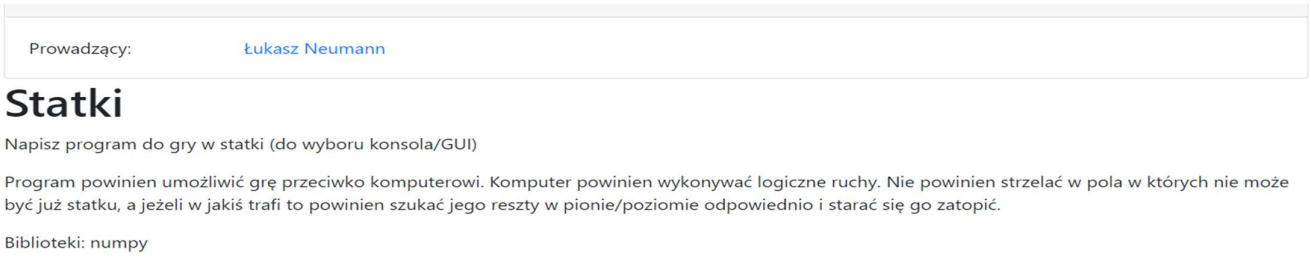
Dokumentacja końcowa
projekt PIPR (statki)

Cel i opis projektu.

Celem projektu było napisanie w języku Python gry w statki.



Prowadzący: Łukasz Neumann

Statki

Napisz program do gry w statki (do wyboru konsola/GUI)

Program powinien umożliwić grę przeciwko komputerowi. Komputer powinien wykonywać logiczne ruchy. Nie powinien strzelać w pola w których nie może być już statku, a jeżeli w jakiś trafi to powinien szukać jego reszty w pionie/poziomie odpowiednio i starać się go zatopić.

Biblioteki: numpy

Założeniem było, że gracz będzie mógł grać przeciwko komputerowi (dalej jako bot). Bot powinien postępować logicznie, co jest opisane na zdjęciu wyżej, to znaczy strzelać w pola, gdzie ma to sens oraz w sensowny sposób dążyć do zatopienia statku.

Podział programu na klasy.

Tworząc program zaprojektowałem 3 klasy: Ship, Fleet oraz Matrix.

Klasa Ship reprezentuje tytułowe statki. Przechowuje ich wielkość (size), czyli liczbę pól, które zajmują, lokację (location), czyli listę koordynatów/pól, na których się znajdują oraz ich kierunek (direction), czyli opis ich ułożenia; w zależności od statku jest to ‘pion’ lub ‘poziom’, a domyślnie wynosi ona None (dla statku jednokadłubowego).

Klasa Fleet reprezentuje flotę gracza lub bota. Przechowuje ona listę statków danego grającego oraz umożliwia dostęp do konkretnych statków.

Klasa Matrix reprezentuje macierz danego grającego. Do jej tworzenia używam array z biblioteki numpy.
Działa to w taki sposób, że początkowo macierz składa się z samych 0, co oznacza, że nie ma tam statków. Podczas tworzenia swoich statków/tworzenia statków przez bota wartości w miejscach znajdowania się statku zamieniane są na 1; przy trafieniu w element

statku jego wartość zamienia się na 2, a przy trafieniu w puste pole (0) jego wartość zamienia się na 3.

Instrukcja użytkownika.

Aby uruchomić grę, należy uruchomić funkcję main() z pliku game.py lub uruchomić plik game.py . Całość gry przeprowadzana jest w konsoli, gdzie użytkownik odczytuje instrukcje, polecenia i obrazki prezentowane przez grę.

Część refleksyjna.

Całość projektu udało mi się wykonać (jak na pierwsze kroki w programowaniu w ogóle) dosyć spójnie. Korzystając z wiedzy zdobytej na laboratoriach oraz dzięki filmom przed laboratoriami udało mi się napisać sprawnie działający kod. Zaimplementowałem logikę działań dla bota, która spełnia swoje zadania oraz wymagania przedstawione przez prowadzącego. Udało mi się stworzyć możliwość wyboru wielkości planszy oraz ilości statków, żeby można było w przyjemny sposób customizować grę wg swojego upodobania. Pracując nad projektem udało mi się znacznie uprościć funkcje strzału przez gracza i bota względem stanu początkowego, gdzie te funkcje były niepotrzebnie rozwlekłe.

Dopuszczałem myśl zwiększenia inteligencji bota poprzez wprowadzenie dodatkowego mechanizmu analizującego planszę gracza, tzn.: jeżeli w danym miejscu są wolne tylko 2 pola, a pozostaje do ‘ustrzelenia’ jeszcze statek 3-kadłubowy, bot nie będzie tam strzelał, jednak przerosło to moje umiejętności programistyczne.

Nieoczekiwanym problemem okazała się potrzebna mocnej analizy matematycznej całego problemu. Analiza taka była potrzebna do mechanizmu stawiania statków przez bota oraz do ustalania minimalnej wielkości planszy dla danej ilości statków, żeby w ogóle istniała możliwość swobodnego ułożenia statków.

Początkowo planowałem stworzyć standardową wersję statków, gdzie plansza ma wymiary 10x10 oraz największym statkiem jest statek 4-kadłubowy, jednak dzięki sugestiom prowadzącego mój kod ma szersze zastosowanie i, jak pisałem wyżej, pozostawia w kwestii planszy i ilości statków większą dowolność dla użytkownika.

Udało mi się spełnić wszystkie wymagania projektowe zapisane w treści zadania oraz zaimplementować funkcje, których istnienia

początkowo nie zakładałem. Gra działa w sposób przyjemny, a bot działa na tyle logicznie, że przy testowaniu gry nieraz poniosłem porażkę.

Jak na moje pierwsze spotkanie z programowaniem jestem wysoce zadowolony z mojego projektu, choć zdaję sobie sprawę, że pewne rzeczy jeszcze dało się poprawić, jednakże wykorzystałem wszystkie moje zasoby wiedzy programistycznej i matematycznej, żeby kod był jak najlepszy, a gra jak najsprawniejsza.

Adam Sudoł
gr. INF203