Szachy – projekt PROI

Program składa się z 10 projektów, w których osobno kodowane były elementy gry:

- Bierki(figury szachowe)
- Testy jednostkowe
- Szchownica
- Silnik gry
- GUI

Chess

Implementację programu zaczęliśmy od stworzenia figur szachowych. W projekcie Chess stworzony została klasa Figure, po której dziedziczą wszystkie sześć bierek. W polu protected tej klasy znajduje się pozycja figury na szachownicy oraz jej kolor. W nagłówkach samych bierek znajduje się funkcja zwracająca wszystkie pola, na które figura, zgodnie ze swoim sposobem ruchu, można się przesunąć - ***get_possible positions()***. Ponieważ figury pełnią ważną rolę w programie, wszystkie posiadają osobne testy jednostkowe.

Board

Kolejnym krokiem w implementacji było stworzenie szachownicy. Klasa Board, znajdująca się w osobnym projekcie, tworzy arraya 8 na 8, w którym ustawione mogą zostać stworzone wcześniej figury.

Funkcje zaimplementowane tutaj umożliwiają m.in. zmiane pozycji figury oraz zastąpienie figury inną – ***make_move()*** oraz ***replace_figure()***.

Funkcja ***get_free_positions_for_figure*** ogranicza zakres ruchu bierek o pola zajmowane przez inne figury, oraz o pola znajdujące się za nimi (***dzięki get_positions_behind()***) – kluczowe dla wieży, hetmana i gońca.

Set_starting_positions() ustawia figury w pozycji początkowej, z białymi na dole, a czarnymi na górze.

Ważną dla działania programu jest funkcja get_positions_behind zwracająca te pozycje na szachownicy, które w teorii są dostępne dla figury, ale przed nimi stoi już

inna figura. Sprawdzane są wszystkie z ośmiu przypadków (lewo, prawo, góra, dół i ukosy), aby znaleźć te pozycje, które nie są faktycznie dostępne

Zaimplementowane jest tam również bicie piona – funkcja sprawdza czy pola, które figura ta może bić znajdują się w zakresie szachownicy. Wymagana jest osobna funkcja, ponieważ w przeciwieństwie do pozostałych figur pion bije w innym kierunku niż się porusza. Dlatego zapisany jest każdy możliwy przypadek – dla czarnego i białego piona, na bicie po lewo i po prawo w funkcji get_strike_positions_for_pawn().

GameEngine

Następny w kolejności napisany został kluczowy kod odpowiadający za przebieg rozgrywki umieszczony został w GameEngine. Zaimplementowane tam zostały dwie nowe klasy – Player oraz Game. Pierwsza z nich posiada informacje o bierkach, które obydwaj gracze obecnie posiadają oraz może je usuwać (***del_figure()***). W drugiej z nich stworzone zostały funkcje takie jak:

restart_game – usuwa z szachownicy (arraya) wszystkie figury zostawiając nullptr, a następnie ustawia bierki na swoje startowe pozycje

capture_figure – umożliwiająca zbijanie bierek znajdujących się na polu dostępnym dla wybranej figury o przeciwnym kolorze

get_possible_checking_figures – iterująca po wszystkich figurach gracza i sprawdzająca jakie pola są przez nie szachowane, a więc gdzie figura może przesunąć się w następnej turze. Jest to wykorzystywane przy ograniczaniu dozwolonych pól króla

get_checking_figures – funkcja, z której korzysta król. Sprawdza czy na pozycję króla w obecnej chwili może przesunąć się figura przeciwnika – jeśli tak to jest szach.

get_possible_checking_figures – sprawdza na jakie pola może przesunąć się każda z figur gracza, tym samym blokując je przeciwnemu królowi

restrict_king_positions – funkcja, która odbiera królowi z dostępnych pól te pozycje na które ruszyć może się figura przeciwnika, ponieważ nie może on przesunąć się na pole, które automatycznie należałoby do ***get_checking_figures()***. Oprócz tego w przypadku szacha usuwa możliwość ruchu króla w lini szacha.

get_positions_beetwen – funkcja, która zwraca pola pomiędzy królem a szachującą go figurą. Wykorzystywana przy ograniczaniu ruchów króla oraz wskazywaniu pól, na które muszą przesunąć się inne figury. Ponieważ jest tylko 8 lini po których król może być szachowany, każdy przypadek jest zaimplementowany w osobnych if-ach i wywoływany gdy zajdzie

get_allowed_moves – sprawdza sytuację konkretnie w momencie, gdy król jest szachowany. Zwraca pozycje, na które król może się przesunąć (korzysta z wcześniej stworzonej ***get_checking_figures()***, aby nie przesunął się na inne szachowane pole) oraz pola w lini prostej między królem a szachującą go figurą (nie dotyczy skoczka).

get_final_moves_for_figure – sprawdzająca na jakie pozycje figura może się przesunąć – z wyłączeniem pól zajętych oraz, dla króla, pól na których byłby szachowany. Funkcja ta korzysta z wymienionych wcześniej funkcji, co umożliwia sprawdzenie czy król jest szachowany, czy ma wolne pozycje na przesunięcie się, możliwość zbicia figury szachującej lub zablokowanie szacha inną figurą. Tylko te pozycje, które skutkować będą końcem szacha są zwracane jako final_moves

check_win_condition – sprawdza warunek na zwyciestwo sprawdzając czy get_final_moves jest równe zero

check_stalemate_condition – sprawdzająca warunek na pat, czyli sytuację, gdy żadna z figur gracza nie ma ruchów - ***get_final_moves_for_figure.size() == 0***, a jednocześnie król nie jest szachowany

do_castling – wykonująca roszadę (długą i krótką) zgodnie z zasadami szachowymi tzn ***get_first_move()*** jest aktywna – ani król ani wieża którą checmy wykonać roszadę nie byli wcześniej przemieszczani. Ponieważ są tylko dwie różne sytuacje, gdy roszada może być wykonana, obydwa przypadki zostały zapisane osobno

check_promote_pawn – sprawdza warunek promocji piona – czy figura ta znjaduje się, w zależności czy biały czy czarny, w wierszu o indeksie 0 lub 7. W przypadku spełnienia warunku dokonana zostaje promocja

ChessGUIV3

Interfejs gry wykonany został w GUI przy użyciu rozszerzenia qt. Szachownica została stworzona z QpushButtons. Do figur zostały dołączone ikony wyświetlające się na

zajmowanych przez nie polach. Funkcja ***show_possible_moves_for_figure()*** podświetla dozwolone pola dla kliknięte figury.

Make_move() w tym projekcie jest funkcją wykorzystującą wcześniej zaimplementowane już metody. Wyboru figury dokonujemy przez clicked_figure – czyli zmienną bezpośrednio powiązaną z interfejsem, zmieniającą się w zależności od ostatnio kliknięte bierki. Dla króla oraz piona sprawdzane tam są warunki na kolejno roszadę i promocję. Po wywołaniu funkcji z klasy game – ***make_move()*** następuje zmiana tury poprzez osobną funkcję i sprawdzane jest czy któryś z warunków na zakończenie gry - mat (***check_win_condition()***) lub pat (***check_stalemate_condtition()***) jest spełniany. Jeśli tak, otwiera się nowe okno stworzone w GUI, gdzie umieszczone są przyciski umożliwiające opuszczenie programu lub prześledzenie partii według zapisu (***show_match_history()***).

Jednym z wymagań programu w zadaniu był zapis przebiegu rozgrywki. Jego implementacja została wykonana w pliku Game.cpp w funkcji get_record. Zamienia ona pozycje pól z programu (0-7) w szachowe (A_H i 1-8). Następnie w wykonanym w GUI polu tekstowym wyświetla się kolor i typ figury oraz pole, na które się przemieściła. Zapis ruchu zaimeplementowany jest w funkcji ***make_move()***. Funkcja ***save_record()***, podpięta pod przycisk w intefejsie umożliwia zapis partii jako plik tekstowy na komputerze.

Ostatnim elementem, który został zaimplementowany była rozgrywka z komputerem. W tym celu stworzona została funkcja ***get_random_move()***. Z listy figur przy uzyciu funkcji rand() pobiera losową bierkę, po czym ponownie z listy ruchów dozwolonych dla tej figury losowo wybiera nową pozycję.

Projekt wykonali:

- Adam Wróblewski

- Krystian Grela

- Aleksander Woś