Warcaby - dokumentacja projektu

Jakub Chomiczewski

Wstęp

Celem projektu było napisanie programu pozwalającego na rozgrywkę dwóch ludzkich graczy lub ludzkiego gracza przeciwko graczowi komputerowemu w warcaby. Rozgrywka może być przeprowadzana jak i w terminalu jak i w trybie okienkowym przy użyciu biblioteki SDL. Zasady meczu będę zgodne z zasadami zamieszczonymi na stan 30.01.2022 roku na stronie FMJD (World Draughts Federation).

Instalacja oraz wymagania

Program wymaga od użytkownika zainstalowania następujących bibliotek:

- SDL 2.0
- SDL_image
- FreeType
- SDL_TTF

Następnie, należy pobrać plik Checkers_project-main.zip lub sklonować repozytorium ze strony https://github.com/MinionJakub/Checkers_project. Po pobraniu, rozpakowaniu (plik .zip) oraz wejściu do folderu z plikami przez terminal należy wpisać *chmod 755 kompilacja Uruchom_warcaby* i uruchomić plik kompilacja, a następnie Uruchom_warcaby. Jeśli będą jakieś błędy to należy wpisać następujące komendy:

- gcc -o checkers_project draw_in_SDL.c checkers_rules_and_logic.c bot.c main.c draw_in_command_console.c -lSDL2 -lSDL2_image -lSDL2_ttf -lm
- ./checkers_project

Po pierwszej kompilacji nie trzeba więcej używać ani pliku kompilacja, ani komendy gcc ani chmod.

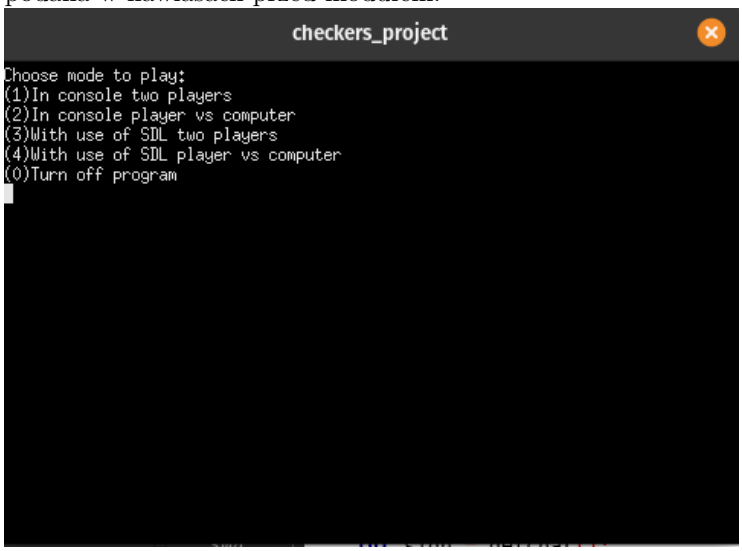
Uwagi: Program był testowany oraz zalecane jest jego uruchamianie na Linuxie (dystrybucja Ubuntu). Warto również sprawdzić, czy terminal obsługuje

UTF-8. Program należy zawsze uruchamiać poprzez wpierw wejście przez terminal do katalogu z grą i uruchomienie pliku Uruchom_warcaby.

Interakcja użytkownika z programem

Wybór trybu

Użytkownik ma prawo do wyboru między graniem w terminal czy w formacie okienkowym. Podjęcie tej decyzji jest możliwe poprzez wybranie odpowiadającej użytkownikowi opcji w konsoli po przez wpisanie odpowiedniej cyfry, która jest podana w nawiasach przed modułem.



Obsługa wersji konsolowej

Gracz do dyspozycji ma następujące komendy:

- surrender - funkcja pozwalająca na poddanie partii
- show moves - funkcja po wpisaniu której są wyświetlane wszystkie legalne ruchy
- A3B4 - przyklad legalnego ruchu po wpisaniu, którego pionek z pola A3 przesunie się na pole B4
- A3C5 - przyklad legalnego bicia po wpisaniu, którego pionek z A3 wykona ruch na C5 i zbije wrogiego pionka na B4

Uwagi: Po zakończeniu partii w terminalu, wyłączany jest również sam program. Program poprawnie wczyta komendę nawet jeśli użytkownik będzie używał zmiennej wielkości liter czy znaki będą oddzielone spacjami.

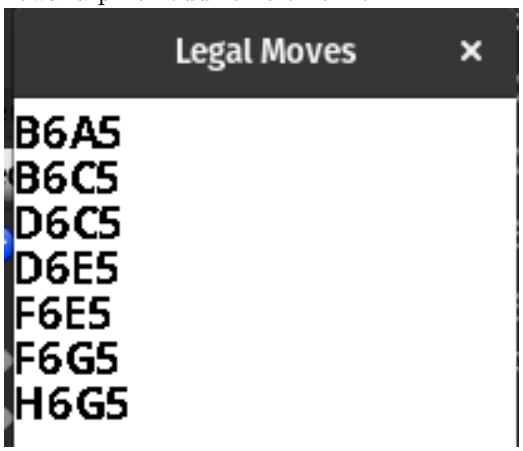
Obsługa wersji okienkowej

Użytkownik ma do dyspozycji następujące komendy:

- surrender - funkcja pozwalająca na poddanie partii
- A3B4 - przyklad legalnego ruchu po wpisaniu, którego pionek z pola A3 przesunie się na pole B4
- A3C5 - przyklad legalnego bicia po wpisaniu, którego pionek z A3 wykona ruch na C5 i zbije wrogiego pionka na B4

jeśli jest wyświetlone okienko:

Zatwierdza się ruch lub komende przez naciśnięcie przycisku ESC lub Enter na klawiaturze numerycznej.
Nie występuje w tym module funkcja showmoves ze względu, że jest ona realizowana przez oddzielne okienko:



Uwagi: W przeciwieństwie do modułu w terminalu zakończenie rozgrywki nie jest równoważne z zakończeniem programu. Program obsługuje jedynie komendy wpisane do okienka Input. Program poprawnie wczyta komendę nawet jeśli użytkownik będzie używał zmiennej wielkości liter czy znaki będą oddzielone spacjami.

Podział programu

Program składa się z następujących modułów:

- checkers_rules_and_logic - elementy gry oraz zasady
- bot - działanie gracza komputerowego

- draw_in_command_console - rysowanie gry oraz jej przeprowadzanie w terminalu
- draw_in_SDL - rysowanie gry oraz jej przeprowadzanie w SDL
- main - wybór trybu rozgrywki

Powyżej wypisane moduły zostaną opisane w poszczególnych podrozdziałach.

Checkers rules and logic

- **int **initiate()** Funkcja ta tworzy planszę 8x8 do gry w warcaby i zwraca wskaźnik do niej.
- **void legal_action(...)** Funkcja ta zapisuje w odpowiednich tablicach (parametry int legal_attack[ ][4] oraz int legal_move[ ][4]), które są podawane przy jej wywołaniu, legalne ruchy dla podanego gracza (int whose_turn) dla zadanego stanu planszy (int **board) i jej rozmiaru (int size_of_board), wymagane jest również podanie wskaźników na dwie zmienne typu int które będą przechowywać ile ruchów danego typu jest (what_number_legal_moves oraz what_number_legal_attacks).
- **void continue_attack(...)** - odpowiada za sprawdzenie czy jeżeli było bicie to czy nie jest ono wielokrotne, jako parametry potrzebuje planszy, wiersza oraz kolumny figury która dokonała zbicia, tablicy legalnych ataków oraz wskaźnika na zmienną typu int odpowiadającą za ilość legalnych ataków oraz jaki typ ataku to był tzn. czy figura biła jak biała figura czy jak czerwona.
- **void promotion (int **board, int size_of_board)** - odpowiada za promocję pionka(man) w króla(king), w odpowiednich modułach jest wykonywana na końcu pętli.
- **void move_action(int **board, int *coordinates)** - pozwala na wykonanie ruchu zadaną figurą na planszy, drugi parametr to wskaźnik mający 4 elementy typu int pierwsze dwa to kolumna i wiersz figury którą chcemy ruszyć natomiast dwa ostatnie to kolumna oraz wiersz pola docelowego.
- **void attack_action(int **board, int *coordinates)** - umożliwia wykonanie bicia, parametr coordinates jest tak samo zdefiniowany jak przy wcześniej wymienionej funkcji move.
- **int value_of_board(int **board, int size_of_board)** - determinuje wartość punktową aktualnego stanu planszy.
- **void who_is_winner(int *board, int size_of_board, int *end_the_game)** - funkcja ta sprawdza czy któraś ze stron rozgrywki przegrała, jeśli tak to zapisuje w zmiennej end_the_game odpowiedni stan.

- **int input_in_attack(int legal_attack[ ][4], int what_number_legal_attacks, int *input)** - funkcja zwraca wartość boolowską jeśli zadany input (zdefiniowany tak jak coordinates) odpowiada jakiemuś legalnemu atakowi.
- **int input_in_move(int legal_move[ ][4], int what_number_legal_move, int *input)** - działanie podobne do input_in_attack.
- **int *read_move (int legal_attack[][4], int legal_move[][4],int what_number_legal_moves, int what_number_legal_attacks, int *correctness, char *entry)** - służy do określenia czy podane entry jest ruchem, i jeśli jest to czy jest poprawny zgodnie z zasadami, po sprawdzeniu tego zwraca wskaźnik ruch.
- **void dynamic_array_of_chars(char *array, int *size)** - funkcja ta zwiększa dwukrotnie rozmiar tablicy char-ów, która jest podawana jako wskaźnik array.
- **void insert_to_dynamic_array(char *array, int *last_element, int *size, char what_to_insert)** - wstawia znak do tablicy char-ów i jeśli nie ma na to miejsca to wywołuje funkcje do zwiększenia zaalokowanej pamięci.
- **int compare_arrays (char *array, const char *with_what_to_compare, int length)** - zwraca wartość boolową jeśli dwa podane ciągi znaków są podobne do określonej długości.
- **char *read_entry(int *operation)** - wczytuje wejście użytkownika i zwraca wartość entry która jest odpowiednio po przetworzeniu, zapisuje również w zmiennej operation odpowiedni stan odpowiadający wczytanej komendzie.
- **char *read_entry_from_text(int *operation,char *text,int last_element)** - jest podobna do funkcji read_entry, jednakże nie wczytuje tekstu tylko go przetwarza i zwraca odpowiednie elementy.

Bot

- **int evaluation_of_move(int **board, int size_of_board, int whose_turn, int enemy_value, int my_value, int deep)** - funkcja ta zwraca relatywną wartość punktową (może zależeć od głębokości) przy założeniu, że gracze perfekcyjnie grają do określonej głębokości (inaczej mówiąc do określonej ilość ruchów w przyszłość), algorytmem odpowiedzialnym głównie za działanie jej jest algorytm minimax.
- **int *bot_make_decision(int **board,int size_of_board,int whose_turn,int enemy_value,int my_value,int deep)** - funkcja ta zwraca najlepszy relatywnie ruch na bazie ewaluacji ich przy pomocy wcześniej wymienionej funkcji, jeśli jest kilka równi dobrych ruchów to zwraca losowo jeden z nich.

Uwagi: Nie jest zalecane zwiększanie głębokości powyżej 10. Parametry int enemy_value czy int my_value są potrzebne do określenia czy gracz jest na górze czy na dole, dla przykładu jeśli bot jest na górze to jako my_value podajesz 1 natomiast enemy_value = -1.

Draw in command console

- **void draw_map_of_game(int **map_of_game,int length_of_map, int size_of_line)** - funkcja służy do wypisania stanu rozgrywki do konsoli, rekomendowane jest podawanie tej samej wielkości do length_of_map oraz do size_size_of_line.
- **void show_legal_moves(int legal_move[][4], int legal_attack[][4],int what_number_legal_moves, int what_number_legal_attacks)** - pozwala na wypisanie wszystkich możliwych ruchów po wpisaniu komendy showmoves.
- **void draw_end(int winner)** - odpowiada za wypisanie stanu kończącego grę, w programie winner jest równy zmiennej end_the_game.
- **void PvP_console()** - funkcja jest odpowiedzialna za wyświetlanie gry za pomocą funkcji draw_map_of_game oraz za jej przebieg dla dwóch graczy ludzkich.
- **void PvComputer_console()** - funkcja jest odpowiedzialna za wyświetlanie gry za pomocą funkcji draw_map_of_game oraz za jej przebieg dla jednego gracza ludzkiego i bota. Gracz jest zawsze na dole a bot zawsze na górze.

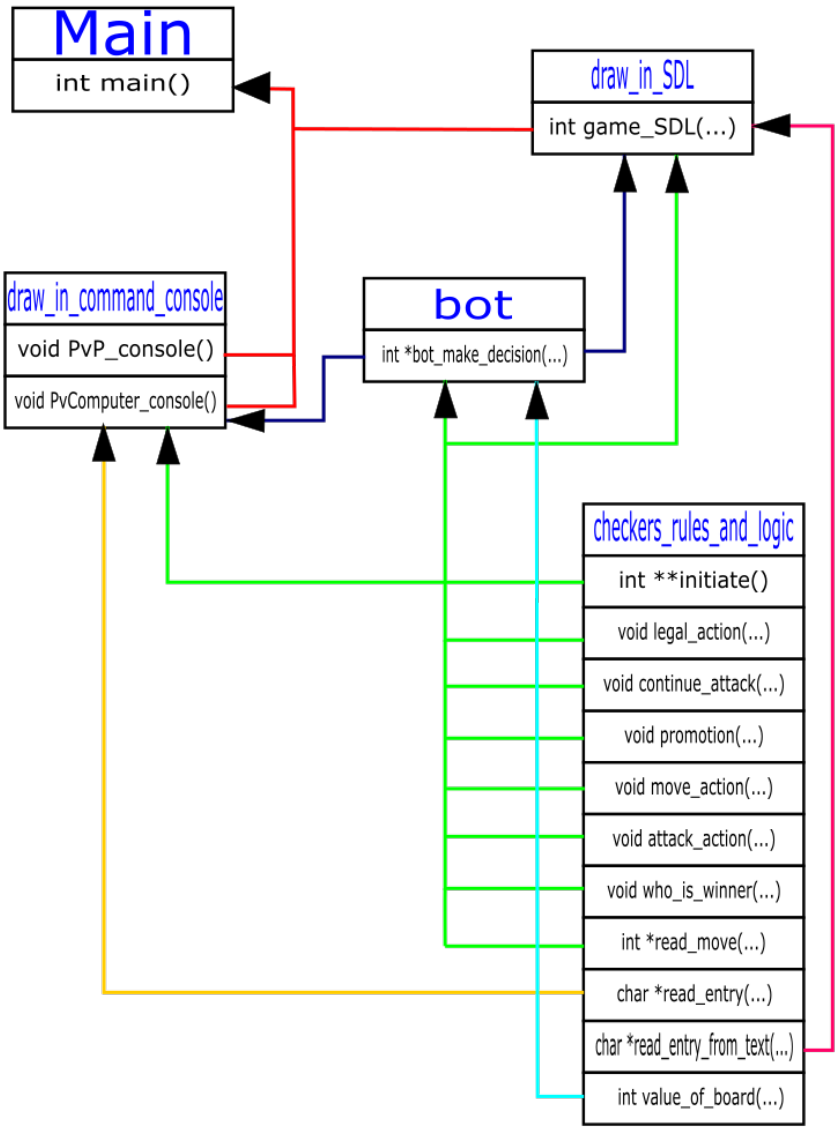
Draw in SDL

int game_SDL(int what_type_of_game), int easter_egg - opdowiada za wyświetlanie wszystkich elementów, trybu okienkowego, jeśli nie będzie wstanie zainicjalizować jakiegokolwiek elementu zwróci odpowiedni błąd oraz zakończy się z nie zerową wartością. Obsługuje grę dwóch ludzkich graczy oraz gracza z botem, odpowiedzialny za to jest parametr what_type_of_game gdzie 0 to gra z botem natomiast 1 to gra między ludzkimi oponentami. Parametr easter_egg jest zmienną która pozwala uruchomić ukrytą w grze niespodziankę przy wyborze trybu.
Uwagi: Kolor biały jest równoważny z graczem, który ma turę pierwszy. Może być lekki lag po wykonaniu ruchu jeśli gra się przeciwko komputerowi jest to spowodowane tym, że komputer od razu wykonuje ruch po graczu.

Main

Składa się z jednej funkcji, której nazwa jest taka sama jak nazwa modułu, odpowiada jedynie za wybór trybu gry.

Mapa powiązań



Dodatkowe pliki

Oprócz modułów do gry jest potrzebny folder z plikami graficznymi (Assets rozszerzenie tych plików to .svg) oraz plik font.ttf, pierwszy służy do przechowywania, liter, cyfr oraz figur. Lista plików z opisem za co odpowiadają:

- Figury
  - Chess_klt45.svg - odpowiada za białą figurę króla bądź królowej
  - Chess_krt45.svg - odpowiada za czerwoną figurę króla bądź królowej
  - Chess_plt45.svg - odpowiada za białą figurę pionka (ang. men)
  - Chess_prt45.svg - odpowiada za czerwoną figurę pionka (ang. men)
- Symbole
  - Litera_(znak).svg - odpowiada za odpowiednią literę gdzie znak to litera w zakresie od A do H
  - Number_(znak).svg - odpowiada za odpowiednią cyfrę gdzie znak to cyfra w zakresie od 1 do 8
- Font - należy w głównym folderze z grą i ma nazwę na font.ttf