DOKUMENTACJA TECHNICZNA

Gra „Chińczyk”

O projekcie

Projekt przedstawia grę komputerową „Chińczyk”

Stępień Kamil
GitHub: KamilStepien

Toczek Dominik
GitHub: Toczq

Rup Adam
GitHub: AdamRup

Spis treści

1. Opis Projektu

Projekt jest grą komputerową opierającą się na zasadach gry planszowej „Chińczyk”.
Projekt składa się z aplikacji WPF, biblioteki klas „ChinczykLib” oraz testów jednostkowych.
Biblioteka klas zawiera 8 klas oraz 3 interface’y potrzebne do realizacji logiki aplikacji.

2. Zasady Gry

Gra dla 2-4 osób, w której celem graczy jest przejechanie dookoła planszy czterema pionkami z pozycji początkowych na końcowe. Pierwszy gracz, który tego dokona, wygrywa.

Wygląd planszy z początkowym ustawieniem pionów:

[tu wstawić zdjęcie planszy]

Ruch w grze polega na przesunięciu na planszy dowolnego z własnych pionów o tyle pól, ile oczek zostanie wyrzuconych na kostce (1-6). Jeśli na docelowym polu znajduje się pion przeciwnika, to jest on usuwany i wraca na jedną z początkowych pozycji, a jeśli znajduje się tam własny pion, to taki ruch jest niedozwolony.

Wyrzucając 6 oczek na kostce, gracz wykonuje dodatkowy ruch, ale nie może wykonać pod rząd więcej niż 3 takie ruchy.

Aby wyjechać pionem z pozycji początkowej konieczne jest wyrzucenie „szóstki”. Dowolny pion jest wtedy umieszczany na pierwszym polu planszy oznaczonym w kolorze pionów gracza. Jeśli gracz posiada wszystkie swoje piony na pozycjach początkowych to przysługują mu trzy próby na wyrzucenie „szóstki” wymaganej do wyjechania dowolnym z pionów.

3. Charakterystyka Interfejsu Użytkownika

Interfejs Użytkownika składa się z 4 okien aplikacji WPF – okno główne, okno z zasadami gry, okno konfiguracji nowej gry oraz okno właściwe z planszą gry.
Po uruchomienia aplikacji pojawia się okno startowe, w którym użytkownik ma do wyboru 3 opcje:

- Nowa gra
- Zasady
- Wyjście

Po wybraniu przycisku „Wyjście” program kończy swoje działanie i aplikacja zostaje zamknięta.

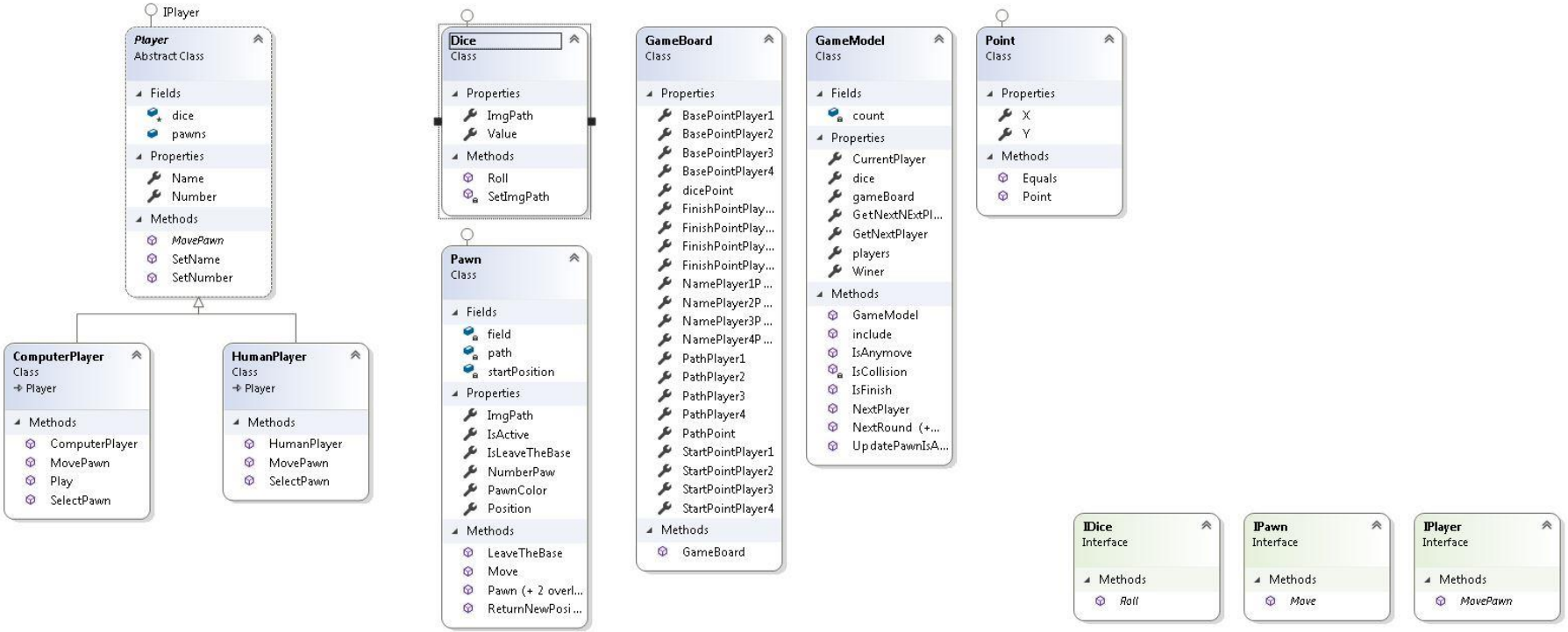
Opcja „Zasady” przenosi użytkownika do nowego okna, w którym jest opis zasad gry w „Chińczyka”. Po zaznajomieniu się z regułami gry, użytkownik może powrócić do głównego menu poprzez kliknięcie przycisku „powrót”.

Gdy użytkownik kliknie na przycisk „Nowa gra” aplikacja otworzy nowe okno, w którym użytkownik będzie mógł wybrać ilość graczy oraz poziom trudności, a następnie rozpocząć grę poprzez kliknięcie przycisku „Start”.

Po rozpoczęciu gry zostanie wyświetlona plansza gry, na której znajdują się piony graczy. Na środku planszy znajduje się przycisk służący do „rzutu kostką”. W każdej fazie gry, użytkownik może przerwać swoją rozgrywkę poprzez wybranie przycisku „Zakończ”, który przeniesie użytkownika do okna głównego menu

4. Charakterystyka Biblioteki Klas „ChinczykLib”

4.1 Diagram Klas



4.2 Klasa Dice

Klasa definiująca obiekt kostki do gry.

4.2.1 Pola i właściwości

- **ImgPath –** Właściwość przechowująca ścieżkę do zdjęcia kostki z odpowiednią wartością.
  Typ danych: string
- **Value –** Właściwość przechowująca wartość na kostce do gry (liczba oczek).
  Typ danych: int

4.2.2 Metody

- **SetImgPath() –** Metoda ustawiająca ścieżkę do zdjęcia z odpowiednią wartością na kostce.
  Typ zwracany: void
- **Roll() –** Metoda symulująca rzut kostką (losowy wybór wartości kostki z zakresu 1-6). Przypisuje wylosowaną wartość do właściwości Value.
  Typ zwracany: void

4.3 Klasa Pawn

Klasa definiująca obiekt pionka w grze.

4.3.1 Pola i właściwości

- **field**
  Typ danych: int
- **path –** Zmienna przechowująca ścieżkę, po której pionek przemieszcza się po planszy
  Typ danych: Point[]
- **startPosition –** Zmienna przechowująca punkt początkowy na planszy od którego rozpoczyna się rozgrywka
  Typ danych: Point
- **IsLeaveTheBase**
  Typ danych: bool
- **IsActive**
  Typ danych: bool
- **ImgPath –** Właściwość przechowująca ścieżkę do zdjęcia pionka.
  typ danych: string
- **Position –** Właściwość przechowująca pozycję pionka na planszy.
  Typ danych: Point
- **PawnColor –** Właściwość przechowująca kolor pionka.
  Typ danych: string
- **NumberPawn –** Właściwość przechowująca unikalny numer pionka.
  Typ danych: int

4.3.2 Metody

- **Pawn(int, Point, Point[], Point, string, string) –** Konstruktor obiektu klasy Pawn 6-argumentowy.
  Parametry:
  - int_NumberPawn
  - Point _Position
  - Point[] _Path
  - Point _StartPosition
  - string _PawnColor
  - string _ImgePath

- **Move(int) –** Metoda przesuwająca pionek o daną liczbę pól na planszy.
  Parametry:
  - int NewPosition
  Typ zwracany: void
- **ReturnNewPosition(int) –** Metoda sprawdzająca, czy ruch pionka o daną liczbę pól jest możliwy i zwracająca nową pozycję pionka.
  Parametry:
  - int NewPosition
  Typ zwracany: Point
- **LeaveTheBase() –** Metoda przesuwająca pionek z pozycji bazowej na pozycję początkową(pierwsze pole na planszy).
  Typ zwracany: void

4.4 Klasa Player

Klasa abstrakcyjna definiująca gracza.

4.4.1 Pola i właściwości

- **Name** – Właściwość przechowująca nazwę gracza.
  Typ danych: string
- **Number** – Właściwość przechowująca unikalny numer gracza
  Typ danych: int
- **pawns** – Zmienna przechowująca pionki gracza
  Typ danych: Pawns[]
- **dice** – zmienna przechowująca kostkę do gry
  Typ danych: Dice

4.4.2 Metody

- **MovePawn(Pawn)** – Metoda abstrakcyjna przesuwająca wskazany pionek gracza na planszy
  Parametry:
  - Pawn
  Typ zwracany: void
- **SetNumber(int)** – Metoda ustawiająca unikalny numer gracza
  Parametry:
  - int playerNumber
  Typ zwracany: void
- **SetName(string)** – Metoda ustawiająca imię/nick gracza
  Parametry:
  - string playerName
  Typ zwracany: void

4.5 Klasa HumanPlayer

Klasa definiująca obiekt gracza użytkownika i dziedzicząca po klasie abstrakcyjnej Player.

4.5.1 Pola i właściwości

-

4.5.2 Metody

- **HumanPlayer(string, int, Point[], Point[], Point, Dice)** – Konstruktor obiektu klasy HumanPlayer 6 argumentowy
  Parametry:
  - string playerName
  - int playerNumber
  - Point[] pawnPosition
  - Point[] pawnPath
  - Point pawnStart
  - Dice
- **MovePawn(Pawn)** – Metoda przesuwająca pionek gracza na planszy
  Parametry**:**
  - Pawn
  Typ zwracany: void
- **SelectPawn(int)** – Metoda zwracająca pionek, który ma zostać przesunięty
  Parametry:
  - int pawnNumber
  Typ zwracany: Pawn

4.6 Klasa ComputerPlayer

Klasa definiująca obiekt gracza komputerowego

4.6.1 Pola i właściwości

-

4.6.2 Metody

- **ComputerPlayer(int, Point[], Point[], Point, Dice)** – Konstruktor obiektu klasy ComputerPlayer 5-argumentowy
  Parametry:
  - int playerNumber
  - Point[] pawnPosition
  - Point[] pawnPath
  - Point pawnStart
  - Dice
- **Play()** – Metoda wywołująca inne metody dla obiektu ComputerPlayer i przyspieszająca realizację ruchu gracza komputerowego
  Typ zwracany: void
- **MovePawn(Pawn)** – Metoda przesuwająca pionek gracza komputerowego na planszy
  Parametry**:**
  - Pawn
  Typ zwracany: void
- **SelectPawn()** – Metoda automatycznie wybierająca pionek gracza komputerowego
  Typ zwracany: Pawn

4.7 Klasa Point

Klasa definiująca pozycję na planszy.

4.7.1 Pola i właściwości

- **X** - Zmienna przechowująca współrzędną X na planszy
  Typ danych: int

- **Y –** Zmienna przechowująca współrzędną Y na planszy
  Typ danych: int

4.7.2 Metody

- **Point(int, int)** – Konstruktor obiektu klasy Point 2-argumentowy
  Parametry:
  - int x
  - int y
- **Equals** – Metoda porównująca 2 obiekty typu Point
  Parametry:
  - Point other
  Typ zwracany: bool

4.8 Klasa GameBoard

Klasa definiująca obiekt planszy gry.

4.8.1 Pola i właściwości

- **PathPoint** – Właściwość przechowująca pola, po których poruszają się pionki
  Typ danych: Point[]
- **PathPlayer1** – Właściwość przechowująca tablicę punktów, po których porusza się gracz 1
  Typ danych: Point[]
- **PathPlayer2** – Właściwość przechowująca tablicę punktów, po których porusza się gracz 2
  Typ danych: Point[]
- **PathPlayer3** – Właściwość przechowująca tablicę punktów, po których porusza się gracz 3
  Typ danych: Point[]
- **PathPlayer4** – Właściwość przechowująca tablicę punktów, po których porusza się gracz 4
  Typ danych: Point[]
- **StartPointPlayer1** – Właściwość przechowująca punkt, w którym gracz 1 rozpoczyna okrążenie planszy
  Typ danych: Point
- **StartPointPlayer2** – Właściwość przechowująca punkt, w którym gracz 2 rozpoczyna okrążenie planszy
  Typ danych: Point
- **StartPointPlayer3** – Właściwość przechowująca punkt, w którym gracz 3 rozpoczyna okrążenie planszy
  Typ danych: Point
- **StartPointPlayer4** – Właściwość przechowująca punkt, w którym gracz 4 rozpoczyna okrążenie planszy
  Typ danych: Point
- **BasePointPlayer1** – Właściwość przechowująca tablice punktów, na których gracz 1 ustawia pionki na początku rozgrywki
  Typ danych: Point[]
- **BasePointPlayer2** – Właściwość przechowująca tablice punktów, na których gracz 2 ustawia pionki na początku rozgrywki
  Typ danych: Point[]
- **BasePointPlayer3** – Właściwość przechowująca tablice punktów, na których gracz 3 ustawia pionki na początku rozgrywki
  Typ danych: Point[]

- **BasePointPlayer4** – Właściwość przechowująca tablice punktów, na których gracz 4 ustawia pionki na początku rozgrywki
  Typ danych: Point[]
- **FinishPointPlayer1** – Właściwość przechowująca tablicę punktów, na których gracz 1 kończy obieg planszy
  Typ danych: Point[]
- **FinishPointPlayer2** – Właściwość przechowująca tablicę punktów, na których gracz 2 kończy obieg planszy
  Typ danych: Point[]
- **FinishPointPlayer3** – Właściwość przechowująca tablicę punktów, na których gracz 3 kończy obieg planszy
  Typ danych: Point[]
- **FinishPointPlayer4** – Właściwość przechowująca tablicę punktów, na których gracz 4 kończy obieg planszy
  Typ danych: Point[]
- **dicePoint** – Właściwość przechowująca punkt, w którym znajduje się kostka do gry
  Typ danych: Point
- **NamePlayer1Point** – Właściwość przechowująca punkt, w którym będzie wyświetlana nazwa gracza 1
  Typ danych: Point
- **NamePlayer2Point** – Właściwość przechowująca punkt, w którym będzie wyświetlana nazwa gracza 2
  Typ danych: Point
- **NamePlayer3Point** – Właściwość przechowująca punkt, w którym będzie wyświetlana nazwa gracza 3
  Typ danych: Point
- **NamePlayer4Point** – Właściwość przechowująca punkt, w którym będzie wyświetlana nazwa gracza 4
  Typ danych: Point

4.8.2 Metody

- **GameBoard(int, int)** – Konstruktor obiektu klasy GameBoard 2-argumentowy
  Parametry:
  - int width
  - int height

4.9 Klasa GameModel

Klasa definiująca model gry.

4.9.1 Pola i właściwości

- **Players** – Właściwość przechowująca tablicę wszystkich graczy
  Typ danych: Player[]
- **GameBoard** – Właścowość przechowująca planszę do gry
  Typ danych: GameBoard
- **Dice** – Właściwość przechowująca kostkę do gry
  Typ danych: Dice
- **Winer** – Właściwość przechowująca gracza-zwycięzcę gry
  Typ danych: Player
- **CurrentPlayer** – Właściwość przechowująca aktualnie grającego gracza
  Typ danych: Player

- **GetNextPlayer** – Właściwość przechowująca kolejnego gracza, który będzie wykonywał ruch
  Typ danych: Player
- **GetNextNextPlayer** – Właściwość przechowująca drugiego w kolejności gracza, który będzie wykonywał ruch
  Typ danych: Player

4.9.2 Metody

- **GameModel(int, string, string, string, string)** – Konstruktor obiektu klasy GameModel 5-argumentowy
  Parametry:
  - int numberHumanPlayer
  - string Player1Name
  - string Player2Name
  - string Player3Name
  - string Player4Name
- **NextRound()** – Metoda pozwalająca przejść do kolejnej rundy dla gracza komputerowego
  Typ zwracany: bool
- **NextRound(int)** – Metoda pozwalająca przejść do kolejnej rundy dla gracza komputerowego
  Parametry:
  - int idPawn
  Typ zwracany: bool
- **NextPlayer()** – Metoda, która zwraca i zapisuje do zmiennej następnego gracza
  Typ zwracany: Player
- **UpdatePawnIsActive()** – Metoda aktualizująca pole IsActive
  Typ zwracany:void
- **IsCollision(Pawn)** – Metoda sprawdzająca czy po wykonaniu ruchu pionek nie będzie na takiej samej pozycji z innym pionkiem
  Parametry:
  - Pawn
  Typ zwracany: bool
- **IsAnyMove()** – Metoda sprawdzająca czy na planszy są obiekty którymi można się poruszać
  Typ zwracany: bool
- **IsFinish()** – Metoda sprawdzająca czy gra dobiegła końca
  Typ zwracany: bool
- **Include(Point, Point[])** – Metoda sprawdzająca czy obiekt jest elementem tablicy
  Parametry:
  - Point p
  - Point[] tab
  Typ zwracany: bool