Projekt gry w Statki (Battleships)

Cel i opis projektu

Celem projektu było stworzenie programu do powszechnie znanej gry w statki. Do wyboru była realizacja gry z poziomu konsoli, bądź za pomocą GUI.

Zdecydowałam się na wykonanie gry w konsoli.

Podczas tworzenia projektu moim głównym celem było stworzenie prostej gry konsolowej w statki z jasnymi, przejrzystymi zasadami. Chciałam zapewnić graczowi możliwość doboru trudności gry, ustawienia własnych statków na planszy.

Do wykonania projektu wykorzystałam bibliotekę numpy.

Podział programu na klasy i ich opis

Wykonany przeze mnie projekt został podzielony na pięć klas:

Klasa Player – będąca klasą nadrzędną dla klas HumanPlayer oraz BotPlayer, która posiada kilka atrybutów oraz metod wspólnych obu tych klas. Stanowi swoistą osłonkę obu tych klas, gdyż klasy podrzędne ze względu na swój konkretny typ gracza (człowiek/ automat) mają różniące się pod względem działania metody. Posiada metody pozwalające na ustawienie statków, weryfikację danego pola (czy zostało ono wcześniej trafione), stwierdzenia, czy gracz wygrał daną rozgrywkę oraz pamięć gracza, która pozwala zapobiec wybraniu podczas rozgrywki współrzędnych, które zostały już wcześniej trafione przez gracza.

Klasa HumanPlayer – klasa dziedzicząca po klasie Player, reprezentująca w grze „żywego" gracza. Oprócz wspomnianych metod i atrybutów klasy Player posiada dodatkowe funkcje pozwalające na ustawienie statku na planszy (funkcje te otrzymują wybrane przez gracza współrzędne), własną reprezentację planszy (z odpowiednio zaznaczonymi polami) oraz swoją osobistą „mechanikę" trafiania w przeciwnika.

Klasa BotPlayer – klasa dziedzicząca po klasie Player, reprezentująca gracza – automat. Oprócz wspomnianych metod i atrybutów klasy Player posiada dodatkowy, cenny dla rozrywki atrybut – pamięć strzałów. Atrybut ten oraz metody związane z nim oraz na nim wykonywane powodują, że „komputerowy" gracz w trakcie rozgrywki dokonuje inteligentnych strzałów (czyli np. po trafieniu jednego pola statku przeciwnika stara się w swych kolejnych ruchach utopić jego statek (a nie wybiera kolejne losowe pole na planszy). Klasa ta, tak samo jak i klasa HumanPlayer posiada własną zdefiniowaną reprezentację graficzną planszy oraz mechanikę atakowania „ludzkiego gracza".

Klasa GameBoard – klasa reprezentująca planszę gracza. Jako swe atrybuty posiada rozmiar planszy, listę znajdujących się na niej statków. Ma metodę potrzebną do ustawienia statku na planszy, „uaktualnienia" statusu planszy - odznaczenia poszczególnych pól liczbą, która reprezentuje dany stan pola/ współrzędnej. Stany te zostały zdefiniowane następująco – „0" – wolne „niecelowane" pole, „1" – pole, na którym ustawiony jest statek, nie odkryte jeszcze przez przeciwnika, „2" – trafione pole statku, „3" – pudło.

Klasa Ship – klasa reprezentująca statek na planszy gracza. Posiada atrybuty, które definiują dany statek (jego nazwę, długość) oraz metody, które określają status statku – czy został on trafiony, czy jest zatopiony.

Program został podzielony na 5 plików ze względu na funkcjonalność/ klasy:

Plik battleships_game.py zawiera funkcje odpowiedzialne za inicjalizację oraz właściwą rozgrywkę/przebieg gry.

Plik game_board.py posiada wspomnianą wcześniej klasę GameBoard wraz ze swoimi metodami.

Plik players.py zawiera wyżej wymienione 3 klasy: Player, HumanPlayer oraz BotPlayer, które reprezentują użytkownika (bądź jego „komputerowego" przeciwnika) w grze. Zawiera wszystkie metody wyżej wspomnianych klas.

Plik ship.py posiada klasę Ship - jej funkcję, atrybuty oraz słownik globalny zawierający nazwy oraz rozmiar konkretnych statków.

Plik game_interface.py zawiera funkcje, które są odpowiedzialne za „komunikację" między programem a użytkownikiem. Znajdują się tu funkcje, które dekodują wpisane przez użytkownika wartości (np. współrzędnych na planszy) oraz te, które informują użytkownika o dalszych etapach rozgrywki (wynikach rundy)/

Instrukcja obsługi programu dla gracza

W celu uruchomieniu programu (gry) użytkownik powinien odpalić plik battleships_game.py przez interpreter, bądź z poziomu konsoli wpisując python3 battleships_game.py. Nie wymagane są żadne pliki konfiguracyjne/wejściowe.

Refleksja

Uważam, że jak na pierwszy „większy" projekt moja gra w statki wypadła całkiem dobrze. Choć myśl o stworzeniu od zera dobrze znanej mi z dzieciństwa gry wydawała się na początku przygnębiająca, to wykonanie jej pod względem koncepcyjnym nie okazało się być zbyt skomplikowane. Stworzenie gry sprowadza się poniekąd do zdefiniowania różnych operacji na matrycy. Projekt ten zdecydowanie „otworzył mi oczy" na to jak rzeczy, znane mi ze świata rzeczywistego, można za pomocą kodu rozbić do całkiem prostych i mało zawiłych operacji – a dzięki czemu przenieść je do świata „komputera". Cele, jakie postawiłam przed sobą na początku projektu, udało mi się w pełni wykonać – choć może na mój relatywny brak doświadczenia nie były one dość wysokie, to jestem zadowolona, że udało mi się je osiągnąć. Podczas tworzenia programu największe problemy przysparzało mi nie tworzenie samej rozrywki (jak mogłoby się zdawać), ale pisaniu kodu odpowiedzialnego za możliwość ustawiania statków na planszy (przez gracza). Szukałam najbardziej optymalnego rozwiązania „komunikacji" z graczem – myślę, że choć zapewne istnieje lepsza metoda na komunikację z graczem za pomocą konsoli, to ta, którą wymyśliłam również nie powinna być najgorsza. Bardzo ciekawe było moim zdaniem pisanie logiki gry dla „komputerowego gracza". Ze względu na to, że musiał on być inteligentny trzeba było wymyśleć konkretny schemat, proces „rozumowania", który kierowałby kolejnym ruchem automatu – jak ma on np. postępować w sytuacji, gdy trafił w statek „ludzkiego" gracza. Na początku wydawało mi się, że rozwiązanie tego aspektu zadania będzie dość proste – komputer przechowuje w „swej pamięci" współrzędną trafionego statku i jego kolejnymi polami planszy „obranymi" za cel, będą pola otaczające daną współrzędną. Podczas testowania gry okazało się, że nie jest to w pełni „mądre" rozumowanie komputera – w sytuacjach, gdy komputer trafił w dwa sąsiadujące ze sobą pola statku, było jasne, że dalsza część statku znajduje się wzdłuż osi, które te dwa punkty wyznaczają – jednak automat przy swych kolejnych ruchach nadal wybierał jedno z 4 możliwych pól sąsiadujących ze współrzędną/współrzędnymi, które znajdowały się w „pamięci trafionych strzałów komputera" (pamięć ta bowiem była za każdym razem odnawiana – przy kolejnym trafionym strzale dodawane były nowe współrzędne trafionego strzału).

Ostatecznie w ramach projektu udało mi się stworzyć grę w statki, której rozgrywka w dużym stopniu zależy od preferencji użytkownika – ma on bowiem możliwość ustawienia własnego imienia, wielkości

planszy (zakres rozmiarów planszy został dobrany tak, aby rozgrywka mogła posiadać kilka różnych poziomów trudności a jednocześnie nie była „za łatwa” ani „za trudna”). Gracz ma również w pełni możliwość doboru miejsca, w którym mogą znajdować się jego statki na planszy. Sposób prezentacji planszy gracza jest czytelny i jak na grę z poziomu konsoli wydaje się być jak najbardziej „przyjazny”. Opisy dalszych kroków w grze mam nadzieję, że jasno wyznaczają przebieg rozgrywki, a gracz nie „pogubi się w grze”.

Choć wiem, że na pewno istnieje wiele „lepszych” wersji gry w Statki, to moja zapewnia bezproblemową rozgrywkę z prostymi i jasnymi zasadami gry.

Jedyne, czego żałuję to to, że nie udało mi się zrobić możliwość ustawiania „prędkości gry” (zmniejszania „time.sleep()”), co również zostało mi zasugerowane podczas konsultacji. Nie ukrywam, że gdyby nie duża ilość obowiązków związanych z uczelnią z chęcią bym tę możliwość zaimplementowała.

Grę stworzyłam w konsoli, gdyż nie ukrywam, że ze względu na moją dość „krótką przygodę” z programowaniem, idea zrealizowania gry w GUI wydawała się na dany moment być wyzwaniem, z którym mogłabym sobie nie poradzić. Mam jednak nadzieję, że w przyszłości uda mi się powrócić do tego projektu i spróbować go wykonać na nowo, implementując lepsze algorytmy, mechanikę gry.

Samo doświadczenie „tworzenia gry” będę mile wspominać – praca we własnym tempie oraz możliwość obserwowania własnych postępów przynosiła mi dużo szczęścia i satysfakcji.

Małgorzata Kozłowska