Gobblet Gobblers

Cel i opis projektu

Gobblet Gobblers to rozszerzona wersja popularnej gry "Kółko i Krzyżyk". Gracz ma do dyspozycji pionki różnych rozmiarów, które może stawiać na planszy (lub przestawiać już postawione) i zakrywać nimi pionki przeciwnika. Zwycięża ten gracz, który jako pierwszy postawi "gobblety" w jednej linii. Projekt ten miał na celu stworzenie aplikacji umożliwiającej grę w Gobblet Gobblers z drugim graczem oraz komputerem. Aplikacja zakładała możliwość wyboru rozmiaru planszy, na której gracz chce grać.

Opis programu

Plik `game.py`

Wyjątek `PieceUnavailableError`

Jeśli program "podnosi" ten wyjątek, oznacza to, że pionek, który gracz chce postawić na planszy, został albo już wcześniej postawiony i gracz nie ma już dostępnego żadnego takiego pionka, albo taki pionek nie istnieje w ogóle.

Wyjątek `NotOnBoardError`

Jeśli program "podnosi" ten wyjątek, oznacza to, że gracz chce przestawić pionek, który nie został jeszcze postawiony na planszy.

Wyjątek `CantCoverPieceError`

Jeśli program "podnosi" ten wyjątek, oznacza to, że gracz chce postawić (lub przenieść) pionek na pozycję, gdzie znajduje się inny pionek większy od niego (nie może być "zakryty").

Klasa `Game`

Jest to klasa reprezentująca pojedynczą rozgrywkę Gobblet Gobblers. Na wejściu pobiera od gracz informację, jakiego rozmiaru ma być plansza (rozmiar musi być liczbą całkowitą od 3 do 9) - na przykład jeśli gracz poda liczbę 4, to zostanie utworzona plansza o rozmiarze 4x4.

Klasa oprócz planszy do gry tworzy zestaw pionków dla obu graczy (po 2 każdego rozmiaru) oraz listę ostatnich układów pionków na planszy. Odpowiada ona też za wykonywanie ruchów oraz ustalanie zwycięzcy rozgrywki.

Plik `ai.py`

Wyjątek `CouldNotMoveError`

Jeśli program "podnosi" ten wyjątek, oznacza to, że komputer nie był w stanie wykonać ruchu swoim pionkiem.

Klasa `Ai`

Jest to klasa reprezentująca gracza komputerowego. Pobiera ona informację o grze, w której ma wykonywać ruchy (jako drugi gracz). Algorytm gracza komputerowego działa w następujący sposób:

1. Komputer sprawdza, czy może wygrać grę,
2. Jeśli nie może wygrać gry, sprawdza, czy przeciwnik ma szansę na wygraną i próbuje go zablokować stawiając swój pionek w jego linii.
3. Jeśli żadna z powyższych opcji nie jest możliwa, wykonuje losowy ruch.

Przy sprawdzaniu szansy na wygraną, komputer priorytetyzuje przestawienie już postawionego pionka.

Plik `ui.py`

Klasa `Interface`

Jest to klasa przetwarzająca informacje zwracane przez klasę `Game` i przedstawiające je w przyjazny dla użytkownika sposób. Pobiera informację o danej rozgrywce, oraz czy była ona przeciwko komputerowi czy drugiemu graczowi. Klasa przetwarza informację o układzie planszy, pionkach graczy i ewentualnej wygranej.

Plik `main.py`

Funkcja `menu`

Jest to funkcja odpowiadająca za wyświetlanie menu głównego gry.

Funkcja `game`

Jest to funkcja odpowiadająca za interfejs użytkownika podczas rozgrywki.

Funkcja `move`

Jest to funkcja odpowiadająca za interfejs użytkownika podczas wykonywania ruchu. Zadaje graczowi pytania czy chce postawić nowy pionek czy przestawić istniejący oraz prosi o koordynaty pozycji, na którą ma być przestawiony pionek.

Funkcja `game_status`

Wyświetla planszę, pionki graczy oraz ewnetualne komunikaty o nieprawidłowym ruchu, podniesionych przez program wyjątkach, itp.

Funkcja `main`

Główna funkcja programu, wywołuje ona menu główne oraz przetwarza komendy użytkownika.

Funkcja `game_creation`

Funkcja pomocnicza do funkcji `main`, pyta gracza o rozmiar planszy.

Instrukcja użytkownika

Aby pobrać program, należy sklonować jego repozytorium, wpisując w terminalu następującą komendę:

```
git clone https://gitlab-stud.elka.pw.edu.pl/mnowak1/gobblet-gobblers.git
```

Do poprawnego działania program wymaga biblioteki **termcolor**. Można ją zainstalować, wpisując następującą komendę:

```
pip install termcolor
```

lub, po uprzednim przejściu do katalogu z programem:

```
pip install -r requirements.txt
```

Po wykonaniu wyżej wymienonych kroków, można uruchomić program, przechodząc do katalogu, do którego zostało sklonowane repozytorium (domyślnie *gobblet-gobblers*) i uruchamiając plik `main.py`.

Refleksje

Aby zrealizować cel projektu, napisałem:

- "silnik" gry, odpowiadający za logikę całego programu
- komputer, który nie wykonuje ruchów w sposób kompletnie losowy
- prosty interfejs użytkownika

Niestety nie udało mi się napisać komputera w sposób taki jak chciałem - w założeniach miał on przeszukiwać drzewo, jednak ze względu na brak wystarczającej wiedzy jak zaimplementować takie drzewo oraz za mało czasu musiałem porzucić ten pomysł. Problemem było też programowanie komputera samego w sobie - podczas testowania programu często napotykałem błędy, których naprawienie zajmowało czasami po kilka godzin, a obecne podejście komputera, żeby najpierw próbować przestawiać postawiony pionek, nie zawsze jest skuteczne. Poza tą jedną niedogodnością, praca nad projektem przebiegła całkiem sprawnie i jestem z niego zadowolony. Przy jego tworzeniu starałem się stosować podstawowe zasady, którymi kieruje się dobry programista - ograniczanie powtarzania tych samych bloków kodu, rozbijanie dużych problemów na mniejsze podproblemy, czy pisanie klas i funkcji w taki sposób, żeby można było je ponownie wykorzystać w innych miejscach w projekcie. Dlatego też uważam, że pomimo wymienionych wcześniej problemów, projekt wciąż zasługuje na wysoką ocenę.