PROJEKT PIPR 2023/2024 – „Statki”

Autor: Jerzy Muszyński, Informatyka gr. lab 202

Cel Projektu:

Głównym celem projektu było stworzenie interaktywnej aplikacji do gry w statki, pozwalającej na rozgrywkę z komputerem. Użytkownik miał zmierzyć się z „wirtualnym” przeciwnikiem, który naśladuje człowieka i jest w stanie wykonywać logiczne ruchy, w celu zatopienia wszystkich statków.

Opis Projektu:

Projekt został wykonany w języku Python a rozgrywka odbywa się poprzez okno konsoli. Do dyspozycji są plansze w rozmiarze od 2x2 do 26x26, na których ulokowanych może być maksymalnie 5 statków, które są zbudowane z 1-5 masztów. Użytkownik na zmianę z komputerem „strzela” w wybrane współrzędne do momentu zatopienia przez któregoś z nich wszystkich statków.

Struktura Plików

```
├── README.md
├── docs
│   └── PIPR-Projekt-dokumentacja.pdf
├── pytest.ini
├── requirements.txt
├── tests
│   ├── test_board.py
│   ├── test_game.py
│   ├── test_player.py
│   └── test_warship.py
└── warships
    ├── classes
    │   ├── board.py
    │   ├── game.py
    │   ├── player.py
    │   └── warship.py
    ├── play.py
    └── utils
        ├── board_io.py
        ├── consts.py
        ├── player_io.py
        └── system_io.py
```

Użyte Biblioteki:

pick 2.2.0
pytest 7.4.4

Użyte klasy:

Warship – klasa opisująca pojedynczy statek na planszy, każdy z nich wyróżnia się w szczególności rozmiarem i „blokami", które reprezentują poszczególne współrzędne tworzące ten statek. Statek może być maksymalnie 5 masztowy i przewidziano jedynie „jednowymiarowe" warianty, tj. zawsze jeden z wymiarów statku będzie wynosił 1 (długość lub szerokość)

Board – klasa opisująca planszę. Plansza jest zawsze macierzą kwadratową a jej możliwe wymiary należą do przedziału <2x2, 26x26>. Plansze NxN, gdzie N <= 4 zawierają N statków, natomiast plansze NxN, gdzie N > 4, zawierają zawsze 5 statków, (statki są obiektami klasy ***Warship***)

BasePlayer – klasa abstrakcyjna opisująca gracza. Każdy gracz charakteryzuje się swoją własną planszą (obiekt klasy ***Board***) oraz słownikiem zawierającym typy statków wraz z ich liczbą na planszy.

Player – klasa opisująca gracza - użytkownika. Dziedziczy po klasie ***BasePlayer*** i umożliwia wybór pozycji do ułożenia statków na planszy.

Ai – klasa opisująca „wirtualnego" gracza, dziedziczy po klasie ***BasePlayer***. Dla jej planszy ułożenie statków jest wybierane w pełni losowo. Wykonuje logiczne ruchy, na podstawie rezultatów poprzednich strzałów. Dokonuje stałego zestawienia wszystkich uprzednio wykonanych strzałów oraz strzałów trafionych aby następnie wybrać możliwe pozostałe „bloki" jeszcze nie zatopionych statków.

Game – klasa opisująca główną pętle gry oraz przechowująca dane dotyczące graczy i ich plansz. Zarządza kolejnością ruchów, pilnuje stanu gry i ustala rezultat na podstawie jej przebiegu.

Instrukcja Obsługi:

Przygotowanie Środowiska

Najpierw należy sklonować repozytorium na własną maszynę i przejść do głównego katalogu projektu

```
git clone https://gitlab-stud.elka.pw.edu.pl/jmuszyns/warships.git
cd warships
```

Program należy uruchomić w środowisku wirtualnym, którego konfiguracja znajduje się w pliku requirements.txt

Aby odpowiednio przygotować środowisko należy

1. Zainstalować moduł *virtualenv*
2. Utworzyć środowisko wirtualne
3. Uruchomić środowisko wirtualne
4. Zainstalować wymagane biblioteki
5. Uruchomić plik z główną pętlą gry

Na systemie **Linux**:

```
pip install virtualenv
python3 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
python3 warships/play.py
```

Na systemie **Windows**:

```
pip install virtualenv
python -m venv .venv
.\venv\Scripts\activate
pip install -r requirements.txt
python warships/play.py
```

Przygotowanie rozgrywki

Po prawidłowej konfiguracji środowiska wirtualnego użytkownik powinien zobaczyć główne menu gry, na którym wyświetlą się 3 dalsze możliwości interakcji z programem:

```
::::::::::::::::::::::::::::
 Welcome to Warships with AI
::::::::::::::::::::::::::::
1. Start Game
2. Instructions
3. Exit

Enter your choice (1-3):
```

Zaleca się aby przed rozpoczęciem rozgrywki użytkownik zapoznał się z krótką instrukcją wbudowaną do programu

Rozgrywka

Aby rozpocząć rozgrywkę, użytkownik powinien wybrać opcje ***1. Start Game*** z głównego menu gry. Na początku zostanie poproszony o wybranie rozmiaru planszy na której będzie chciał grać (**UWAGA**: wprowadza się tylko jedną liczbę, która musi być z zakresu <2,26>).

Następnie użytkownik otrzyma możliwość ulokowania swoich statków na wcześniej wybranej planszy. W zależności od systemu operacyjnego na którym będzie uruchomiony program, użytkownik będzie mógł wybrać pozycje swoich statków za pomocą klawiszy UP/DOWN lub wprowadzając odpowiednią liczbę przy której wyświetli się wybrana pozycja statków, dokonany wybór należy potwierdzić przyciskiem ENTER.

```
 Place your 2 mast warship

 -> ['A0', 'A1']
    ['B0', 'B1']
    ['A0', 'B0']
    ['A1', 'B1']
```

```
Place your 2 mast warship

  0: ['A0', 'A1']
  1: ['B0', 'B1']
  2: ['A0', 'B0']
  3: ['A1', 'B1']

Enter your choice (0-3):
```

Po ułożeniu statków rozpocznie się „właściwa" gra, polegająca na wymianie strzałów między komputerem a użytkownikiem. Podczas swojej tury użytkownik musi wprowadzić współrzędne w które będzie chciał oddać swój strzał. Współrzędne muszą być w formacie „A0", gdzie A jest indeksem pierwszej kolumny, a 0 indeksem pierwszego wiersza, w przeciwnym wypadku użytkownik otrzyma informacje o nieprawidłowym wczytaniu współrzędnych i będzie musiał je wprowadzić jeszcze raz (wprowadzone współrzędne należy naturalnie potwierdzić przyciskiem ENTER).

Gra toczy się do momentu zatopienia wszystkich statków przeciwnika, a informację o stanie gry użytkownik otrzymuje po każdym strzale w postaci wyświetlonej planszy

- '[#]' oznacza że w strzał w dane współrzędne był chybiony,
- '[x]' oznacza że strzał w dane współrzędne trafił w jakiś statek,
- '[o]' oznacza że w danych współrzędnych znajduje się jakiś nieukryty statek,
- '[ ]' oznacza że w danych współrzędnych nie ma statku bądź jest ukryty.

Testowanie

Testy jednostkowe należy uruchomić również z poziomu uprzednio przygotowanego środowiska wirtualnego, za pomocą komendy `.venv/bin/pytest` (Linux) lub `.venv\Scripts\pytest` (Windows). Powodem umieszczenia biblioteki ***pytest*** wewnątrz środowiska wirtualnego były konflikty jego konfiguracji i zewnętrznych bibliotek, co uniemożliwiało prawidłowe uruchomienie plików testowych

```
├── tests
│   ├── test_board.py
│   ├── test_game.py
│   ├── test_player.py
│   └── test_warship.py
```

Refleksje:

Rozważania dotyczące projektu chciałbym skupić przede wszystkim wokół zestawienia osiągnięć, błędów oraz obszarów do dalszego rozbudowywania i refleksji na temat ogólnego rezultatu końcowego.

✔ Tak jak było to planowane wcześniej, utworzony program miał umożliwić użytkownikowi grę z„komputerem”. Utworzona aplikacja pozwala na wymianę strzałów między użytkownikiem a „wirtualnym” przeciwnikiem. "Komputer" wykonuje swoje ruchy na podstawie logicznych „zasad”, tj. trafiając w jakiś statek, próbuje go zatopić analizując co może znajdować się w otoczeniu. Dodatkowo jest świadomy tego w których miejscach może znajdować się jakiś, jeszcze nietrafiony, statek bądź pozostały fragment, dzięki czemu przy kolejnych próbach omija współrzędne gdzie takiego statku na pewno być nie może.

✕ Niestety nie udało mi się zaimplementować poziomów trudności rozgrywki, które zaplanowałem na początku swojej pracy. Wymagało by to utworzenia kilku wariantów algorytmu strzelania przez „komputer”, które charakteryzowałyby się inną skutecznością w wyborze współrzędnych. Stwierdziłem, że problemem mogłaby być ocena takiej skuteczności i próba podtrzymania jej na stałym poziomie w każdej rozgrywce, dlatego ostatecznie zrezygnowałem z implementacji tego pomysłu i obecne rozwiązanie polega na pojedynczym algorytmie wyboru możliwych miejsc do strzału przez "komputer".

✕ Dodatkowym ograniczeniem obecnego rozwiązania może być fakt stałej liczby statków danego typu. Początkowo chciałem umożliwić użytkownikowi wybór liczby i typów statków którą będzie chciał zaalokować na swojej planszy, dzięki czemu rozgrywka mogłaby być ciekawsza i bardziej nietuzinkowa. Natomiast podczas implementacji tej funkcjonalności natknąłem się na problem w postaci nieprawidłowego wypełniania tej planszy, co często kończyło się tym, że na planszy pojawiało się mniej statków niż zostało początkowo zadeklarowanych.

✔ Podczas tego projektu przede wszystkim udało mi się nauczyć jak odpowiednio rozplanowywać zadaniaaby praca nad nimi była efektywna. Zanim zacząłem implementacje tego rozwiązania poświęciłem dość sporo czasu na dogłębną analizę wymagań i samej istoty przedstawionego problemu. Dopiero po zestawieniu przykładowych pomysłów i sprecyzowanych założeń przeszedłem do wstępnego budowania fundamentalnych struktur mojego projektu. Naturalnie, czasem pojawiały się błędy, przez które trzeba było wrócić jeszcze raz do analizy danej funkcjonalności, czasami też, dochodziło do wielokrotnej modyfikacji uprzednio wybranego podejścia, aby zapewnić pełną wartość końcowego „produktu”. Mimo to, uważam, że udało mi się uzyskać dość szerokie spojrzenie na cały proces budowania projektu i doświadczyć jak kompleksowo tworzyć oprogramowanie „od zera”.