Etap Check - statyczna analiza kodu z użyciem narzędzia **cppcheck**, m.in. z flagą *--enable=all*.
2. Etap Build - etap kompilacji z użyciem ***cmake*** i kompilatora ***gcc11***. Podczas kompilacji ustawione były flagi *-Wall -Wextra -pedantic -Werror*. Dodatkowo dołączono wykorzystanie narzędzia **clang-tidy** wkomponowanego w **cmake**.
3. Etap Test - etap, w którym przeprowadzone zostały testy jednostkowe (użyto framework ***google test***)
4. Etap Merge - nowo utworzony kod łączony był z całością projektu wyłącznie po przejściu poprzednich etapów. Oprócz tego każdy merge poprzedzony był **code review** innego autora projektu.

Dla całego projektu przygotowano ponad 90 testów jednostkowych, w tym niektórych o rozbudowanych scenariuszach. Dodatkowo po ukończeniu projektu wielokrotnie testowano manualnie wykonanie programu i rozgrywkę, a także pozostałe funkcjonalności.

W projekcie wykorzystywano wyłącznie inteligentne wskaźniki, więc ryzyko błędów pamięci zostało zdecydowanie ograniczone, ale dla sprawdzenia uruchomiono wszystkie testy jednostkowe przy użyciu programu **valgrind**, co nie wykazało żadnych wycieków pamięci.

Opis aplikacji

Opcje

Uruchomienie aplikacji

Po uruchomieniu aplikacje możliwe jest skorzystanie z następujących opcji:

- ➢ Rozpoczęcie nowej gry
  W celu rozpoczęcia nowej gry należy wprowadzić polecenie:
  1. Rozgrywka komputer vs komputer[1]: *start-new;*
  2. Rozgrywka gracz vs komputer: *start-new -player1-name: NazwaGracza;*
  3. Rozgrywka komputer vs gracz: *start-new -player2-name: NazwaGracza;*
  4. Rozgrywka gracz vs gracz: *start-new -player1-name: NazwaGracza1 -player2-name: NazwaGracza2;*

  Po wprowadzeniu jednego z wyżej wymienionych poleceń następuje przejście do **trybu gry.**
- ➢ Załadowanie gry z pliku
  W celu załadowania gry należy wprowadzić następujące polecenie:
  load-from-file -path SCIEZKA/DO/PLIKU/Z/ZAPISEM/ROZGRYWKI;
  Dodatkowo można podać parametry *-player1-name,-player-2-name*, które mają analogiczne działania jak w przypadku rozpoczynania rozgrywki, domyślnie gra jest załadowana dla trybu komputer vs komputer.
  Przygotowane zostały dwa pliki *precheckmate.txt* oraz plik *pat.txt* w katalogu *save_games*, aby wczytać je w trybie gracz vs komputer można użyć następujących poleceń:
  load-from-file -path ./../save_games/pat.txt -player1-name Gracz;
  load-from-file -path ./../save_games/precheckmate.txt -player1-name Gracz;

- ➢ Zakończenie działania aplikacji
  W celu zakończenia działania aplikacji należy użyć polecenia: *exit;*

Tryb gry

Po przejściu do trybu gry możliwe jest skorzystanie z następujących opcji:

- ➢ Wyświetlenie szachownicy ukazującej aktualny stan gry z pomocą polecenia: *display-board;*
- ➢ Zapisania rozgrywki do pliku (wskazanego w poleceniu) z pomocą polecenia: *save-game -path SCIEZKA/DO/PLIKU/GDZIE/ZAPISAC/PLIK;*
- ➢ Wywołanie następnego ruchu z pomocą polecenia *next-move;*
  W przypadku gracza: Komputer spowoduje to wywołanie ruchu wybranego przez zaimplementowany silnik szachowy.

[1] Przyjęto konwencję, że pierwszy podawany graczy rozpoczyna figurami białymi.

W przypadku gracza: Gracz wyświetlona zostaje plansza oraz zapytanie o ruch. W celu wykonania ruchu należy użyć polecenia: *move -from LiteraCyfra -to LiteraCyfra;*
Gdzie para (litera, cyfra) wyznacza współrzędne na szachownicy. Litery od a-h oznaczają kolumny, cyfry od 1-8 oznaczają numer rzędu

- ➢ Możliwość zrezygnowania z rozgrywki z pomocą polecenia: *resign-game;*
- ➢ Możliwość przejścia do trybu przeglądania z pomocą polecenia: *enter-preview;*

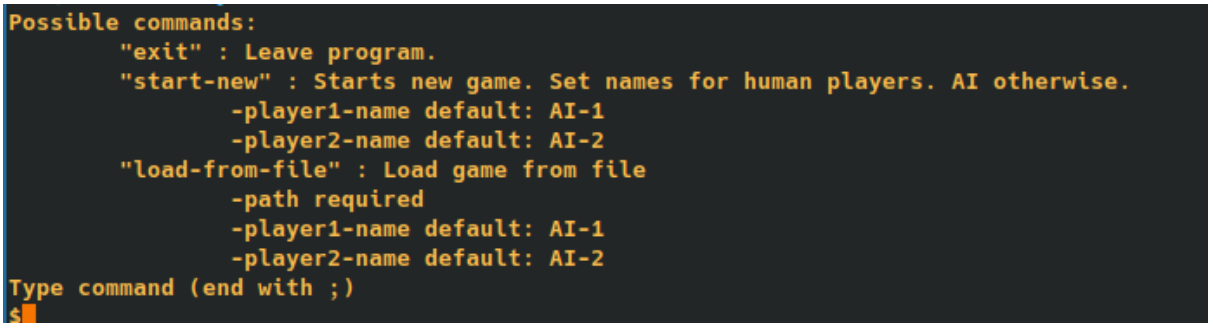
Tryb preview

Podczas rozgrywki możliwe jest wejście w oddzielny tryb *preview,* który zawiera następujące opcje podglądu rozgrywanej partii:

- ➢ Przewijanie o zadaną wartość zarówno w tył jak i przód wszystkich ruchów w grze za pomocą polecenia: *set-steps -steps LiczbaKroków*
- ➢ Wyświetlenie szachownicy ukazującej stan gry dla konkretnego ustawionego kroku z pomocą polecenia: *display-board*
- ➢ Opuszczenie trybu *preview* i kontynuowanie gry od konkretnego ustawionego momentu, przy pomocy analogicznego polecenia jak przy starcie nowej gry, pozwalającego dowolnie przypisać graczy.
- ➢ Zakończenie działania trybu *preview*
  W celu zakończenia działania aplikacji należy użyć polecenia: *exit-preview;*
- ➢ Zakończenie działania aplikacji
  W celu zakończenia działania aplikacji należy użyć polecenia: *exit;*

Zrzuty z działającej aplikacji

Uruchomienie aplikacji



Tryb gry

```
Possible commands:
        "exit" : Leave program.
        "display-board" : Displays current state of board.
        "save-game" : It saves your game in given location
                -path required
        "next-move" : Invokes next move.
        "resign" : I want to resign
        "enter-preview" : Preview other state of game.
Type command (end with ;)
```

Wybór ruchu

```
Running command next-move
Turn of player named: Gracz1
  A  B  C  D  E  F  G  H
8 R  N  B  Q  K  B  N  R 8
7 P  P  P  P  P  P  P  P 7
6                       6
5                       5
4                       4
3                       3
2 P  P  P  P  P  P  P  P 2
1 R  N  B  Q  K  B  N  R 1
  A  B  C  D  E  F  G  H
Possible commands:
        "move" : Give move coords. (e.g. move -from a2 -to a3)
                -from required
                -to required
Type command (end with ;)
$
```

Tryb preview

```
Running command enter-preview
Possible commands:
        "exit" : Leave program.
        "exit-preview" : Leave preview mode
        "display-board" : Displays state of board for selected step
        "set-steps" : Set amount of steps forward (> 0) or backward (< 0)
                -steps required
        "continue-game" : Continue current game
                -player1-name default: AI-1
                -player2-name default: AI-2
Type command (end with ;)
```

Dalszy rozwój oprogramowania

Projekt może być dalej rozwijany w wielu kierunkach kilka z nich przedstawiono poniżej.

Sztuczna Inteligencja

W pierwszej wersji projektu nie została zastosowana żadna metoda Sztucznej Inteligencji, ruchy wykonywane przez gracza Komputer są losowymi ruchami, w takim przypadku rozgrywka staje się prosta. Dlatego w przyszłości należałoby rozwinąć moduł oprogramowania związany z silnikiem szachowym, np. poprzez wybór ruchu zgodny z algorytmem MIN-MAX.

System Logowania

Pierwsza wersja projektu nie przewiduje możliwości równoległego istnienia wielu różnych kont, do których logowanie odbywałoby się przy użyciu loginu i hasła, każde konto mogłoby posiadać swoje zapisane stany gry czy statystyki dotyczące rozegranych partii. Kolejnym krokiem mogłoby być umożliwienie połączenia przez sieć i gry z innym użytkownikiem.

Wyświetlanie, interakcja z użytkownikiem

W pierwszej wersji projektu wyświetlanie zostało zaimplementowane konsolowo. W kolejnych iteracjach programu można by wprowadzić wyświetlanie okienkowe, np. przy użyciu bibliotek **Qt**. Ponadto do gry można by jeszcze zaimplementować zegar, odpowiednio dla obydwu graczy, odliczający czas jaki pozostał do końca partii.