Politechnika Śląska
Wydział Automatyki, Elektroniki i Informatyki

Programowanie Komputerów 3

Szachy

Autor	Dominik Uszok
Prowadzący	dr inż. Piotr Pecka
Rok akademicki	2020/2021
Kierunek	Informatyka
Rodzaj studiów	SSI
Semestr	3
Termin laboratorium	czwartek, 11:15 – 12:45 piątek, 8:15 – 9:45
Sekcja	21
Termin oddania sprawozdania	2020-11-10

---

1 Treść zadania

Celem projektu jest napisanie programu służącego do gry w szachy z dwoma ludzkimi graczami (bez komputera).

Każdy z graczy posiada 16 figur - 8 pionków, 2 gońce, 2 skoczki, 2 wieże, hetmana i króla. Każda z figur może poruszać się tylko w charakterystyczny dla siebie sposób.

Rozgrywka odbywa się na planszy o rozmiarze 8 x 8 pokolorowanej na przemian. Gracz pierwszy posiada figury białe i one rozpoczynają rozgrywkę, gracz drugi posiada figury czarne. Celem gry jest zbicie króla gracza przeciwnego (szach mat).

Interfejs będzie realizowany z użyciem biblioteki SFML. Plansza jest rozmiaru 800x800px z jednym polem o rozmiarze 100x100px.

2 Analiza zadania

Zagadnienie przedstawia problem stworzenia grafiki szachownicy i jej figur z użyciem biblioteki SFML. Reprezentacji w sposób obiektowy szachownicy z jej figurami, przy użyciu klas, dziedziczenia i polimorfizmu. Finalnie na połączeniu tych dwóch elementów - grafiki i obiektowego back-endu.

2.1 Struktury danych

W programie wykorzystano strukturę tablicy dwuwymiarowej, rozmiaru 8 na 8 do przedstawienia szachownicy. Jest to tablica typu wskaźnik na klasę Piece (Piece*). Tablica ta jest zawarta w klasie Board.

2.2 Algorytmy

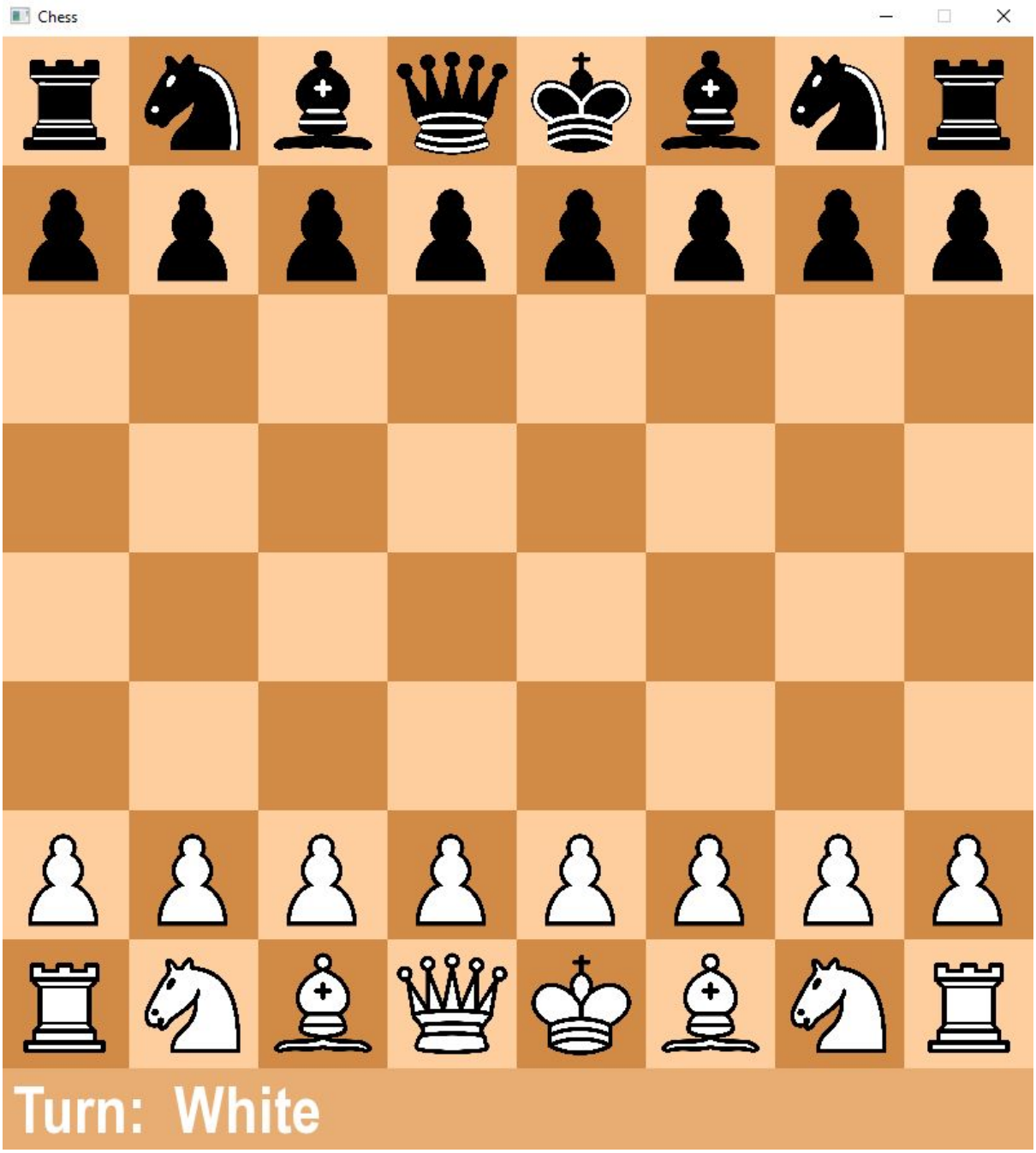
Program do wykonania wielu funkcji iteruje 8 razy przez pętle for, w której jest zawarta druga pętla for, przez którą również iteruje 8 razy. W ten sposób program przeszukuje całą szachownicę w celu znalezienia na danym polu figury lub jej braku. W taki sposób przedstawiane są grafiki figur szachowych, graficzna reprezentacja możliwości wykonania ruchu (w postaci zielonej kropki) oraz wyzerowanie tablicy po stworzeniu obiektu Board jak i usunięciu obiektów po zakończeniu pracy programu.

Algorytm przemieszczenia figury jest następujący: tymczasowe zapisanie aktualnych koordynatów figury do przemieszczenia, jeżeli na koordynatach do przemieszczenia znajduje się inna figura - usunięcie jej, zamiana koordynatów figury przemieszczanej, inkrementacja zmiennej oznaczającej ilość wykonanych ruchów przez figurę, umieszczenie figury w odpowiednim miejscu tablicy reprezentującej szachownicę, usunięciu z tablicy poprzedniego położenia figury poruszanej, zamiana tury na drugiego gracza, wyłączenie figury aktywnej.

Algorytmy ograniczające ruchy poszczególnych rodzajów figur są na tyle różnorodne że nie dają się opisać w sposób ogólny.

3 Specyfikacja zewnętrzna

Po uruchomieniu programu ukazuje się okno z szachownicą pod którą znajdują się informacje dla użytkownika:

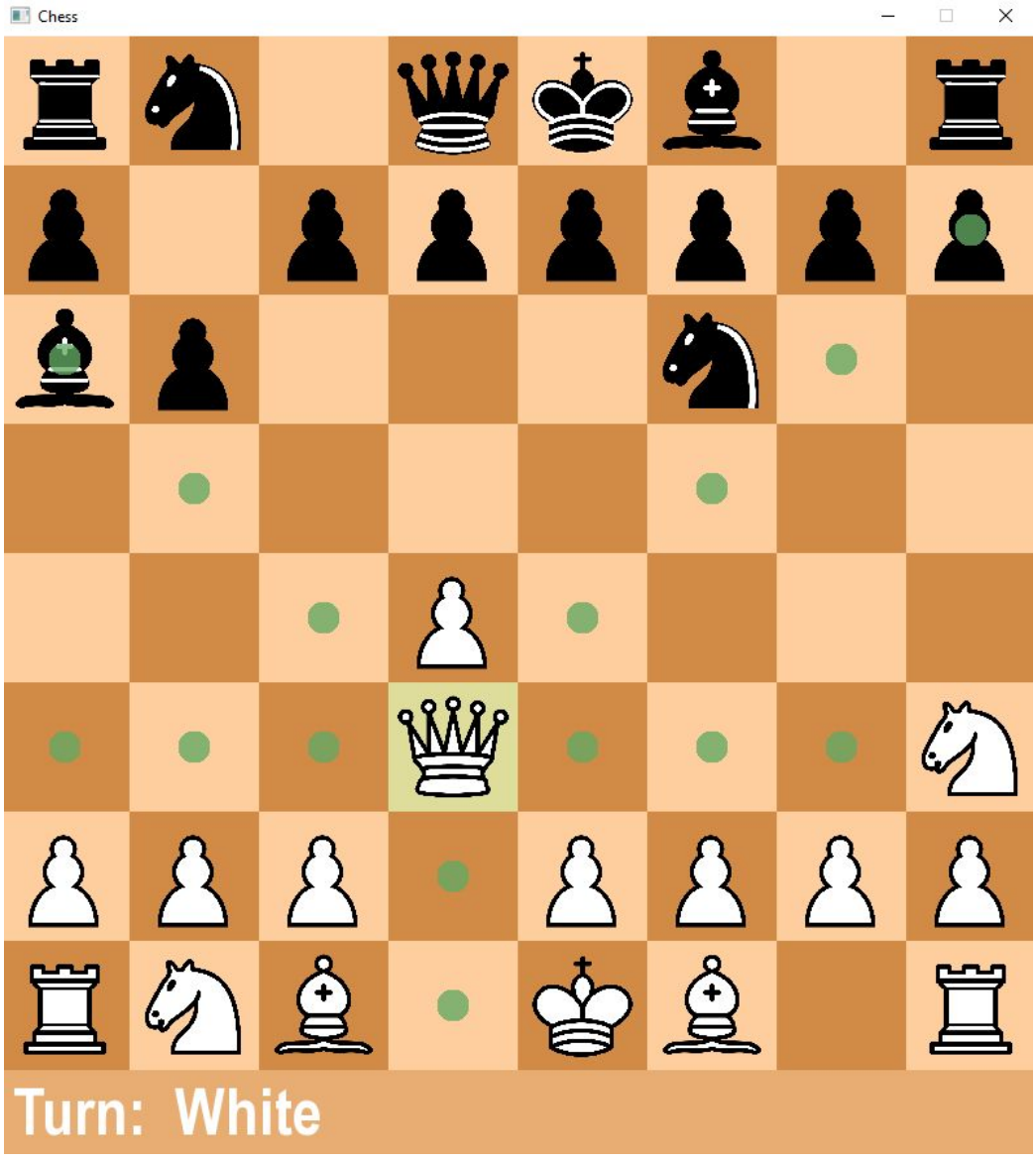


Zawarte są tam informacje o:

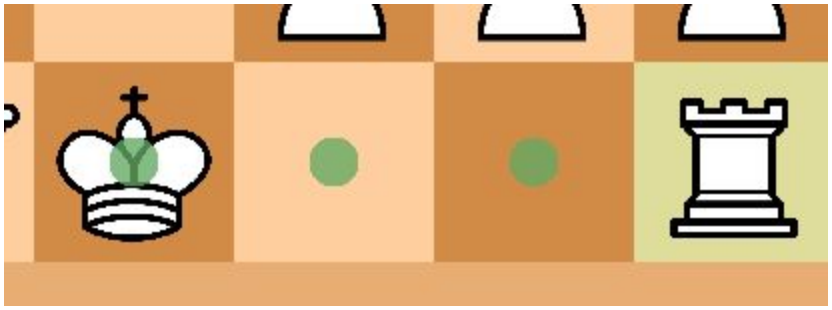
- Turze

- Zwycięstwie i możliwości rozpoczęcia nowej gry klawiszem “Enter”

Pole, na którym znajduje się aktualnie poruszana przez gracza figura, jest podświetlone na zielono. Zielonymi kropkami są oznaczone wszystkie możliwe ruchy które aktualnie aktywna figura może wykonać:



Możliwość wykonania roszady jest przedstawiona w następujący sposób:



4 Specyfikacja wewnętrzna

Program został zrealizowany zgodnie z paradygmatem strukturalnym, w sposób obiektowy. W programie rozdzielono interfejs (komunikację z użytkownikiem) od logiki aplikacji - poruszanie się i zbijanie figur szachowych, rozpoczynanie nowej gry i wyznaczanie zwycięzcy.

4.1 Ogólna struktura programu

Funkcja główna zaczyna się od stworzenia elementów graficznych: załadowania tekstur z plików, utworzenia zmiennych biblioteki SFML, przyporządkowanie tym zmiennym odpowiednich tekstur i rozmiarów oraz położenie w oknie programu.

Następnym etapem jest utworzenie obiektów: szachownicy i jej figur oraz przyporządkowanie tym obiektom odpowiednich im wartości: koordynatów, koloru, tekstur.

Ostatecznie w pętli while realizowana jest graficzna reprezentacja programu, który można zamknąć klikając znak X w prawym górnym rogu okna.

Jeżeli kliknięty zostanie lewy przycisk myszy, funkcja klasy Board *isMouseClickInsideWindow* sprawdzi czy kliknięcie odbyło się w oknie programu. Jeżeli tak, koordynaty kliknięcia w pikselach przy użyciu funkcji *setClickedBoardSquare* zostaną przekonwertowane na koordynaty na szachownicy i zapisane w zmiennych klasy Board *xPressed* i *yPressed*. Jeżeli kolor klikniętej figury odpowiada aktualnej turze gracza funkcja *setClickedPieceAsActive* ustawi figurę jako aktywną i gotową do ruchu. Jeżeli istnieje aktywna figura następne kliknięcie będzie ruchem. Jego możliwość zostanie sprawdzona przez *isMoveCorrect* - funkcję polimorficzną podklas dziedziczących z klasy Piece, a następnie ruch zostanie zrealizowany przez *moveActivePiece* po czym nastąpi zmiana tury gracza.

Jeżeli zostanie wciśnięty klawisz “Enter” rozpocznie się nowa gra funkcją *startNewGame*.

Obiekt szachownica zawiera zmienną *winType*, której wartość na początku gry wynosi 0. Funkcja *removePiece* po zbiciu króla czarnego ustanawia tę wartość na 1 a po zbiciu króla białego na 2. W zależności od tej wartości zostanie wyświetlona informacja o zwycięstwie i grę można kontynuować tylko gdy jej wartość wynosi 0 (po wciśnięciu klawisza “Enter”).

4.2 Szczegółowy opis klas i metod

Szczegółowy opis klas i metod zawarty w załączniku.

5 Testowanie

Program został przetestowany używając różnych kombinacji ustawień figur na szachownicy. Próbując przemieścić się w miejsca, w które dana figura nie powinna mieć dostępu, próbując ruszyć figurę gracza, który nie ma aktualnie tury oraz próbując zbić figury własnego koloru. Warunki zajścia roszady zostały przetestowane dla obu graczy i dla obu wież. Kliknięcia myszy i konwersja koordynatów z pikseli na koordynaty szachowe jest wyświetlana w oknie konsoli co umożliwiło łatwe sprawdzenie poprawności.

Program został sprawdzony pod kątem wycieków pamięci z użyciem biblioteki CRT.

6 Wnioski

Program “Szachy” jest programem którego zadaniem jest odwzorowanie tradycyjnej gry z użyciem biblioteki graficznej SFML i programowania obiektowego. Wykorzystuje nieskomplikowaną strukturę tablicy dwuwymiarowej, która bardzo dobrze nadaje się do przedstawienia szachownicy. Wykorzystuje również dwie klasy Board i Piece oraz 6 podklas klasy Piece odpowiadającej każdej z figur. Pomimo że szachy są grą tradycyjną, której zasady nie ulegają znacznym zmianom, to użycie dziedziczenia i polimorficznej funkcji do ograniczania ruchów umożliwia łatwą możliwość wprowadzenia nowych figur szachowych dla graczy chcących doświadczenia własnej wersji szachów.

Literatura

- https://www.sfml-dev.org/learn.php
- https://en.wikipedia.org/wiki/Chess

Dodatek
Szczegółowy opis klas i metod

Chess

Wygenerowano przez Doxygen 1.8.20

1 Indeks hierarchiczny

1.1 Hierarchia klas

Ta lista dziedziczenia posortowana jest z grubsza, choć nie całkowicie, alfabetycznie:

2 Indeks klas

2.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

3 Indeks plików

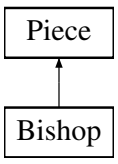
3.1 Lista plików

Tutaj znajduje się lista wszystkich udokumentowanych plików z ich krótkimi opisami:

4 Dokumentacja klas

4.1 Dokumentacja klasy Bishop

Diagram dziedziczenia dla Bishop



Metody publiczne

- Bishop (Board *board_, int x_, int y_, Color pieceColor_, sf::RectangleShape pieceShape_)

  Tworzy nowy objekt typu Bishop dziedziczący z klasy Piece.
- bool isMoveCorrect (sf::Vector2i coordinatesToMoveTo)

  Metoda wirtualna - każda z sześciu podklas klasy Piece zawiera swój polimorficzny odpowiednik tej metody. Określa czy ruch wykonany przez figurę jest możliwy.

Dodatkowe Dziedziczone Składowe

4.1.1 Dokumentacja konstruktora i destruktora

4.1.1.1 Bishop()

```
Bishop::Bishop (
        Board * board_,
        int x_,
        int y_,
        Color pieceColor_,
        sf::RectangleShape pieceShape_ )  [inline]
```

Tworzy nowy objekt typu Bishop dziedziczący z klasy Piece.

Parametry

*board_*	Wskaźnik na szachownicę. Każda z figur wie na jakiej szachownicy się znajduje.
*x_*	Koordynat X szachownicy na którym znajduje się figura - szerokość.
*y_*	Koordynat Y szachownicy na którym znajduje się figura - wysokość.
*pieceColor_*	Kolor figury.
*pieceShape_*	Grafika figury szachowej.

4.1.2 Dokumentacja funkcji składowych

4.1.2.1 isMoveCorrect()

```
bool Bishop::isMoveCorrect (
        sf::Vector2i clickedCoordinates )  [virtual]
```

Metoda wirtualna - każda z sześciu podklas klasy Piece zawiera swój polimorficzny odpowiednik tej metody. Określa czy ruch wykonany przez figurę jest możliwy.

Parametry

clickedCoordinates	Koordynaty do których poprawność przemieszczenia figury ma zostać sprawdzona.

Zwraca

true - ruch prawidłowy.

false - ruch nieprawidłowy.

Implementuje Piece.

4.2 Dokumentacja klasy Board

Metody publiczne

- Board (float boardSizeToSet, sf::RectangleShape pieceActiveVisualsToSet)

  Tworzy nowy objekt typu Board.
- ~Board ()

  Niszczy objekt typu Board. Iteruje przez wszystkie pola szachownicy i usuwa wszystkie figury.
- bool isMouseClickInsideWindow (sf::Vector2i coordInPixels)

  Sprawdza czy koordynaty kliknięcia myszą znajdują się w obrębie szachownicy. Kliknięcie interfejsu pod szachownicą zwraca false.
- void setClickedBoardSquare (sf::Vector2i coordInPixels)

  Ustawia wskaźnik klikniętego pola w zmiennej clickedBoardSquare - może wynosić NULL.
- void setClickedPieceAsActive (Piece *clickedPiece)

  Aktywuje figurę - przygotowuje ją do ruchu, jeżeli nie ma aktualnie aktywnej innej figury. Jeżeli kliknięta figura jest już aktywna następuje jej deaktywacja.
- sf::RectangleShape getActivePieceVisuals ()

  Zwraca grafike pola na którym znajduje się aktywowana figura.
- Piece * getActivePiece ()

  Zwraca wskaźnik na aktywną figurę.
- Color getCurrentTurn ()

  Zwraca zmienną opisującą aktualną turę gracza.
- Piece * getClickedBoardSquare ()

  Zwraca wskaźnik klikniętego pola szachownicy - może wynosić NULL.
- Piece * getPiecePointer (int x, int y)

  Zwraca wskaźnik na figurę szachową która znajduje się na koordynatach przekazanych w parametrach metody.
- sf::Vector2i getClickedCoordinates ()

  Zwraca kliknięte koordynaty w postaci wektoru.
- void startNewGame (Piece **p)

  Resetuje szachownicę do stanu początkowego: turę ma gracz biały, wszystkie figury na swoich początkowych koordynatach, brak gracza który wygrał, brak aktywnej figury.
- void setKings (Piece *whiteKing, Piece *blackKing)

  Przekazuje objektowi typu Board informację o tym które ze wskaźników na szachownicy są królami.
- int getWinType ()

  Zwraca typ wygranej.

Przyjaciele

- class Piece

  Zaprzyjaźnienie umożliwia realizacje wielu powiązanych metod pomiędzy tymi klasami.

4.2.1 Dokumentacja konstruktora i destruktora

4.2.1.1 Board() `Board::Board ( float boardSizeToSet, sf::RectangleShape pieceActiveVisualsToSet ) [inline]`

Tworzy nowy objekt typu Board.

Parametry

boardSizeToSet	Wartość w pikselach określająca wielkość szachownicy.
pieceActiveVisualsToSet	Grafika pola na którym znajduje się aktywowana figura.

4.2.2 Dokumentacja funkcji składowych

4.2.2.1 getActivePiece() `Piece * Board::getActivePiece ( )`

Zwraca wskaźnik na aktywną figurę.

Zwraca

Piece∗ wskaźnik na aktywną figurę.

4.2.2.2 getActivePieceVisuals() `sf::RectangleShape Board::getActivePieceVisuals ( )`

Zwraca grafike pola na którym znajduje się aktywowana figura.

Zwraca

sf::RectangleShape Grafika pola na którym znajduje się aktywowana figura.

4.2.2.3 getClickedBoardSquare() `Piece * Board::getClickedBoardSquare ( )`

Zwraca wskaźnik klikniętego pola szachownicy - może wynosić NULL.

Zwraca

Piece∗ Wskaźnik klikniętego pola szachownicy - może wynosić NULL.

4.2.2.4 getClickedCoordinates() `sf::Vector2i Board::getClickedCoordinates ( )`

Zwraca kliknięte koordynaty w postaci wektoru.

Zwraca

sf::Vector2i Kliknięte koordynaty w postaci wektoru.

4.2.2.5 getCurrentTurn() `Color Board::getCurrentTurn ( )`

Zwraca zmienną opisującą aktualną turę gracza.

Zwraca

Color Zmienna opisująca aktualną turę gracza.

4.2.2.6 getPiecePointer() `Piece * Board::getPiecePointer ( int x, int y )`

Zwraca wskaźnik na figurę szachową która znajduje się na koordynatach przekazanych w parametrach metody.

Parametry

x	Koordynat X szachownicy - szerokość.
y	Koordynat Y szachownicy - wysokość.

Zwraca

Piece∗ Wskaźnik figury szachowej na podanych koordynatach.

4.2.2.7 getWinType() `int Board::getWinType ( )`

Zwraca typ wygranej.

Zwraca

int 0 - aktualnie nikt nie wygrał, 1 - wygrał gracz biały, 2 - wygrał gracz czarny.

4.2.2.8 isMouseClickInsideWindow() `bool Board::isMouseClickInsideWindow ( sf::Vector2i coordInPixels )`

Sprawdza czy koordynaty kliknięcia myszą znajdują się w obrębie szachownicy. Kliknięcie interfejsu pod szachownicą zwraca false.

Parametry

coordInPixels	Koordynaty w pikselach kliknięte myszą.

Zwraca

true - gdy koordynaty są od 0 do boardSize

false - gdy koordynaty wychodzą poza szachownicę

4.2.2.9 setClickedBoardSquare() `void Board::setClickedBoardSquare ( sf::Vector2i coordInPixels )`

Ustawia wskaźnik kliknięty pola w zmiennej clickedBoardSquare - może wynosić NULL.

Parametry

coordInPixels	Koordynaty w pikselach kliknięte myszą.

4.2.2.10 setClickedPieceAsActive() `void Board::setClickedPieceAsActive ( Piece * clickedPiece )`

Aktywuje figurę - przygotowuje ją do ruchu, jeżeli nie ma aktualnie aktywnej innej figury. Jeżeli kliknięta figura jest już aktywna następuje jej deaktywacja.

Parametry

clickedPiece	Wskaźnik figury która ma zostać aktywowana.

4.2.2.11 setKings() `void Board::setKings ( Piece * whiteKing, Piece * blackKing )`

Przekazuje objektowi typu Board informację o tym które ze wskaźników na szachownicy są królami.

Parametry

whiteKing	Wskaźnik na króla białego.
blackKing	Wskaźnik na króla czarnego.

4.2.2.12 startNewGame() `void Board::startNewGame ( Piece ** p )`

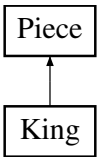
Resetuje szachownicę do stanu początkowego: turę ma gracz biały, wszystkie figury na swoich początkowych koordynatach, brak gracza który wygrał, brak aktywnej figury.

Parametry

p	Wskaźnik na tablicę wszystkich figur. Ma 32 elementy.

4.3 Dokumentacja klasy King

Diagram dziedziczenia dla King



Metody publiczne

- King (Board *board_, int x_, int y_, Color pieceColor_, sf::RectangleShape pieceShape_)

  Tworzy nowy objekt typu King dziedziczący z klasy Piece.
- bool isMoveCorrect (sf::Vector2i coordinatesToMoveTo)

  Metoda wirtualna - każda z sześciu podklas klasy Piece zawiera swój polimorficzny odpowiednik tej metody. Określa czy ruch wykonany przez figurę jest możliwy.

Dodatkowe Dziedziczone Składowe

4.3.1 Dokumentacja konstruktora i destruktora

4.3.1.1 King() `King::King ( Board * board_, int x_, int y_, Color pieceColor_, sf::RectangleShape pieceShape_ ) [inline]`

Tworzy nowy objekt typu King dziedziczący z klasy Piece.

Parametry

*board_*	Wskaźnik na szachownicę. Każda z figur wie na jakiej szachownicy się znajduje.
*x_*	Koordynat X szachownicy na którym znajduje się figura - szerokość.
*y_*	Koordynat Y szachownicy na którym znajduje się figura - wysokość.
*pieceColor_*	Kolor figury.
*piece Shape_*	Grafika figury szachowej.

4.3.2 Dokumentacja funkcji składowych

4.3.2.1 isMoveCorrect() `bool King::isMoveCorrect ( sf::Vector2i clickedCoordinates ) [virtual]`

Metoda wirtualna - każda z sześciu podklas klasy Piece zawiera swój polimorficzny odpowiednik tej metody. Określa czy ruch wykonany przez figurę jest możliwy.

Parametry

clickedCoordinates	Koordynaty do których poprawność przemieszczenia figury ma zostać sprawdzona.

Zwraca

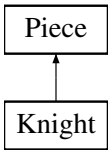
true - ruch prawidłowy.

false - ruch nieprawidłowy.

Implementuje Piece.

4.4 Dokumentacja klasy Knight

Diagram dziedziczenia dla Knight



Metody publiczne

- Knight (Board *board_, int x_, int y_, Color pieceColor_, sf::RectangleShape pieceShape_)

  Tworzy nowy objekt typu Knight dziedziczący z klasy Piece.
- bool isMoveCorrect (sf::Vector2i coordinatesToMoveTo)

  Metoda wirtualna - każda z sześciu podklas klasy Piece zawiera swój polimorficzny odpowiednik tej metody. Określa czy ruch wykonany przez figurę jest możliwy.

Dodatkowe Dziedziczone Składowe

4.4.1 Dokumentacja konstruktora i destruktora

4.4.1.1 Knight() `Knight::Knight ( Board * board_, int x_, int y_, Color pieceColor_, sf::RectangleShape pieceShape_ ) [inline]`

Tworzy nowy objekt typu Knight dziedziczący z klasy Piece.

Parametry

*board_*	Wskaźnik na szachownicę. Każda z figur wie na jakiej szachownicy się znajduje.
*x_*	Koordynat X szachownicy na którym znajduje się figura - szerokość.
*y_*	Koordynat Y szachownicy na którym znajduje się figura - wysokość.
*pieceColor_*	Kolor figury.
*pieceShape_*	Grafika figury szachowej.

4.4.2 Dokumentacja funkcji składowych

4.4.2.1 isMoveCorrect() `bool Knight::isMoveCorrect ( sf::Vector2i clickedCoordinates ) [virtual]`

Metoda wirtualna - każda z sześciu podklas klasy Piece zawiera swój polimorficzny odpowiednik tej metody. Określa czy ruch wykonany przez figurę jest możliwy.

Parametry

clickedCoordinates	Koordynaty do których poprawność przemieszczenia figury ma zostać sprawdzona.

Zwraca

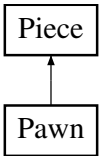
true - ruch prawidłowy.

false - ruch nieprawidłowy.

Implementuje Piece.

4.5 Dokumentacja klasy Pawn

Diagram dziedziczenia dla Pawn



Metody publiczne

- Pawn (Board *board_, int x_, int y_, Color pieceColor_, sf::RectangleShape pieceShape_)

  Tworzy nowy objekt typu Pawn dziedziczący z klasy Piece.
- bool isMoveCorrect (sf::Vector2i coordinatesToMoveTo)

  Metoda wirtualna - każda z sześciu podklas klasy Piece zawiera swój polimorficzny odpowiednik tej metody. Określa czy ruch wykonany przez figurę jest możliwy.

Dodatkowe Dziedziczone Składowe

4.5.1 Dokumentacja konstruktora i destruktora

4.5.1.1 Pawn()

```
Pawn::Pawn (
        Board * board_,
        int x_,
        int y_,
        Color pieceColor_,
        sf::RectangleShape pieceShape_ )  [inline]
```

Tworzy nowy objekt typu Pawn dziedziczący z klasy Piece.

Parametry

*board_*	Wskaźnik na szachownicę. Każda z figur wie na jakiej szachownicy się znajduje.
*x_*	Koordynat X szachownicy na którym znajduje się figura - szerokość.
*y_*	Koordynat Y szachownicy na którym znajduje się figura - wysokość.
*pieceColor_*	Kolor figury.
*pieceShape_*	Grafika figury szachowej.

4.5.2 Dokumentacja funkcji składowych

4.5.2.1 isMoveCorrect()

```
bool Pawn::isMoveCorrect (
        sf::Vector2i clickedCoordinates )  [virtual]
```

Metoda wirtualna - każda z sześciu podklas klasy Piece zawiera swój polimorficzny odpowiednik tej metody. Określa czy ruch wykonany przez figurę jest możliwy.

Parametry

clickedCoordinates	Koordynaty do których poprawność przemieszczenia figury ma zostać sprawdzona.

Zwraca

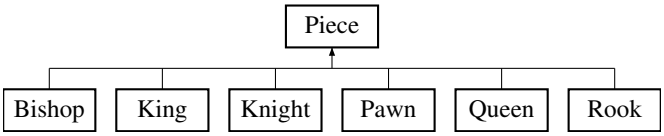
true - ruch prawidłowy.

false - ruch nieprawidłowy.

Implementuje Piece.

4.6 Dokumentacja klasy Piece

Diagram dziedziczenia dla Piece



Metody publiczne

- Piece ()

  Tworzy nowy objekt typu Piece.
- virtual bool isMoveCorrect (sf::Vector2i clickedCoordinates)=0

  Metoda wirtualna - każda z sześciu podklas klasy Piece zawiera swój polimorficzny odpowiednik tej metody. Określa czy ruch wykonany przez figurę jest możliwy.
- void placePieceOnBoard ()

  Umieszcza wskaźnik figury w odpowiednim miejscu tablicy boardArray klasy Board oraz przemieszcza grafikę figury w odpowiednie miejsce szachownicy.
- sf::RectangleShape getVisualOfPiece ()

  Zwraca grafike figury szachowej.
- void moveActivePiece ()

  Metoda realizuje przemieszczenie się figur, jest ograniczona przez metodę isMoveCorrect. Przemieszczanie się obejmuje:
- void removePiece (int x, int y)

  Usuwa figurę z boardArray klasy Board z koordynatów podanych w parametrach metody. Wykrywa czy usuniętą figurą jest któryś z królów i określa zwycięzce.
- std::string getPieceName ()

  Zwraca nazwę figury.
- int getMovesMade ()

  Zwraca ilość wykonanych ruchów przez figurę.
- void changePieceLocation (int x, int y)

  Zmienia lokacje figury. Zmiana odbywa się w dwóch miejscach: na zmiennych x i y objektu na którym metoda jest wykonywana oraz w tablicy boardArray reprezentującej szachownicę w klasie Board.

Atrybuty publiczne

- Color pieceColor = Color::White

  Kolor figury.

Atrybuty chronione

- Board ∗ board = NULL

  Wskaźnik na szachownicę. Każda z figur wie na jakiej szachownicy się znajduje.
- int **x** = 0
- int y = 0

  Koordynaty na których znajduje się figura.
- int movesMade = 0

  Ilość wykonanych ruchów przez figurę.

- sf::RectangleShape visualOfPiece

  Grafika figury szachowej.
- std::string pieceName = "Abstract piece"

  Nazwa figury.
- PieceType pieceType = PieceType::Abstract

  Typ figury.

4.6.1 Dokumentacja funkcji składowych

4.6.1.1 changePieceLocation() `void Piece::changePieceLocation ( int x, int y )`

Zmienia lokacje figury. Zmiana odbywa się w dwóch miejscach: na zmiennych x i y objektu na którym metoda jest wykonywana oraz w tablicy boardArray reprezentującej szachownicę w klasie Board.

Parametry

x	Koordynat X szachownicy - szerokość.
y	Koordynat Y szachownicy - wysokość.

4.6.1.2 getMovesMade() `int Piece::getMovesMade ( )`

Zwraca ilość wykonanych ruchów przez figurę.

Zwraca

int Ilość wykonanych ruchów przez figurę.

4.6.1.3 getPieceName() `std::string Piece::getPieceName ( )`

Zwraca nazwę figury.

Zwraca

std::string Nazwa figury.

4.6.1.4 getVisualOfPiece() `sf::RectangleShape Piece::getVisualOfPiece ( )`

Zwraca grafike figury szachowej.

Zwraca

sf::RectangleShape Grafika figury szachowej.

4.6.1.5 isMoveCorrect() `virtual bool Piece::isMoveCorrect ( sf::Vector2i clickedCoordinates ) [pure virtual]`

Metoda wirtualna - każda z sześciu podklas klasy Piece zawiera swój polimorficzny odpowiednik tej metody. Określa czy ruch wykonany przez figurę jest możliwy.

Parametry

clickedCoordinates	Koordynaty do których poprawność przemieszczenia figury ma zostać sprawdzona.

Zwraca

true - ruch prawidłowy.

false - ruch nieprawidłowy.

Implementowany w Pawn, Bishop, Knight, Rook, Queen i King.

4.6.1.6 moveActivePiece() `void Piece::moveActivePiece ( )`

Metoda realizuje przemieszczenie się figur, jest ograniczona przez metodę isMoveCorrect. Przemieszczanie się obejmuje:

- ruch do miejsca pustego
- ruch do miejsca już zajętego
- roszada pomiędzy królem a wieżą

4.6.1.7 removePiece() `void Piece::removePiece ( int x, int y )`

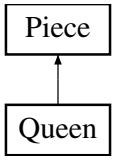
Usuwa figurę z boardArray klasy Board z koordynatów podanych w parametrach metody. Wykrywa czy usuniętą figurą jest któryś z królów i określa zwycięzce.

Parametry

x	Koordynat X szachownicy - szerokość.
y	Koordynat Y szachownicy - wysokość.

4.7 Dokumentacja klasy Queen

Diagram dziedziczenia dla Queen



Metody publiczne

- Queen (Board *board_, int x_, int y_, Color pieceColor_, sf::RectangleShape pieceShape_)

  Tworzy nowy objekt typu Queen dziedziczący z klasy Piece.
- bool isMoveCorrect (sf::Vector2i coordinatesToMoveTo)

  Metoda wirtualna - każda z sześciu podklas klasy Piece zawiera swój polimorficzny odpowiednik tej metody. Określa czy ruch wykonany przez figurę jest możliwy.

Dodatkowe Dziedziczone Składowe

4.7.1 Dokumentacja konstruktora i destruktora

4.7.1.1 Queen()

```
Queen::Queen (
        Board * board_,
        int x_,
        int y_,
        Color pieceColor_,
        sf::RectangleShape pieceShape_ )  [inline]
```

Tworzy nowy objekt typu Queen dziedziczący z klasy Piece.

Parametry

*board_*	Wskaźnik na szachownicę. Każda z figur wie na jakiej szachownicy się znajduje.
*x_*	Koordynat X szachownicy na którym znajduje się figura - szerokość.
*y_*	Koordynat Y szachownicy na którym znajduje się figura - wysokość.
*pieceColor_*	Kolor figury.
*pieceShape_*	Grafika figury szachowej.

4.7.2 Dokumentacja funkcji składowych

4.7.2.1 isMoveCorrect() `bool Queen::isMoveCorrect ( sf::Vector2i clickedCoordinates ) [virtual]`

Metoda wirtualna - każda z sześciu podklas klasy Piece zawiera swój polimorficzny odpowiednik tej metody. Określa czy ruch wykonany przez figurę jest możliwy.

Parametry

clickedCoordinates	Koordynaty do których poprawność przemieszczenia figury ma zostać sprawdzona.

Zwraca

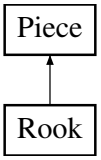
true - ruch prawidłowy.

false - ruch nieprawidłowy.

Implementuje Piece.

4.8 Dokumentacja klasy Rook

Diagram dziedziczenia dla Rook



Metody publiczne

- Rook (Board *board_, int x_, int y_, Color pieceColor_, sf::RectangleShape pieceShape_)

  Tworzy nowy objekt typu Rook dziedziczący z klasy Piece.
- bool isMoveCorrect (sf::Vector2i coordinatesToMoveTo)

  Metoda wirtualna - każda z sześciu podklas klasy Piece zawiera swój polimorficzny odpowiednik tej metody. Określa czy ruch wykonany przez figurę jest możliwy.

Dodatkowe Dziedziczone Składowe

4.8.1 Dokumentacja konstruktora i destruktora

4.8.1.1 Rook() `Rook::Rook ( Board * board_, int x_, int y_, Color pieceColor_, sf::RectangleShape pieceShape_ ) [inline]`

Tworzy nowy objekt typu Rook dziedziczący z klasy Piece.

Parametry

*board_*	Wskaźnik na szachownicę. Każda z figur wie na jakiej szachownicy się znajduje.
*x_*	Koordynat X szachownicy na którym znajduje się figura - szerokość.
*y_*	Koordynat Y szachownicy na którym znajduje się figura - wysokość.
*pieceColor_*	Kolor figury.
*pieceShape_*	Grafika figury szachowej.

4.8.2 Dokumentacja funkcji składowych

4.8.2.1 isMoveCorrect() `bool Rook::isMoveCorrect ( sf::Vector2i clickedCoordinates ) [virtual]`

Metoda wirtualna - każda z sześciu podklas klasy Piece zawiera swój polimorficzny odpowiednik tej metody. Określa czy ruch wykonany przez figurę jest możliwy.

Parametry

clickedCoordinates	Koordynaty do których poprawność przemieszczenia figury ma zostać sprawdzona.

Zwraca

true - ruch prawidłowy.

false - ruch nieprawidłowy.

Implementuje Piece.

5 Dokumentacja plików

5.1 Dokumentacja pliku chess.h

```
#include <iostream>
#include "SFML/Graphics.hpp"
```

Komponenty

- class Board
- class Piece
- class King
- class Queen
- class Rook
- class Knight
- class Bishop
- class Pawn

Wyliczenia

- enum **Color** { **White**, **Black** }
- enum **PieceType** {
  King, **Queen**, **Rook**, **Knight**,
  Bishop, **Pawn**, **Abstract** }

Indeks